Carlos Brito-Loeza
Arturo Espinosa-Romero (Eds.)

Communications in Computer and Information Science 820

Intelligent Computing Systems

Second International Symposium, ISICS 2018
Merida, Mexico, March 21–23, 2018
Proceedings

Springer

Communications in Computer and Information Science 820

Carlos Brito-Loeza · Arturo Espinosa-Romero (Eds.)

Intelligent Computing Systems

Second International Symposium, ISICS 2018
Merida, Mexico, March 21–23, 2018
Proceedings

Springer

Editors
Carlos Brito-Loeza
Autonomous University of Yucatán
Merida
Mexico

Arturo Espinosa-Romero
Autonomous University of Yucatán
Merida
Mexico

ISSN 1865-0929 ISSN 1865-0937 (electronic)
Communications in Computer and Information Science
ISBN 978-3-319-76260-9 ISBN 978-3-319-76261-6 (eBook)
https://doi.org/10.1007/978-3-319-76261-6

Library of Congress Control Number: 2018934326

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Recent technological advances such as the increase in computing power and wider and faster communication networks have made computing systems ubiquitous. Intelligent systems are computing systems capable of simulating the intelligent behavior unique to living beings, and researchers in this field take advantage of this technological growth to propose new algorithms and applications.

Intelligent systems are important, since the application of this field of study takes on a main role in many disciplines, where the development of automatic systems capable of making decisions on complex environments or assisting humans in tasks where the unique human perceptual capabilities are difficult to replace.

In this volume you will find many novel examples of research in intelligent systems: algorithms capable of learning from examples to recognize and classify objects in images, novel sensors and the computational methods needed to process and interpret their readings, neural network-based methods to predict time series and control and manage energy generation systems, variational image processing methods systems to segment and denoise images, as well as the processing of multimodal signals to use in the context of human–robot interaction.

This book contains the written contributions of the Second International Symposium on Intelligent Computing Systems (ISICS) that was held in Merida (Mexico), during March 21–23, 2018. To further increase the body of knowledge in this specific area of computer science was the aim of the ISICS 2018, by providing a forum in which to exchange ideas and discuss state-of-the-art results. ISICS 2018 was committed to the promotion, preservation, and collaboration of research and practice, focusing on the fields of artificial intelligence, computer vision, and image processing.

We received 28 submissions from seven countries around the world. Each submission was evaluated by at least three members of the Program Committee and external reviewers. Based on these reviews, 12 papers were selected for long oral presentation. In addition to the contributed papers, four keynote speaker presentations were included in the conference program.

We want to thank the authors for their contributions, the scientific Program Committee members for their reviews, and especially our invited speakers, Prof. Andrew Barto (University of Massachusetts, USA), Prof. Leo Joskowicz (Hebrew University of Jerusalem, Israel), Prof. Reinhard Klette (Auckland University of Technology, New Zealand), Prof. Jesús Savage Carmona, (Bio-Robotics Laboratory, Mexico), and Prof. Sajjad Mohsin (University of Islamabad, Pakistan). We are very grateful to the Universidad Nacional Autónoma de México (UNAM), the Centro de Investigaciones Matemáticas (CIMAT), and the Universidad Autónoma de Yucatán (UADY) for their support in the organization of the ISICS 2018.

March 2018

Carlos Brito-Loeza
Arturo Espinosa-Romero

Organization

Scientific Advisory Committee

Walterio Mayol-Cuevas	University of Bristol, UK
Adolfo Sánchez Valenzuela	Centro de Investigación en Matemáticas A. C., Mexico

Program Committee

Bassam Ali	Universidad Autónoma de Yucatán, Mexico
Noor Badshah	University of Engineering and Technology, Pakistan
Carlos Brito-Loeza	Universidad Autónoma de Yucatán, Mexico
Morgado Dias	Universidade da Madeira, Portugal
Arturo Espinosa-Romero	Universidad Autónoma de Yucatán, Mexico
Jorge Gomez-Montalvo	Universidad Autónoma de Yucatán, Mexico
Benjamín Gutierrez-Becker	Technische Universität München, Germany
Jean-Bernard Hayet	Centro de Investigación en Matemáticas A. C., Mexico
Francisco Javier Hernández López	Centro de Investigación en Matemáticas A. C., Mexico
Nidiyare Hevia-Montiel	IIMAS-UNAM, Mexico
Angel Kuri-Morales	Instituto Tecnológico Autónomo de México, Mexico
Ricardo Legarda-Saenz	Universidad Autónoma de Yucatán, Mexico
Stacey Levine	Duquesne University, USA
Eduardo Lleida	Universidad de Zaragoza, Spain
Elena Loli Piccolomini	University of Bologna, Italy
José Luis López Martínez	Universidad Autónoma de Yucatán, Mexico
Salvador Mandujano	Google Inc., USA
Anabel Martin-Gonzalez	Universidad Autónoma de Yucatán, Mexico
Rolando Medellín	University of Dundee, UK
Raúl Monroy Borja	ITESM-CEM, Mexico
Francisco Moo-Mena	Universidad Autónoma de Yucatán, Mexico
Nicolás Navarro-Guerrero	Aarhus University, Denmark
Luis A. Pineda Cortés	IIMAS-UNAM, Mexico
Lavdie Rada	Bahçesehir University, Turkey
Alonso Ramirez-Manzanares	Centro de Investigación en Matemáticas A. C., Mexico
Eduardo Rodriguez-Martinez	Universidad Autónoma Metropolitana, Mexico
Israel Sánchez-Domínguez	IIMAS-UNAM, Mexico
Asad Safi	COMSATS Institute of Information Technology, Pakistan
Sai Deep Tetali	Google Inc., USA
Victor Uc-Cetina	Universidad Autónoma de Yucatán, Mexico

Flavio Javier Vigueras Gómez	Universidad Autónoma de San Luis Potosí, Mexico
Fernando Von Borstel Luna	Centro de Investigaciones Biológicas del Noroeste, Mexico

Organizing Committee

Anabel Martin-Gonzalez	Universidad Autónoma de Yucatán, Mexico
Víctor Uc-Cetina	Universidad Autónoma de Yucatán, Mexico
Carlos Brito-Loeza	Universidad Autónoma de Yucatán, Mexico
Bassam Ali	Universidad Autónoma de Yucatán, Mexico
Arturo Espinosa-Romero	Universidad Autónoma de Yucatán, Mexico
Ricardo Legarda-Saenz	Universidad Autónoma de Yucatán, Mexico
Nidiyare Hevia-Montiel	Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas - UNAM, Mexico
Israel Sánchez-Domínguez	Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas - UNAM, Mexico
Eric Molino-Minero-Re	Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas - UNAM, Mexico
Francisco J. Hernández López	Centro de Investigación en Matemáticas A. C., Mexico

Contents

Object Recognition Using Hierarchical Temporal Memory

Fabián Fallas-Moya[1(✉)] and Francisco Torres-Rojas[2]

[1] Universidad de Costa Rica, Cartago, Costa Rica
fabian.fallasmoya@ucr.ac.cr
[2] Instituto Tecnológico de Costa Rica, Cartago, Costa Rica
torresrojas@gmail.com

Abstract. At this time, great effort is being directed toward developing problem-solving technology that mimic human cognitive processes. Research has been done to develop object recognition using Computer Vision for daily tasks such as secure access, traffic management, and robotic behavior. For this research, four different machine learning algorithms have been developed to overcome the computer vision problem of object recognition. Hierarchical temporal memory (HTM) is an emerging technology based on biological methods of the human cortex to learn patterns. This research applied an HTM algorithm to images (video sequences) in order to compare this technique against two others: support vector machines (SVM) and artificial neural networks (ANN). It was concluded that HTM was the most effective.

Keywords: Machine learning · Computer vision · HTM

1 Introduction

One relevant issue in computer science is to try to interpret an image for a specific purpose. This research focuses on recognizing objects in an image with different occlusion percentages. Right now, computers have the computational power to achieve this objective with an acceptable running time.

Singh [1] defines *video tracking* as the process to detect one moving object throughout a video sequence, and a fundamental task of this tracking is to recognize the object in every frame. An object of interest can be: an animal, a machine, a person, etc. A system to detect the existence of objects in a frame was developed. The dataset consisted of video sequences of people, and this gave different challenges related to shape transformation (because of object movements) and occlusion (when the object is partially occluded by other objects). Figure 1 shows an example of occlusion.

The human brain has a *neocortex*. This element is in charge of visual pattern recognition and many other cognitive processes. Hierarchical temporal memory (HTM) is a technology developed to try to mimic the neocortex. It is very similar to artificial neural network (ANN), but with some architectural and conceptual

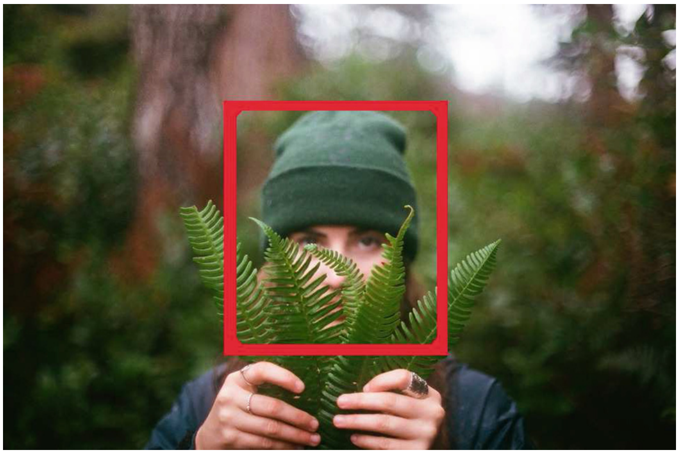


Fig. 1. An example of an occluded face.

differences. For example, the cells[1] are connected to different cells all over the region (see Fig. 2). These connections or *synapses* can change during the training period. Even more, they can change during classification. It depends on the input data. The HTM architecture have cells that are arranged into regions, and regions that can form a complex hierarchy.

This HTM technology was developed by Hawkins [2] and it is closely related to the biological structure of the neocortex that has six layers of regions. This explains why HTM uses the concepts of regions, cells, and hierarchies. The regions can be constructed using different numbers of cells. An important aspect is that the connections inside regions can change over time.

The idea behind hierarchies is to learn complex patterns. This is similar to the idea behind convolutional neural networks, as explained by Fan *et al.* [3]. There, every layer tries to detect gradient features to have an accurate classification process. In the case of HTM, every defined level[2] will be learning different degrees of an image, to get to the final layer. The idea is to learn complex patterns. See Fig. 3.

1.1 HTM and Pattern Learning

As well as in ANN, HTM network has a specific region for the input data. The outcome of this first layer is passed to the following region to be trained, and the process continues for every region until the last one.

HTM uses the concept of sparse distributed representations (SDR). It is a technique to represent different patterns, where a small number of cells are picked to be active.

[1] In the HTM terminology, cells are synonyms of neurons.
[2] The terms level and region will be used interchangeably.

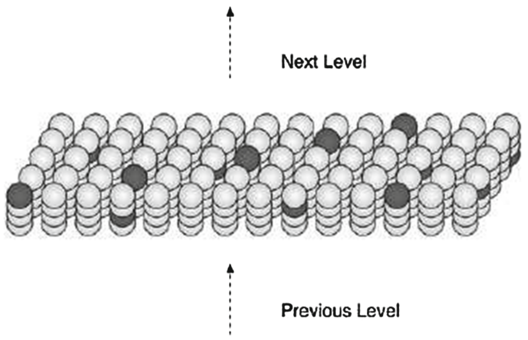


Fig. 2. An HTM region with columns of cells, the cells are connected to different cells. Also, the concept of sparse distributed representations (SDR), the patterns are just a few and distributed cells along the region.

HTM can receive an enormous input. However, every layer needs just a few cells to represent it. Higher layers apply the same concept until the final layer is reached. This process is not uniform or deterministic. This explains why different cells can be picked. In Fig. 2 shows that it is not mandatory for these cells to be close to each other. To summarize, only a few cells are need and these cells can be distributed along the region.

As Schlag [21] explains, the way HTM learns patterns is in a statistical manner, very similar to Bayesian Networks; however, they differ in the hierarchy construction and the usage of time. First, it converts raw data into proper HTM input, using some decoders (libraries developed for HTM). Second, from the data it looks for activations which occur together (spatial patterns). It then searches sequences of these patterns over time (temporal patterns). So, there is not a back propagation step, it does some kind of clustering classification. These learned patterns are used to perform inference on new inputs. In HTM the concept of *time* is important, the order in which the patterns enter the architecture impacts the connection construction in the network.

Just like a biological neuron, an HTM cell has dendrites, proximal and a distal dendrites. The dendrites contain many synapses in order to receive signals. The proximal dendrites receive a feed-forward input from regions in a lower layer. The distal receive their input from neighboring cells which belong to the same HTM region. Every cell is made up of a number of synapses, which can be potential or permanent. During the training process these synapses can change due to the classification result. Every cell can be in a different state: predictive, active and fully active. These states will defined the synapses. This cell flexibility to easily change its connections will help to make a fast training step. According to Schlag [21] (2016), “This increase and decrease in permanence is an important aspect of the Hebbian[3] learning ability of HTMs”.

[3] Hebbian Theory defined by Gerstner [22].

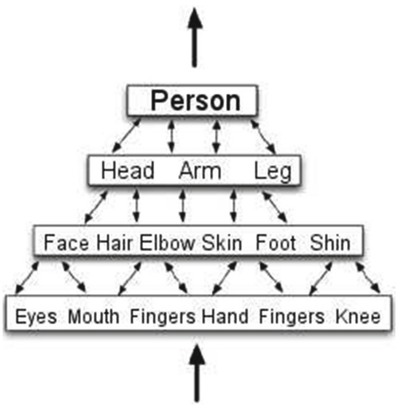

Person
Head Arm Leg
Face Hair Elbow Skin Foot Shin
Eyes Mouth Fingers Hand Fingers Knee

Fig. 3. An HTM hierarchy: in this example, the hierarchy has four levels of regions.

To have a full HTM implementation one can use the online prediction framework (OPF). It provides all components needed to create an architecture. A short description of these components follows.

- Raw input: for example integers.
- Encoder: it turns data into SDR.
- Spatial Pooler: it creates an SDR over a region.
- Temporal Pooler: it links the connections between cells (synapses).
- Cortical Learning Algorithm (CLA): it generates a prediction.

Another way to use HTM technology is to use isolated components. For example, using the *spatial pooler* and the *temporal pooler* to make patterns and these patterns can be connected to a classifier algorithm. The proposed approach involves using the two previous options. First the OPF tool, second, the temporal pooler combined with a high level classifier.

2 The Recognition Algorithms

Occlusion is an interesting challenge, see Fig. 1. The proposed algorithms were developed to overcome this problem. To measure this aspect accuracy[4] was used. In fact, the occlusion success rate (OSR) was used to referred to the accuracy. If the OSR has a high value, it means that with an occlusion challenge, the algorithm has a good performance. Therefore, the OSR was implemented to measure and to compare the different algorithms.

[4] "The accuracy is the proportion of true results (both true positives and true negatives) among the total number of cases examined" [5].

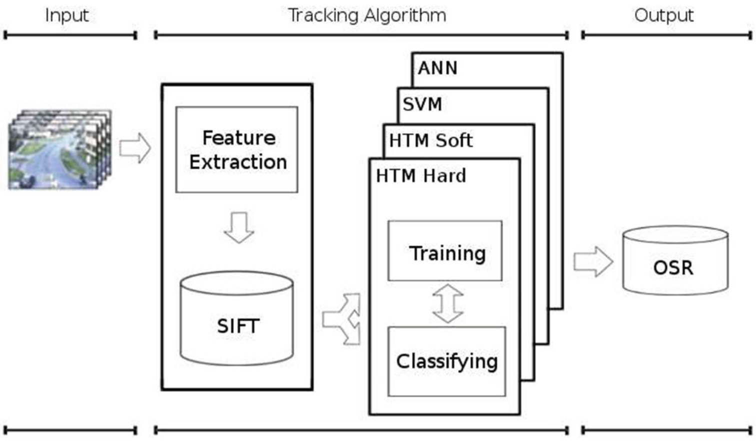


Fig. 4. The main components of the proposed algorithms.

2.1 Feature Extraction

As far as the algorithms are concerned, Fig. 4 shows the proposal. Every algorithm has the same pre-processing step. Scale invariant feature transform (SIFT) [6] was used for extracting features. Solem [7] gives a definition for SIFT: "SIFT features are invariant to rotation, intensity, and scale. In fact, they can be tracked in the presence of noise and 3D rotation". It performs two tasks: interest point detection and description of each point with data. These two tasks generate data to avoid rotation and scale differences, which are common during video frame analysis. The descriptor obtained is a vector with the following data: 16 values that describe the *interest point*, it is a matrix 4×4. And, every cell of the matrix is represented by histogram of 8 values. At the end, there are 128 floating values ($4 \times 4 \times 8 = 128$) as shown in Fig. 5.

The SIFT process can be seen in Fig. 6. It illustrates a SIFT process over an image. The results are shown on the right of Fig. 6, with the following results: 3365 detected points and since every point has a descriptor of 128 floating values, the total is 430720 values. This quantity was not manageable by the available hardware. As a result, a constraint was implemented: the videos have a resolution of 50 pixels with only 12 interest points, as shown in Fig. 7.

Figure 7 gives an example of one frame. On the left of the image, a person with a red t-shirt is raising their hands. An important aspect is that a variant of the SIFT algorithm was used. It is called *Dense SIFT*. It is different because it allows to choose static points and to set the radiuos of each one. 12 points were chosen to cover the whole image to form a grid over it, for a total of 1536 values. Some advantages of this technique are: the SIFT information for every frame of the video will have the same size for the classifications task. Also, static points will allow to detect changes on the image. This is a well-known technique used by Hassner *et al.* [8] in a similar problem of classification. Also Han *et al.* [9] developed an algorithm using SIFT to classify people.

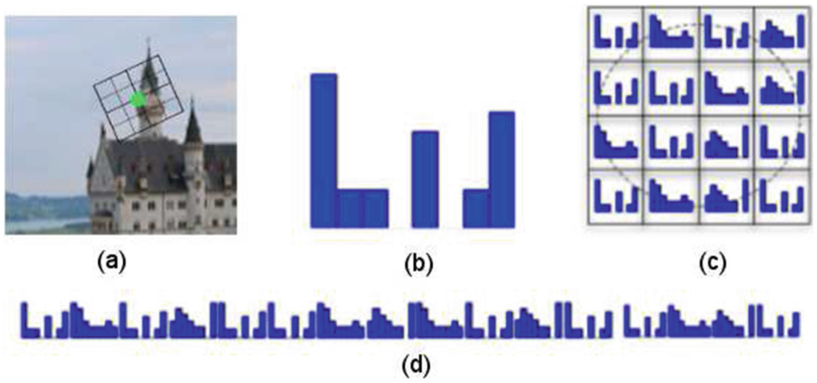


Fig. 5. A point detected by the SIFT algorithm and the descriptor generated around the point. (a) A grid generated around the interesting point. Its orientation is according to the gradient direction. (b) Every cell of the grid is represented by an 8 bin histogram. (c) Get all histograms from the grid. (d) All histograms are concatenated to have a vector. Image from [7].

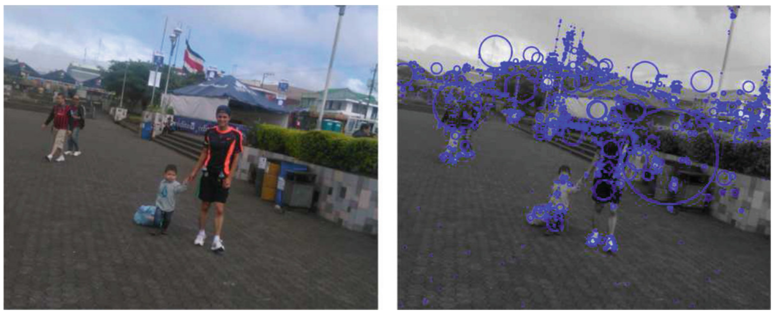


Fig. 6. SIFT process over an image: it detects about 3365 points.

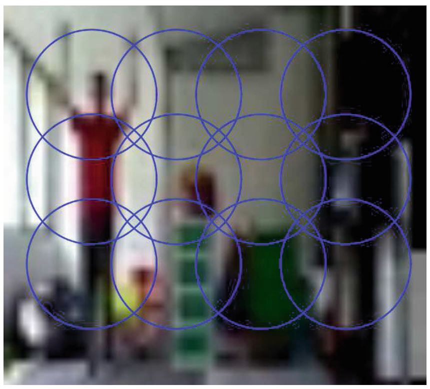


Fig. 7. Dense SIFT: there are 12 static points and they have a predetermined radius.

2.2 Proposed Algorithms

As shown in Fig. 4 there are 4 algorithms. A brief explanation of each algorithm follows.

1. HTM Hard: the word "Hard" means a full or complete implementation of an HTM algorithm. It does mean that all pieces of HTM were used. That is, Encoders, Spatial Pooler, Temporal Pooler, and Cortical Learning Algorithm. A modified version of OPF was used. This modification was implemented because of the streaming data, and it was based on Costa's proposal [10]. Numenta[5] does not recommend the use of streaming data over OPF. Anyway this modification was needed. This explains the long execution time (it lasts 60 times more). This modification was made in the input layer. Basically, it receives only one input value. There is an OPF file called *model params*, there, an for loop was added to read more than one parameter. Here, the 1536 values were read. Algorithm 1 shows the relationship between all the steps in this algorithm.
2. HTM soft: it is the recommended version for image processing. Numenta recommends this algorithm [15], where a single component (in this case the spatial pooler) is connected to a classifier. In this research a k-Nearest Neighbors [11] classifier was implemented with the spatial pooler.
3. SVM: a strong SVM library for the classification task was used, in this case the LibSVM [16]. A linear kernel function was used for this implementation.
4. ANN: the library PyBrain [12] was chosen for implementing this algorithm. A feedforward backpropagation neural network was implemented. The architecture was chosen from previous research, the number of hidden layers by Bishop [13] and the numbers of hidden neurons by Blum [14].

3 Experiments

Design of Experiments (DoE) was used as a statistical model with the R [17] language. It tries to answer if predefined factors have an influence over a study response variable, and, if there is a significant influence, the important issue is to quantify that influence [18]. The ANalysis Of VAriance (ANOVA) is an instrument of DoE. It measures the variance over the factor values against the variance of other predefined factors. OSR is the response variable and it has four factors. Table 1 shows the factors: machine learning technique, training-set size, the percentage of occlusion and the complexity of the scenario. On the one hand, a simple scenario is composed of white background and some objects. On the other hand, a complex scenario has more objects, light contrast, and different backgrounds.

As shown in Table 1, there are 4 techniques, 3 training-sets, 4 occlusion percentages, and 2 scenarios, for a total of 96 combinations. 4 replicas were picked, for a number of 384 runs ($4 \times 3 \times 4 \times 2 = 96 \times 4 = 384$). It is important to

[5] Numenta is the enterprise behind of the HTM technology.

Algorithm 1. HTM Hard

```
Require: $videoFrames$ are Dense SIFT files

function CLASSIFICATION-TASK($videoFrames$, $groundTruth$)
    $resultingFrames \leftarrow videoFrames$                ▷ Encode all entries
    for $i \leftarrow 1$ to $resultingFrames.size$ do
        $spatialPoolerResult \leftarrow SP(resultingFrames_i)$
        $temporalPoolerResult \leftarrow TP(spatialPoolerResult)$
        $prediction \leftarrow CorticalLearningAlgo(temporalPoolerResult)$

        if $prediction = groundTruth_i$ then
            $\delta \leftarrow \delta + 1$
        end if
    end for
    $accuracy \leftarrow \frac{groundTruth.size}{\delta}$                ▷ accuracy = OSR
    return $accuracy$
end function
```

highlight that every run is a sequence of frames. For instance, the training phase has $38 + 176 + 474 = 688$ images for training. With 2 scenarios, that is 1376 images. Also there are 4 replicas, for a total of 5504 images to train our algorithms. Regarding to the testing phase, there are 96 runs of 360 frames, for a total of 34560. Since there are 4 replicas, the total number of classification tasks is 138240.

Table 1. Factors and levels for this research.

	Selected factors			
	Techniques	Training sets	Occlusion(%)	Scenarios
Levels	HTM (Hard)	38	75	Simple
	HTM (Soft)	176	50	Complex
	ANN	474	25	
	SVM		0	

The data are composite of low-quality videos (in a resolution of 50 × 50 pixels). There was a hardware constraint, the OPF requires more RAM than the one available on the machine used (16 GB of RAM). The OPF builds a full structure of HTM which requires a lot of memory.

Another important aspect is regarding to the videos used in this research. These videos were recorded with some important rules to have good statistical results.

1. There are 3 video classes: two different persons moving around, one class per person. And the background with no motion, the final class. These persons were moving with different percentages of occlusion, this percentage does not change throughout the video. See Table 1 for the occlusion percentages.
2. One replica has 4 occlusion percentages, 2 different scenarios (completely different videos), and 3 different classes (8 videos per class). That is, 24 different videos. Since there are 4 replicas, there were 96 videos.
3. Every video has 36 s. Every second has 15 frames, that is, a total number of 540 frames per video. In Table 1 the number of frames taken for training the algorithms can be different (38, 176 and 474).

4 Results and Analysis

DoE needs to accomplish two assumptions. First, the residuals are normally distributed. Second, they are independent with a constant variance, as explained by Anderson and Whitcomb [19]. Fortunately, these assumptions were accomplished and no data transformation was needed. Table 2 shows the output of an ANOVA (Acronym explanation: Df: *Degrees of Freedom*, Sum Sq: *Sum of Squares*, Mean Sq: *Mean of Squares*, F val: *F values*, Pr(>F): *Probability values*).

Table 2. The ANOVA results.

	Df.	Sum-Sq.	Mean-Sq.	F-val	Pr(>F)
Tech	3	0.96	0.32	4.97	**0.0022**
Train	1	0.03	0.03	0.54	0.4623
Occlu	1	0.51	0.51	8.02	**0.0049**
Scenario	1	0.16	0.16	2.50	0.1148
Tech:Train	3	0.41	0.14	2.12	0.0970
Tech:Occlu	3	1.44	0.48	7.49	**7.15e-05**
Train:Occlu	1	0.01	0.01	0.11	0.7369
Tech:Scenario	3	0.31	0.10	1.62	0.1840
Train:Scenario	1	0.06	0.06	0.95	0.3303
Occlu:Scenario	1	0.08	0.08	1.20	0.2746
Tech:Train:Occlu	3	0.06	0.02	0.33	0.8012
Tech:Train:Scenario	3	0.20	0.07	1.05	0.3695
Tech:Occlu:Scenario	3	0.06	0.02	0.30	0.8247
Train:Occlu:Scenario	1	0.03	0.03	0.40	0.5255
Tech:Train:Occlu:Scenario	3	0.06	0.02	0.30	0.8256
Residuals	352	22.57	0.06		

The first important fact of Table 2, is that the probability of *Technique:Occlusion* has a value of 7.15^{-5}, which gives us a significance of 0.0. This means that there is confidence that the interaction of Technique and Occlusion is very significant. With this information, the null research hypothesis can be rejected, which says that no factor affects the response variable.

There are two other interesting facts: the factor *Technique* has a confidence of 99.78%, because its probability value is 0.0022. Similarly with the factor *Occlusion* that gives the confidence of 99.51%, its probability is about 0.0049. Consequently, the factors *Occlusion*, *Technique* and their interaction has a significant affectation over the response variable, the OSR value. With the information about the ANOVA process, the following step is to analyze the significant factors and their interaction.

4.1 Technique (Tech)

Box plots [20] were used to analyzed some aspects. On the one hand, Fig. 8 shows that ANN has the lowest quartile[6] and the mean with the lowest value, which means that ANN fails more than the others.

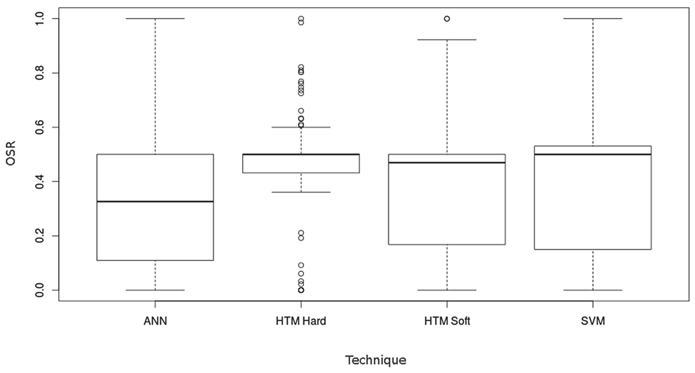


Fig. 8. Box plot for the Technique factor.

On the other hand, the HTM Hard algorithm has its 50% of data enclosed in the thinnest box, also the upper and lower quartile boundaries have the smallest range. The mean is near to 0.5 and it has a lot of outliers, this situation is normal because the upper and lower quartiles form a small range. The lower quartile shows that this algorithm had fewer failures than the others. This algorithm had a stable performance.

[6] Quartile: they are the values that divide a list of numbers into quarters.

4.2 Occlusion (Oclu)

In Fig. 9, there is the box plot of the factor *Occlusion* against OSR. It can be seen that all algorithms fail if the occlusion value is incremented, on 75% the lower boundary of the box is almost 0.0. This is a expected behavior, because the more the object is occluded, the more difficult to hit.

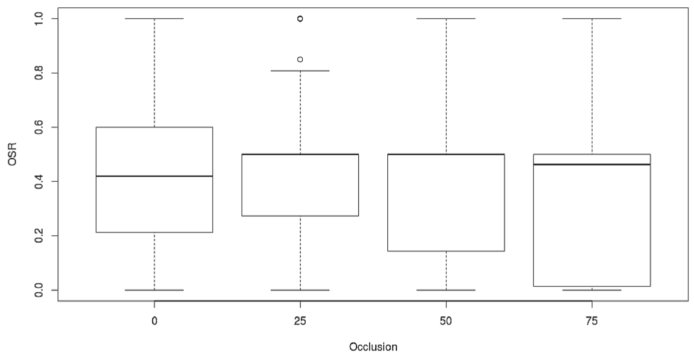


Fig. 9. Box plot for the Occlusion factor.

It can be appreciated that the mean value for 25% and 50% have almost the same value, which indicates it is easier for an algorithm to hit. This can be explained because people have more characteristics from the hips towards the head. Also, there is another interesting aspect: the values for 25%, 50%, and 75%, have their upper box boundary close to 0.5 of accuracy, but, the box of 0% has it close to 0.6, which indicates that with 0% the algorithms tend to fail less. However, its mean has the lower value. This situation tells that under a specific condition, it produces more fails with 0%. An explanation is that with 0% the objects (people) had more freedom to move, and did not have to be behind of an object to catch the occlusion.

4.3 Interaction: Occlusion (Oclu) and Technique (Tech)

The final aspect to analyze is one of the most interesting ones. The interaction between two factors: technique and occlusion. Whitcomb and Anderson give the definition of interaction [19]: “Interactions happen when the effect of one of the factors depends on another factor”. Figure 10 shows many relevant situations. The first one is that all algorithms tend to decrease. It is obvious because as the occlusion raises its value, it is more difficult for all algorithms to hit.

Another aspect to note is that HTM Soft and ANN do not have an acceptable performance, unlike SVM and HTM Hard. It was mentioned that with 0% of occlusion, there is a huge challenge: the targets had more freedom to move all over the scene. It is considered that ANN and HTM Soft do not have sensibility under this condition (target movements). Hence, the good performance of SVM and HTM Hard shows that these algorithms are better implementations for real world systems.

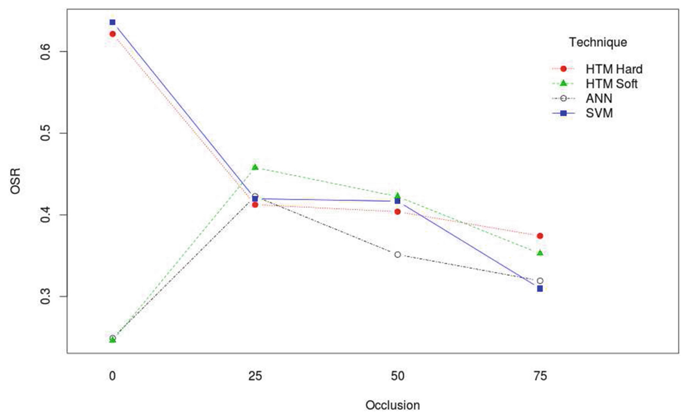


Fig. 10. Interaction of two significant elements (factors): occlusion and technique.

The third aspect analyzed is that SVM has a remarkable behavior on 0%. A stable performance on 25% and 50%, it is similar to HTM Hard. However, it fails on the 75% value. This situation indicates that it could not classify well when the object was almost totally occluded.

Lastly, there is the HTM Hard implementation. It has an excellent performance on 0%, a good sensibility to target movements. Also, good results on 25% and 50%. As seen in Fig. 10, HTM Hard had the best performance on the value of 75% of occlusion. As a result, this is the best algorithm for the occlusion challenge.

5 Conclusions

A modified version of an OPF algorithm was used (HTM Hard). Based on the statistical results, one conclusion is that this algorithm is better to recognize objects than the others. But, it was experienced an unexpected situation: it had the largest running time. It lasts 60 times more comparing with the other implementations. It is not feasible to implement with streaming data.

Another aspect is that SVM had a good performance. During tests, it was the fastest implementation. However, it had a poor performance on 75%. Also, there are the ANN and the HTM Soft performance, which had the worst results. As a result, their implementation is not recommended with the proposed architectures.

All algorithms used SIFT to classify. This technique provides enough information to feed a machine learning algorithm. For classifying purposes, Dense SIFT was used to have a fixed amount of data per frame. Only 12 SIFT points were used. However, with a limited amount of data the results were acceptable and satisfactory.

Finally, it was discovered that the implementations of HTM Hard and SVM are more sensitive to object movements. They have an acceptable performance with a small quantity of data. HTM Hard is the recommended version for this kind of problem because in real world systems the objects are moving all over the scene.

6 Future Work

The next step is to test different combinations of algorithms. For instance, recurrent neural networks (RNN), convolutional neural networks (CNN), bayesian networks, etc. It would be interesting to use CNN, due to its similarity to HTM (many layers, feature patterns reduction, etc.).

Another interesting aspect is to test different HTM combinations. For example, to implement the usage of a temporal pooler with a classifier (in this research kNN was implemented) such as SVM, ANN, RNN, CNN, etc. Furthermore, high-quality videos can be used and get more information.

Also, to use more points (Dense SIFT values). It can be treated as an hyper-parameter and adjust it to get better results.

Finally, another feature extraction technique can be used as well as different preprocessing techniques.

Acknowledgements. We would like to thank our colleagues from the Happy Few Research Group for their support during the development of this research. In addition, we thank the PARMA Group for its guidance and support publishing this research.

References

1. Jalal, A.S., Singh, V.: Visual object tracking: state of art. Int. J. Comput. Inform. **3**, 227–247 (2011). Ljubljana, Slovenia
2. Hawkins, J., Blakeslee, S.: On Intelligence. St. Martin's Griffin, USA (2005)
3. Fan, J., Xu, W., Gong, Y.: Convolutional neural networks: human traffic. IEEE Trans. About Neural Netw., 1610–1623 (2010). https://doi.org/10.1109/TNN.2010.2066286
4. Hawkins, J., Ahmad, S., Dubinsky, D.: Cortical Learning Algorithms and HTM. Numenta Inc., California (2011)
5. Metz, C.E.: Principles of ROC Analysis. University of Chicago and the Franklin McLean Memorial Research Institute, Chicago, USA, pp. 283–298 (1978). 0001-2998/78/0804-0003S02.00/0
6. Lowe, D.G.: Object recognition based on local scale invariant features. In: Computer Vision: International Conference, pp. 1150–1157. IEEE, Canada (1999). https://doi.org/10.1109/ICCV.1999.790410
7. Solem, J.E.: Python: Basics on Computer Vision. O'Reilly Medias, Sebastopol (2012)

8. Hassner, T., Mayzels, V., Zelnik-Manor, L.: About SIFT and its scale. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Rhode Island, USA (2012)
9. Han, B., Li, D., Ji, J.: DSIFT Algorithm for People Detection. Stanford University, California (2011)
10. Costa, A.: A MNIST Classifier Using OPF (2016). http://github.com/allanino/nupic-classifier-mnist
11. Altman, N.S.: Introduction to kernel and nearest-neighbor nonparametric regression. Am. Stat. **46**, 175–185 (1992)
12. Schaul, T., Bayer, J., Wierstra, D., Sun, Y., Felder, M., Sehnke, F., Rückstieß, T., Schmidhuber, J.: The PyBrain library. J. Mach. Learn. Res. **11**, 743–746 (2011)
13. Bishop, C.M.: Pattern Recognition Using Neural Networks. Oxford University Press, Oxford (1995)
14. Blum, A.: Neural Networks (C++). Wiley, New York (1992)
15. Numenta: Nupic: managing vision tasks (2017). http://github.com/numenta
16. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2017). http://www.csie.ntu.edu.tw/cjlin/libsvm/
17. Ihaka, R.: History of R: Past and Future. The University of Auckland, Auckland (1998)
18. Ferré, J., Rius, X.: Introduction to the Statistical Design of Experiments. Universitat Rovira i Virgili, Tarragona (2001)
19. Anderson, M., Whitcomb, P.: Design of Experiments: A Simplified Approach. Taylor and Francis Group, Boca Raton (2007)
20. Massart, D.L., Smeyers-Verbeke, J., Capron, X., Schlesier, K.: Means of Box Plots: Visual Presentation of Data. Vrije Universiteit Brussel, Brussel, Belgium (2005)
21. Schlag, I.: On Hierarchical Temporal Memory (2016). http://ischlag.github.io/2016/04/25/on-hierarchical-temporal-memory
22. Gerstner, W.: Hebbian learning and plasticity. From Neuron to Cognition via Computational Neuroscience, Chap. 9. MIT Press, Cambridge (2011)

Tree-Structured CNN for the Classification of Surgical Instruments

Paula Catalina Useche Murillo(✉), Javier Orlando Pinzón Arenas(✉), and Robinson Jiménez Moreno(✉)

Faculty of Engineering, Nueva Granada Military University, Bogotá D.C, Colombia
{u3900235,u3900231, robinson.jimenez}@unimilitar.edu.co

Abstract. This paper presents the development of an algorithm of object classification focused on the detection and categorization of surgical instruments using tree-structured convolutional neural networks for the classification of initially 5 different tools and later evaluated with 10, managing to develop a new way of classifying objects capable of differentiating elements with a high resemblance to each other. Through partial division of the categories and kernel training focused on the most relevant changes of each instrument, percentages of about 70% accuracy were reached for a total of 10 elements, and 96% for 5 instruments.

Keywords: Convolutional Neural Network · Layer activations Tree-structured network · Surgical instrument recognition Detailed feature extraction

1 Introduction

The use of convolutional neural networks (CNN) [1] for the classification of objects in different types of applications has been made possible by the different architectures that can be proposed for each network, the use of certain types of layers, and to the dimensions defined for each of their filters, factors that have allowed CNNs with different depths to acquire the ability to extract and learn specific characteristics of different training categories, as shown in [1].

The implementation of this type of networks has allowed to develop numerous applications of recognition that cover diverse subjects like the detection of faces [2, 3], the detection and classification of positions of the hands [4], text classification in written documents [5], the classification of traffic signals [6], and even applications for speech recognition [7]. Also, they have been extended to video object recognition [8, 9] and to handwriting recognition [10].

However, all of them present a particular characteristic and it is the classification of categories with numerous differences between them, where diverse characteristics like the background, the color, or certain proportions of the object allow each category to be classified by a single convolutional neural network with error rates lower than 5%, as shown in [6], while the problem that is sought to solve in this article is focused on the classification of surgical instruments through CNNs, where only small details of each tool are the ones that allow to differentiate one from the others, forcing the network to acquire the ability to detect small changes in the instruments in order to classify them properly.

A novel method of classification by means of tree-structured convolutional neural networks is presented in this work, focused on the classification of 10 surgical instruments, whose main characteristics for the classification of each category cannot be extracted with a single CNN architecture, as it is demonstrated during the development of the application, but with different networks focused on the main characteristics of each category.

The following article is divided into four main sections, where the first one focuses on the "Classification Tree" within which the classification method with tree-structured CNN is explained, and the problem to be faced is described. In the next section the tests made to obtain the final classification structure are presented. In the third section the behavior of the different trained networks is analyzed. Finally, in the fourth section, the conclusions reached are presented.

2 Classification Tree

2.1 Database

For the development of the surgical instrument classification algorithm, 10 tools were selected, which are presented in Fig. 1, and a tolerance of ± 15 degrees of inclination with respect to the central vertical axis of each tool was set for the taking of the training and test databases, in order to give a degree of tolerance to the classification algorithm that allows it to recognize the instruments even if there is a slight inclination of the element.

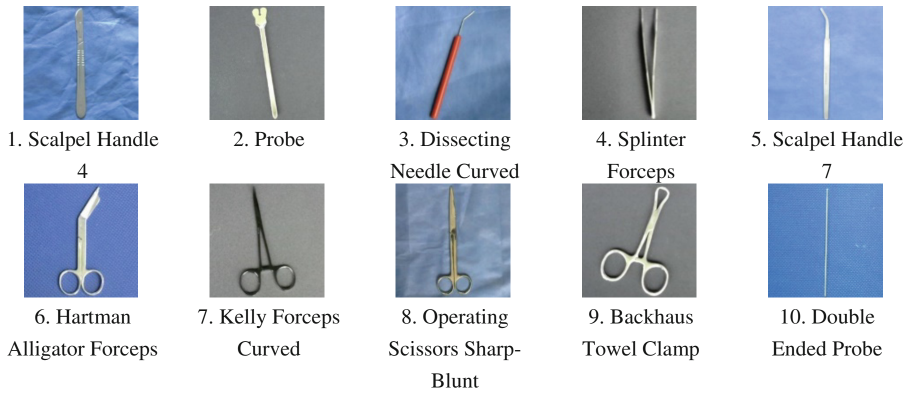


Fig. 1. Ten surgical instruments selected.

With this tolerance limit, that eliminates the need for the network to learn the same characteristic of each instrument under different rotation situations, allowing to focus the classification on the shape of the element and not on its inclination, 300 training color images and 50 test images with size of 128 × 128 pixels were taken for each instrument, different backgrounds were used for the capture of the database, whose shades vary between blue and gray, as shown in Fig. 1, and the lighting conditions were varied to capture the changes of tonality that appear on the material of each instrument.

2.2 Structure of the Classification Algorithm

Due to the great similarities between the surgical instruments and the results obtained in the classification of the ten tools with a single convolutional neural network (CNN), as presented in the Sect. 3, it was necessary to establish a tree-structured classification, as illustrated in Fig. 2, for the classification of 10 tools (Tree 10), and in Fig. 3 for the classification of 5 tools (Tree 5), where CNNs were used for the classification of the categories of each branch, and different architectures for each CNN, focused on capturing the differential details between each category.

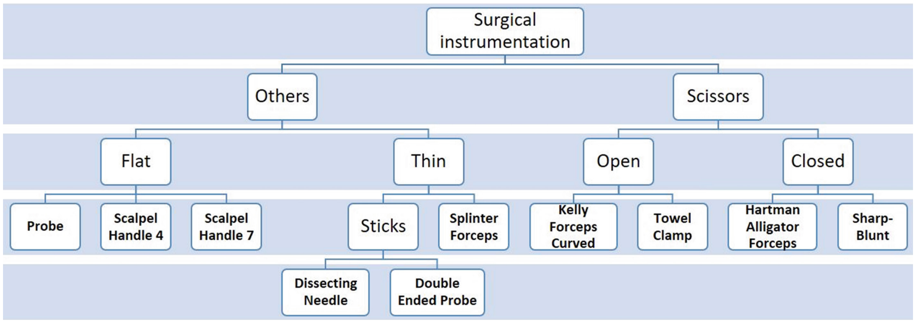


Fig. 2. CNN classification tree for 10 surgical instruments (Tree 10).

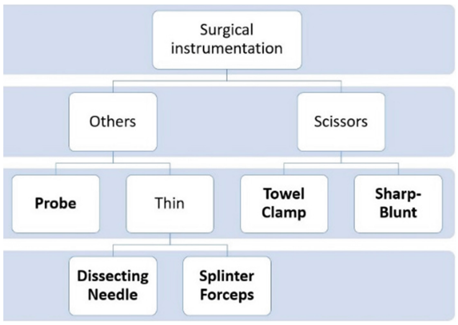


Fig. 3. CNN classification tree for surgical instruments (Tree 5).

The initial classification of the tree consists of a main category called “Scissors” which contains all the scissors-type surgical instruments, and a secondary category called “Others” containing more varied surgical instruments, from cutting elements such as “Scalpel Handle 4” to apprehension tools such as “Splinter Forceps”. The categories “Flat” and “Thin” are composed of tools with mainly flat bodies, in the case of “Flat”, or thin instruments, in the case of “Thin”, and the category “Sticks” contains tools with shapes similar to rods. On the other hand, the category “Open” contains those scissors whose eye rings are separated from each other, while the “Closed” category contains the scissors that have the eye rings together.

An example of the two categories set for the first branch of the tree is shown in Fig. 4, where Fig. 4a contains the instruments of the “Others” category and Fig. 4b, the instruments of the “Scissors “category. When comparing both categories it is possible to appreciate that the great difference between the two lies in the existence or not of eye rings, a factor that becomes a characteristic of the category “Scissors” that cannot be found in the “Others” category. On the other hand, within each category are great similarities, such as a thin body for the category “Others”, and the presence of eye rings for scissors, with small differences in the points that characterize each instrument.

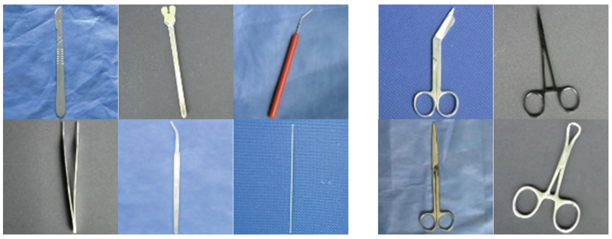


Fig. 4. Categories of the first branch, (a) Others and (b) Scissors.

The tree-structured object classification works as follows: the image of the surgical instrument to be classified is taken as an input and, by means of a CNN, the instrument is placed in one of the two categories of the first branch of the tree, either “Others” or “Scissors”. Subsequently, the image enters another CNN that classifies it into one of the next two or three categories in which the tree is divided, changing the image of branch to a final classification, where it is named after any of the surgical instrument categories. Figure 5 shows two examples of successful path classification for the case of Tree 5.

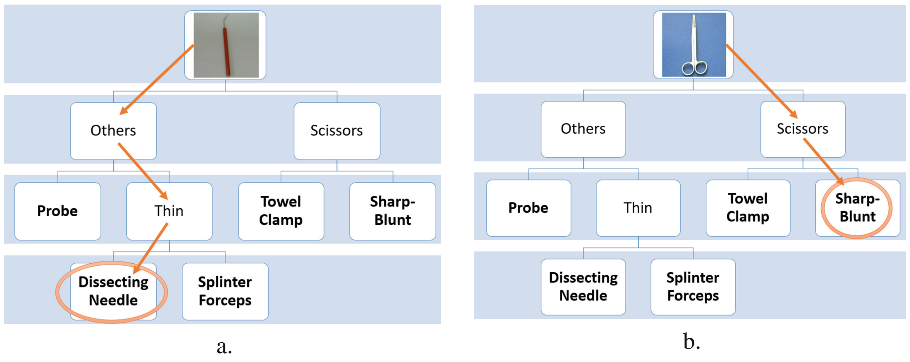


Fig. 5. Operation of the classification tree for (a) “Dissecting Needle Curved” and (b) “Operating Scissors Sharp-Blunt”.

The use of a tree-structured classifier allows to reduce the number of categories to be classified in each CNN, helping the networks to reduce the amount of features to learn during the training, requiring fewer learned characteristics to be able to classify of 2 or 3 categories than a general network whose filters must focus on extracting an enormous amount of details from each element that may allow the adequate classification of some of them, but a great confusion among the most similar.

On the other hand, each one of the databases of the categories that compose the CNNs tree was organized in such a way that they always had the same amount of images with each other, for instance, in the case of “Others” and “Scissors” in the Tree 10, the first category contains 6 instruments while the second only 4, so that random images of each item belonging to the “Others” category were eliminated, until the number of images between the two categories was equal. This adjustment was made to avoid that CNN tended to categorize all the instruments within a single category due to the large training database used.

2.3 Guided Classification

In addition to classifying surgical instruments into tree-structured categories, a relevant feature detection algorithm was developed, where the programmer defines in which section are the characteristics to be extracted, either the upper or lower half of the instrument, and the algorithm is in charge of finding the tool in said section and generating a square crop on it to extract only a fraction of the image, as shown in Fig. 6b.

Figure 6 shows an example of the trimming generated on the Hartman Alligator Forceps, where the section of the image within the square cutout is used to classify the item into the next category, allowing complex tasks such as sorting the four types of medical scissors selected for the application, whose differences between them are minimal. The characteristics to be extracted for each instrument are defined by the programmer according to their classification criteria, in the case of the “Scissors” category, an initial classification focused only on the eye rings (lower half of the instrument), and a final classification focused on the scissors tips (upper half of the instrument).

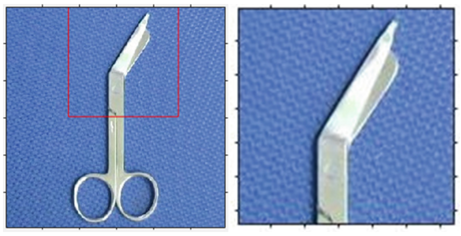


Fig. 6. (a) Tool with cropping section and (b) copped image.

The developed algorithm receives as input the image to be cropped, the desired output dimensions for the cropping, and the section of the image to be cropped, i.e. whether it is to be focused on the top or bottom of the instrument, as set by the programmer. Subsequently, the input image is taken and converted to grayscale, then the contours are obtained using the Prewitt contour detection method which determines the presence of a contour at those points where the image gradient is maximum, removes noise from the background, divides the image in half (taking the top or bottom as previously selected), and proceeds to look for the surgical instrument.

To determine the position of the crop, a square window with a side equal to half the dimension of the original image is slid, and the number of white pixels enclosed inside the box is added, storing in a variable the position where the highest result was found. Then an offset (equal to half the size of the search window) is subtracted from the position found to center the tool in the middle of the cut, and proceed to crop the original image in that position until a cutout is obtained as in Fig. 6b (if the upper section of the instrument has been selected).

Figure 7 shows an example of the top section of the Hartman Alligator Forceps, after having obtained contours, eliminated the noise, and slid the window over the image, where it is possible to observe that the area where the instrument is located is the one with the greatest number of contours or white pixels.

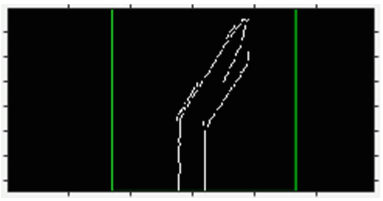


Fig. 7. Sliding window for detection of the instrument "Hartman Alligator Forceps".

2.4 Final Architectures of the CNNs for Each Branch

Eight CNNs were trained for the classification of ten surgical instruments, and four to classify only five of them. Each of the networks was designed specifically to capture certain characteristics of the images, so square filters can be found for general classifications such as categorizing the surgical instruments (Others and Scissors) into two groups, and rectangular filters for a more detailed classification at the end of the tree branches, such as with the "Sticks" network to differentiate the "Double Ended Probe" from the "Dissecting Needle Curved" in Tree 10, as shown in Table 1, or in the case of the "Thin" network to differentiate the "Splinter Forceps" from the "Dissecting Needle Curved", as presented in Table 2 for Tree 5.

In Tables 1 and 2 the CNN architecture used for each branch of the tree is presented vertically, where the cells with darker shades are the MaxPooling and the clearer are the convolution layers. The letters H and W represent the height and width respectively, while F indicates the size of the filter and s the stride, in case of not specifying F or s it means that the stride is equal to 1 and the value of the box is the size of the filter. At the end of the tables is shown the amount of Fully-Connected layers used for each network and the accuracy of each network individually. The number of filters (NF) used satisfies $NF = 16 * 2^n$ (1), where n increases by a ratio of one with each convolution starting at 0 for the first convolution. In the case of deep networks like "Scissors", n increases every two convolutions.

$$NF = 16 * 2^n \tag{1}$$

The relationship presented in $NF = 16 * 2^n$ (1) was used to generate an increase in the number of filters as depth increases, in order to have a greater number of filters to learn more detailed characteristics.

Table 1. CNN architectures for the Tree 10.

Network	CNN ARCHITECTURE									
	General	Others	Scissors	Flat	Thin		Sticks		Open	Closed
Dimensions	W, H	W, H	W, H	W, H	H	W	H	W	W, H	W, H
Tree 10	11F 3s	15	5	15	2	16	16	2	5F 3s	5F 3s
	2F 2s	13	4	13	2	14	14	2	4	4
	5	**2F 2s**	**3F 2s**	**2F 2s**	**2F 2s**	**2F 2s**	**2F 2s**	**2F 2s**	**3F 2s**	**3F 2s**
	2F 2s	11	3	11	2	12	12	2	3	3
	3	9	3	9	2	10	10	2	3	3
	3	**3F 2s**	**2F 2s**	**3F 2s**	**3F 2s**	**2F 2s**	**2F 2s**	**3F 2s**	**3F 2s**	**3F 2s**
	3	7	3	7	2	8	8	2	3	3
		5	3	5	2	6	6	2	2	2
			2F 2s							
			3							
			3							
			2F 2s							
Number of fully-connected	2	2	2	2	2	2	2	2	2	2
Accuracy	**96.40%**	**91.30%**	**91%**	**82%**	**82%**		**100%**		**88%**	**90%**

Table 2. CNN architectures for the Tree 5.

CNN ARCHITECTURE					
Network	General	Others	Scissors	Thin	
Dimensions	W, H	W, H	W, H	H	W
Tree 5	11F 3s	15	5	2	16
	2F 2s	13	4	2	14
	5	**2F 2s**	**3F 2s**	**2F 2s**	**2F 2s**
	2F 2s	11	3	2	12
	3	9	3	2	10
	3	**3F 2s**	**2F 2s**	**3F 2s**	**2F 2s**
	3	7	3	2	8
		5	3	2	6
			2F 2s		
			3		
			3		
			2F 2s		
Number of fully-connected layers	2	2	2	2	2
Accuracy	**96.40%**	**95.00%**	**91%**	**91%**	

Between the two Fully-connected layers was added a ReLU (Rectified Linear Units) layer and a Dropout with a disconnection percentage of 50%, to avoid overfitting.

In Table 3, it is specified by an X which networks were trained with upper or lower sections of the surgical instruments using the cropping algorithm presented in Sect. 2.3, and which were trained with the complete image, for both Tree 10 and Tree 5. The section of the instrument selected for the training of each network was set according to the location of the most significant differences between the instruments of each category, based on the criterion of the programmer, as explained for the example of the category "Scissors" in the Fig. 4.

To classify tools very similar to each other, as in the case of scissors, networks were trained focused on the details that differentiate them. For the case of Tree 10, two cascading classifications were made to differentiate the scissors, one focused on the eye

Table 3. Section of the image trained for each network.

Tree	TREE 10								TREE 5			
Network	General	Others	Scissors	Flat	Thin	Sticks	Open	Closed	General	Others	Scissors	Thin
Upper section					**X**	**X**	**X**	**X**		**X**		**X**
Lower section			**X**								**X**	
Complete instrument	**X**	**X**		**X**					**X**			

rings ("Scissors" network) and another focused on the tips. Initially, scissors were separated in groups that had separate eye rings from those that had them together, allowing to generate the categories "Open" and "Closed", within which a final classification was made focused on the tips of the scissors in order to reduce the amount of similar information between the instruments, reaching accuracies greater than 87% in each network.

3 Tests

Initially, 300 photos of each tool were taken, varying not only their rotation, background and illumination, but also their position in space, as shown in the example in Fig. 8, and they were performed more than 10 different trainings until finding the architecture that best managed to classify the surgical instruments.

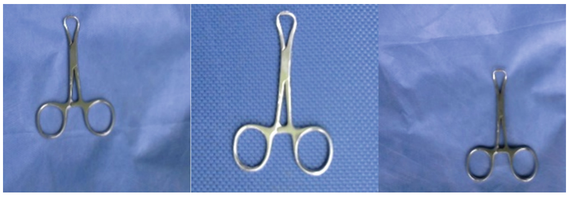


Fig. 8. Initial database.

The accuracy of each trained network ranged from 40% to 55%, where the best network achieved 54.6% accuracy, using the same "Others" network architecture in Table 1 but without the last convolution layer. Due to the poor performance of the network, the database was modified until the final database of Fig. 1 was reached, in which only the tool is framed, without size variations or significant position changes, to focus the problem to the training of the tool and subtract the relevance to the position. With the new images, the training was repeated and an improvement of about 10% with the same architecture of the previous network was achieved, reaching an accuracy of 63.4%.

Faced with the difficulty of generating an increase in the accuracy of the network, it was decided to compare the activations of the instruments classified erroneously with respect to those classified correctly in order to analyze the behavior of CNN with each instrument. It was found that, as shown in Fig. 9, the feature extraction is focused on the tool body, generating the loss of relevant information from some elements with the passage of the convolutions, such as the eye rings of the scissors or the head of "Probe", which entails to obtain two different classifications for images very similar to each other by very small changes of location or rotation.

In Fig. 9, the white areas represent the most relevant characteristics extracted by the network, the first row represents the tools correctly classified while the second shows the wrong classifications, and each column contains two images of the same instrument. Activations were organized from left to right, starting with the first row, where

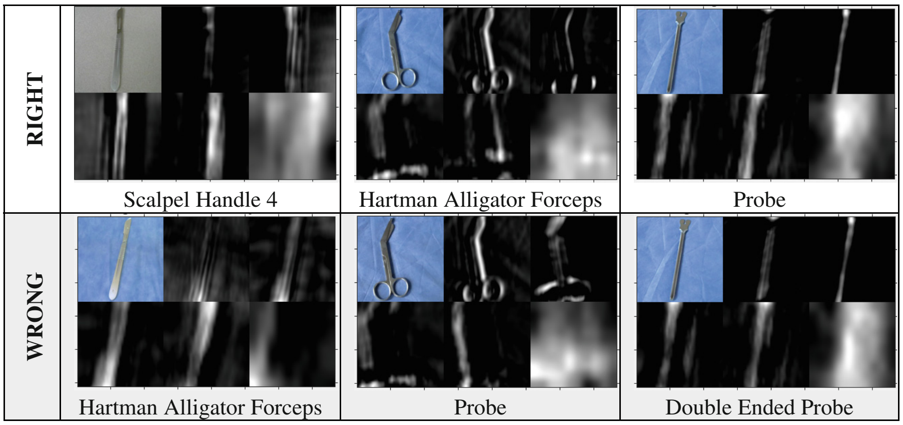


Fig. 9. Activations for CNN with 63.4% accuracy.

the activations to the right of the instrument image were obtained from the first convolution, and the activations in the lower right corner are from the fifth convolution.

In order to improve the quality of the activations and to guide the CNN to the learning of the most relevant details of each category instead of the more generalized aspects, the tree-structured classification algorithm was developed as it was presented in Sect. 2.

The categories were chosen according to the similarity between the instruments that compose them and the most relevant characteristics that differentiate them from other categories. With this, unlike the classification with a single network, a CNN focused on the scissors' eye rings allows to obtain, almost in its entirety, the whole body of the thimbles as it is shown in the activations of the first convolution layer of Fig. 10, and maintained during the following convolutions, as shown in Fig. 11, where, despite starting with a majority detection of the background in the first convolution (upper central image), in the second, the body of the scissors (upper right corner) is taken and maintained through the following three convolutions.

The quality of the activations allows to determine the depth of the CNN necessary to classify a group of categories, where the most distorted activations of the image, such as the last activation for the examples in Fig. 9, represent layers that do not adequately extract the characteristics and that can be eliminated to improve the

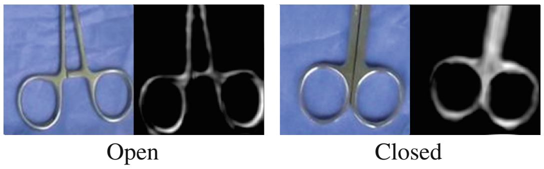


Fig. 10. Classification according to scissors' eye rings.

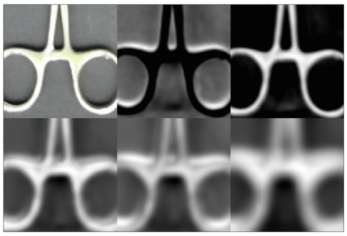


Fig. 11. Eye rings activations.

behavior of the network, while more detailed activations like those in Fig. 11, represent an adequate extraction of characteristics and are necessary layers for classification.

On the other hand, training CNNs by categories of instruments allowed to observe in more detail the characteristics that generate confusion between two or more tools, as shown below in the case of the "Thin" category for Tree 10, where the network presented problems to separate the "Sticks" from the "Splinter Forceps", despite having focused its training to the upper section where the difference between both is more marked.

The activations were plotted in order to see what features the network extracted from each image to classify the instruments as "Stick" or "Splinter Forceps", finding that the "Double Ended Probe" in the "Sticks" category generates more confusion in the network due to its thin body which begins to resemble more to the activations of the Splinter Forceps than to the ones of Sticks with the passage through the convolutions, as shown in Fig. 12, while the "Dissecting Needle Curved" has a more marked activation due to its wide body.

On the other hand, with the activations of the "Sticks" network it was verified that the most important feature that CNN used to differentiate the "Double Ended Probe" from the "Dissecting Needle Curved" was its thickness and the amount of background in the image, as illustrated in Fig. 13, which allowed to reach an accuracy of 100% for that category.

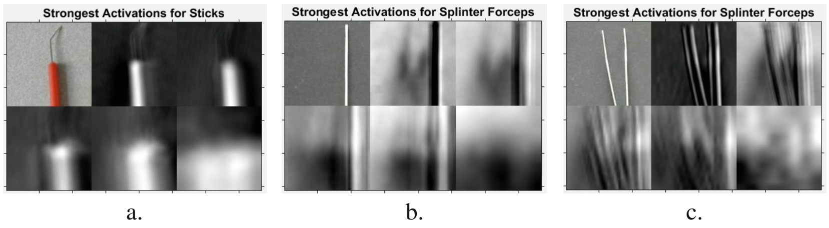

Strongest Activations for Sticks
Strongest Activations for Splinter Forceps
Strongest Activations for Splinter Forceps
a.
b.
c.

Fig. 12. "Thin" network activations.

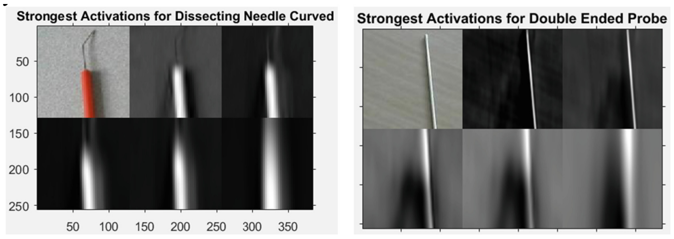
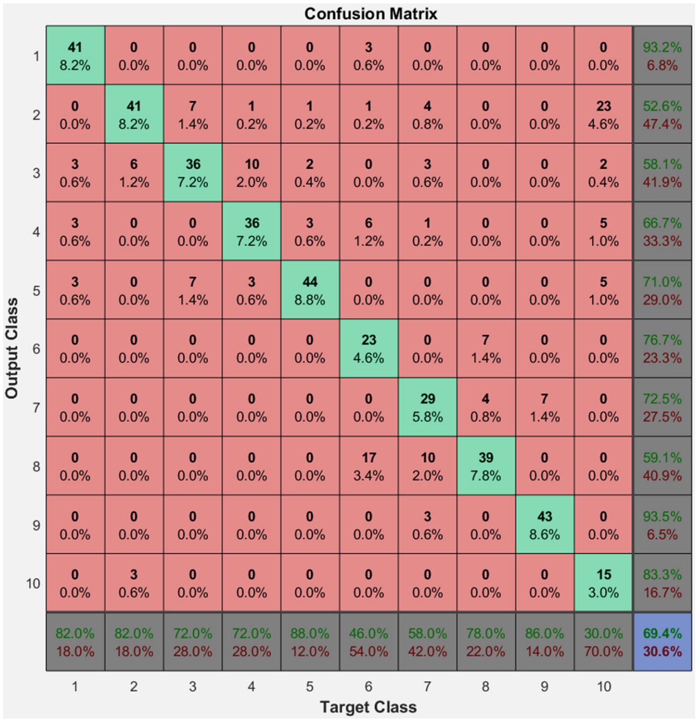


Fig. 13. "Sticks" network activations with 100% accuracy

4 Results and Analysis

Comparing the networks that compose the Tree 10, it can be observed that the best results were obtained with filters of greater or smaller size depending on how deep the network was located in the tree structure, finding that for the first networks, like "General" and "Others", the big filters work properly, since they emphasize very general characteristics, while other networks such as "scissors", "open" and "closed" scissors use smaller filters, allowing them to extract more detailed features as seen in the activations of Figs. 10 and 11.


Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	6	7	8	9	10	
1	41 8.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	3 0.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	93.2% 6.8%
2	0 0.0%	41 8.2%	7 1.4%	1 0.2%	1 0.2%	1 0.2%	4 0.8%	0 0.0%	0 0.0%	23 4.6%	52.6% 47.4%
3	3 0.6%	6 1.2%	36 7.2%	10 2.0%	2 0.4%	0 0.0%	3 0.6%	0 0.0%	0 0.0%	2 0.4%	58.1% 41.9%
4	3 0.6%	0 0.0%	0 0.0%	36 7.2%	3 0.6%	6 1.2%	1 0.2%	0 0.0%	0 0.0%	5 1.0%	66.7% 33.3%
5	3 0.6%	0 0.0%	7 1.4%	3 0.6%	44 8.8%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	5 1.0%	71.0% 29.0%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	23 4.6%	0 0.0%	7 1.4%	0 0.0%	0 0.0%	76.7% 23.3%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	29 5.8%	4 0.8%	7 1.4%	0 0.0%	72.5% 27.5%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	17 3.4%	10 2.0%	39 7.8%	0 0.0%	0 0.0%	59.1% 40.9%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	3 0.6%	0 0.0%	43 8.6%	0 0.0%	93.5% 6.5%
10	0 0.0%	3 0.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	15 3.0%	83.3% 16.7%
	82.0% 18.0%	82.0% 18.0%	72.0% 28.0%	72.0% 28.0%	88.0% 12.0%	46.0% 54.0%	58.0% 42.0%	78.0% 22.0%	86.0% 14.0%	30.0% 70.0%	69.4% 30.6%

Fig. 14. Confusion Matrix of the Tree 10

On the other hand, in the case of the “Thin” and “Sticks” networks, the use of rectangular filters was required, the first with greater width than high, and the second with greater height than width. Although the filter sizes are similar for both networks, the “Thin” network worked better with wide filters than with high or squares, since the distortion generated by the horizontal convolution application highlighted some features such as the thickness of the “Dissecting Needle Curved” or the two tips of the “Splinter Forceps”, shown in Fig. 12, while in the case of the “Sticks” network, the minimum horizontal distortion generated by the vertical convolutions helped to highlight the difference in thickness between the two elements, as illustrated in Fig. 13.

Figure 14 shows the final confusion matrix obtained for Tree 10, where it was possible to increase the accuracy up to 69.4%, using the networks in Table 1. Each number of the matrix represents each of the trained categories, numbered as indicated in Fig. 1, the columns indicate in which categories the network recognizes each tool, and the rows show the desired category.

The most difficult instruments for classification were the “Double Ended Probe” and the “Hartman Alligator Forceps”, where the former were greatly confused by their geometry, as previously explained, and the latter were classified erroneously in several networks, causing it to go through the wrong branches of the tree and to be classified as “Scalpel Handle 4” or “Splinter Forceps”.

After observing and finding the greatest difficulties of the network during the classification, it was proceeded to take only 5 instruments of the 10 trained to generate another classification tree in order to obtain better percentages of accuracy and to reduce the number of confusions between the medical tools.

The 5 instruments were selected according to the following parameters:

- The quality of the activations, as it was the case of the “Dissecting Needle Curved” and the “Splinter Forceps” with the “Thin” network.
- The number of images mistakenly classified between two categories, such as the “Sharp-Blunt” and “Towel Clamp”, which did not show any confusion with each other within the confusion matrix of Fig. 14 and had the best recognition rates among the scissors.
- The last tool was selected according to its degree of similarity between the instruments already selected, their percentage of recognition in the confusion matrix of Tree 10 and the number of images mistakenly classified among the 4 instruments already selected and the fifth to be selected, choosing “Probe”.

After choosing the 5 instruments to be classified, the remaining branches of Tree 10 were removed until the Tree 5 reaches the structure shown in Fig. 3, where some networks of Tree 10 were reused, such as “Scissors” to classify the scissors according to the position of their eye rings, and “General” to divide into the “Others” and “Scissors” categories. In the case of the “Others” and “Thin” networks, the corresponding network obtained for Tree 10 was taken and re-trained only with the images to be classified, in order to recalculate the CNN weights to focus the recognition only on the desired tools, achieving a 3.7% and a 9% improvement in the accuracy of said networks respectively, resulting in a 96% accuracy for the final classification of the 5 instruments, as shown in the confusion matrix of Fig. 15, whose tools are numbered as shown in Table 4.

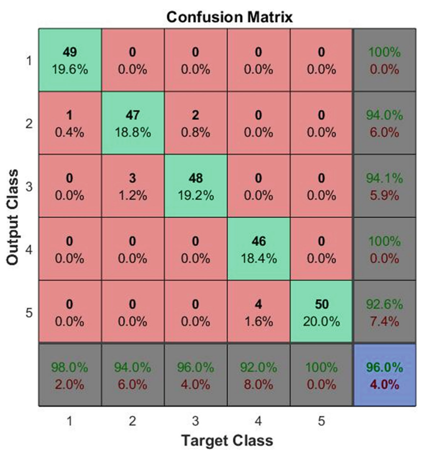


Fig. 15. Confusion Matrix of the Tree 5

Table 4. The five trained instrument categories

1	Dissecting Needle Curved
2	Splinter Forceps
3	Probe
4	Operating Scissors Sharp-Blunt
5	Backhaus Towel Clamp

5 Conclusions

The architecture of a CNN and the dimensions of its filters can facilitate the extraction and learning of certain characteristics of the elements to be classified, however, when these characteristics are not very noticeable and the differential aspects between categories are very small or reduced, the network requirement increases and the distortion generated by the convolutions and pooling layers can lead to the elimination of the most relevant characteristics of any of the categories, as observed with the activations of Fig. 9 for the Hartman Alligator Forceps, leading to a loss of information due to any variation in the instrument to be classified that may generate a change in the result obtained by the network.

On the other hand, for the classification of elements very similar to each other as the case of the category “Sticks”, it was possible to obtain a 100% of accuracy using vertical rectangular filters in the architecture of the network, since they allow to extract the form and the proportions characteristic of both elements to generate a classification, however, these filters did not work in the same way for the classification of other instruments as the case of the category “Scissors”, where it was required a feature extraction less focused to the elongated body of the instrument, and more focused on other aspects such as the scissors’ eye rings, where small and square filters worked best.

The above as well allows to conclude that the training of a single network for the classification of surgical instruments, whose elements present more similarities to each other than differences, becomes a problem of difficult solution, since not all the specific characteristics of each element can be extracted with the same type of filters or architectures, causing that a single network is able to classify easily some elements, but to have great difficulties with others.

For this reason, a tree-structured CNN and feature extraction centered only on the most relevant details of the instrument (through upper or lower image cutouts), becomes a feasible solution to achieve classification of surgical instruments, where the networks need to extract less features for each category. However, although the network of each branch has accuracy percentages above 80%, the final accuracy percentage of the tree will be strongly affected by the quality of the networks of the first branches, because a first misclassification triggers an error in the following branches, making accuracy not as high as expected.

In the case of instruments very similar to each other, it is fundamental to choose adequately the characteristics that allow to discriminate one category from another. For example, visually clear differences such as the position of the eye rings on the scissors allow to guide the training solely to that characteristic, as well as the characteristics related to thickness, as happens with the classification of the elements of the category "Sticks" for Tree 5, where, with the passage of the convolutions, the network begins to generalize some aspects and to highlight certain characteristics until the activations of each category differ considerably, despite having similar geometries.

Acknowledgment. The authors are grateful to the Nueva Granada Military University, which, through its Vice chancellor for research, finances the present project with code IMP-ING-2290 and titled "Prototype of robot assistance for surgery", from which the present work is derived.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
2. Farfade, S.S., Saberian, M.J., Li, L.J.: Multi-view face detection using deep convolutional neural networks. In: Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, pp. 643–650. ACM, China (2015). https://doi.org/10.1145/2671188.2749408
3. Lawrence, S., Giles, C.L., Tsoi, A.C., Back, A.D.: Face recognition: a convolutional neural-network approach. IEEE Trans. Neural Netw. **8**(1), 98–113 (1997). https://doi.org/10.1109/72.554195
4. Nagi, J., Ducatelle, F., Di Caro, G.A., Cireşan, D., Meier, U., Giusti, A., Gambardella, L.M.: Max-pooling convolutional neural networks for vision-based hand gesture recognition. In: 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), pp. 342–347. IEEE, Malaysia (2011). https://doi.org/10.1109/ICSIPA.2011.6144164
5. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: ICDAR, vol. 3, pp. 958–962 (2003)

6. Ciregan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3642–3649. IEEE, Rhode Island (2012). https://doi.org/10.1109/CVPR.2012.6248110
7. Abdel-Hamid, O., Mohamed, A.R., Jiang, H., Deng, L., Penn, G., Yu, D.: Convolutional neural networks for speech recognition. IEEE/ACM Trans. Audio Speech Lang. Process. **22**(10), 1533–1545 (2014). https://doi.org/10.1109/TASLP.2014.2339736
8. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1725–1732. IEEE, Ohio (2014)
9. Ahn, B.: Real-time video object recognition using convolutional neural network. In: 2015 International Joint Conference on Neural Networks (IJCNN), pp. 1–7. IEEE, Ireland (2015). https://doi.org/10.1109/IJCNN.2015.7280718
10. Ciresan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Convolutional neural network committees for handwritten character classification. In: 2011 International Conference on Document Analysis and Recognition (ICDAR), pp. 1135–1139. IEEE, China (2011). https://doi.org/10.1109/ICDAR.2011.229

Knee-Ankle Sensor for Gait Characterization: Gender Identification Case

Fabiola Monrraga Bernardino[1], Eddy Sánchez-DelaCruz[1](✉), and Iván Vladimir Meza Ruíz[2]

[1] Departamento de Posgrado, Instituto Tecnológico Superior de Misantla, Veracruz, Mexico
esanchezd@itsm.edu.mx, eddsacx@gmail.com

[2] Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México, Mexico City, Mexico

Abstract. Classification based on gait biomarkers is an area of study that includes approaches aimed at monitoring (vigilance), education and health. A correct classification is achieved depending on algorithms that serve that purpose, however, an accuracy must be available during the data acquisition of gait. In this study, a sensor network is proposed that allows to capture, in children, data of knee and right ankle. Results shows acceptable percentages of correct classification when implementing various machine learning algorithms, especially, combining the LogitBoost+Random Forest algorithms.

Keywords: Sensor network · Gait biomarkers · Classification

1 Introduction

In recent years there has been an increase interest in the diagnosis of degenerative diseases based on the gait [7]. In this field it is assumed a direct relationship between the disease and the way of walking, the disease has an effect on the motor system that impacts the gait. This work explores the construction of a sensor network of gait that allows the recording biomarkers that can later be analyzed by a professional (expert) and can support the diagnosis by her or him. Furthermore, a data analysis methodology can be associated with the process in which a predictive model can be generated so it can be used for automatically detect an aspect of gait: for instance to automatize the diagnostic of a patient. In addition, in order to test the device and the data pipeline we propose to use the measurements taken by the device in a task of gender detection.

Previous work have proposed sensors to record the gait in patients but with a single accelerometer, which is usually located at the ankle [3,4,8]. In this work we propose to use two sensor one in the ankle and the second one in the knee. We hypothesize that aspects of the gait can be better sensor in the knee, however since we keep the one in the ankle we can consider not just each characterization of both, but the relations that arise between both: ankle and knee. Both sensors are connected to the control unit which in this case is an Arduino and which is located in the lower back of the patient.

In order to test the capabilities of the recording device and the analysis pipeline which give us a predictive model, we suggest a straight forward task: gender identification through gait for children. The relation between gain and gender identification has been suggested before and visual characteristics has been used for discrimination among both genders [4]. In this work we focus our attention to machine learning techniques, we propose LogitBoost and Random Forest methodologies to model the data and create a predictive model. We hypothesize that a good characterization of the gait through the measurements of accelerations in the ankle and the knee should be enough to discriminate among both gender.

2 Previous Work

There has been several works that focus on gait to provide an insight on children health problems. Some of the approaches determine weight problems, other are proposed to prevent accidents, or to diagnose some underlaying illness. The following are some of these examples that are relevant for our work.

Sove of the previous research has been focused on sensing the gait with the goal of creating a description of the gait that can latter be interpreted by an expert, moreover in this same line there has been several works focused on children gait. In [3] the authors focus on the children physical shape, they use an accelerometer in the ankle to register the gait to classify children in four groups: low, normal, overweight and obese. Following a similar research line, in [4] the authors present a supervised machine learning technique to classify the obesity and overweight in children, in this case the recordings were done by children in the day to day activities. In contrast Nam [8] rather than focusing on classify the obesity they focus on activity detection in children. Their device also use one accelerometer, but this time it is used in combination with a barometer. Both sensors were put in the children hip. They aimed to help to prevent accidents at home. They collected data from the accelerometers in children from 16 and 29 months of age. They used statistics from the signal to classify the activities such as: media, standard deviation and the slope of the data on windows of the signal.

In [6] the authors face a more challenging scenario. In their case they tried to differentiate cases of ataxia from those of motor coordination disorder, since symptoms are similar. They suggest that gait might be able to better differentiate among the illnesses. In their research they used inertial sensors and a supervised classifier to obtain produce a predictive model. All these methodologies have the problem that require training data that potentially could be large in order to model the phenomenon right. However this premise is mostly no true since there is not a lot of gait data and there is a great variety of gait sensors. In order to tackle this problem, Taborri [10] proposes an unsupervised methodology based on Hidden Markov Models to recognize the gait in real time of two or four phases, this was implemented to control the active pediatric ortesis in lower limbs. The data in this case was senses using a three-axis gyroscope (Magnetic Inertial Measurement Units).

This methodology has been proposed to deal with other illness that not necessary are directly related to the motor system. In [5], the gait is proposed as medium to diagnose autism spectrum disorder in children. They presented a preliminary review of this viability of this diagnose. Although there are few studies focused on the psychological mental state of the patient's gait this idea has been further explored in [2], in which they focus on variance in the gait to diagnose autism spectrum disorders. In this case the system uses eight cameras and four floor plates to calculate the movements of the joints while walking and to characterize the gait. In this study they found significant difference between the diagnosed children and the control group, in particular in the ankle and the hip.

In this stage of the work we are focused on the design of the device which has the goal to be cheap, easy to wear for the patients and provide data which will be able to be used in the future with other aspect. In the meantime, we focus in gender identification to test our design.

3 Methodology

We propose to create a gait recording device to characterize the walking of a person, the goal is to record acceleration of two joins of the leg of a patient so the medical expert latter can use this data to guide her o his diagnose. In order to reach this goal we propose a network of two sensors to be collocated in the ankle and the knee, we aim to capture gait biomarkers that cal help to only as a guide to the expert but when enough information is collected a machine learning methodology could be use to create a predictive model.

Figure 1 shows the main elements on our methodology and how they relate among them. In the side of the user (i.e., patient) the network of the sensors (i.e., the device) is wore by she or he. This consist of two accelerometers, an Arduino and an harness, they are light weighted and comfortable to wear by means of being mounted in an elastic harness which could be easily adjusted in several type of users. The measurements of the sensor network are saved into a database per patient and path suggested to the patient. For instance, the patient can be suggested to walk in a straight line until certain point, turn and comeback. The collected data will be accessed by an expert which will perform an analysis on the data and produce an diagnose.

Additional to a traditional diagnose by an expert, in a machine learning framework the expert will have access to a predictive model which will help him or her with the diagnose. In this case, we can think on the predictive model as a support for the expert in order to complement her or his diagnose. In this work, we explored with various classifiers and the better result was obtained when combining two machine learning methodologies: LogiBoost and Random Forest classifiers.

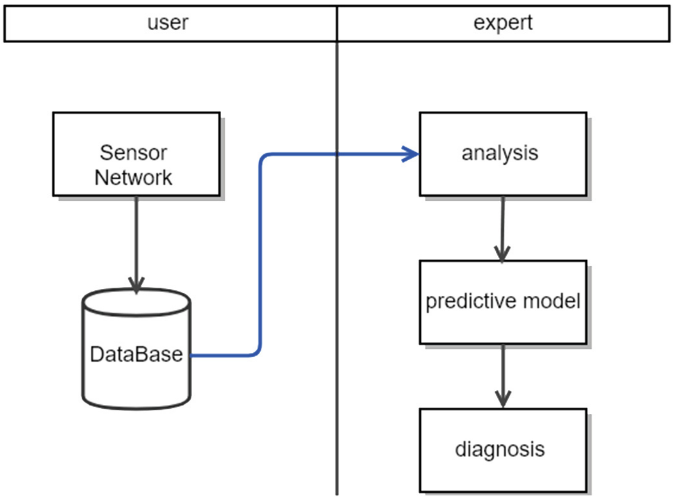


Fig. 1. Sensor network methodology

LogitBoost it is a boost algorithm. This extends AdaBoosts with a probabilistic framework in which the exponential loss is replaced by a conditional Bernoulli probability. We can think of LogitBoosts as an additive regression in which there are three main components: loss, the model function and the optimization algorithm. The lost is a LogitBoost, this is a logistic regression, the model is additive decision trees. The following algorithm shows the inner workings of LogitBoost [9].

```
**Input:** $Z = \{z_1, z_2, ..., z_n\}$ with $z_i = (x_i, y_i)$ as training set.
M, the maximum number of classifiers.
Star with $F^0(x_i) = 0$ and $p(x_i) = 1/2, i = 1, ..., n$
**for** *m=1,...,M;* **do**
    $a$)compute the wights and working response
    $w_i = p(x_i)(1_p(x_i))$, $z_i = (y_i^*)/\ p(x_i)(1 - p(x_i))$.
    $b$) Fit the function $F^m(x)$, using the tree-based learner, by a weighted
    least-squares regression $z_i$ to $x_i$ using weights $x_i$
    $c$) Update $F^{(m)}(x_i) = F^{(m-1)}$and
    $p(x_1) = exp(F^{(m)}(x_i)/(1 + exp(F^{(m)}(x_i))$
**end**
```

Algorithm 1. LogitBoost

In the case of Random Forest, this is a technique which improves the precision of the classification by incorporating randomness in the construction of individual classifiers of the decision tree type. There is also randomness in the bootstrapping of the data for each decision tree, and there is randomness in the way each dimension is choose. The following algorithm summarizes the inner workings of random forest.

```
Data: dataset T = (x, y), number of trees m, number of random levels k
Result: RF, a set of grown trees  Initialization RF
for i = 1 to m do
    T' ← boostrap(T)
    Tree ← trainDT (T'k)
    add Tree to RF
end
```

Algorithm 2. Random Forest

In this technique, the input are the data (T) which are previous recordings (x) with their corresponding diagnose (y), the number of decision trees to generate (m), and the maximum number of levels that each tree will have (k). For a each tree a new set (T') of training data is created by means of bootstrapping (i.e., sampling with replacement). Once each the data is selected a tree is selected by randomly selecting two features and choosing the best partition among both features, this is repeated until all the features have been analyzed or it reaches the maximum number of levels [1].

3.1 Sensor Network

As explained before the sensor network consist of two sensor, one located in the ankle and the other in the knee. The position of the sensors can be seen in Fig. 2 The sensor are of the type $MPU6050$ and can record acceleration and gyroscopic measurements. Both sensor are connected to an Arduino controller which is in charge to record the measurements.

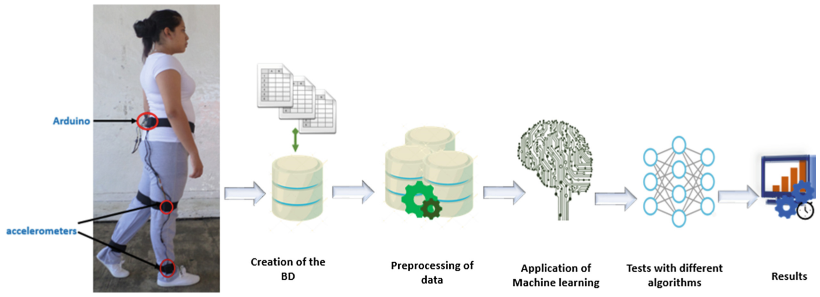


Fig. 2. DB analysis pipeline

The schematic of our device can be observed in Fig. 3. As can be seen the connection is straight forward and can be easily replicated. The information from the sensor flows through the SDA y SCL ports of the Arduino, this configurations allows to differentiate both of the sensors and register each one independently.

One of the goals of our research was to develop an accessible sensor network in terms of easy to wear and to collect the recordings. As can been seen in Fig. 2, it can be easily wear and manipulated by the person responsible to register the

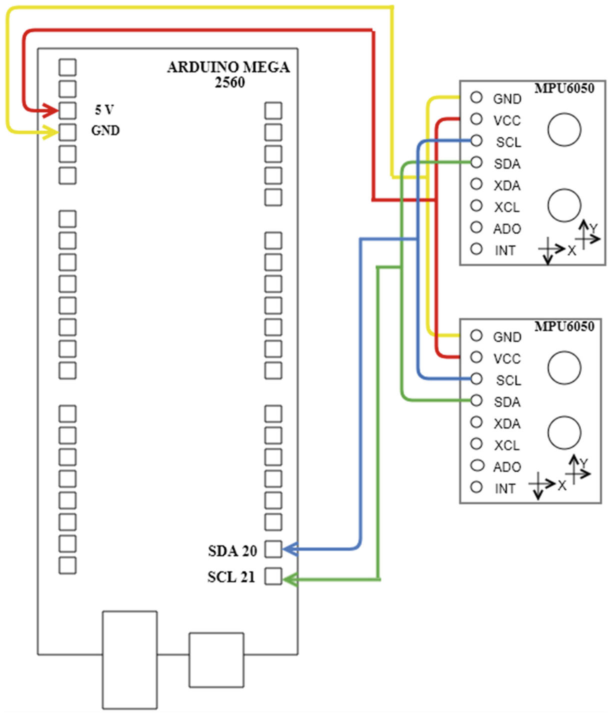


Fig. 3. Topology of network sensors

information. The cables have are leave hanging so the patient is free of movement and walks as natural as possible. But additionally our research was focus in creating a cheap device the use of an Arduino and the *MPU* sensor warranty this situation. The extra cost from the rest of the materials and manufacture of the harness was very little. In total we estimate the device could cost in this version 40 USD or 700 MXN. All the material with its cost is listed in Table 1.

Table 1. Material and costs of knee-ankle sensor

Material	Price (in MXN)
Arduino mega	$ 300.00
Sensor MPU6050	$ 80.00
Velcro	$ 50.00
Elastic	$ 40.00
Case	$ 120.00
Dupont cables	$ 100.00

3.2 Data Recolection

The procedure for the data was collected in the following manner, the children were explained in what the experiment consisted of and where each of the sensor was locates, as can be appreciates in Fig. 2. Once the network of sensors was put on the child, a test was performed in order to identify any problem with working with the sensor or the thighness of the bands. After the test the child was commanded to walk in a wide area, in this case was a court in the school. The child was asked to walk in a straight line. This walk took approximately 3 min and data was colected in a csv file for each child. The obtained records was joined in a single csv file to carry out experiments with various machine learning algorithms.

4 Experiments

In order to test our device and to test the proposed machine learning methodology we performed a gender identification task, in which the goal is to determine the gender of a children based on data from their gait. The machine learning pipeline can be observed in the Fig. 2. We record 10 children walking. We set the Arduino to record 400 points in 3 min, which is the maximum amount of time that children take walking the predefine path.

Table 2 present the statistics details of the ten children which participated in the experiment. They were 5 girls and 5 boys, the group has an average age of 13.6 years, average height of 1.58 m, average weight of 57 Kg and a average body mass of 21.4. The older children has 15 years old while the youngest 12 years old. Their physical shape is normal for their age and the stage of development.

Table 2. Children statistics details

Gender	Age	Height	Weight	Body mass
Boy	14	1.60 m	60 kg	23.43
Boy	14	1.65 m	61 kg	22.51
Girl	12	1.59 m	48 kg	19.22
Girl	13	1.62 m	56 kg	21.33
Boy	13	1.58 m	62 Kg	24.83
Boy	15	1.58 m	53 kg	21.23
Boy	14	1.65 m	61 kg	22.71
Girl	14	1.50 m	43 kg	19.28
Girl	15	1.54 m	47 kg	19.86
Girl	12	1.50 m	44 kg	19.55

Once we obtain the recordings form the children walking we proceed to perform a data analysis using the Weka system. This provides a great collection of algorithms, including LogitBoost and Random Forest, we perform several experiments with this collections of machine learning methods. Given the amount of data we used a cross validation setting in which we leave out 30% of the data for testing. Algorithms evaluation was done with standard configuration of software Waikato Environment for Knowledge Analysis (WEKA). The modeling was done in a computer with the following characteristics: SO Windows 10 Pro, an Intel (R) Core (TM) i3-3000 CPU a 2.00 GHz 2.60 GHz processor 8.00 GB RAM memory and the WEKA V.3.8 system.

5 Results

After a stage of development in which different combinations algorithms we identified that the combination of LogitBoost+Random Forest presented the highest performance. We found that we could discriminate the gender of the children with an accuracy of 92.04%, which shows that the recording of the gait and the machine learning pipeline are adequate to perform this task and furthermore to start researching other aspects such as diagnose of illnesses. Figure 4 shows the different performances from other combination of algorithms we can see that different algorithms in different combination had a good performance, however there are few which do not significantly improve the baseline which would be guess between both gender. However, there are several which have a performance over 85% which is good and signals that the task is in fact discriminative.

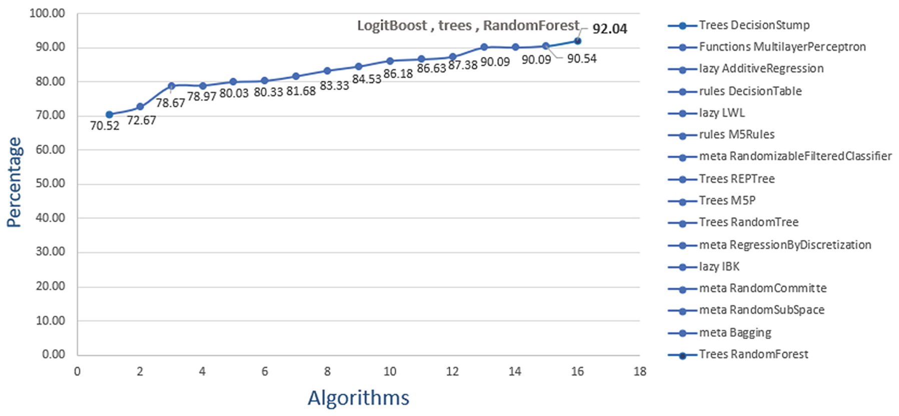


Fig. 4. Performance of algorithms.

Table 3 shows the confusion matrix from this discriminative experiments. We could observe that there are few confusion errors for the data we collected, girls are the most confussed but we have more data for this class, so we consider this normal, this is corroborated by the average value of ROC area: 0.972, which indicates a high rate in proportion of patients and healthy people correctly identified.

Table 3. Confusion matrix

a	b	Class
282	35	a = boy
18	331	b = girl

6 Conclusion and Future Works

In this work we present the design of an accessible network of sensors to record gait and the corresponding machine learning pipeline in order to create a predictive model from it. The device is aimed to help experts to diagnose some illness related to the gait. However, in the stage of our research we propose to use the sensor to discriminate between genders since it has studied before and it was a more straight forward setting. We record 10 children walking and obtained data from them which latter was used to predict the performance of the predictive model.

From our experiments we found that it was possible to discriminate between girls and boys based on the gait with an accuracy of 92. This are good results since they validate our sensor and the proposed pipeline. However there is still open question that we will address in future research:

- Focus on a specific illness, in particular we will focus on diabetic neuropathy. This illness is a complication of diabetes and consist in damage on the nervous system because of the diabetes. Currently, we have collected the data from several patients and we are looking forward to create different predictive models.
- Following, the line from previous research we could apply the methodology to children in the autistic spectrum and help to localize their degree of autism.
- Improve the network of sensors by incorporating more sensors to it.

In all this application, we aim to create a open database for other to look to the data and come up with other machine learning mechanism to improve the diagnose.

Acknowledgments. The authors are very grateful to the National Council of Science and Technology (CONACYT) for supporting this work.

References

1. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
2. Calhoun, M., Longworth, M., Chester, V.L.: Gait patterns in children with autism. Clin. Biomech. **26**(2), 200–206 (2011)
3. Clark, C.C., Barnes, C.M., Holton, M., Summers, H.D., Stratton, G.: Profiling movement quality and gait characteristics according to body mass index in children 9–11 years. Hum. Mov. Sci. **49**, 291–300 (2016)

4. Fergus, P., Hussain, A.J., Hearty, J., Fairclough, S., Boddy, L., Mackintosh, K., Stratton, G., Ridgers, N., Al-Jumeily, D., Aljaaf, A.J., Lunn, J.: A machine learning approach to measure and monitor physical activity in children. Neurocomputing **228**, 220–230 (2017). Advanced Intelligent Computing, Theory and Applications
5. Jamil, N., Khir, N.H.M., Ismail, M., Razak, F.H.A.: Gait-based emotion detection of children with autism spectrum disorders: a preliminary investigation. Procedia Comput. Sci. **76**, 342–348 (2015). 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IEEE IRIS2015)
6. Mannini, A., Martinez-Manzanera, O., Lawerman, T.F., Trojaniello, D., Croce, U.D., Sival, D.A., Maurits, N.M., Sabatini, A.M.: Automatic classification of gait in children with early-onset ataxia or developmental coordination disorder and controls using inertial sensors. Gait Posture **52**, 287–292 (2017)
7. Muro-De-La-Herran, A., Garcia-Zapirain, B., Mendez-Zorrilla, A.: Gait analysis methods: an overview of wearable and non-wearable systems, highlighting clinical applications. Sensors **14**(2), 3362–3394 (2014)
8. Nam, Y., Park, J.W.: Child activity recognition based on cooperative fusion model of a triaxial accelerometer and a barometric pressure sensor. IEEE J. Biomed. Health Inform. **17**(2), 420–426 (2013)
9. Otero, J., Sánchez, L.: Induction of descriptive fuzzy classifiers with the logitboost algorithm. Soft Comput. Fusion Found. Methodol. Appl. **10**(9), 825–835 (2006)
10. Taborri, J., Scalona, E., Rossi, S., Palermo, E., Patané, F., Cappa, P.: Real-time gait detection based on hidden Markov model: is it possible to avoid training procedure? In: 2015 IEEE International Symposium on Medical Measurements and Applications (MeMeA) Proceedings, pp. 141–145, May 2015

Methodology for Learning Multimodal Instructions in the Context of Human-Robot Interaction Using Machine Learning

Saith Rodríguez[1], Carlos A. Quintero[1(✉)], Andrea K. Pérez[1], Eyberth Rojas[1], Oswaldo Peña[1], and Fernando De La Rosa[2]

[1] Universidad Santo Tomás, Bogotá D.C., Colombia
{saithrodriguez,carlosquinterop,andrea.perez,eyberthrojas,oswaldopena}@usantotomas.edu.co
[2] Universidad de los Andes, Bogotá D.C., Colombia
fde@uniandes.edu.co

Abstract. This work shows the design, implementation and evaluation of a human-robot interaction system where a robot is capable of learning multimodal instructions through gestures and voice issued by a human user. The learning procedure can be performed in two ways: an instruction learning phase, where the human aims at teaching one instruction to the robot by performing several repetitions and an instruction receiving phase where the robot reacts to the instructions given by the human and possibly asks for feedback from the user to strengthen the instruction's model.

1 Introduction

Every day more robots are immersed in daily-life tasks and disciplines such as home tasks, medical-care and industries, among others. Moreover, every day this human-robot interaction becomes more personal and accessible to everyone, from children to elders, from engineers to farmers. Therefore, it is paramount for robots to interact with people using types of communication familiar to anyone with whom it would interact. In other words, robots should understand and use natural and intuitive interactions with people.

To succeed in the aforementioned task, it is necessary to tackle the problem from multiple areas of expertise such as human-robot interaction, navigation, mapping, mechanics, object manipulation, among others. This article is focused in human-robot interaction, which includes sensor integration to capture data, artificial intelligence systems development, and implementation of mechanisms to interact with humans in a natural, reliable and simple manner. Specifically, we deal with the problem of one robot that learns how to interact with a human in a natural way by using inputs from varied nature, i.e., audio and video. The instruction learning process is achieved by several repetitions of the instruction

© Springer International Publishing AG, part of Springer Nature 2018
C. Brito-Loeza and A. Espinosa-Romero (Eds.): ISICS 2018, CCIS 820, pp. 41–55, 2018.
https://doi.org/10.1007/978-3-319-76261-6_4

performed by the human. Afterwards, the human can execute the instruction and the robot will understand and will react to it. The proposed model uses the input data to build two classifiers; one for the audio data and one for the video stream. Both classifiers consists of ensembles of one-class SVMs. The final decision of which instruction the input data belongs to, is taken based on each sample confidence.

This article is organized as follows, the first section presents the state-of-art in instructions learning in human-robot interaction. The following section presents the proposed methodology, first we show the system's overall architecture for learning multi-modal instructions in the human-robot interaction context; then, every module is explained in detail. The next section shows the experimental setup and the results obtained. Finally, the last section summarizes the conclusions of our work.

2 Related Work

Recently, there have been many research and application works around the problem of Human-Robot Interaction (HRI), especially, in proposals that aim at recognizing instructions issued by a human to a robot that makes the interaction as natural as possible to the human. The work in [1] show a proposal that uses gesture classification to orient a telepresence robot to direct its attention to specific targets using a Kinect. The gesture recognition is achieved by processing visual information that allows the system to track the human head as well as audio information from the robot's microphone to locate the source of sound. In their work, they use Localist Attractor Network for the gesture recognition task and Short-Term Memory to obtain the attention direction.

Other works also propose the application of machine learning techniques in the context of HRI to recognize only gestures by using the information of the human body motion by tracking its joints using a Kinect [2]. The authors propose three phases. In the initial phase, gesture classification is performed on a set of known human movements using an edited video stream. Each gesture consists of a set of frames where the Kinect detects the position of the 20 human joints for each frame. The second phase attempts to detect unknown movements by extending the techniques proposed in the first phase, detecting movements in an unedited video stream which also captures random gestures. In the third phase, a set of rules are applied in order to filter gestures that were classified into the known vocabulary but were not really intended to be instructions for the robot. Using this methodology, the authors build a proposal to classify gestures over unedited video streams.

In [3,4], a proposal is presented to create an automatic gesture classifier capable of discriminate between known gestures and signals with no meaning. In this work, a Kinect is also used to acquire the input data and a set of 4 gestures with 300 samples each is considered. Additionally, a set of random gestures is also acquired and included in the classifier's accuracy calculation during the validation process. In [5], an audio classification system is proposed using SVMs and RBF-NN. In this work, the authors propose a feature extraction phase for the audio data using Mel-frequency Cepstral Coefficients (MFCCs). Such coefficients have shown to be a key step in the construction of classification systems that use audio inputs [6] since they accurately represent the acoustic phenomena by highlighting low frequencies.

In [7] the authors introduce a classification system with 5 different labels by analyzing visual features. This work is the base for [8,9], where the instructions given by the human are captured, not only using visual information, but also audio features. These works propose a first attempt to combine both data sources (audio and visual) to classify the 5 labels (news, advertisement, sport, serial and movies). Like in the previous works, the authors also use the MFCCs to extract audio features. For the gesture data they propose the use of color histograms for video segmentation. The information obtained from both sources is used to construct a different classifier, i.e., one for audio and one for video, using Support Vector Machines. By weighted sum of each individual classifier, the output is combined into one single answer. Many authors have proposed the use of individual classifiers, mostly for audio and video classification and the general idea is to create several classifiers and combine them in different ways to improve the general system's performance [10].

One of the most interesting applications of HRI is that of collaboration between humans and robots to execute specific tasks. However, several factors need to be taken into account to successfully accomplish such interaction. Specifically, the robot and humans goals may not be completely aligned when long-term interactions are established. The authors in [11] claim that online learning is of paramount importance to alleviate this situation. However, it is not yet widely used in most human-robot interactions for two main reasons: On one hand, typical learning algorithms are incapable of learning in time frames required to interact effectively with humans. On the other hand, there is a random exploration property present in most online learning algorithms that may cause inappropriate results when a robot interacts with humans. In their work, they propose the use of effective communication strategies to accelerate the learning process during the interaction.

3 Proposed Methodology

In the following sections we show the proposed methodology to design, develop and test a HRI system capable of learning instructions issued by a human using multimodal inputs, i.e., audio and gesture signals.

3.1 Solution Architecture

Figure 1 shows our proposed architecture. The process starts by using the Kinect to capture both the audio signal from the human voice and the gestures from the video stream. The overall design consist of 5 main modules, namely Motion

Detection, Voice Activity Detection, Feature Extraction (skeleton features), Feature Extraction (MFCC) and Multimodal Classifier. The first two modules allow the detection and identification of possible instructions given by the human. These modules allow our system to discriminate between small movements and audio that may be considered noise, from actual instructions issued by the human. Afterwards, a feature extraction process is performed over the filtered data in order to extract and keep the important information required for the classification task. Finally, a multimodal classifier provides an answer that is transmitted to the user through an action performed by the robot. This validation is made in order to improve the learning model while the interaction runs.

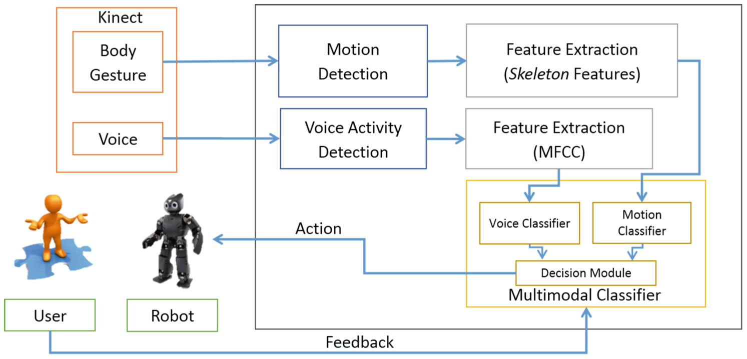


Fig. 1. Overall system architecture

The following sections show a detailed description of each module within the overall architecture.

3.2 Inputs Description

One of the main important contributions within our proposal is the input's multimodal nature, i.e., it comes from two different sources for the same instruction. One of them is the video stream to detect gestures and the other one is audio data to detect a voice instruction. For this purpose, we have integrated the Microsoft Kinect sensor which provides us with both data sources.

Table 1 shows the sampled frequency and format of each input captured using the Kinect.

3.3 Gestures Detection and Voice Activity

The main target of the gestures and voice activity detection modules is to effectively detect and extract the pieces of information, provided by the Kinect, where

Table 1. Sampling frequency and format of each input captured using the Kinect

Input	Format	Frequency
Body gesture	(x, y, z) datapoints that indicate the spatial coordinate (in meters) of certain human body joints	30 fps
Audio	Data array from the 24 bit ADC	16 kHz

presumably a human has issued an instruction, either spoken or by using a gesture. In both cases, the modules are implemented through a finite state machine especially designed to follow the behavior of each type of input signal: for the audio input, the signal energy is computed and processed, while for the gesture signal a change of coordinates is performed in order to obtain an estimated "amount of movement" from the user in certain periods of time. For the audio signal, if the calculated energy does not exceed a threshold value, it is considered noise. Similarly, if the calculated "amount of movement" is not high enough, it is not considered an actual gesture.

3.4 Feature Extraction for Gesture Instructions

This module is in charge of extracting useful information from the data provided by the Kinect in the form of skeleton joint coordinates. The main idea is to build a set of fixed-size distinctive features that may be used to classify between different body gestures issued by different users.

The first step consists of performing a sub-sampling procedure on the data in order to match the number of samples between gestures, since it may vary according to the user and the specific body movement. The general procedure is shown in Fig. 2 for two different body gestures, one with n_1 samples and another with $n_2 < n_1$ samples. Both gestures are sub-sampled to match a desired number of m samples.

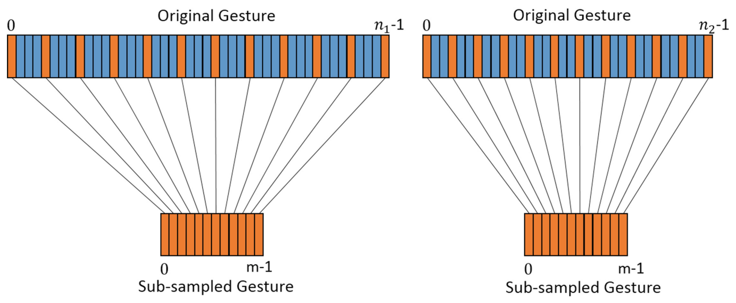


Fig. 2. Sub-sampling procedure applied to body gesture data for two gestures with a different amount of original sample numbers.

Initially, the new sampling frequency is computed using the original number of samples and the desired number of samples as shown in Eq. (1).

$$f_m = \frac{n}{m} \tag{1}$$

where
f_m is the new sampling frequency
n is the number of samples from the original gestsure
m is the desired number of samples for the sub-sampled gesture

The new dataset of size m is obtained by extracting the original samples as shown in Eq. (2) for $i = 0, 1, ..., m - 1$.

$$\hat{S}_i = S_{i * f_m} \tag{2}$$

where:
$\hat{S}_i$ is the i-th sub-sampled gesture sample
S_i is i-th original gesture sample
f_m is the new sampling frequency

This sub-sampling process guarantees that every gesture performed by the user is represented with m samples that are in turn composed of 9 skeleton (x, y, z) coordinate joints. However, these gesture representations are static coordinates that do not represent the motion of the gesture itself. To alleviate this situation we have proposed to create vectors that represent the gesture motion among time using the available joints coordinates. The procedure consists on calculating origin-centered vectors for each joint where the vector tail is located in the current sample and its head is in the next corresponding joint sample. A diagram of the general procedure is shown in Fig. 3 for two consecutive samples.

The amount of features for the body gesture input corresponds to the $(m - 1)$ joint motion vectors times the 9 skeleton joints times the 3 (x, y, z) spatial coordinates for a total of $27(m - 1)$ features.

3.5 Feature Extraction for Voice Instructions

This module will receive audio data captured by the Kinect's microphone. The idea is to obtain a set of features from such data stream useful to classify between instructions given orally by a human.

For this purpose, we used the approach followed by [12], where the signal is split in smaller audio windows where the coefficients of the Discrete Fourier Transform (DFT) are calculated for each one, in order to estimate the power spectral density. This density is then filtered using a set of overlapped triangular filters to highlight the importance of low frequencies imitating the behavior of the human ear. Afterwards, the signal is scaled to a logarithmic scale and the Discrete Cosine Transform (DCT) is applied in order to eliminate the dependency of adjacent bands. Finally, the signal is split into several windows where the MFCCs are calculated for each one and then averaged for all windows. A summary of the whole procedure is shown in Fig. 4.

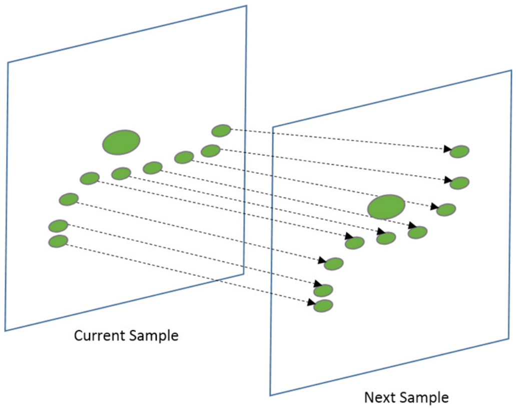


Fig. 3. Transformation from joint coordinates to joint motion vectors using the current and the next samples

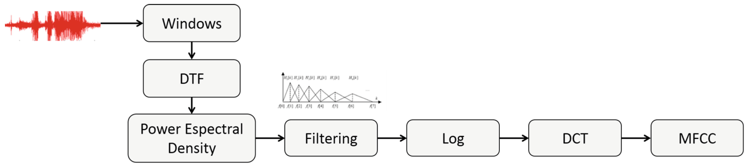


Fig. 4. Procedure for the extraction of MFCCs

3.6 Classifier

The classifier is the module responsible for receiving the processed data from the feature extraction modules and providing an answer of which instruction, if any, has been issued by the human user.

We have proposed the architecture shown in Fig. 5 for the multiclass multimodal classification task. A specific classifier is created based on the nature of the input data, i.e., one classification for the audio input and another for the gesture motion input. A decision output module takes the outputs of both classifiers and provides the general classifier output.

This module is capable of performing two different functions depending on the phase of interaction between the human and the robot, namely the **instruction learning phase** and the **instruction receiving phase**. On the first phase, the human attempts to teach one instruction to the robot by means of a training procedure, while on the latter, the human user issues instructions that the robot attempts to understand and execute. The steps performed by the classifier in each interaction phase are described in Fig. 6.

During the *instruction learning phase*, the human presents to the robot a set of repetitions of the specific gesture and voice command that determines one instruction. The classification module receives such multimodal inputs and uses them to build, through a training process, a model capable of accurately classify such instruction when presented again. In this phase, the classifier

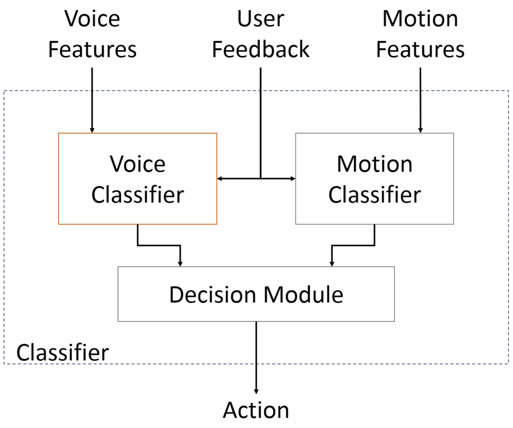


Fig. 5. Proposed architecture for the automatic classifier

performs a training function (Fig. 6(a)). During the *instruction receiving phase*, the human issues an instruction which is processed by the classifier to determine the user's intention. According to its result, the robot provides an action and then receives feedback from the user in order to enlarge the training database which in turn further strengthen the existing classification model. During this phase, the module acts mostly as a classifier (Fig. 6(b)).

Gesture and Voice Classifiers. As shown in Fig. 5, the general classifier is integrated by two independent classifiers; one for the audio data and one for the motion data. Each classifier is built independently using Support Vector Machines (SVMs), as these have shown high success in classification problems for audio and motion data [13]. In Fig. 7 we show our proposed architecture for each separate classifier using a special class of classifiers known as *One-class SVM*.

The One-class SVMs are learning machines capable of discriminate whether a given input sample belongs to one class or not [14]. Similar to other SVM-based algorithms, they rely on the kernel trick, a mathematical concept that allows the algorithms to create non-linear classification boundaries. In this work we use the widely used Gaussian and Polynomial kernels that have shown improved performance for general applications. Unlike binary classifiers, the One-class SVMs do not require training samples from other classes other than the one they are built for in their training process. Using a set of One-class SVMs, i.e., one for each class, as shown in Fig. 7, it is possible to build complete multiclass classifiers using one classifier per instruction. This architecture has the advantage that each new instruction provided by the user is independent from the others and does not require retraining the whole classifier, but only the one in charge of the new class.

The training procedure of the One-class SVMs is performed following a typical methodology for building statistical learning models [15]. The complete

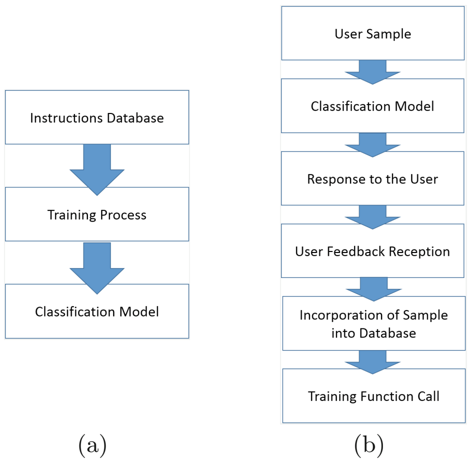


Fig. 6. Functions performed by the classifier (a) During the training function, the database for each instruction is built while the human interacts with the robot and is used in a training process that finally allows the creation of a classification model. (b) During the classification function, a user provides a sample to the robot which passes through the classification model. Depending on the classifier's output, the robot provides a reaction and asks (or not) for feedback to identify the real data label that is finally incorporated into the dataset to retrain the classifier.

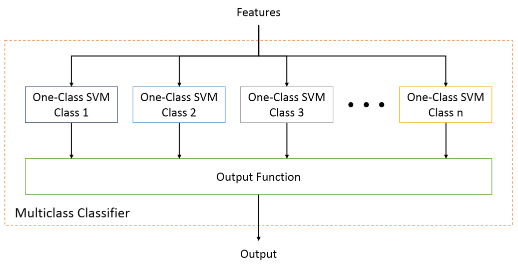


Fig. 7. General architecture for the multiclass classifiers for audio and motion data. The input data is passed through all one-class SVMs and then, the answer is merged into one single output.

dataset is split into two disjoint sets; one for the training process and other for testing the generalization capabilities of the classifier. In this case, the training data for each One-class SVM is made of samples that belong only to one instruction. However, the testing set for each One-class SVM classifier is composed of samples from the own class, but also of data samples from other classes.

In order to obtain classifiers with high generalization capabilities, the values of the kernel parameter and regularization parameter C are chosen such that the empirical generalization error is minimized [15]. For this, a model selection methodology is applied creating a grid search of the discretized parameter space.

In each parameter combination, a model is created using the training set and tested using the testing set. The chosen model is the one that shows the lowest classification error (i.e., the fraction of missclassified samples) on the testing dataset.

Finally, our methodology includes a grid refinement step where we begin with a coarse grid and end with a recursively finer grid. The recursion is achieved by creating new parameter values the half closer than the previous grid size, until a desired generalization error is achieved or a maximum number of iterations is reached.

Classification Confidence. One additional parameter from our proposal is the calculation of a classification confidence value δ for each classifier for an input sample. This value provides the classifier's certainty about its output and is closely related to the concept of classification margin for each input data point. The general idea is that the higher the margin, the higher its classification certainty. Finally, we propose a margin normalization that will allow us to compare the margins between different classifiers for one single input sample.

The confidence value of the j-th classifier when presented the input sample x is as shown in Eq. (3)

$$\delta_j(x) = \frac{m_j(x)}{\hat{m_j}} \tag{3}$$

where $m_j(x)$ is the margin of the input sample x and $\hat{m_j}$ is the average margin of the j-th classifier in the training dataset.

This confidence value may be understood as follows:

- If $\delta_j(x) \approx 1$, it means that the margin is similar to the average margins for the classification set.
- If $\delta_j(x) > 1$, the margin is better than the average, i.e., there is a high certainty on the classifier's output.
- If $\delta_j(x) < 1$, the margin is lower than the average and hence there is a high uncertainty regarding the classified sample.
- If $\delta_j(x) < 0$, the input data point does not belong to the class and its magnitude represents the certainty of not belonging to the class, i.e., the higher the margin's magnitude, the lower uncertainty of the classified sample.

Multiclass Output Function. The j-th classifier outputs two values when presented an input sample x: (1) its classification label $y_j(x)$, whose value is $y_j(x) = 1$ if it labels the input sample as belonging to its class or $y_j(x) = 0$ otherwise and (2) its classification confidence $\delta_j(x)$. The last module within the classifier is the output function that receives the outputs of all One-class SVMs and merge them into one single answer (classification label and confidence). This module implements the functions shown in Eqs. (4) and (5) in the pressence of N classifiers.

$$\hat{y}(x) = \begin{cases} y_j \text{ if } \sum_{i=0}^{N} y_i(x) = 0 \\ y_k \text{ if } \sum_{i=0}^{N} y_i(x) \neq 0 \end{cases} \tag{4}$$

$$\hat{\delta}(x) = \begin{cases} 0 & \text{if } \sum_{i=0}^{N} y_i = 0 \\ \max_{i=1:N} \delta_i(x) & \text{if } \sum_{i=0}^{N} y_i \neq 0 \end{cases} \tag{5}$$

where $j = \arg\min_{i=1:N} \|\delta_i(x)\|$ and $k = \arg\max_{i=1:N} \delta_i(x)$.

The general idea is to assign to the input data point x the label of the class "closer" to the classification boundary, when all the classifiers claim that it does not belong to any class. In the opposite case, where one or more classifiers assign the input sample as belonging to their own class, the assigned label is the one given by the classifier with highest confidence.

It is noteworthy that this architecture is feasible in the context of HRI, specifically in our proposed framework, since each classifier is anticipated to have high bias due to the low amount of data available for training, compared to other applications and the complexity of the problem at hand.

3.7 Decision Module

Both classifiers, the one related to the audio data and the one that processes the gesture actions, finally produce a label and a confidence value $(\hat{y}_a(x), \hat{\delta}_a(x))$ and $(\hat{y}_v(x), \hat{\delta}_v(x)))$ respectively. There is a final module that receives the answer provided by the two classifiers and makes the final decision regarding the class of the input sample. The *decision module* implements the function in Eq. (6) to produce the final classification output $\hat{y}_g(x)$.

$$\hat{y}_g(x) = \begin{cases} \hat{y}_v(x) & \text{if } \hat{y}_v(x) = \hat{y}_a(x) \\ \hat{y}_v(x) & \text{if } \Big((\hat{y}_v(x) \neq \hat{y}_a(x)) \wedge ((\hat{\delta}_v(x) - \hat{\delta}_a(x)) \geq l)\Big) \\ \hat{y}_a(x) & \text{if } \Big((\hat{y}_v(x) \neq \hat{y}_a(x)) \wedge ((\hat{\delta}_v(x) - \hat{\delta}_a(x)) \leq -l)\Big) \\ 0 & \text{if } \Big((\hat{y}_v(x) \neq \hat{y}_a(x)) \wedge (|\hat{\delta}_v(x)) - \hat{\delta}_a(x))| < l)\Big) \end{cases} \tag{6}$$

The decision module output can be understood as follows:

– If both classifiers agree on the class, then such output is the final decision
– If the classifiers disagree on their outputs, and the difference in their confidences is greater than certain threshold value, then the overall output is the one with largest confidence
– If the classifiers disagree on their outputs and the difference in their confidence *do not* exceed certain threshold value, then the overall classifier output is "don't know", meaning that the instruction has not been recognized.

3.8 Human-Robot Interaction

The last step in our methodology is the actual interaction between the robot and the human. The robot will execute an action according to the output given by the classifier module and hence two possibilities arise: (1) The robot may be highly certain of its answer or (2) the robot has high uncertainty about it. In both cases, the system may use the current sample to further strengthen the classifier's model (i.e., by including it into the training process and retraining the one-class SVM). In the first case, the label assigned to the input sample is the one assigned by the robot (since it is highly certain of its answer). However, in the latter, a feedback from the user is required. This feedback is provided in the form of *approved* or *not approved* signals, meaning that the robot's action is the one expected by the human or not, respectively. If the robot's output was not the one expected by the human, the input sample is not included into the training process (since its true label is unknown).

4 Experiments and Results

We have prepared an experimental setup to test our proposed methodology using a Kinect sensor and a DARwIn-OP robot. Figure 8 shows a human user interacting with the robot in 3 possible steps. Initially, the user performs several repetitions of the instruction that needs to be learned. Afterwards, the user shows to the robot the action that needs to be done when such instruction is issued. Finally, the user issues the instruction and waits for the robot's action.

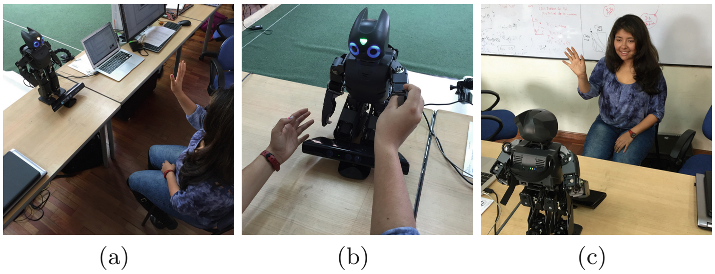
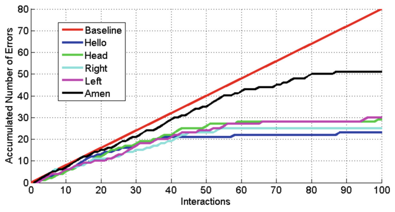


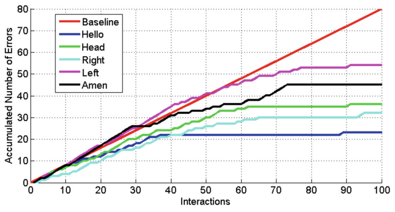
Fig. 8. Experimental setup for the human-robot interaction test: (a) The human performs several repetitions of the instruction to the robot. (b) The human shows the robot the expected answer (c) The human issues the instruction

For the experiment we have asked 4 different users to interact with the robot to teach the system 5 different instructions, namely: **Hello**, **Head**, **Right**, **Left** and **Amen**. Every instruction is composed of a voice command and a body gesture performed by the human. In the *instruction learning phase* the user performs 25 repetitions of each class and these data are used by the robot to build a model for each one. Afterwards, the user starts interacting with the robot 100 times for each instruction, for a total of 500 interactions. During each interaction, the robot has the opportunity of asking for feedback from the user and hence strengthen the classifier's model. In Fig. 9 we show the results of the interaction between the human and the robot for two users when performing the 100 interactions for each class.

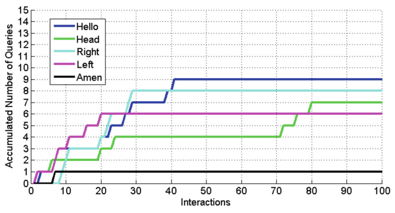
In Fig. 9(a) and (b) we present an accumulated count on the number of errors made by the learning system as the interactions pass. In the plots, we show the



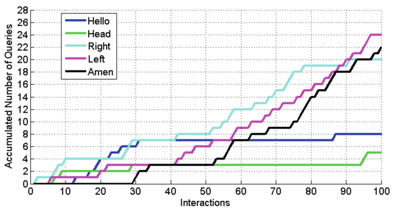
(a)



(b)



(c)



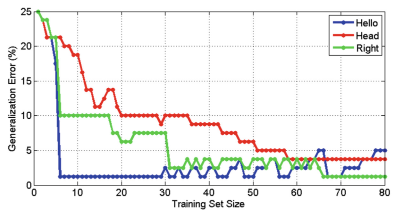
(d)

Fig. 9. (a) Accumulated number of errors during the human-robot interaction for user 1. (b) Accumulated number of errors during the human-robot interaction for user 2. (c) Accumulated number of queries during the human-robot interaction for user 1. (d) Accumulated number of queries during the human-robot interaction for user 2.

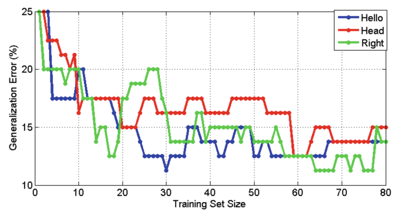
behavior of a Baseline classifier which randomly (uniformly distributed) chooses one of the 5 classes regardless of the input sample. It is noteworthy that as the interaction progresses, for all classes, the accumulated number of errors moves away from the classifier baseline behavior, showing that our system is, in fact, learning throughout the interaction with the user for all classes. We can also see that, for all classes, the number of accumulated errors, for both users, flattened after a certain number of iterations, meaning that the number of data samples included in the training process during the interaction is in fact improving each classifier model.

In Fig. 9(c) and (d), we also show the accumulated number of queries made by the robot during its interaction with the human. Recall from Sect. 3.8 that the robot may ask for feedback to the human only when its output confidence is not high enough. The results show that, for most classes, the robot requests for queries mostly at the beginning of the interaction and that these queries tend to decrease during the interaction. Note that this is an expected result since the models for each One-class SVM become more accurate when new samples are included into the training process and hence the confidence values of each classifier tend to increase. Nevertheless, for some classes, of certain users, it is possible that more queries are required before their confidence is high enough to avoid asking for user feedback.

Finally, we show in Fig. 10 an approximation to the generalization error for three out of the five classes for both classifiers, motion and voice, separately, as the training set grows.



(a)



(b)

Fig. 10. (a) Generalization error for the classes "Hello", "Head" and "Right" for the gesture classifier. (b) Generalization error for the classes "Hello", "Head" and "Right" for the audio classifier.

For both cases, it is possible to note that the generalization error decreases as the training dataset size increases. However, the gesture classifier shows lower generalization error than the audio-based classifier.

5 Conclusions

We have proposed an instruction learning system capable of using multimodal inputs in the context of human-robot interaction that makes it possible for a human with no knowledge or experience in programming, robotics and/or engineering to control or receive help from a robot in a natural way.

Our proposed architecture for the learning system is based on the ensemble of several One-class SVMs to create a multiclass classifier, one for each input mode. This architecture allows the complete system to improve its model every time a new data sample is available without retraining the complete multiclass model, but only the one-class classifier corresponding to the class of the input data point. We believe this is an important contribution since it allows the incorporation of the training process as the interaction with the user progresses in a moderate computational time.

Finally, we have proposed a possible feedback query from the robot, when the certainty of its output is not high enough by computing an overall confidence value of the learning and classifying system. This confidence value is related to a normalized value of the classification margin for each data sample, that allows the multiclass classifier to decide the output label for the input data.

References

1. Tee, K.P., Yan, R., Chu, Y.: Gesture-based attention direction for a telepresence robot: design and experimental study. In: Proceedings 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE (2014)
2. Bhattacharya, S., Czejdo, B., Perez, N.: Gesture classification with machine learning using kinect sensor data. In: Third International Conference on Emerging Applications of Information Technology (EAIT) (2012)

3. Huang, J., Lee, C.W., Ma, J.: Gesture recognition and classification using the microsoft kinect. In: 5th International Conference on Automation, Robotics and Applications, ICARA 2011 (2009)
4. Zhang, H., Du, W.X., Li, H.: Kinect Gesture Recognition for Interactive System (2009)
5. Dhanalakshmi, P., Palanivel, S., Ramalingam, V.: Classification of audio signals using SVM and RBFNN. Expert Syst. Appl. **36**(3), 6069–6075 (2009)
6. Sarchaga, G., Sartori, V., Vignoli, L.: Identificacin automtica de resumen en canciones (2006)
7. Suresh, V., Mohan, C.K., Swamy, R.K., Yegnanarayana, B.: Content-based video classification using support vector machines. In: Pal, N.R., Kasabov, N., Mudi, R.K., Pal, S., Parui, S.K. (eds.) ICONIP 2004. LNCS, vol. 3316, pp. 726–731. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30499-9_111
8. Subashini, K., Palanivel, S., Ramaligam, V.: Audio-video based segmentation and classification using SVM (2012)
9. Subashini, K., Palanivel, S., Ramalingam, V.: Audio-video based classification using SVM and AANN. Int. J. Comput. Appl. **44**(6), 33–39 (2012)
10. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Trans. Pattern Anal. Mach. Intell. **20**(3), 226–239 (1998)
11. Babushkin, V., Oudah, M., Chenlinangjia, T., Alshaer, A., Crandall, J.: Online learning in repeated human-robot interactions. In: Artificial Intelligence for Human-robot Interaction: Papers from the AAAI Fall Symposium, pp. 42–44. AAAI Press (2014)
12. Molau, S., Pitz, M., Schluter, R., Ney, H.: Computing mel-frequency cepstral coefficients on the power spectrum. In: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 73–76. IEEE Press (2001)
13. Rodriguez, S., Pérez, K., Quintero, C., López, J., Rojas, E., Calderón, J.: Identification of multimodal human-robot interaction using combined kernels. In: Snášel, V., Abraham, A., Krömer, P., Pant, M., Muda, A.K. (eds.) Innovations in Bio-Inspired Computing and Applications. AISC, vol. 424, pp. 263–273. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-28031-8_23
14. Schölkopf, B., Williamson, R.C., Smola, A.J., Shawe-Taylor, J., Platt, J.C.: Support vector method for novelty detection. In: Solla, S.A., Leen, T.K., Müller, K. (eds.) Advances in Neural Information Processing Systems 12, pp. 582–588. MIT Press (2000)
15. Hsu, C.W., Chang, C.C., Lin, C.J.: A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University (2003)

Deep Learning for Photovoltaic Power Plant Forecasting

E. J. Alejos Moo[1](✉), J. Tziu Dzib[1], Jorge Canto-Esquivel[2], and A. Bassam[1]

[1] Facultad de Ingeniería, Universidad Autónoma de Yucatán, Av. Industrias no Contaminantes, Mérida, Yucatán, México
e.alejos15@gmail.com, josetziu@me.com, baali@correo.uady.mx

[2] Instituto Tecnológico de Mérida, Av. Tecnológico km. 4.5 S/N, Mérida, Yucatán, México
jorge.canto@itmerida.mx

Abstract. Deep Learning is getting a relative importance in the field of machine learning due to better performance in fields of classification and pattern recognition. However, deep models have seen little use in time series forecasting. Thus, the purpose of this work is to investigate the performance of such models in power plant output forecasting. A classical Artificial Neural Network with one hidden layer and two Deep Learning models were developed to forecast the output from a photovoltaic power plant. A Recurrent Deep Neural Network with Long Short Term Memory and a Deep Neural Network were proposed to predict future values; trained by the Adam algorithm and validated using R, $RMSE$ and $MAPE$ statistical criteria. Using deep models improves the accuracy of forecasting better than models without a large hidden layer size. This improvement is demonstrated by training several structures of Deep Models and feed forward Neural Networks models. Correlation coefficient of 1.0 is achieved using a deep architecture for this case study.

Keywords: Machine learning · Artificial Neural Networks
Recurrent Deep Neural Network · Solar energy · Gradient descent
Adam algorithm · Statistical comparison

1 Introduction

The world's energy consumption grows as human population does, thus increasing the emissions of CO2, causing global warming and biosphere contamination. Alternatives for energy generation became public concern since 1991 to substitute fossil fuels with a cheap and abundant energy source. Solar energy is inexhaustible, renewable and free to use in a lot of applications. Photovoltaic (PV) cells for electricity generation is the fastest growing energy technology since 2002 [1]. The international Energy Agency's Photovoltaic Power System Programmers latest report found that 75 gigawatts of PV modules were installed globally in 2016, a grow up of 32.89% in one year [2].

PV power forecasting holds a crucial role in the planning process of a PV power plant's schedule, maintenance, and market operation. Due to the variability of the energy generation, PV power forecasting is challenging. There are many influencing issues such as climate factors and social activities which cause the data to be highly nonlinear [3]. The power output from a PV system has a high correlation with the solar radiation, but the nature of the weather and the uncertainties of the atmospheric variables such as temperature, cloud amount, dust and relative humidity makes solar irradiation hard to predict [4]. Common time-based PV power forecasting is classified into short (STF), medium (MTF) and long term (LTF) forecasting. STF is associated with power prediction from few hours to days ahead. MTF hands forecasting from days to months and LTF deals with prediction trough years.

In the literature there exists many studies about PV power forecasting methods based on physical models, statistical equations and artificial intelligence (AI). Chu *et al.* [5] presented the linear stationary model Auto Regressive Moving Average (ARMA) to short-term predict the power output from a solar plant. Linear non-stationary model as the Auto-Regressive Integrated Moving Average (ARIMA) were used by Pedro and Coimbra [6] for predicting the power generated by a PV plant operating in Merced, California. Both works compare results against forecasting models of Artificial Intelligence. The ANN evaluated by the authors achieved better accuracy than the statistical models. Antonanzas *et al.* [7] presented a summary of techniques about solar power forecasting, including statistical and AI methods, demonstrating that ANNs performs better.

Not long ago, diverse architectures of ANN with two or more hidden layers, called Deep Neural Networks (DNN) have emerged in the field of machine learning with tremendous success. The Deep Learning models were classically applied to problems of image processing, speech recognition, natural language processing and transfer learning. Due to the state-of-art success of DNN, they are proposed to others fields of work like forecasting time series [8]. Anderson and Ludemir [9] presented a Deep Learning model to forecast Wind Speed in the Northeastern Region of Brazil. Qiu *et al.* [3] proposed for the first time an ensemble of Deep Learning Belief Networks (DBN) for time series forecasting of load demand. Gensler *et al.* [10] used DBN, auto encoders and Recurrent Neural Networks (LSTM) for energy power forecasting. The results using Deep Learning show a superior forecasting performance compared to ANN.

This works aims to investigate the use of DNN models applied to time series forecasting, predicting the output from a PV power plant in the north region of Mexico. We compare an Artificial Neural Network (ANN), a Deep Neural Network (DNN) and a Recurrent Deep Neural Network (R-DNN), both deep models trained with ADAM algorithm, and a Neural Network of a single hidden layer trained with a classical algorithm named Gradient Descendent. This paper is organized as follows: for the first section, we present this brief introduction, followed by an overview of Artificial Intelligence Models, next, the organization for the experimental setup is presented. After that, we present the training process and architecture selection for the Artificial Intelligence Models.

2 Artificial Intelligence Models

2.1 Artificial Neural Networks

Artificial Neural Networks (ANN) are a set of interconnected processing units that uses mathematical and computational techniques to solve problems from complicated, imprecise or missing data. Each of these units is called a perceptron or neuron and has an incoming weight, bias and an output given by the transfer function of the sum of the inputs [11]. The function of the output neuron can be mathematically expressed as:

$$u(x, w) = \sum_{i=1}^{n} f(w_i x_i + b) \tag{1}$$

where $u(x, w)$ is the output of the neuron, w_i is the synaptic weights, x_i is the input data and b is the bias value. An ANN is generally organized in three layers: Input, hidden and output layer, as show in Fig. 1. The ANN training can be divided in two phases: the first steps consists of updating the neuron activation values with a chosen learning algorithm, the second phase updates weights to minimize the function error measuring the difference between the desired and actual output [12]. Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks. Mathematically are expressed as:

$$\theta_{t+1} = \theta_t - \eta * \nabla_\theta * J(\theta) \tag{2}$$

where θ is the weights, η the learning rate and ∇ the gradients of the cost function J. We then update our parameters in the direction of the gradients with the learning rate determining how big of an update we perform. Batch gradient descent is guaranteed to converge to the global minimum for convex error surfaces and to a local minimum for non-convex surfaces [14].

2.2 Deep Artificial Neural Network

The basic architecture of a multilayer perceptron (showed in Fig. 1) consist of one hidden layer fully connected to other neurons, but a Deep Neural Network has two or more interconnected hidden layers. In the past, many researchers failed to train neural networks with multiple layers, networks were often trapped in local minimal when initialized with random weights [9]. To overpass vanishing and over fitting, some optimization algorithms and techniques were applied. The dropout prevents the over fitting and the Adam algorithm reduces the vanishing problem, improving accuracy and performance. The Adam algorithm computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients; the name Adam is derived from Adaptive Moment estimation [15]. Adam stores an exponentially decaying average of past squared gradients v(t) and an exponentially decaying average of past gradients m(t), both similar as a momentum:

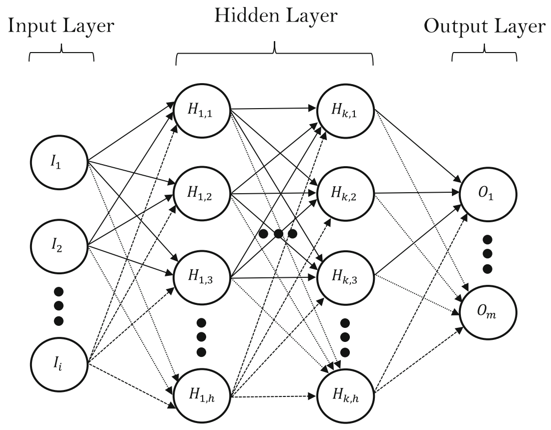


Fig. 1. ANN architecture.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{4}$$

and then estimates the first and second momentum with the decay rates β_1 and β_2:

$$\hat{m_t} = m_t / 1 - \beta_1^t \tag{5}$$

$$\hat{v_t} = v_t / 1 - \beta_2^t \tag{6}$$

Finally, parameters are updated with Adam rule:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v_t}} + \epsilon} \hat{m_t} \tag{7}$$

2.3 Recurrent Deep Neural Network

The Long Short-Term Memory networks (LSTM) are based on Recurrent Neural Networks (RNN). Unlike "normal" MLP, RNN use temporal information of the input data. An RNN realizes this ability by recurrent connections between the neurons. A LSTM has a special neuron structure called memory cell. These memory cells have the ability to store information over an arbitrary time. Three gates control the information flow in and out of the neuron's memory cell: the input, output, and forget gate. Each gate in the LSTM gets the same input as the input neuron. Furthermore, each gate possesses an activation function. For example, if the input gate notes (Gensler, Henze and Sick 2016).

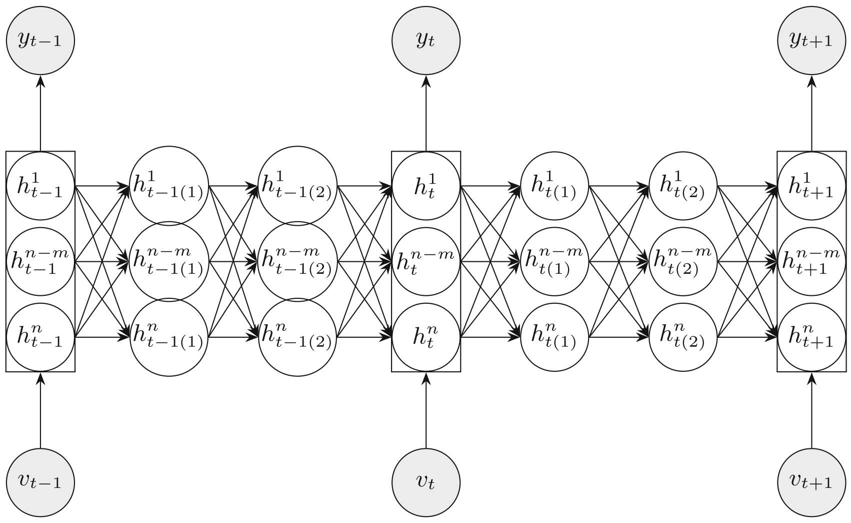


Fig. 2. RNN

The topology for a R-DNN with LSTM showed in Fig. 2 is mathematically expressed as:

$$Q(t) = f_1(X(t) * W_i + Q(t-1) * (W_{td}) \tag{8}$$

$$Y(t) = f_2(Q(t) * (W_1) \tag{9}$$

where $X(t)$ is input at time t, $Q(t)$ is output, $Q(t-1)$ is input at time t, W_i is weight for input layer, W_{td} represents weight for time delay input and W_1 is the weight for output layer, f_1 and f_2 are hidden and output layer transfer functions respectively.

3 Experimental Evaluation

This section explains the experimental evaluation of the forecasting methods described in Sect. 2. First, we describe the data set and plot some figures. After, we explain the experimental setup used by the models to forecast. Finally, in the section of analysis we present graphs and tables with results of the trained models.

3.1 Data Set Description

The power output data set was collected from a Power Plant on the north of Mexico. The data set recorded the power output from January 1 to December 31 of year 2015. The data was recollected every half hour, resulting in 17,520 data points. Due to the influence of chaotic elements, the Fig. 3 shows an irregular

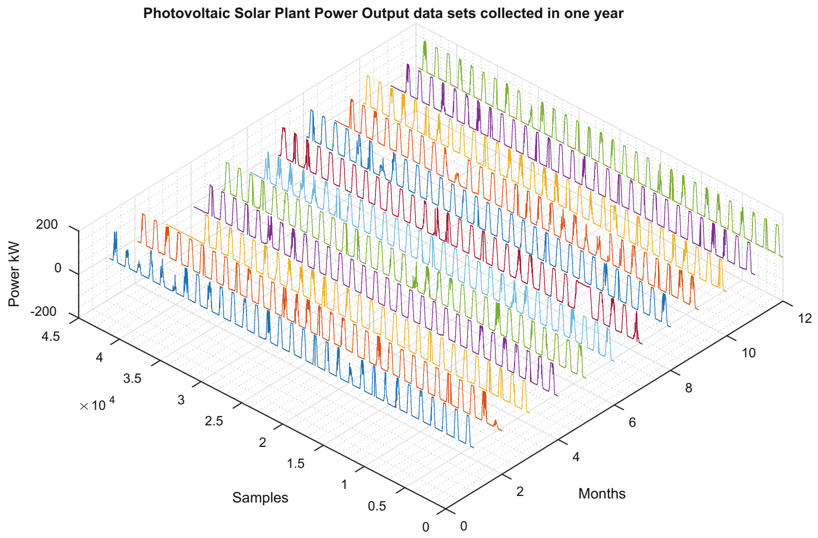


Fig. 3. Data set divided by months for a full view.

behaviour of the solar plant output through the year. Empirical observations about the power output concludes that it is highly complex and statistical non linear. All time series in the data set are normalized, except the measured power output. In order to take data sets for training, testing and validating, the data set is divided in 70% for training and 15% for testing and validating purposes.

3.2 Statistical Criteria

For training, testing and validation of the comparison process for ANN, DNN and R-DNN models, a statistical analysis is performed and applied using the following statistical test parameters: Correlation Coefficient (R), Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE), see Eqs. (10, 11 and 12). R provides information on the linear relationship between the measured and estimated values. RMSE parameter is a frequently-used measure of the differences between values predicted by a model and the actual values observed. MAPE parameter is the absolute computed average of errors (%) by which estimated predictions of a variable differ from their actual values. The knowledge of this statistical parameter aids to evaluate whether the estimated predictions are underestimated or overestimated with respect to actual or expected data [13].

$$R = \frac{\sum_{i=1}^{N}(P_{real} - \overline{P_{real}})(P_{sim} - \overline{P_{sim}})}{\sqrt{\sum_{i=1}^{N}(P_{real} - \overline{P_{real}})^2 (P_{sim} - \overline{P_{sim}})^2}} \tag{10}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(P_{real} - P_{sim})^2} \tag{11}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}|\frac{P_{real} - P_{sim}}{P_{real}}| \tag{12}$$

3.3 Experimental Setup

Three Solar Power forecasting models will be trained in this study: Artificial Neuron Network (ANN), Deep Artificial Neural Network (DNN) and Recurrent Deep Neural Network (R-DNN) with Long Short Term Memory (LSTM) architecture. Each of these models will be trained and tested using the Solar Power database.

The data set is split into 70% samples for training, 15% for testing and 15% of validation. The error is calculated after each training iteration with R, $RMSE$ and $MAPE$ statistical criteria. All models had to predict the minute-ahead forecast horizon of 15 min. All models were implemented using Keras. Each model was trained 50 times in total with a different topology.

The methodology used in this study was showed in Fig. 4 and consist in the following steps:

1. Select the training, testing and validation data sets.
2. Define possible values for the number of neurons for each layer.
3. Change hidden layers depending of the chosen model.
4. Define the loss function.

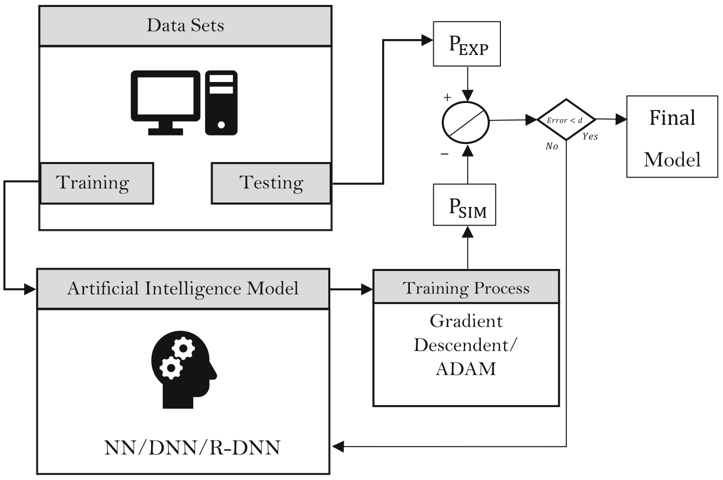


Fig. 4. Schematic for the experimental setup.

5. Determine the algorithm for training.
6. Train the model with 5 epochs.
7. Use the test data set to compare performance with statistical criteria.
8. Return to point 2 until a better performance is reached, else stop iterations.

4 Results and Discussions

Different neural network model architectures were developed and tested for the best performance. For the evaluation of the models, RMSE, MAPE and R^2 have been calculated for each training, testing and validation process. In Table 1, we summarize the results for various ANN architectures trained with Gradient Descent algorithm. The number of neurons in hidden layer was increased to up to five neurons in order to archive a better performance. The training process of the ANN shows that four neurons was enough to stop the training process. The best architecture was obtained at 4 neurons with smaller $RMSE = 3.46$, $MAPE = 0.09$ and higher $R^2 = 1$.

Table 1. Statistical criteria for the training and testing process with ANN.

Hidden layers	No. neurons	RMSE		MAPE		R^2	
		Training	Testing	Training	Testing	Training	Testing
1	1	4.01	5.45	0.09	0.09	0.99	0.97
1	2	4.01	5.45	0.09	0.09	0.99	0.98
1	3	3.55	4.83	0.07	0.07	0.99	0.98
1	**4**	**2.58**	**3.46**	**0.08**	**0.09**	**0.99**	**1.00**
1	5	2.59	3.46	0.08	0.09	0.99	0.98

Table 2 shows how the performance with a deep architecture trained with Adam algorithm changes with the depth and same number of hidden neurons as the ANN trained. It is possible to see that increasing the number of hidden layers does not improve the overall performance. The best results was reached with a $RMSE = 3.27$, $MAPE = 0.12$ and $R^2 = 0.99$. Its observed that the

Table 2. Statistical criteria for the training and testing process with DNN.

Hidden layers	No. neurons	RMSE		MAPE		R^2	
		Training	Testing	Training	Testing	Training	Testing
1	4	2.57	3.46	0.11	0.11	1.00	0.99
2	**4**	**2.46**	**3.27**	**0.12**	**0.12**	**1.00**	**0.99**
3	4	2.48	3.30	0.31	0.33	1.00	0.99
4	4	4.71	6.40	0.14	0.15	0.99	0.98

deep architecture trained with Adam algorithm performs better than the ANN trained with gradient descent for the same number of hidden layers and number of hidden neurons.

In order to observe the performance of a R-DNN with the same depth as the DNN architecture, Table 3 summarizes the comparison for the validation data sets of the three models. For this case study, having more layers or more memory cells does not reduce error parameters, and we see a over fitting problem increasing the depth. The best result was reached with the shallow neural network trained with Adam algorithm with two hidden layer.

Table 3. Statistical criteria for ANN, DNN and R-DNN validation.

IA Architecture	RMSE	MAPE	R^2
ANN	2.59	0.15	0.99
DNN	2.46	0.12	0.99
R-DNN	3.58	0.20	0.98

To illustrate the behavior of the estimated Power of the DNN with 2 hidden layers with 4 neurons for each layer in comparison with the measured data, Fig. 5 presents samples of the validation data set and DNN predictions. It is observed that the models follow the periodic behaviour of the samples for validation.

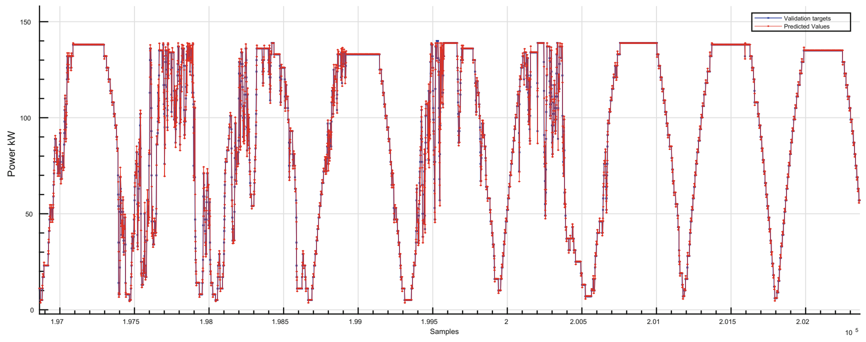


Fig. 5. Validation data sets and predicted DNN data.

The best linear regression equation for the DNN model was given by $Y = 0.99 * x + 0.31$ as shown in Fig. 6.

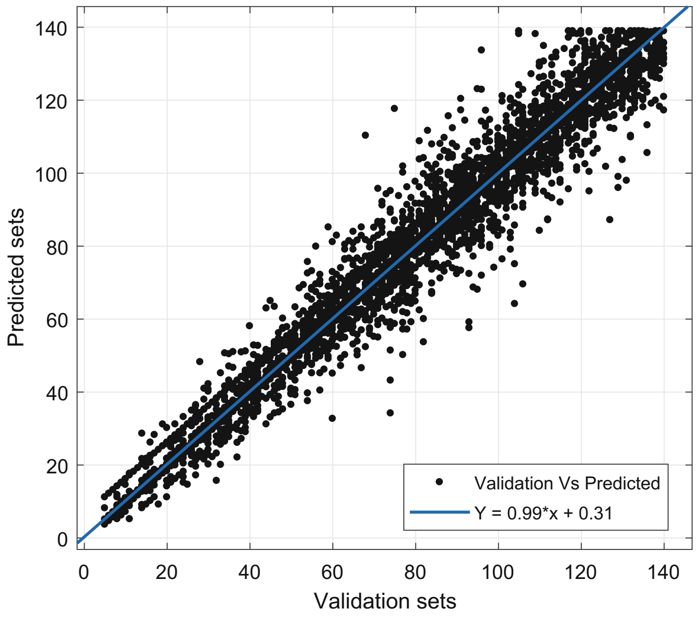


Fig. 6. Regression for DNN.

5 Conclusion

Application and comparison of ANN, DNN and R-DNN for Power Plant forecasting were investigated. The results in Table 3 showed that the DNN models have a lower $RMSE$, $MAPE$ and higher R^2 than the other models. The best performing model is the Deep Neural Network with two hidden layers with four neurons for each layer. Comparing the performance of both Gradient Descent and Adam algorithms, the last one had better performance in this case study.

Acknowledgements. This work was carried out thanks to the support of CONACYT on project CONACYT-SENER-254667 and the project FOMIX/YUC/2017/073.

References

1. Yadav, H.K., Pal, Y., Tripathi, M.M.: Photovoltaic power forecasting methods in smart power grid. In: 2015 Annual IEEE India Conference (INDICON) (2015)
2. IEA International Energy Agency: Snapshot of Global Photovoltaic Markets (2016)
3. Qiu, X., Zhang, L., Ren, Y., Suganthan, P.N., Amaratung, G.: Ensemble deep learning for regression and time series forecasting. In: IEEE SSCI 2014–2014 IEEE Symposium Series on Computational Intelligence (2014)
4. Wan, C., Zhao, J., Song, Y., Zhao, X., Lin, J., Zechun, H.: Photovoltaic and solar power forecasting for smart grid energy management. CSEE J. Power Energy Syst. **1**, 38–46 (2015)

5. Chu, Y., Urquhart, B., Gohari, S.M.I., Pedro, H.T.C., Kleissl, J., Coimbra, C.F.M.: Short-term reforecasting of power output from a 48 MWe solar PV plant. Sol. Energy **112**, 68–77 (2015)
6. Pedro, H.T.C., Coimbra, C.F.M.: Assessment of forecasting techniques for solar power production with no exogenous inputs. Sol. Energy **86**, 2017–2028 (2012)
7. Antonanzas, J., Osorio, N., Escobar, R., Urraca, R., Martinez-de-Pison, F.J., Antonanzas-Torresa, F.: Review of photovoltaic power forecasting. Sol. Energy **136**, 78–111 (2016)
8. Din, G.M.U., Marnerides, A.K.: Short term power load forecasting using Deep Neural Networks. In: 2017 International Conference on Computing, Networking and Communications (ICNC), pp. 594–598 (2017)
9. Sergio, A.T., Ludermir, T.B.: Deep learning for wind speed forecasting in Northeastern Region of Brazil. In: 2015 Brazilian Conference on Intelligent Systems (BRACIS), pp. 322–327 (2015)
10. Gensler, A., Henze, J., Sick, B., Raabe, N.: Deep Learning for solar power forecasting - an approach using AutoEncoder and LSTM neural networks. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 2858–2865 (2016)
11. Dzib, J.T., Moo, E.J.A., Bassam, A., Flota-Bañuelos, M., Soberanis, M.A.E., Ricalde, L.J., López-Sánchez, M.J.: Photovoltaic module temperature estimation: a comparison between artificial neural networks and adaptive neuro fuzzy inference systems models. In: Martin-Gonzalez, A., Uc-Cetina, V. (eds.) ISICS 2016. CCIS, vol. 597, pp. 46–60. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30447-2_4
12. Malluhi, Q.M., Bayoumi, M.A., Rao, T.R.N.: An application-specific array architecture for feedforward with backpropagation ANNs. In: International Conference on Application-Specific Array Processors, Proceedings, Venice, vol. 86, pp. 333–344 (1993)
13. Bassam, A., Álvarez del Castillo, A., García-Valladares, O., Santoyo, E.: Determination of pressure drops in flowing geothermal wells by using artificial neural networks and wellbore simulation tools. Appl. Therm. Eng. **75**, 1217–1228 (2015)
14. Ruder, S.: An overview of gradient descent optimization algorithms. arXiv:1609.04747 (cs.LG) (2017)
15. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv:1412.6980 (cs.LG) (2014)

Identification of Multimodal Signals for Emotion Recognition in the Context of Human-Robot Interaction

Andrea K. Pérez[1], Carlos A. Quintero[1](✉), Saith Rodríguez[1], Eyberth Rojas[1], Oswaldo Peña[1], and Fernando De La Rosa[2]

[1] Universidad Santo Tomás, Bogotá D.C., Colombia
{andrea.perez,carlosquinterop,saithrodriguez, eyberthrojas,oswaldopena}@usantotomas.edu.co
[2] Universidad de los Andes, Bogotá D.C., Colombia
fde@uniandes.edu.co

Abstract. This paper presents a proposal for the identification of multimodal signals for recognizing 4 human emotions in the context of human-robot interaction, specifically, the following emotions: happiness, anger, surprise and neutrality. We propose to implement a multiclass classifier that is based on two unimodal classifiers: one to process the input data from a video signal and another one that uses audio. On one hand, for detecting the human emotions using video data we have propose a multiclass image classifier based on a convolutional neural network that achieved 86.4% of generalization accuracy for individual frames and 100% when used to detect emotions in a video stream. On the other hand, for the emotion detection using audio data we have proposed a multiclass classifier based on several one-class classifiers, one for each emotion, achieving a generalization accuracy of 69.7%. The complete system shows a generalization error of 0% and is tested with several real users in an sales-robot application.

1 Introduction

In the last decades, multiple research groups have focused in developing the technology for constructing humanoid robots that interact actively in different aspects of daily life activities, which would generate a high social and industrial impact. One renowned initiative is RoboCup [1], which was established at the end of the last decade and where universities from 45 different countries participate every year. Such initiative has different research areas, one of them is human-robot interaction. Robocup, together with other well-known initiatives such as [2–4], show the relevance given among the academia and scientific community to achieve robots that can cooperate with humans in daily-life activities, in a medium-term period. However, such challenge implies the interaction of multiple disciplines and integration among all the different technical advances. Then, the common goal is to develop intelligent systems robust enough so that

C. Brito-Loeza and A. Espinosa-Romero (Eds.): ISICS 2018, CCIS 820, pp. 67–80, 2018.
https://doi.org/10.1007/978-3-319-76261-6_6

anyone can interact with robots in a natural manner, i.e., the robot should be able to understand different types of human communication.

Therefore, this work focuses in the problem of human emotion detection by a robot. Specifically, in the context of social interaction such as a sales agent robot promoting a product. In such environment, the robot starts the conversation presenting the product and consulting the interest of the potential client (user) through an emotion recognition system that is executed remotely while the person answers to a series of questions. Depending on the detected emotional state of the user a command is sent to the robot via Ethernet, which alters the responses according to the user's emotional response.

The emotion recognition system identifies four emotions: happiness, anger, surprise and neutral; which represent the possible emotional state of a user in front of a sales agent. The proposed system implements a multimodal classifier composed by the combination of two unimodal classifiers, one per type of input data: facial expressions and voice.

The rest of the article is organized as follows. The next section presents a summary of works related to emotion recognition systems. Section 4 contextualizes the proposed methodology under an application case of study. In Sect. 4 the proposed system's overall architecture in the human-robot interaction context is explained in detail. Section 5 details the selection process for the architecture and unimodal classifier parameters. In Sect. 6 the obtained results with the audio and video classifier are shown and analyzed; subsequently, the results obtained of the multimodal classifier are shown. Finally, in Sect. 7 we present the conclusions of the preceding results.

2 Related Work

In recent years, human-robot interaction (HRI) has been highly studied among researchers in artificial intelligence and computer science. In [5], the authors present a recompilation of the main problems to solve in HRI, the fundamental problems and a discussion regarding the short-term challenges that need to be addressed. This article first describes the history context of human-robot interaction, beginning in late 90's and early 2000's where multiple disciplines such as robotics and psychology noticed the importance to work together. Subsequently, the HRI problem is described as the interaction between one or more humans with one or more robots; understanding the interaction as an intrinsic part of robotics, mainly because robots are built to assist humans in their tasks.

Additionally, Goodrich [5] defines five features to consider while designing solutions in HRI context: autonomy level, type of information exchange, equipment structure, adaptability, learning and training for humans and robots, and the task. Finally, a series of challenges are described for HRI, one of them is assistive robotics, which seeks to provide physical, mental, or social support to persons in such need. This challenge includes special attention in features intrinsic in human communications, such as speech recognition, multimodal interaction and cognitive analysis.

On the other hand, the researching relevance of HRI is reflected in the @Home league of RoboCup world initiative [6]. The main aim of such league is to develop technology cooperatively among the participant universities to create a robot that is able to do the domestic chores and have a natural interaction with humans. To achieve such objective, research needs to focus in navigation and mapping in dynamic environments, artificial vision and object recognition in natural lighting environments, object manipulation, and human-robot interaction and cooperation.

One approximation to the HRI problem is from the machine learning perspective. From the aforementioned works, it can be concluded that it is paramount to develop intelligent systems that allow a natural interaction between the human and the robot. According to this, [7] presents a framework for learning desirable tasks in HRI, based on the illustration of the task to the robot and its decomposition in sequences of spatial points. Such approach is called KLfD and the proposed model consists of an organized set of groups of spatial points. Another approach is presented in [8], which focuses on the recognition of human hand gestures in the HRI context. The presented methodology consists of combining an algorithm to recognize hand gestures with two classifiers, one built for hand skeleton recognition (HSR) and the other is based on a SVM.

As previously mentioned, in HRI different approaches are considered in which several challenges are tackled, this is exemplified in the aforementioned articles. Furthermore, in [9] an emotion multimodal recognition system is proposed for human-robot interaction. This work is developed under a RDS (Robotics Dialog System) and implements two modules for emotion recognitions, one for the voice (GEVA) and one for the facial expression (GEGA). These modules are based on a fast recognition algorithm for objects and a toolbox for expressions recognitions (CERT).

In this area, other approaches are found such as the one in [10], in which Machine Learning techniques are used for recognition of integrated audio-video signals, where supervised learning is usually used. However, recently, recommendation systems are being used in robotics, specifically in human-robot interaction.

Several works have been developed either in facial gestures recognition or emotion recognition. However, in [11] an integrated approach is presented, where facial and emotion recognition is implemented simultaneously. This approach, used Soft computing, specifically Fuzzy rule based system (FBS), to reduce road traffic accidents due to driver somnolence. In order to do so, this work proposes to include the facial gesture and recognized emotion of the driver, so that if any fatigue signal is detected in the driver, the vehicle switches to autopilot mode. The solution architecture consists of a camera for image acquisition, face recognition analysing images in RGB, then feature extraction for eye movement and lips. This is the input for the emotion and gesture recognition systems. The former systems where developed using Fuzzy inference, which allows to distinguish between 4 emotional states (happy, angry, sad and surprised). As a result, a precision of 91.66% is obtained for facial gestures recognition and 94.85% for the combination of facial gestures and emotion recognition.

3 Social Robot: Application Case Study

Figure 1 shows the general scheme for our case study to validate the proposed emotion recognition system. The scenario consists of a social robot that acts as a sales agent. The idea is that the robot will attempt to sale a specific product to the human and during their interaction, it will constantly detect the user's emotional state, using her facial gestures and voice, to adapt to the user's response. Similar to traditional *call centers*, the system will use the user emotions, detected through her answers during the interaction, to guide the sales process (i.e., to choose the verbal script to persuade the user to perform the purchase).

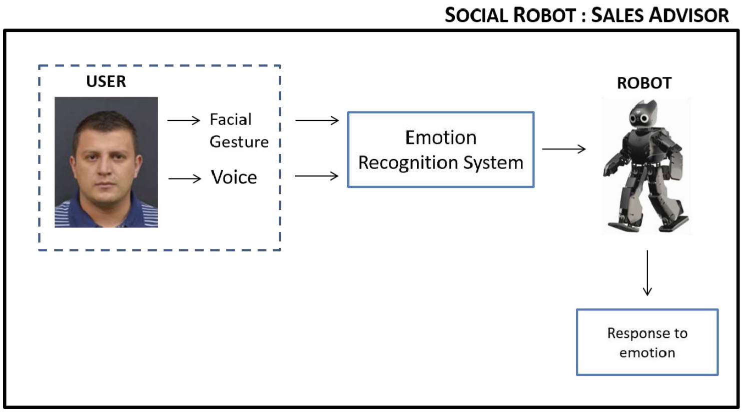


Fig. 1. General scheme of the proposed test case scenario for the detection of human emotions: robot sales advisor

4 Automatic Multimodal Emotion Recognition System Design

Our emotion recognition system is based on an automatic classifier that combines two different and independent classifiers, one for each input type, namely voice and facial gesture. The idea is that the classification system will detect the user's emotions through their interaction using data that comes from a video stream and also data collected with a microphone.

In Fig. 2 we show a diagram of the overall emotion recognition system. Initially, the input is the data acquired by the robot while it interacts with the human. This input is multimodal since it comes from different sources an hence there is one classifier per source. In the end, the outputs of both individual classifier is combined to identify one unique emotion. To build the emotion recognition model, we have performed experiments to collect data of different users interacting with the robot and such data have been split into two disjoint sets: the training set and the testing set. The former will be used to create the classification module while the latter will be used to evaluate the model's accuracy. All the modules of the overall architecture will be explained in detail in the following sections.

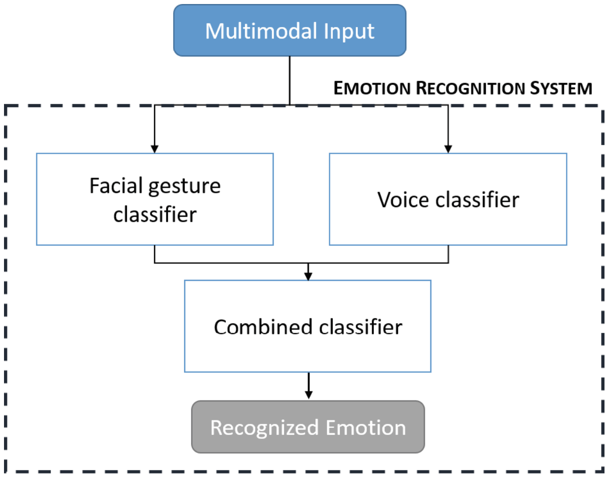


Fig. 2. Diagram of the proposed automatic emotion recognition system

4.1 Facial Gesture Classifier

We have selected a convolutional neural network (CNN) to implement a classifier capable of automatically recognize the user emotion based on a video data stream that captures the user facial gestures. In artificial vision systems particularly, deep learning and more specifically the CNNs have been successfully used, especially for their ability to automatically extract valuable features of input images for classification tasks [12,13].

Figure 3 illustrates the general scheme of the facial expression classifier. The video stream is considered as a set of independent image frames that are fed into the convolutional neural network, one at a time to be classified into one of the possible emotions.

Each frame is converted from its original RGB space to greyscale since the key features used to recognize the user's facial expressions are mostly related to local relationships such as borders and corners.

The CNN will output one detected emotion for each input frame of the video stream. All the detected emotions need to be merged together to define the whole video stream emotion using a voting procedure, which simply chooses the emotion decided by the majority. We define the video stream confidence q_{cv} found by the classifier as shown in Eq. (1), where FD is the number of frames labeled with the selected emotion and FV is the total number of frames of the video stream.

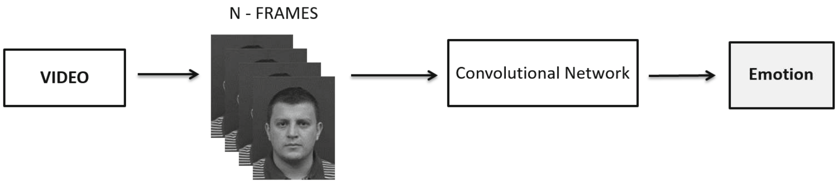


Fig. 3. General scheme of the facial expression classifier

$$q_{cv} = \frac{FD}{FV} \tag{1}$$

4.2 Voice Classifier

For the emotion recognition classifier that uses voice data we will use a similar approach to the one shown in [14] where One-class SVMs are used to discriminate one class each and then merged into one single output to create a multiclass classifier. Figure 4 shows the general voice-based classifier where a set of features need to be extracted from the audio source to feed the classifier. This feature extraction process, for the application at hand, is performed by computing the signal's Mel-frequency cepstral coefficients (MFCC): a set of coefficients that contain frequency relations of small windows that allows us to represent the important voice features from the user.

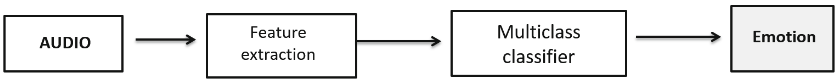


Fig. 4. Block diagram for the detection of emotions using audio data

The architecture for the multiclass classifier is shown in Fig. 5. An ensemble of one class classifiers process the features extracted from the original voice data and determine whether the given sample belongs to their class or not. Each SVM also outputs the classification margin for the current data sample, which provides an estimation of the classification confidence of the classifier for such data. In order to compare the classification margins output by each one-class classifier, we normalized the given margin $m(x)$ by the maximum margin obtained by each classifier during its training process $\hat{m}$. Therefore, the confidence value $q_{ca}(x)$ of each classifier when presented the input data x is as shown in Eq. (2).

$$q_{ca}(x) = \frac{m(x)}{\hat{m}} \tag{2}$$

The final output function for the multiclass classifier will select the class of the one-class SVM with highest confidence value.

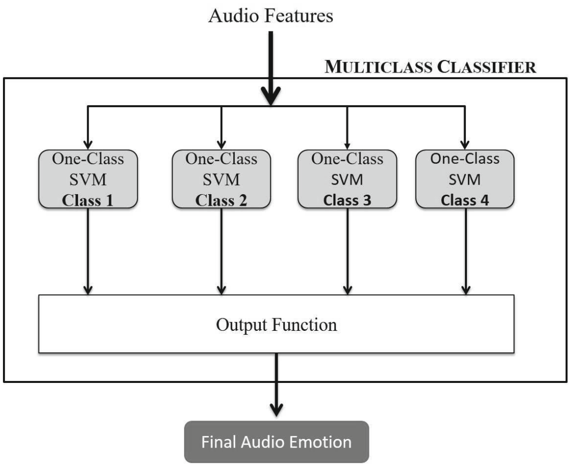


Fig. 5. General architecture of the emotion classifier using the voice commands

4.3 Combined Classifier

As described in previous sections, we have built a specific classifier for the video data and another for the voice data, each one, trained to automatically detect the possible user emotions during the interaction. Furthermore, each classifier also computes the classification confidence of a data sample and is characterized by the model's accuracy. Using this information we will assign the final label according to the following conditions:

– If both classifiers agree on the detected emotion, the general system's output will be the emotion detected by both classifiers. In this case, we calculate the agreement between both classifiers taken into account the confidence value for each classifier and their accuracies as shown in Eq. (3)

$$Agreement_{(cv,ca)} = \frac{Acc_{cv}q_{cv} + Acc_{ca}q_{ca}}{Acc_{cv} + Acc_{ca}} \tag{3}$$

where Acc_{cv} and Acc_{ca} are the classifier accuracies of the video and audio classifier respectively and q_{cv} and q_{ca} are the confidence values output by the video and audio classifiers respectively. The confidence values are assumed to be normalized between 0 and 1.

– If both classifiers output a different emotion, we must take into account the classification confidence of each classifier and also the model's accuracy. Therefore, the final emotion will be the one given by the classifier with highest weighted confidence value as shown in Eq. (4).

$$\max(Acc_{cv}q_{cv}, Acc_{ca}q_{ca}) \tag{4}$$

5 Experimental Selection and Tuning of Unimodal Classifiers

We have designed an experiment that allows us to test our proposed methodology. For this we have asked 11 users to perform 10 different repetitions, using voice and facial gestures, of the four emotions of interest. These signals are captured using a Kinect sensor finally obtaining a video with the facial gesture and an audio stream with the corresponding voice.

Using this method we have obtained a total of 110 videos and 110 audio samples. Each video is made of 640 × 480 pixel frames for a total of 30.994 images, for all emotions and users. Figure 6 shows one example of the collected image data for each emotion.

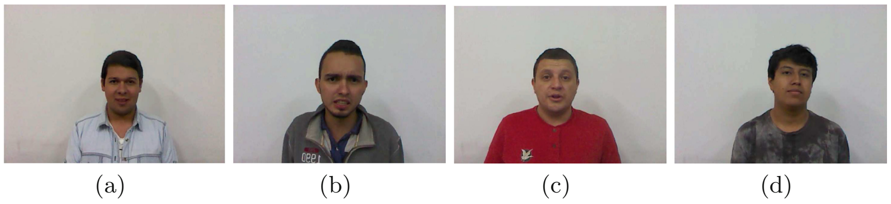


Fig. 6. Example of images captured per emotion: (a) Happiness (b) Anger (c) Surprise (d) Neutral

The collected data is split into training and testing set, leaving 77 video and audio data for training and 33 for testing. However, each video has a different number of frames. Table 1 shows the exact number of data reserved for training and testing in each emotion and for each classifier:

Table 1. Data distribution for training and testing per emotion

	Number of samples	Happiness	Anger	Surprise	Neutral
Training	Image	6465	5231	5040	5215
	Audio	77	77	77	77
Validation	Image	2415	2179	2244	2205
	Audio	33	33	33	33

5.1 Facial Expressions Classifier

The selection of the convolutional neural network architecture is achieved through an experimental approach by the variation of the number of receptive fields and their sizes. Initially, we have adopted the general network architecture shown in [15], where a similar problem is solved using CNNs, and implemented it using the Matlab toolbox **MatConvNet**, which contains many pre-trained CNNs for image classification and other tasks [16].

The first step consisted on shrinking the images size on the input layer. This decision was taken based on the fact that the captured images contained large background regions. The images were cropped and resized to 44 × 44 for the first two experiments and 43 × 43 for the last two scenarios. The general network contains 9 layers as shown in Fig. 7, whose area and number of receptive fields (RF) were modified for each experimental scenario (Test 1, Test 2, Test 3 and Test 4) as shown in Table 2.

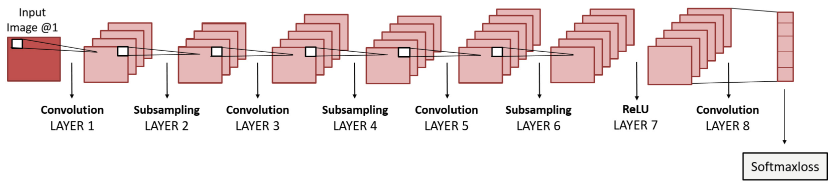


Fig. 7. Architecture of the convolutional neural network for emotion recognition in images

Table 2. Architecture parameters in the convolutional neural network for each experimental scenario

	Test 1		Test 2		Test 3		Test 4	
	Area RF	Number of RF	Area RF	Number of RF	Area RF	Number of RF	Area RF	Number of RF
Layer 1	2×2	50	3×3	50	4×4	50	5×5	50
Layer 2	2×2	50	2×2	50	2×2	50	2×2	50
Layer 3	2×2	100	2×2	100	3×3	100	3×3	100
Layer 4	2×2	100	2×2	100	2×2	100	2×2	100
Layer 5	5×5	200	3×3	200	2×2	300	2×2	200
Layer 6	2×2	200	2×2	200	2×2	300	2×2	200
Layer 8	3×3	4	4×4	4	4×4	4	4×4	4

The convolutional neural network for each experimental setup is trained and the training parameters and main results are shown in Table 3.

According to these results, the selected CNN architecture is the one of Test 3 since it attained the highest accuracy over the testing dataset $Acc_{cv} = 86.4\%$.

Table 3. Training parameters and classification accuracy for the different CNN architectures

CNN architecture	Training epochs	Training time	Accuracy (Acc_{cv})
Test 1	7000	49.6 h	60.26%
Test 2	10000	78.47 h	81.97%
Test 3	**1500**	**11.77 h**	**86.40%**
Test 4	1500	12.60 h	84.50%

5.2 Voice Classifier

The initial stage of the voice classifier consists on the feature extraction phase, where the MFCCs must be computed. To this end, we have used the C++ library Aquila DSP [17]. For each audio input we computed the first 13 coefficients since it has been shown that they represent good enough the signal features [14]. The audio data is split as shown in Table 4, leaving 70% for training and 30% for testing, each sample with a dimensionality of 13.

Table 4. Audio data distribution for each emotion

	Number of samples	Happiness	Anger	Surprise	Neutral
Dataset	Training	77	77	77	77
	Validation	33	33	33	33
	Total	110	110	110	110

Each One-class SVM of the multiclass audio classifier is trained for each class using a model selection methodology where a grid is built with the values of the generalization and kernel parameters and in each combination of the parameters, a model is trained and tested using both datasets and the chosen model is the one that presents higher generalization accuracy. To this end, we have used the open source C++ package libSVM, a widely and efficient SVM implementation [18].

Table 5. Training parameters and results for each voice classifier

Classifier	Selected kernel		Accuracy
	Sigmoid		
	γ	ν	
Anger	0.0082	0.581	**75%**
Happiness	0.02025	0.2147	**68.75%**
Surprise	0.0000189	0.6291	**81.25%**
Neutral	0.0000125	0.7181	**81.25%**

After performing the grid search technique using the three more commonly used kernel functions, polynomial, gaussian and sigmoid, the chosen kernel was the sigmoid. The results for the three One-class SVMs are shown in Table 5.

6 Results

In this section we show the most important results of our emotion recognition system. Initially, we show the performance of each separate classifier and finally the performance of the complete system. The accuracy of the voting procedure to detect the emotion from the video stream on the testing data was 100% in all cases, meaning that the classifier was capable of detecting all the emotions without mistake. For the voice classifier the 4 One-class SVM outputs are merged into one single multiclass output as described in Sect. 4.2 and the resulting multiclass voice classifier attained a 69.7% of accuracy in the testing set.

Table 6 shows the labels that were assigned by both, the convolutional neural network and the voice classifier during the testing process. The rows correspond to the real class and each column contains the number of data points that were classified as the emotion shown in each column. Recall that the total number of samples for each classifier is shown in Table 1.

Table 6. Labels assigned by each unimodal classifier for each emotion

	Recognized emotion							
	Happiness		Anger		Surprise		Neutral	
	Image	Voice	Image	Voice	Image	Voice	Image	Voice
Happiness	1987	22	104	6	307	2	17	3
Anger	110	4	1936	20	101	2	32	7
Surprise	205	2	45	3	1951	24	43	4
Neutral	149	2	37	4	80	1	1939	26

In the final step, both classifiers are combined as described in Sect. 4.3 taking into account their accuracies and confidence values. The results show that all the input data in the testing set were correctly classified by the multimodal classifier, even though the unimodal classifiers do not agree for all the samples. Table 7 shows the amount of input samples where the unimodal classifiers agree and disagree.

When both classifiers agree on their output, we measure the level of *agreement* (normalized from 0 to 1) between the classifiers as shown in Eq. (3). Figure 8 shows the distribution of the *agreement* coefficient. This shows that the agreement between the classifiers is mostly greater than 0.6.

The other scenario considers that the classifiers do not agree on the emotion that should be assigned to the input data. In this case, the chosen output is the one given by the classifier with highest weighted confidence. Figure 9 shows the

Table 7. Number of testing data samples with agreement and disagreement in the recognized emotions between the two unimodal classifiers

	Amount of data	
	Agreement between the classifiers	Disagreement between the classifiers
Happiness	22	11
Anger	20	13
Surprise	24	9
Neutral	26	7

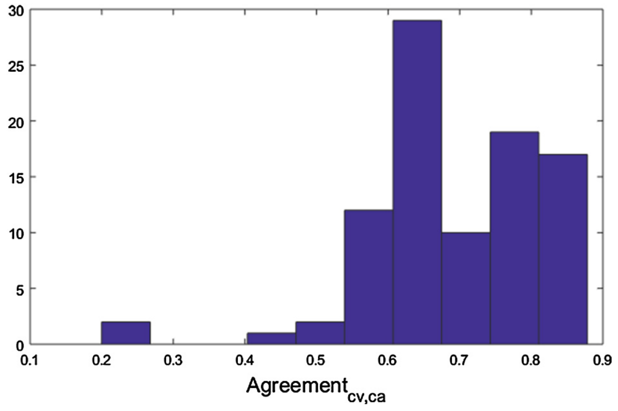


Fig. 8. Distribution of the *agreement* between the video classifier and voice classifier when both classifiers recognize the same emotion.

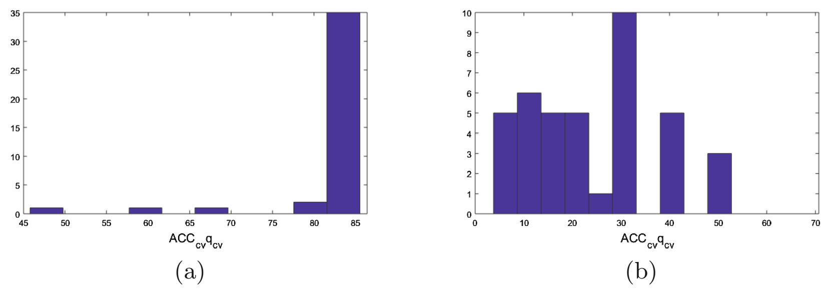


Fig. 9. (a) Distribution of the weighted confidence value for the video classifier in the testing set when the classifiers recognize different emotions (b) Distribution of the weighted confidence value for the audio classifier in the testing set when the classifiers recognize different emotions

distribution of the confidence values for both, the video (Fig. 9(a)) and audio (Fig. 9(b)). It is noteworthy that most of the confidence values are higher than 80 for the video classifier while the audio classifier shows confidence values lower than 30.

7 Conclusions

We have proposed an automatic classification model to recognize emotions in the context of human-robot interaction using multimodal inputs, i.e., audio and video. Specifically, the proposed case study is a social robot that acts as a sales agent and uses the information captured by the emotion recognition system to drive its speech during her interaction with the human. In this paper we show the behavior of the detection system.

The proposed architecture shows an accuracy of 100% when the classifier uses both data types (audio and video). On one hand, the video classifier is created using a convolutional neural network that is trained with images containing the facial gestures of the human. The individual outputs are merged using a voting procedure to finally decide the total video label. This voting procedure has shown to have a very high accuracy in data that were not used during the training process. On the other hand, the voice classifier uses an audio signal from the processed voice of the human during the interaction. The accuracy of this classifier is of 69.7% for the testing dataset showing that it is possible to use this information to accurately discriminate between the 4 human emotions. Finally, we have proposed a system that provides a confidence value for the classification task.

Currently, we are performing experiments in more open environments, such as with higher levels of noise for both data sources. Preliminary results show that the classifier is more robust to noise due to its use of the multimodal inputs. For future works, the presence of the two different classifiers could be used not only to detect the class, but complimentary classes that could be used by the robot to improve the interaction experience.

References

1. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., Matsubara, H.: Robocup: a challenge problem for AI. AI Mag. **18**(1), 73 (1997)
2. Christensen, H.I., Batzinger, T., Bekris, K., Bohringer, K., Bordogna, J., Bradski, G., Brock, O., Burnstein, J., Fuhlbrigge, T., Eastman, R., et al.: A roadmap for us robotics: from internet to robotics. Computing Community Consortium (2009)
3. Multi-Annual Roadmap. For horizon 2020. SPARC Robotics, eu-Robotics AISBL, Brussels, Belgium (2017)
4. Dhall, A., Ramana Murthy, O., Goecke, R., Joshi, J., Gedeon, T.: Video and image based emotion recognition challenges in the wild: Emotiw 2015. In: Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, pp. 423–426. ACM (2015)

5. Goodrich, M.A., Schultz, A.C.: Human-robot interaction: a survey. Found. Trends Hum. Comput. Interact. **1**(3), 203–275 (2007)
6. van Beek, L., Chen, K., Holz, D., Matamoros, M., Rascon, C., Rudinac, M., des Solar, J.R., Wachsmuth, S.: Robocup@ home 2015: Rule and regulations (2015)
7. Akgun, B., Cakmak, M., Jiang, K., Thomaz, A.L.: Keyframe-based learning from demonstration. Int. J. Soc. Robot. **4**(4), 343–355 (2012)
8. Luo, R.C., Wu, Y.C.: Hand gesture recognition for human-robot interaction for service robot. In: 2012 IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pp. 318–323. IEEE (2012)
9. Alonso-Martín, F., Malfaz, M., Sequeira, J., Gorostiza, J.F., Salichs, M.A.: A multimodal emotion detection system during human-robot interaction. Sensors **13**(11), 15549–15581 (2013)
10. Subashini, K., Palanivel, S., Ramalingam, V.: Audio-video based classification using SVM and AANN. Int. J. Comput. Appl. **53**(18), 43–49 (2012)
11. Agrawal, U., Giripunje, S., Bajaj, P.: Emotion and gesture recognition with soft computing tool for drivers assistance system in human centered transportation. In: 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 4612–4616. IEEE (2013)
12. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
13. Deng, L., Dong, Y.: Deep learning: methods and applications. Found. Trends Signal Process. **7**(3–4), 197–387 (2014)
14. Rodriguez, S., Pérez, K., Quintero, C., López, J., Rojas, E., Calderón, J.: Identification of multimodal human-robot interaction using combined kernels. In: Snášel, V., Abraham, A., Krömer, P., Pant, M., Muda, A.K. (eds.) Innovations in Bio-Inspired Computing and Applications. AISC, vol. 424, pp. 263–273. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-28031-8_23
15. Kahou, S.E., Bouthillier, X., Lamblin, P., Gulcehre, C., Michalski, V., Konda, K., Jean, S., Froumenty, P., Dauphin, Y., Boulanger-Lewandowski, N., et al.: Emonets: multimodal deep learning approaches for emotion recognition in video. J. Multimodal User Interfaces **10**(2), 99–111 (2016)
16. Vedaldi, A., Lenc, K.: Matconvnet – convolutional neural networks for MATLAB. In: Proceeding of the ACM International Conference on Multimedia (2015)
17. Django: Aquila digital signal processing C++ library (2014). https://aquila-dsp.org/
18. Libsvm – a library for support vector machines (2015). https://www.csie.ntu.edu.tw/~cjlin/libsvm/

Forecast and Energy Management of a Microgrid with Renewable Energy Sources Using Artificial Intelligence

E. Cruz May[1(✉)], L. J. Ricalde[1], E. J. R. Atoche[2], A. Bassam[1], and E. N. Sanchez[3]

[1] Faculty of Engineering, Autonomous University of Yucatan (UADY), Av. Industrias no Contaminantes por Periférico Norte, Apdo. Postal 150 Cordemex1, Mérida, Mexico
edcm91@hotmail.com, {lricalde,baali}@correo.uady.mx
[2] Instituto Tecnológico de Mérida, Av. Tecnológico km. 4.5 S/N, 971182 Mérida, Mexico
[3] Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional, 45019 Zapopan, Mexico

Abstract. This paper presents a design of an artificial neural network algorithm for prediction and management of electric loads for the optimal operation of a microgrid with sources of renewable energy. The hybrid power generation system is composed of a photovoltaic array, wind turbines, public power grid, electric loads and battery bank as a storage system. A dynamic neural network is implemented to determine the optimal amounts of energy that must be obtained from the sources, to reduce costs and improve efficiency. Simulation results demonstrate that generation of each energy source can be reached in an optimal form using the proposed design.

Keywords: Artificial neural networks · Microgrid · Power measurement Prediction · Renewable energy

1 Introduction

Nowadays, an energy crisis exists since reserves of coal-based fuels are being depleted, due to the increment of their exploitation and consumption. When generating energy with this type of fuel greenhouse gases are emitted, causing: global warming, the alteration of the climate, and the habitat [1]. The costs of production and storage of various low capacity renewable energy sources have decreased considerably, and currently, the world is working on integrating them into public power grids. The first steps in the integration of renewable energy sources came with the implementation of hybrid photovoltaic-wind systems as complementary sources for rural applications and weak connections to the grid.

Research is currently underway on the integration of various small-scale energy sources such as solar thermal, biomass, fuel cells and tidal power, under new and advanced control schemes constituting what is called a smart grid. Since the production costs of photovoltaic and wind farms have been significantly reduced, they have become the primary choice for power generation in smart grids [2, 3]

A smart grid is a kind of grid that can efficiently integrate the behavior and actions of all users connected to it. It ensures a sustainable and efficient energy system, with low losses and high levels of quality and security of supply. Humanity has experienced in the last century an immense development based on the energy of fossil origin consumption. These energies have been exploited assuming an unlimited availability, and without assessing at any time the environmental costs caused.

The human being has focused to date on an energy model in which a rigorous chain is followed in the following order: generation, distribution, transport, and consumption. However, changing this model is more than a necessity today, with the new model tending to the diversification of energy sources, greater use of renewable energies, efficiency, and energy saving. The new energy model aims to transform the current system into a distributed system, in which any agent that is connected to the network has the possibility of contributing energy, making possible the creation of microgenerators, so that there is no such direct dependence on the current energy generation [4–7].

Although there is no standard general definition of a smart grid. The European Smart Grids Platform (Smart Grids: European Technology Platform) defines a smart grid as "A power grid capable of intelligently integrating behavior and the actions of all the users connected to it, generators, consumers and those who carry out both actions, in order to safely and efficiently distribute electricity from a sustainable and economic point of view" [8].

2 Energy Management System

An energy management system (EMS) can be defined as a methodology to achieve sustained and continuous improvement of energy performance. The implementation of an EMS should not be understood as an objective itself, what really matters is the results of the whole system. Understood in this way, the effectiveness of an EMS will depend, to a large extent, on the commitment and willingness of all the factors involved to manage the use and cost of energy. The EMS aim at the continuous improvement in the use of energy through a more efficient use of energy, reducing consumption, associated financial costs and emissions of greenhouse gases, as well as by making better use of renewable energies [9–11].

An EMS brings the following benefits:

- Helps identify, prioritize and select actions to improve energy performance, based on their potential savings and the level of investment required.
- Reduce costs by making the most of the energy resources.

Boosts productivity and growth (greater use, less waste).

- Ensures the trust and quality of the information used for making decisions.
- It facilitates the integration of existing management systems.

2.1 Power Measurement Device

A power measurement device allows the measurement of current and voltage variables. This device replaces the use of voltmeters and amperemeter. It has the capacity to measure power incorporating additional benefits such as wireless communication through protocols ZigBee, with which we can integrate this equipment into the software to perform power management by implementing a monitoring system. An energy monitoring system can record electrical variables of interest that at some point provide information to establish the behavior of a power system. It is important to have these real-time data storage systems to monitor the operational performance.

The proposed measurement device has two voltage transducers and four current traducers, which convert the input power variables into other output variables of very small values. The resulting data is passed through six low pass filters, designed to condition the signal and finally direct them to the analog inputs of an "Arduino DUE". A Xbee module is connected to carry out communication with the computer and obtain the measurement data in real time.

The main components of the wind generation system are illustrated in Fig. 1. This system is part of the microgrid installed in the Faculty of Engineering of the UADY. The power from turbines passes through the control panel, where it is verified that it is within the allowed operating ranges, otherwise, the turbine is protected by shutting down the entire system. Subsequently, the power produced is directed to rectifiers to regulate resistive loads. The resistive loads come into operation when there is an excessive surplus of power. From the resulting power passing through the rectifiers, two currents and the same voltage are generated for each pair of rectifiers. At this stage of the generation process of the wind system is where the measurement of voltages and currents is taken for the proposed device.

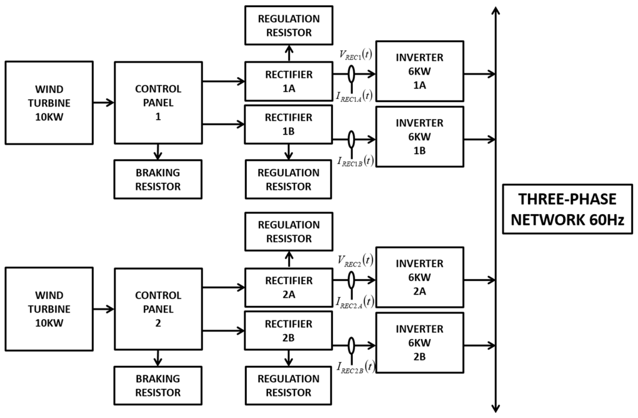


Fig. 1. Power measurement scheme in wind power systems

3 Artificial Neural Networks

An Artificial Neural Network (ANN) is a distributed parallel processor made up of simple processing units (neurons), which stores knowledge obtained experimentally, and makes it available for use. Knowledge is acquired by the network through a learning process and the connection weights between neurons, known as synaptic, are used to store it. The procedure used to perform the learning process is called training algorithm, whose function is to modify the synaptic weights of the network in an orderly way to achieve the desired design goal [12].

An ANN can be seen like a black box in which enters a database conformed by input variables. Each of the input variables is assigned with an appropriate weighting factor (W). The sum of the weighted inputs and the bias (b) produces the input for a transfer function which will generate an output value. The main characteristic of this model is that specific information about the physical behavior system or the way in which the data were obtained are not required [13, 14].

3.1 Model of an Artificial Neuron

The neuron is the fundamental information processing unit for the operation of a neural network. Figure 2 shows the model of a neuron, which forms the basis for the design of artificial neural networks [16, 17]. In the neuronal model presented four basic elements are identified:

- Connection links $\left(w_{kj}\right)$. Characterized by their own weight. Specifically, a signal x_j at the input of the synapse j connected to neuron k is multiplied by the synaptic weight w_{kj}. It is important to note the notation: the first subscript refers to the receiving neuron and the second refers to the input of the synapse to which the weight is concerned. If $w_{kj} > 0$ the connection is excitatory; Also, if $w_{kj} < 0$, the connection is inhibitory.
- Summing Junction (Σ). Add the input signals multiplied by w_{kj}. The operations described in this point constitute a linear combination.
- Activation function (φ). It limits the amplitude of the output of a neuron to a finite value. Usually, the normalized amplitude range of the output of a neuron is written as the closed unit interval [0,1] or alternatively [−1,1].
- Threshold b_k. It has the effect of increasing (positive value) or decreasing (negative value) the total input to the activation function.

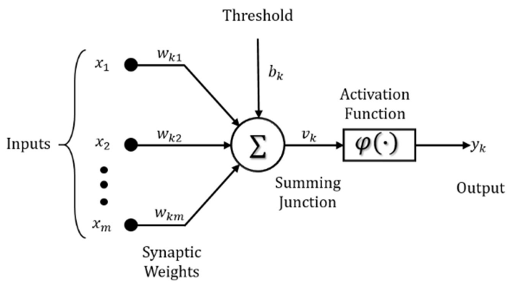


Fig. 2. Model of an artificial neuron

3.2 Extended Kalman Filter

The Kalman filter is based on the state space formulation of linear dynamic systems, providing a recursive solution to the optimal linear filtering problem. It is applied in stationary environments as non-stationary. The solution is recursive; so, each update of the estimated state is calculated from the previous estimate and the new input data, so only the previous estimate requires storage. In the application of the Kalman filter for training neural networks, the synaptic weights of the network are the states that the Kalman filter estimates and the output of the neural network is the measurement used by this filter [17, 18].

With the algorithm based on the Extended Kalman Filter (EKF), learning convergence is improved. EKF training of neural networks, both static and recurrent, has proven to be reliable for many applications in the last twenty years. However, this training requires the heuristic selection of some design parameters which is not always an easy task [19].

Consider a nonlinear dynamic system described by the following state space model

$$w(k+1) = f(k, w(k)) + u(k) \tag{1}$$

$$y(k) = h(k, w(k)) + v(k) \tag{2}$$

$u(k)$ and $v(k)$ are independent noises, empty, zero-mean and with covariance matrices $Q(k)$ and $R(k)$, respectively. $f(k, w(k))$ denotes the non-linear matrix function of transition, which can be variant in time and $h(k, w(k))$ denotes the nonlinear measurement matrix function, which can also be variant in time.

The idea of the extended Kalman filter is to linearize the state space model of (1) and (2) at each sampling instant around the most recent estimated state, which can be taken as $\hat{w}(k)$ o $\hat{w}^-(k)$. Once the model is obtained, the Kalman filter equations are applied. The approximation proceeds:

Step 1. The following matrices are computed

$$F_{k+1,k} = \left.\frac{\partial f(k, w(k))}{\partial w}\right|_{w=\hat{w}(k)} \tag{3}$$

$$H(k) = \left.\frac{\partial h(k, w(k))}{\partial w}\right|_{\hat{w}^-(k)} \tag{4}$$

Step 2. Once the matrices $F(\cdot)$ and $H(\cdot)$ are evaluated, they are used in a first-order Taylor series approximation for the nonlinear functions $f(k, w(k))$ and $h(k, w(k))$ around $\hat{w}(k)$ and $\hat{w}^-(k)$ respectively. Specifically, they approximate as follows

$$f(k, w(k)) \approx F(w, \hat{w}(k)) + F_{k+1,k}(w, \hat{w}(k)) \tag{5}$$

$$h(k, w(k)) \approx H(w, \hat{w}^-(k)) + H_{k+1,k}(w, \hat{w}(k)) \tag{6}$$

With (3) and (4), the nonlinear state Eqs. (5) and (6) are approximated as:

$$w(k+1) \approx F_{k+1,k} w(k) + u(k) + d(k) \tag{7}$$

$$\bar{y}(k) \approx H(k)w(k) + v(k) \tag{8}$$

where

$$\bar{y}(k) = y(k) - (h(w, \hat{w}^{-}(k)) - H(k)\hat{w}^{-}(k)) \tag{9}$$

$$d(k) = f(w, \hat{w}^{-}(k)) - F_{k+1,k}\hat{w}(k) \tag{10}$$

Although there are some differences between the equations that define the neural networks of this work, both can be represented by the model

$$w(k+1) = w(k) + \Delta w(k) \tag{11}$$

$$\hat{y}(k) = h(w(k),\, \varrho(k)) \tag{12}$$

Which is a simplification of the state space model given by (1) and (2). $w(k)$ is the vector of synaptic weights, and $\hat{y}(k)$ is the output vector of the neural network, $\varrho(k)$ represents the input vector to the neural network and $h(\cdot)$ is the nonlinear output function of the network. Considering the model (11–12) and the extended Kalman filter, the following set of equations are used to update the synaptic weights of the neural networks at each iteration

$$K(k) = P(k)H^{T}(k)\left[R + H(k)P(k)H^{T}(k)\right]^{-1} \tag{13}$$

$$w(k+1) = w(k) + K(k)[y(k) - \hat{y}(k)] \tag{14}$$

$$P(k+1) = P(k) - K(k)H(k)P(k) + Q \tag{15}$$

Where $P(k) \in \mathbb{R}^{N \times N}$ and $P(k+1) \in \mathbb{R}^{N \times N}$ are the covariance matrices of the prediction error in iterations k and $k+1$, respectively; N represents the total number of synaptic weights in the neural network; $w(k) \in \mathbb{R}^{N}$ is the vector of weights (states); And $y(k) \in \mathbb{R}^{o}$ is the vector with the desired output of the network; Or is the total number of outputs of the neural network; And $\hat{y}(k) \in \mathbb{R}^{o}$ is the output vector produced by the neural network; $K(k) \in \mathbb{R}^{N \times o}$ is the Kalman gain matrix; $Q \in \mathbb{R}^{N \times N}$ is the covariance matrix of the process noise; $R \in \mathbb{R}^{o \times o}$ is the covariance matrix of the measurement noise and $H(k) \in \mathbb{R}^{o \times N}$ is the matrix containing the derivatives of each output of the neural network $\hat{y}_i$ with respect to each of the weights w_j of the network, as indicated by:

$$H_{ij}(k) = \left[\frac{\partial \hat{y}_i(k)}{\partial w_j(n)}\right]_{w(k)=\hat{w}(k+1)}, i = 1, \ldots, o; j = 1, \ldots, N \tag{16}$$

3.3 Embedding Dimension

To build a prediction model of a time series, it is important to determine the embedding dimension of the series. The embedding dimension defines the number of previous values in the time series known as regressors that will determine the next value. The method used in the present work to determine the embedding dimension is called the Cao method [20].

The reconstruction of the dynamics immersed in the time series was performed, for this purpose two parameters must be calculated, the first is the optimal delay of embedding τe, and the second is the embedding dimension de. The method is based on the construction of the delay vector of the series. The dimension vector de is defined as:

$$y(n) = [s(n), s(n+T), s(n+2T), \ldots, s(n-(d_e-1)\tau_e)] \tag{17}$$

In this method, the quantity is defined as:

$$a(n,d) = \frac{\|y_n(d+1) - y_{k(n,d)}(d+1)\|}{\|y_n(d) - y_{k(n,d)}(d)\|} \tag{18}$$

The number $k(n,d)$ is an integer such that the vector of dimension d, is the nearest value of the vector $y_n(d)$. The advantage of this method lies in defining the average:

$$E(d) = \frac{1}{N - d\tau_e} \sum\nolimits_{n=1}^{N-d\tau_e} a(n,d) \tag{19}$$

And define the parameter

$$E_1(d) = \frac{E(d+1)}{E(d)} \tag{20}$$

While d increases, the value of parameter $E_1(d)$ is stabilized in d_0, in this case it, is expected that d_0 approaches to 1. When this happens the minimum dimension of embedding will be when $d_0 \approx 1$. For the calculation of the optimal delay of embedding as shown in Fig. 3(A), an autocorrelation function was used, where the first value close to zero was selected and the delay was calculated for each database. The embedding dimension was obtained, using the autocorrelation function previously obtained. Figure 3(B) shows the convergence of the method for the time series of wind speed.

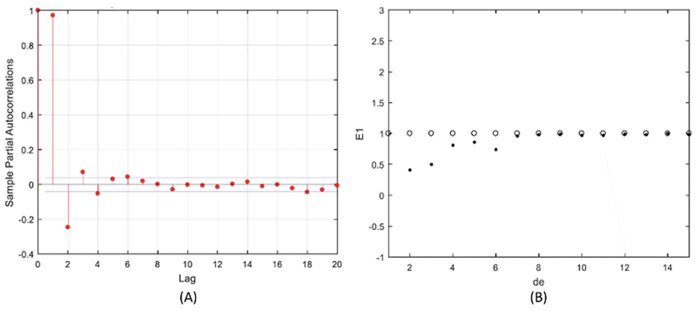


Fig. 3. (A) Autocorrelation function, (B) Embedding convergence using Cao's method

4 Optimization Problem Formulation

In this section, the optimization problem solution of a microgrid operation, interconnected to the public grid of the Engineering Faculty of the Autonomous University of Yucatan (UADY), is developed, besides there are batteries for storage and supply of energy. The objective is to determine the optimal amounts of energy for the wind, solar, battery and public power supply systems, to satisfy the demand generated by a building. A scheme is proposed that minimizes the purchase of energy from the public grid and the operating costs of the hybrid system's energy sources.

The problem of optimization to be solved is the type of linear programming, and for its solution, a recurrent neural network is proposed capable of solving the problem of optimization proposed by [8]. Computer programs are designed in Matlab to simulate the optimal operation of the hybrid system, using the proposed neural network. The simulation results show the performance of the neural network. In addition, it is necessary to define the operating costs of each energy source that forms the system.

The electric microgrid installed in the Engineering Faculty of the Autonomous University of Yucatan is illustrated in Fig. 4. It is composed of the public power grid, wind generators, photovoltaic systems, batteries for energy storage and a load management system. The objective is to determine the optimal amounts of power supplied for each energy source, over a time horizon, to satisfy the electrical demand, with respect to a time horizon. Minimizing the energy acquired from the public grid subject to constraints and power balance equations, as shown below intervals and the short-term generation scheduling problem can be formulated as follows:

$$CT = \sum\nolimits_{t=1}^{N} FC(t) \tag{21}$$

where

CT	Total energy cost acquired from the microgrid
FC	Energy cost of the microgrid at each time interval

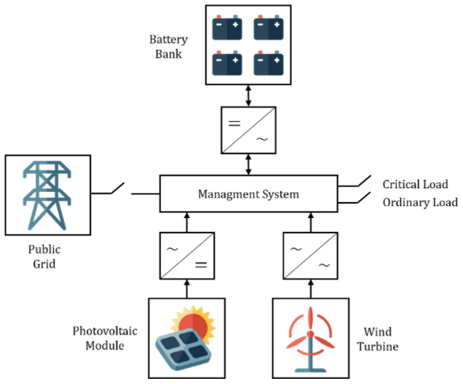


Fig. 4. Scheme of the microgrid installed in the Faculty

The optimization problem minimizes the purchase of energy from the public grid and the operating costs of the microgrid, the objective function arises as follows

$$FC(t) = C_G P_G(t) + C_W P_W(t) + C_S P_S(t) + C_{BD} P_{BD}(t) + C_{BC} P_{BC}(t) \tag{22}$$

where $P_B(t) = P_{BD}(t) - P_{BC}(t)$. The optimization problem is subject to constraints, which are also defined in each subsystem that compose the electric microgrid as

$$P_G(t) + P_W(t) + P_S(t) + P_{BD}(t) - P_{BC}(t) = L_c(t) + L_o(t) \tag{23}$$

where

$P_G(t)$	Power output of the public grid at time t
$P_W(t)$	Power output of the wind system at time t
$P_S(t)$	Power output of the solar system at time t
$P_{BD}(t)$	Power input of the battery bank at time t
$P_{BC}(t)$	Power output of the battery bank at time t
$L_c(t)$	Power demand of the critical load at time t
$L_o(t)$	Power demand of the ordinary load at time t
$L_T(t)$	Total power demand at time t

The total power demand is considered as follows $L_T(t) = L_c(t) + L_o(t)$

To solve the energy management problem a canonical form of a linear programming problem is described in the following way

$$min \quad c^T v \tag{24}$$

$$Av \leq b \tag{25}$$

$$v \geq 0 \tag{26}$$

where $v \in \mathbb{R}^n$ is a vector column of decision variables, $c \in \mathbb{R}^n, b \in \mathbb{R}^m$ are column vectors of parameters and $A \in \mathbb{R}^{mxn}$ is a coefficients matrix. For operative reasons, it is assumed that the feasible region denoted $\hat{V}$ is not empty and limited. Therefore, an upper bound v_{max} exists $0 \leq v \leq v_{max} for\, i = 1, 2, \ldots, n$. Moreover, the inequality of (25) can easily be converted to equality by the addition of m slack variables. Without loss of generality, the linear programming problem is described as follows

$$min \quad c^T v \tag{27}$$

$$Av = b \tag{28}$$

$$0 \leq v_i \leq v_{max}, \quad i = 1, 2, \ldots, n. \tag{29}$$

To solve an optimization problem through neural computation, the key is to propose the problem in a neural network whose steady state represents the solution to the optimization problem [21]. The state equations of the analog neural network are presented as follows

$$\dot{u}(t) = -\eta A^T A v(t) + \eta A^T b - c\xi(t) \tag{30}$$

$$v_i(t) = \frac{v_{max}}{1 + exp[-\beta u_i(t)]}, i = 1, 2, \ldots, n \tag{31}$$

where $u(t) \in \mathbb{R}^n$ is the instantaneous input vector of the network to the neurons and $v(t) \in R^n$ is the state vector of the activation respectively. The initial conditions $u(0)$ y $v(0)$ are initialized randomly, $\xi(t) \in \mathbb{R}^n$ is an auxiliary state vector and $\xi(0) > 0, \eta, \beta, yT$, are positive scalar parameters. The dynamic neural network proposed for linear programming consists of n connected artificial neurons massively. The connection weights between the neurons are given by $-\eta A^T A$ and the polarization thresholds of the neurons are given by $\eta A^T b$. Figure 5 represent the block diagram of (30) and (31).

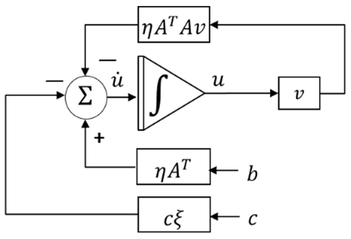


Fig. 5. Proposed dynamic neural network

For a neural network to perform a solution procedure for a linear programming problem, the steady state of a neural network must represent at least one feasible solution in the feasible region. To this end, two theorems are derived from [21].

Theorem 1. The steady state of the proposed neural network represents a feasible solution to the problem of linear programming (27–29), $lim_{t\to\infty} v(t) \in \hat{V}$.

Theorem 2. If $T \gg \frac{4}{\beta V_{max}\lambda_{min}}$ $where\ \lambda_{min} = min\{\lambda_i;\ i = 1, 2, \ldots, min\{m, n\}\}$ and λ_i is the $i-$ieth of $\eta A^T A$, then the steady state of the proposed neural network represents an optimal solution to the linear programming problem (27–29), $lim_{t\to\infty} v(t) = arg\, min_{v\in\hat{v}} c^T v$.

According to Theorem 1, the asymptotic stability of the proposed neural network implies the viability of the solution generated by the neural network.

5 Simulation Results

To achieve the optimization of the operation of a microgrid it is necessary to know the generation capacities of the renewable energy sources, as well as to estimate the energy demand in a determined period with the ability to make estimates from previously measured data. The simulation results are presented for the operation optimization of the microgrid installed in the Faculty. To perform the microgrid simulations, predictions of wind and photovoltaic power, wind speed and energy demand are considered, using high order neural networks. Each sample of the time series used in each prediction process is the average measurement at fifteen-minute intervals.

A neural network predictor for wind speed and wind power is implemented based on the high order recurrent neural network training method with Kalman filter. First, the optimum vector dimension is determined, then the number of hidden units for both hidden layers are selected. The training of the network is performed using measured data every fifteen-minutes for five days.

To train the high order recurrent neural network for each variable corresponding to wind generation, the following design parameters for each neural network was followed:

- 7 regressors
- 10 elements in the hidden layer
- 1 neuron in the output layer
- 250 maximum iterations
- The synaptic weights values were initialized randomly

The wind system has two wind turbines of 10 kW each, and the total wind power is applied to the inverter mathematical model, the result is illustrated in Figs. 6 and 7. A good prediction horizon was obtained with an average square error of 0.0048 for the prediction of wind speed and 0.0050 for wind power.

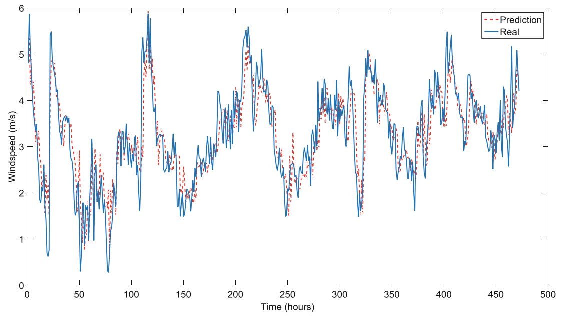


Fig. 6. Wind speed prediction of the wind system area (five days)

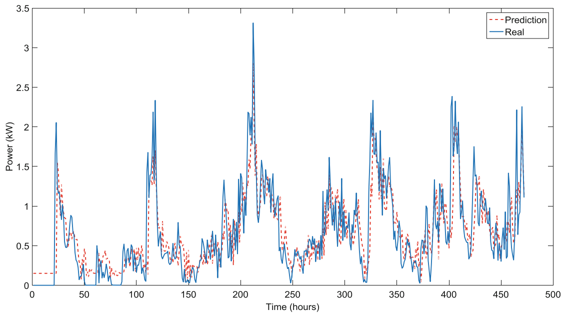


Fig. 7. Wind power prediction of the system installed in the Faculty (five days)

The total photovoltaic power generated by 86 panels installed in the Faculty is applied to the mathematical model of the inverter to obtain the power of the photovoltaic system, and the result is illustrated in Fig. 8. For the prediction of photovoltaic generation, the data was normalized before being processed by the network and the output of the network was scaled, the following design parameters for the neural network was followed:

- 8 regressors
- 8 neurons in the hidden layer
- 1 output neuron
- 200 maximum iterations
- The synaptic weights values were initialized randomly

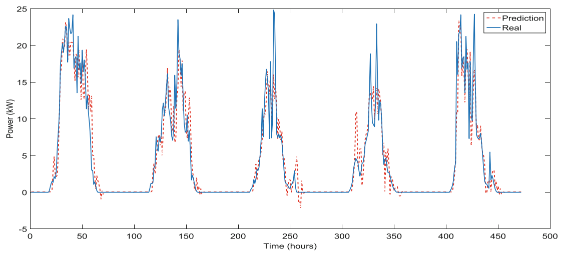


Fig. 8. Photovoltaic power prediction (five days)

For the prediction of energy demand, the following design parameters for the neural network was followed:

- 6 regressors
- 8 neurons in the hidden layer
- 1 neuron in the output layer
- 200 maximum iterations
- The synaptic weights values were initialized randomly

The prediction of the energy demand used in the simulation is presented in Fig. 9. Previously obtained data corresponding to the energy consumption of the mechatronics building in the Faculty. The prediction was successful with a good prediction horizon and a mean square error of 0.00012.

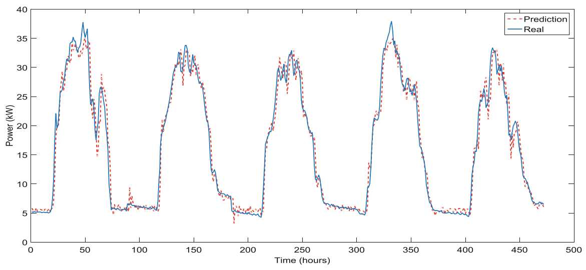


Fig. 9. Energy demand of the building (five days)

From the predictions and the tariff scheme, the algorithm determines if it is necessary to perform energy storage in the battery bank and when it should be used, taking as constraints the maximum and minimum state of charge of the batteries. On the other hand, in the case where the current power delivered by the photovoltaic and wind system is not enough to satisfy the demand, the management system determines the amount of power that it must acquire from the public grid.

After obtaining the predictions corresponding to the power and wind speed, photovoltaic generation and energy demand, they were used in a high order recurrent neural network, where the energy loads management is performed. Figure 10 shows the graph of five days of generation and consumption respectively. According to the results obtained in the simulation, it is observed that the energy demand is always higher than the generation of the wind and photovoltaic systems, and the battery bank, so the system determines the amount of power that is lacking and this is acquired from the public grid.

In Mexico de the information of the energy cost from the public grid is available on its website, in this work, a maximum rate of $ 3.85 per kWh of energy consumed to the public grid was established. According to the results obtained in the simulation, it is obtained that the total operation cost of the microgrid is $ 1395.70, where the daily power consumption of the public grid in kWh is 172.85 and the monthly energy consumption in kWh is 5185.73.

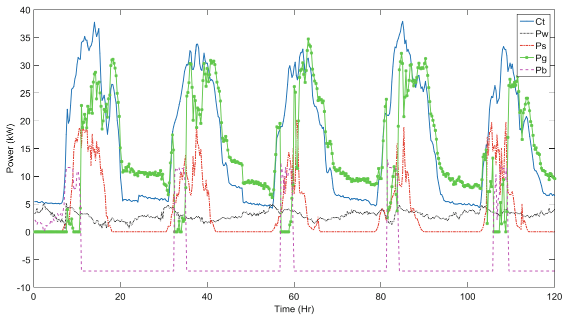


Fig. 10. Power flow in the microgrid with the management system (five days)

The state of charge of the batteries in the system operation, as is illustrated, the loading and unloading behavior is very linear, this behavior is because the energy demand is always higher than the energy generated and stored, so the batteries are in constant operation. The system behaves correctly within the range of the parameters set.

The main advantage of the computational intelligence techniques presented in this chapter is the availability of a reliable tool for forecasting the energy production and the corresponding distribution of the charges and storage of the batteries using artificial neural networks.

6 Conclusions

In this work, the optimization of the operation of a microgrid based on recurrent neural networks was performed. According to the results obtained, it is established that the proposed network configuration proves to be functional. In addition, based on the proposed configuration, it is possible to implement mathematical models of the different elements of the system. The goal was to determine the optimum amounts of energy for wind, solar, battery bank and public grid systems.

High order recurrent neural networks were applied to predict the energetic variables presented in the microgrid with good estimation results. The network used has a compact structure but considering the dynamic nature of the system that is required to predict the behavior, high order neural networks demonstrated in the simulation to be a tool that adequately models the complexity associated with the dynamics of energy generation. The importance of having a permanent measurement system is that it leads users sooner or later to achieve improvements in the electrical system and obtaining energy savings based on a real base of comparison, as well as energetic.

The number of iterations in the range of 200 to 250 was chosen to improve the speed of convergence, since a greater number of iterations does not significantly improve the error, but the computation time increases. Likewise, it was tested with a greater number of neurons, however, the result did not improve, so it was decided to use as few neurons as possible, without losing fidelity at work.

Acknowledgments. This work was carried out thanks to the support of CONACYT on project CONACYT-SENER 254667 and the project FOMIX/YUC/2017/073.

References

1. Ricalde, L.J., Ordonez, E., Gamez, M., Sanchez, E.N.: Design of a smart grid management system with renewable energy generation. In: 2011 IEEE Symposium on Computational Intelligence Applications in Smart Grid, pp. 1–4 (2011)
2. Carta González, J.A.: Centrales de energías renovables: generación eléctrica con energías renovables. Pearson: UNED, Madrid (2013)
3. Bhandari, B., Lee, K.-T., Cho, Y.-M., Ahn, S.-H.: Optimization of hybrid renewable energy power system: a review. Int. J. Precis. Eng. Manuf. Technol. **2**(1), 99–112 (2015)
4. Hernández, L., Baladrón, C., Aguiar, J.M., Carro, B., Sánchez-Esguevillas, A., Lloret, J.: Artificial neural networks for short-term load forecasting in microgrids environment. Energy **75**, 252–264 (2014)
5. Hidaka, Y., Kawahara, K.: Modeling of a hybrid system of photovoltaic and fuel cell for operational strategy in residential use. In: Proceedings of Universities Power Engineering Conference (2012)
6. Bugała, A., Zaborowicz, M., Boniecki, P., Janczak, D., Koszela, K., Czekała, W., Lewicki, A.: Short-term forecast of generation of electric energy in photovoltaic systems. Renew. Sustain. Energy Rev. **81**, 306–312 (2018)
7. Ata, R.: Artificial neural networks applications in wind energy systems: a review. Renew. Sustain. Energy Rev. **49**, 534–562 (2015)
8. Gamez, M.E., Sanchez, E.N., Ricalde, L.J.: Optimal operation via a recurrent neural network of a wind-solar energy system. In: Proceedings of International Joint Conference on Neural Networks, no. 491, pp. 2222–2228 (2011)
9. Li, Y., Sun, Z., Han, L., Mei, N.: Fuzzy comprehensive evaluation method for energy management systems based on an Internet of Things. IEEE Access **5**, 21312–21322 (2017)
10. Byrne, R.H., Nguyen, T.A., Copp, D.A., Chalamala, B.R., Gyuk, I.: Energy management and optimization methods for grid energy storage systems. IEEE Access **1**(99), 1–31 (2017)
11. Rodriguez-diaz, E., Palacios-garcia, E.J., Anvari-moghaddam, A., Vasquez, J.C., Guerrero, J.M.: Real-time energy management system for a hybrid AC/DC residential microgrid. In: IEEE Conference on DC Microgrids, pp. 1–6 (2017)
12. Haykin, S.S.: Neural Networks and Learning Machines. Pearson Education, Harlow (2009)
13. Bassam, A., May Tzuc, O., Escalante Soberanis, M., Ricalde, L.J., Cruz, B.: Temperature estimation for photovoltaic array using an adaptive neuro fuzzy inference system. Sustainability **8**(8), 1399 (2017)
14. Dzib, J.T., Moo, E.J.A., Bassam, A., Flota-Bañuelos, M., Soberanis, M.A.E., Ricalde, L.J., López-Sánchez, M.J.: Photovoltaic module temperature estimation: a comparison between artificial neural networks and adaptive neuro fuzzy inference systems models. In: Martin-Gonzalez, A., Uc-Cetina, V. (eds.) ISICS 2016. CCIS, vol. 597, pp. 46–60. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30447-2_4
15. El Hamzaoui, Y., Ali, B., Alfredo Hernandez, J., Aburto, O.C., Oubram, O.: Search for optimum operating conditions for a water purification process integrated to a heat transformer with energy recycling using artificial neural network inverse solved by genetic and particle swarm algorithms. Chem. Prod. Process Model. **7**(1) (2012)

16. Bassam, A., Del Castillo, A.Á., García-Valladares, O., Santoyo, E.: Determination of pressure drops in flowing geothermal wells by using artificial neural networks and wellbore simulation tools. Appl. Therm. Eng. **75**, 1217–1228 (2015)
17. Haykin, S.S.: Kalman Filtering and Neural Networks. Wiley, New York (2001)
18. Rizal, A.A., Hartati, S.: Recurrent neural network with extended kalman filter for prediction of the number of tourist arrival in Lombok. In: 2016 International Conference on Informatics and Computing, pp. 180–185 (2016)
19. Burger, E.M., Moura, S.J.: Recursive parameter estimation of thermostatically controlled loads via unscented Kalman filter. Sustain. Energy Grids Netw. **8**, 12–25 (2016)
20. Xiaoke, X., Xiaoming, L., Xiaonan, C.: The Cao method for determining the minimum embedding dimension of sea clutter. In: CIE International Conference on Radar Proceedings (2007)
21. Wang, J.: Analysis and design of a recurrent neural network for linear programming. IEEE Trans. Circuits Syst. I Fundam. Theory Appl. **40**(9), 613–618 (1993)

Intelligent Recognition System of Myoelectric Signals of Human Hand Movement

Sergio A. Vicario Vazquez[1](✉), Outmane Oubram[1], and Bassam Ali[2]

[1] Facultad de Ciencias Químicas e Ingeniería, Universidad Autónoma del Estado de Morelos, Av. Universidad 1001 Col. Chamilpa, 62209 Cuernavaca, Mexico
vicario17_@hotmail.com, oubram@uaem.mx

[2] Facultad de Ingeniería, Universidad Autónoma de Yucatán, Av. Industrias no Contaminantes por Periférico Norte, Apdo. Postal 150, Cordemex, Mérida, Yucatán, Mexico
baali@correo.uady.mx

Abstract. This paper presents the recognition of myoelectric signal's algorithm design thru artificial neural network architecture, for the manufacture of a prototype of human hand prosthesis. At the beginning of the project the myoelectric sensor was designed to help capture the signals that correspond to the movement of each finger of a human hand. A database was generated with captured myoelectric signals, which was used for the training of artificial neural networks (ANN), obtaining the weights and bias. The performance of the architecture was evaluated with statistical criteria for the validation of ANN, comparing between simulated data and experimental data. It was found, that the best architecture in this project has 7 neurons in the hidden layer, one in the output layer and 96% correlation coefficient, this architecture is the number 7 in Table 1 which contains a performance report learning algorithm of the different architectures proposed.

Keywords: Myoelectric signals · Intelligent recognition · Prosthesis Artificial neural network · Myoelectric sensor

1 Introduction

Prosthesis is an artifact developed in order to improve the quality of life of people who by some accident lost some limb of their body. Through the advancement of technology in the last decades prostheses have becoming more advanced. The human body is able to generate electrical signals in its muscles, these signals are known as bioelectric signals. The bioelectrical signals are divided into different types depending on the origin of this signal. The myoelectric signals are those generated by the contraction of some muscle of any extremity such as the arms and legs, can be measured with a suitable equipment and thus use the information that these provide us in the design of prostheses. The work of Parimal performs a micro controlled system, based on the microcontroller 68HC11 (Parimal et al. 1988). The system amplifies the myoelectric signals, filters them, digitizes them and the control algorithm decides the movement of a robot of two degrees of freedom. Another similar work in which, unlike the previous

C. Brito-Loeza and A. Espinosa-Romero (Eds.): ISICS 2018, CCIS 820, pp. 97–112, 2018.
https://doi.org/10.1007/978-3-319-76261-6_8

one, a myoelectric prosthesis is activated (Romero et al. 2001). In systems oriented to assistance in industrial environment, is the work of López, di Sciasco, Orosco, Ledesma, Echenique, and Valentinuzzi (López et al. 2006), in this paper, a myoelectric sensor was designed, which is responsible for capturing, amplifying and filtering the signals generated by the contraction of the muscles responsible for the movement of each finger. The recognition of the myoelectric signals is performed through neural network architecture, allowing a more reliable result. This architecture could be developed and validated thanks to the Matlab tool, to later generate a function which represents the operation of said neural network, obtaining a signal recognition algorithm and entering it in the Arduino Mega 2560 development board in order to obtain a Prototype of prosthesis of a human hand shown in Fig. 1.

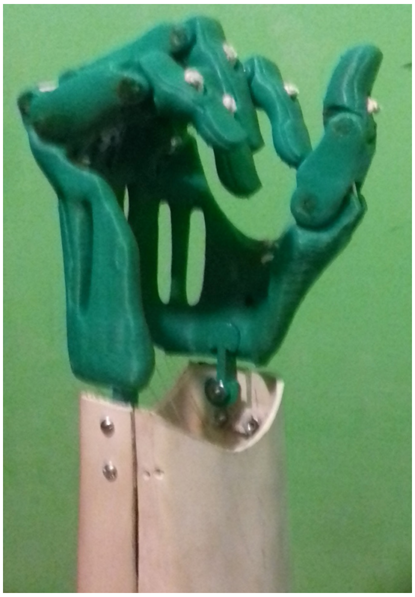


Fig. 1. Prototype of prosthesis of a human hand.

2 Experimental System

The development methodology of this research project begins with the design of a myoelectric sensor for the acquisition of the signals. A database was created from the myoelectric signals generated by the contraction of the muscles responsible for the movements of each finger of a human hand.

2.1 Myoelectric Sensor Design

The parameters of the myoelectric signals to be captured are:

- The frequency of the myoelectric signal is 50 and 200 Hz.

- Sleep state: is the state where the excitable cells maintain a potential -90 mV < V < -50 mV.
- Active state: in this state excitable cells have an electrical potential between -55 mV < V < 30 mV.

A myoelectric sensor was designed with the following characteristics:

- Pre-amplification
- High pass filter
- Low pass filter
- Amplification and signal coupling
- Selection of cables for electromyography (EMG)

An instrumentation amplifier with biomedical applications AD620 was used, which requires less components to adjust the gain and presents a rather high CMRR (Common Mode Rejection Ratio) value at low frequencies. To calculate the components of this stage, the formula recommended in the AD620 circuit data sheet was used:

$$G = 1 + \frac{49.9\,\mathrm{K}\Omega}{R_G}$$

In this case it is desired to calculate R_G, since it was postulated as $G = 250$, therefore the result is the following:

$$R_G = \frac{49.4\,\mathrm{K}\Omega}{250 - 1} = 198.88\,\Omega;\ \frac{R_G}{2} = \frac{200\,\Omega}{2} = 100\,\Omega$$

High Pass Filter: This was calculated for a cut-off frequency of 5 Hz with a commercial capacitor of $C = 0.1$ μF.

$$F_C = \frac{1}{R * 2\pi * c};\ R = \frac{1}{2\pi * (0.1\,\mu F) * (5\,\mathrm{Hz})} = 318.3\,k\Omega$$

$$R_6 = 610\,k\Omega // 610\,k\Omega = 305\,k\Omega$$

Inverter Amplifier: The purpose of this amplifier that is connected between the R_6 of the AD620 is to provide patient safety also helps to maintain a stability in the signals obtained in the EMG.

$$\frac{V_0}{V_1} = -\frac{R_F}{R_1};\ \frac{V_0}{V_1} = -\frac{470\,\mathrm{K}\Omega}{12\,\mathrm{K}\Omega} = 39.16$$

Low pass filter: A Butterworth low pass filter of order 2 was designed with a Sallen-key topology, with a quality factor of 0.71, gain 1, and with a cutoff frequency of 980 Hz. Obtaining the following results.

$$R_1 = 22\,\mathrm{K\Omega};\, R_2 = 1.2\,\mathrm{K\Omega};\, C_1 = 10\,\mathrm{nF};\, C_2 = 100\,\mathrm{nF}$$

Once the amplification problem of the small signals has been solved, it is necessary to design additional stages for conditioning the system, which are aimed at filtering and cleaning the signal being collected and amplifying the filtered signal. The circuits used are shown in Fig. 2.

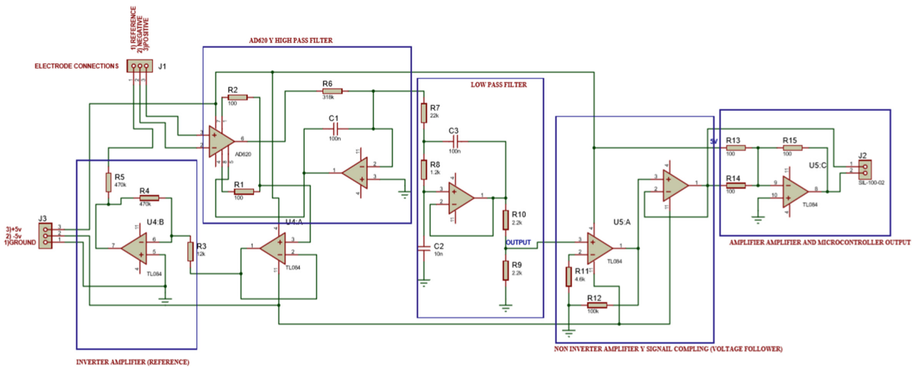


Fig. 2. Sensor of myoelectric signals with all its stages.

2.2 Acquisition

When the myoelectric sensor was obtained, the EMG was performed, in other words, to capture signals from the muscles responsible for the movement of the fingers on the hand. The EMG consists of placing surface electrodes on the required extremity as shown in Fig. 3, these sensors are connected by wires to the input of the myoelectric sensor, and the output of the myoelectric sensor is connected to the Arduino Mega 2560 which sends the data to a PC.

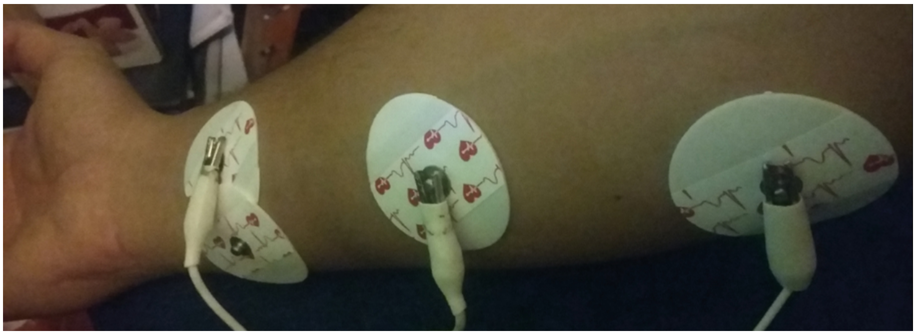


Fig. 3. Placement of the surface electrodes for the realization of EMG.

With the help of the Matlab tool the signals obtained were plotted and saved, in order generate a database necessary for the training of the neural network. In this work the movements on which they worked were the contraction of each of the fingers and the full fist at the same time, namely, the movement of opening and closing the complete hand. The signals are shown in the following Figs. 4, 5, 6, 7, 8 and 9.

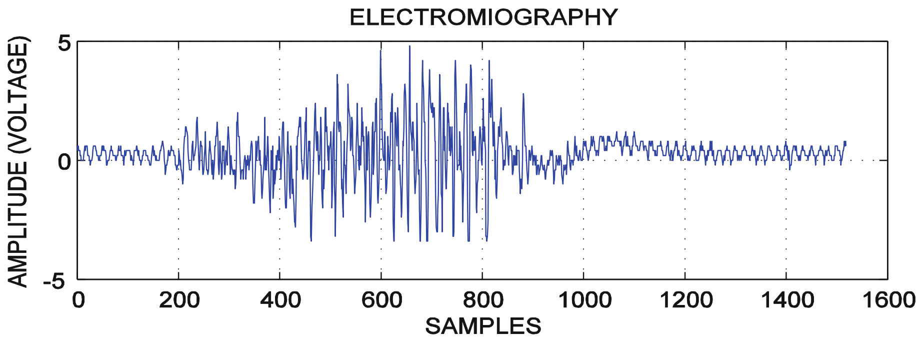


Fig. 4. Myoelectric signal produced by the full fist.

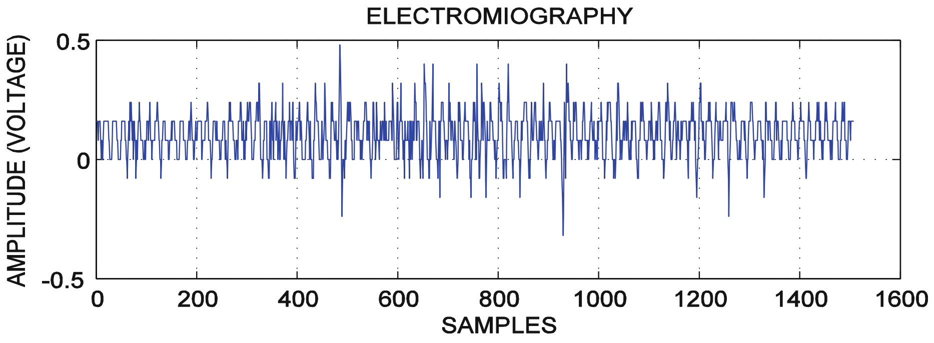


Fig. 5. Myoelectric signal produced by the index finger

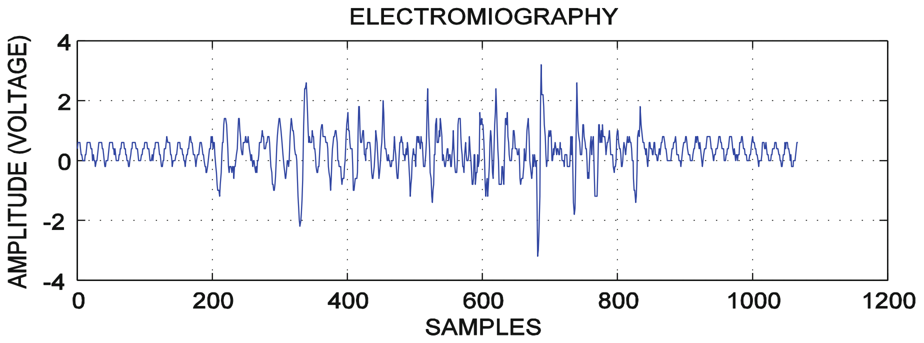


Fig. 6. Myoelectric signal produced by the middle finger

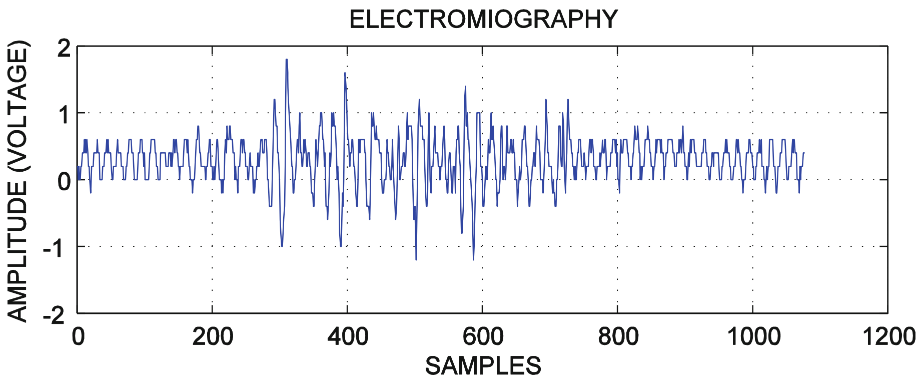


Fig. 7. Myoelectric signal produced by the ring finger

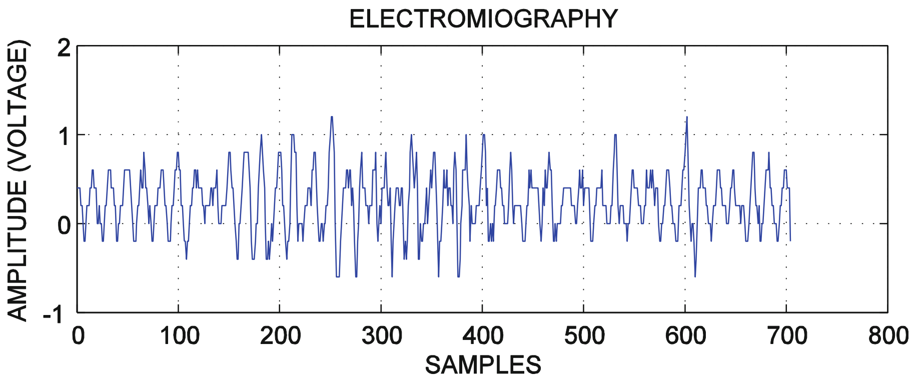


Fig. 8. Myoelectric signal produced by the little finger

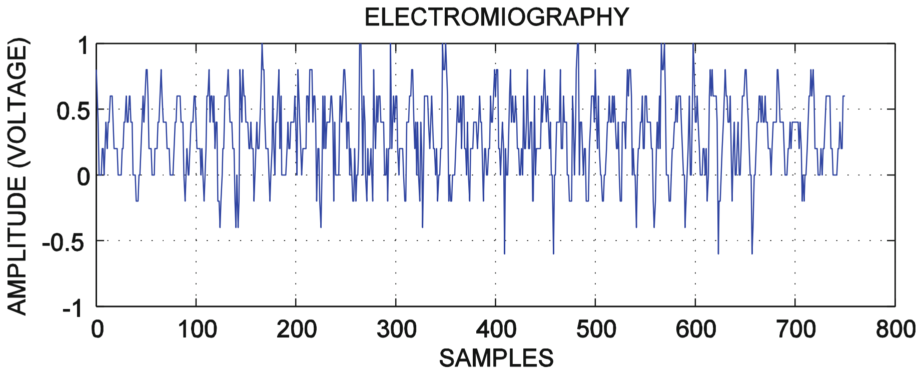


Fig. 9. Myoelectric signal produced by the thumb.

3 Artificial Neuronal Networks

An Artificial Neuronal Networks (ANN) is defined as a nonlinear mapping system. They consist of a number of simple processors connected by connections with weights. Processing units are called neurons. Each unit receives inputs from other nodes and generates a simple scalar output that depends on the available local information, and stored internally or arriving through the weighted connections.

Simple artificial neurons were introduced by McCulloch and Pitts in 1943. An artificial neural network is characterized by the following elements:

1. A set of processing units or neurons.
2. An activation state for each unit, equivalent to the output of the unit.
3. Connections between units, generally defined by a weight that determines an input signal to the unit.
4. A propagation rule, which determines the effective input of a unit from the external inputs.
5. An activation function that updates the new activation level based on the actual input and previous activation.
6. An external entry that corresponds to a term determined as bias for unit.
7. A method for gathering information, corresponding to the learning rule.
8. An environment in which the system will operate, with input signal and even error signals.

An abstract and simple model of an artificial neuron (Fig. 10) composed of a set of inputs $X = (x_1, x_2, x_3, \ldots, x_i)$ which simulates the function of the dendrites; A vector of $W = (w_1, w_2, w_3, \ldots, w_i)$ which emulates the function of a synapse; An action threshold or also called bia θ; An activation function which is considered the equivalent of soma and an output (Pedro and Inés 2004).

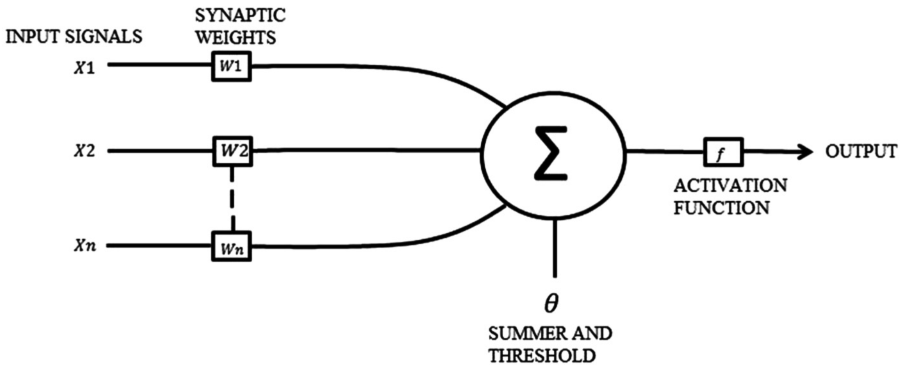


Fig. 10. Natural elements of a neuron.

The descriptions of the components that make up an artificial neuron are:

- Input vector: this vector $X = (x_1, x_2, x_3, \ldots, x_i)$ contains the information that the neuron is going to process; Where "i" is the number of inputs to the neuron.
- Vector of synaptic weights: Each of the elements of the input vector $X = (x_1, x_2, x_3, \ldots, x_i)$ is multiplied by an adjustable value called the synaptic weight and is represented by the vector $X = (x_1, x_2, x_3, \ldots, x_i)$. This vector is adjusted during training, in order to minimize the error of the neuron output with respect to the expected output.
- Threshold or bia: It is denoted by the symbol θ and is considered as an additional weight that receives an input with a value equal to unity.

Activation or transfer function: The rule that establishes the effect of the total input u_t on the activation of the unit is called the activation function F_k (Pedro 2010).

Some of the most used functions are the linear, sigmoid and hyperbolic tangent (Sergio 2017). The equations are listed as follows:

$$f(x) = x$$

$$f(x) = \frac{1}{1 + e^{-x}}$$

$$f(x) = \frac{2}{1 + e^{-2x}} - 1$$

Starting from a series of random synaptic weights, the learning process looks for a set of weights that allow the network to develop a certain task. Most of the training methods used in neural networks with forward connection consist of proposing an error function that measures the current performance of the network as a function of the synaptic weights. The goal of the training method is to find the set of synaptic weights that minimize (or maximize) the function. There are two types of learning, if the network learns during its normal operation or if the disconnection of the network until the process terminates on-line and off-line networks, respectively (Pedro and Inés 2004).

- Learning algorithms are listed below:
- Descending batch gradient (traingd).
- Gradient batch and traingdm.
- Variable learning rate (traingdx).
- Conjugate gradient algorithms.
- Scaled conjugate gradient (trainscg).
- BFGS algorithm (trainbfg).
- Levenberg-Marquardt (trainlm).

After the training were submitted to different statistical processes in order to compare the performance of the different ANN architectures obtained. Some of the statistical equations used to compare groups of data generated with artificial neural networks are described below. The output of the ANN is OUT_{SIM} the simulated data, the experimental data are named with OUT_{EXP} and n represents the number of values of the samples (Bassam 2014).

Coefficient of determination measures the degree of dependence between variables, taking the value 0 in case of null correlation or the value 1 in case of total correlation. Equivalent to the square of the correlation coefficient.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(OUT_{\exp(i)} - OUT_{sim(i)}\right)^2}{\sum_{i=1}^{n} \left(OUT_{\exp(i)} - OUT_{\exp}\right)^2}$$

The Root Mean Square Error (RMSE) is a measure of the degree of dispersion of the data with respect to the average value. It is known as the standard deviation for a discrete probability distribution:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} \left(OUT_{sim(i)} - OUT_{\exp(i)}\right)^2}{n}}$$

MPE (Mean Percentage Error) is the mean of the percentage error. It is a simple metric, used to see if the forecast error has a positive or negative bias. It is also said that the forecast is underestimated or overestimated

$$MPE = \frac{\sum_{i=1}^{n} \left(\frac{OUT_{\exp(i)} - OUT_{sim(i)}}{OUT_{\exp(i)}}\right)}{n} \times 100(\%)$$

4 Artificial Neural Network Design and Training

Once the signals were captured and a database was generated, the data was subsequently debugged. This was done by submitting the database to a statistical process which gave us the variables that were considered for the input vector of RNA, these variables are: variance, standard deviation, mean square value, RMS value, symmetry coefficient and kurtosis. For the training of the neural network we used the variables derived from the database debugging and the Matlab package, which offers us a series of predefined functions which facilitate the training process (Demuth et al. 2007). We performed different tests dividing the database into 3 in different percentages, that is, 60% for training, 20% validation and 20% for test. Before training, it was considered to define different criteria, such as the number of hidden layer neurons, the learning factor among others. The Levenberg-Marquardt algorithm was used for the training of the tests, since in comparative studies (Sergio 2017) it has shown a better performance with respect to the algorithms of backpropagation, descending gradient, among others.

An algorithm was proposed which helped to calculate the following parameters:

- Number of neurons in the hidden layer.
- Learning factor.
- Linear correlation coefficient.
- The RMSE.
- Synaptic weights and bias.
- Storage of results of each training.

The artificial neural network which has the following characteristics (Fig. 11):

- Network type: perceptron.
- 8 inputs, 7 neurons in the hidden layer, one neuron in the output layer and one output.
- Hyperbolic function in the hidden layer of RNA.
- Linear function in the RNA output layer.
- A Levenberg-Marquardt optimization algorithm.

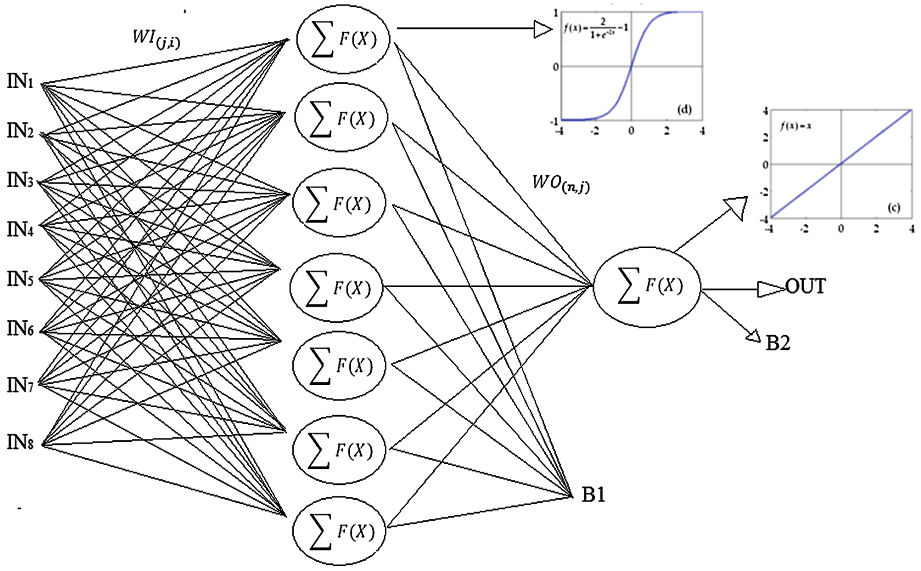


Fig. 11. Neural network graphical model.

In the Table 1 an extract of a report generated from the training is presented, in the Fig. 12 is shown an outline of which are the stages that were followed in the work. This report was obtained using the data base of the myoelectric signals and with the 8 variables derived from the debugging of the database with the help of statistical methods. In this example the architecture No. 7 exhibits the best correlation (0.9622). The cases were calculated the R^2 (test), RMSE (global and test) and the C_R of the variables. The global RMSE includes 100% of the training data and the test data only 20%.

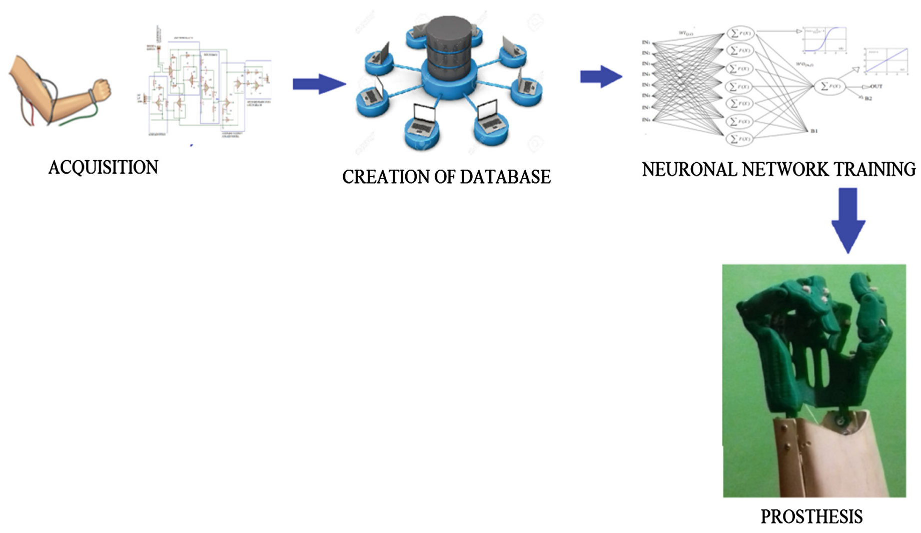


Fig. 12. Diagram of the stages by the project is constituted.

Table 1. Report extract generated by the learning algorithm.

Number of architecture	Architecture ANN	Numbers of neurons	Epoch	Root Means Square Error (*RMSE*)	Means Percentage Error (*MPE*)	R^2 Coefficient of determination	Best linear equation
1	8–1–1	1	1000	0.554209372	17.36174307	0.9004	0.81*T* + 0.71
2	8–2–1	2	1000	0.285012061	12.64856695	0.9502	0.88*T* + 0.43
3	8–3–1	3	1000	0.283046879	13.77088346	0.9504	0.89*T* + 0.04
4	8–4–1	4	1000	0.27890845	11.34057548	0.9516	0.89*T* + 0.35
5	8–5–1	5	1000	0.213592679	11.66547187	0.9628	0.92*T* + 0.3
6	8–6–1	6	1000	0.230296651	10.53668406	0.9599	0.94*T* + 0.2
7	8–7–1	7	1000	0.21687368	10.19372467	0.9622	0.93*T* + 0.27
8	8–8–1	8	1000	0.264669891	12.85161597	0.9539	0.9*T* + 0.33
9	8–9–1	9	1000	0.275548412	11.76859909	0.9517	0.92*T* + 0.28
10	8–10–1	10	1000	0.267190521	11.66423641	0.9535	0.94*T* + 0.22

After completing the training of the network with the architecture No. 7, the validation stage was started in which the database compiled in the electromyography was tested. The objective is to compare the experimental and simulated outputs, in order to bring the ANN architecture to a development card (Arduino) and thus give an application to the ANN.

5 Results and Implementation of the Neural Network

The results obtained with this architecture were satisfactory obtaining the following parameters:

- *RMSE*: 0.2168 (Root Means Square Error).
- *MPE*: 10.194% (Percentage Error).
- Alignment: 0.93*T* + 0.27.
- Synaptic weights.
- Pathways.

The linear correlation between the measured and the estimated ANN obtained in test 7 is shown in Fig. 13, the values of *WI, WO, B1* and *B2*, respectively, are shown in Tables 2 and 3.

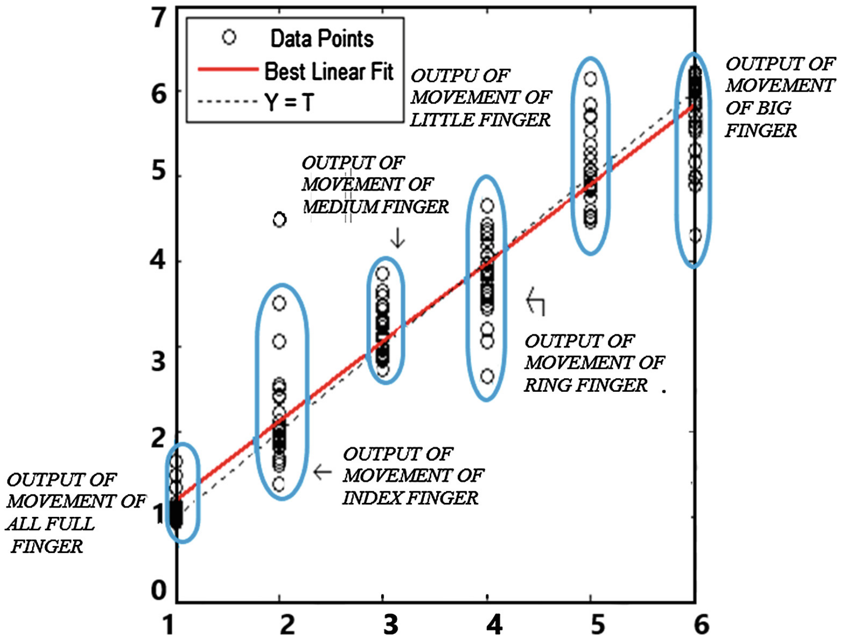


Fig. 13. Graph of linear correlation between the experimental and simulated results.

Table 2. Synaptic weights of architecture #7, of the 8 inputs for the 7 neurons of the hidden layer.

Synaptic Weight Matrix WI							
0.8475	2.2854	2.1476	0.1976	−0.5232	−3.9432	−2.5756	−3.3085
3.7104	−9.9741	−10.029	−5.5328	−3.5506	−5.6621	−7.5008	−6.102
14.8244	13.2162	−0.6775	4.2638	−3.1937	−3.8813	−6.7063	0.6947
−7.9776	−15.0781	3.212	6.7119	1.209	1.3704	9.7667	1.7998
1.5851	0.6765	−0.6552	2.0935	−1.1748	1.1137	−1.5177	−0.7433
−3.9121	−41.0433	−11.7778	8.4399	−1.998	10.4343	−22.3041	0.3034
6.7575	3.4153	1.9836	−19.1374	1.4781	−6.1953	8.4232	−3.6328

Table 3. Synaptic weights and paths of the hidden layer and the layer of exit.

Synaptic weights and bias of the hidden layer and exit layer							
Vector WO	−1.9619	1.0649	1.2106	1.004	1.8337	−1.2558	0.9283
Vector B1	6.8032	7.7847	−3.5825	3.3002	5.0344	10.8505	2.6925
Vector B2	1.4902						

5.1 Equation Resulting from ANN

Finally we present the proposed equation obtained from architecture #7, given by:

$$\boldsymbol{OUT} = \sum_{j=1}^{7}\left[\boldsymbol{WO}_{(\boldsymbol{n},\boldsymbol{j})}\left(\frac{2}{1+\boldsymbol{e}^{-2\left(\sum_{i=1}^{8}\left(\boldsymbol{WI}_{(j,i)}*\boldsymbol{In}_{(i)}\right)+\boldsymbol{B}2\right)}}-1\right)\right]+\boldsymbol{B}1$$

WHERE:

- $WI_{(j,i)}$ = Matrix of synaptic weights of the inputs.
- $WO_{(n,j)}$ = Matrix of synaptic weights of the hidden layer.
- $In_{(i)}$ = Input data.
- $B1$ = Paths of the hidden layer.
- $B2$ = Path of the output layer.

5.2 Application with Arduino

The final application of the previous processes, an ARDUINO MEGA 2560 development board was used, it was possible to carry out a demonstrative application in which the proposed ANN equation was entered in a simplified way. This application was achieved with an algorithm show in Fig. 14 hat has the purpose of making a selection of the input data, which were obtained from the database debugging, it is worth mentioning that each input data is a result of a Statistical process applied to each signal, produced by the movement of each of the fingers. After performing the process of data entry, these are introduced to the ANN equation obtaining an output value which will trigger the movement of one of the fingers of the prototype of the prosthesis.

In order to validate the proper functioning of the training and the result of the neural network, an arduino program was performed, in which a part of the database, the synaptic weights and the bias are entered.

The program that was developed consists of three steps which will be explained in detail:

- Step one: the program generates a random number, that number represents a row of the database which is an 8 × 24 matrix (columns per row), the data in that row is entered into the resulting function of the training of the neural network.
- Step two: data entered into the function of the neural network results in a predicted output according to the row that was selected each result obtained will generate an action.

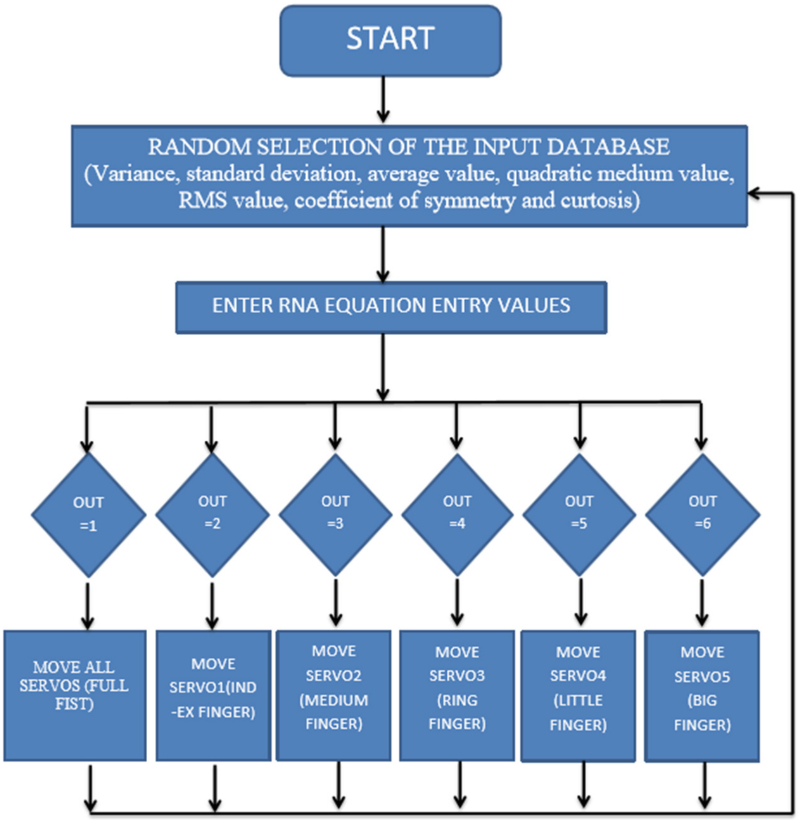

START
RANDOM SELECTION OF THE INPUT DATABASE (Variance, standard deviation, average value, quadratic medium value, RMS value, coefficient of symmetry and curtosis)
ENTER RNA EQUATION ENTRY VALUES
OUT =1
OUT =2
OUT =3
OUT =4
OUT =5
OUT =6
MOVE ALL SERVOS (FULL FIST)
MOVE SERVO1(INDEX FINGER)
MOVE SERVO2 (MEDIUM FINGER)
MOVE SERVO3 (RING FINGER)
MOVE SERVO4 (LITTLE FINGER)
MOVE SERVO5 (BIG FINGER)

Fig. 14. Algorithm used for the application in ARDUINO MEGA 256.

- Step three: each action is based on the calculated output, it is worth mentioning that these outputs have an error rate of 10.94% (*MPE*). The order of the actions is as follows.

- *Out* = 1: All servos are actuated, imitating how the complete fist closes (Fig. 15a).
- *Out* = 2: the servo is actuated by mimicking the contraction of the index finger (Fig. 15b).
- *Out* = 3: the servo two is actuated by mimicking the contraction of the medium finger (Fig. 15c).
- *Out* = 4: the servo three is actuated by mimicking the contraction of the ring finger (Fig. 15d).
- *Out* = 5: the four servo is actuated by mimicking the contraction of the little finger (Fig. 15e).
- *Out* = 6: the five servo is actuated by mimicking the contraction of the thumb (Fig. 15f).

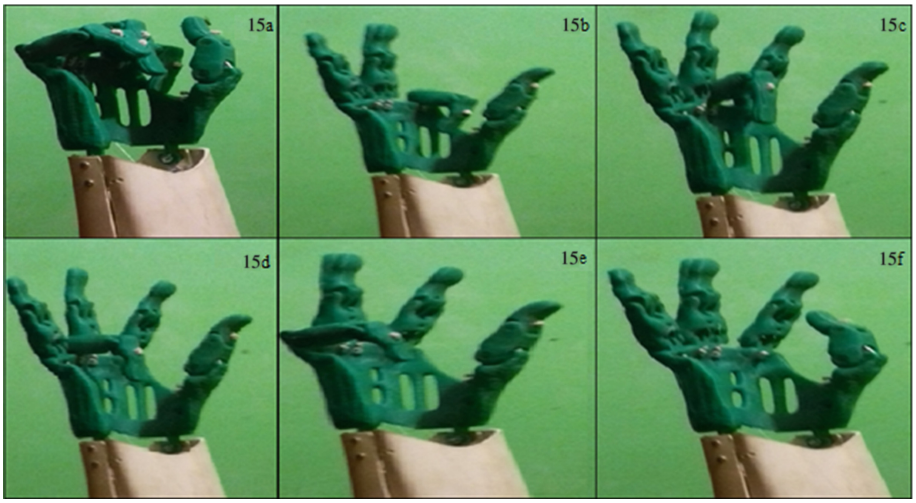


Fig. 15. (a) *Out* = 1 full closed fist, (b) *Out* = 2 contracted index finger, (c) *Out* = 3 contracted middle finger, (d) *Out* = 4 contracted annular finger, (e) *Out* = 5 contracted pinky finger, (f) *Out* = 6 contracted thumb.

6 Conclusions

Through the design of a myoelectric sensor it was possible to obtain myoelectric signals, the contraction of the muscles responsible for the movements of each finger, of a human hand, as well as to develop a way of selection and characterization of myoelectric signals. Using neural networks we can obtain a selection of signals with a small degree of error, with this we offer a good performance of the prototype of the prosthesis.

References

Parimal, P., Allen, S., Rapach, G.: Microcontroller based prosthetic device using myoelectric signal. In: Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 1427–1428 (1988)

López, N., di Sciasco, F., Orosco, E., Ledesma, N., Echenique, A., Valentinuzzi, M.: Control Mioelétrico de un brazo robótico. In: IV Congreso Iberoamericano sobre Tecnologías de Apoyo a la Discapacidad. Vittoria-Espiritu Santo-Brasil (2006). ISBN: 8496023451

Alonso, A., Hornero, R., Espino, P., de la Rosa, R., Liptak, L.: Entrenador Mioeléctrico de Prótesis para Amputados de Brazo y Mano. Universidad de Valladolid (2001)

Pedro, V.I., Inés, G.L.: Redes de Neuronas Artificiales, Un enfoque Práctico. Pearson Education, Madrid (2004)

Pedro, C.P.: Inteligencia Artificial con Aplicaciones en Ingeniería. Alfaomega, México D.F. (2010)

Bassam, A., Conde-Gutiérrez, R.A., Castillo, J., Laredo, G., Hernández, J.A.: Direct neural network modeling for separation of linear and branched paraffins by adsorption process for gasoline octane number improvement. Fuel **124**, 158–167 (2014)

Sergio, V.V.A.: Adquisición, Procesamiento e Interpretación de Señales Mioeléctricas para el Diseño y Fabricación Robótica, tesis de licenciatura. Universidad Autónoma del Estado de Morelos, Cuernavaca (2017)

Analyzing Students Reviews of Teacher Performance Using Support Vector Machines by a Proposed Model

G. Gutiérrez[1](✉), J. Ponce[2], A. Ochoa[3], and M. Álvarez[1]

[1] Universidad Politécnica de Aguascalientes, Aguascalientes, Mexico
guadalupe.gutierrez@upa.edu.mx
[2] Universidad Autónoma de Aguascalientes, Aguascalientes, Mexico
[3] Universidad Autónoma de Ciudad Juárez, Ciudad Juárez, Mexico

Abstract. Sentiment Analysis arises from areas such as natural language processing and data mining, it has become a key area for society because it is possible to identify emotions expressed in texts. Teacher evaluation is considered an important process in higher education institutions to measure teacher performance and implement constructive strategies to benefit students in their education. This paper describes the design and development of a Model with the purpose of analyzing the students reviews of teacher performance. The collection of comments was done in two ways, the first is that comments were collected from a teacher evaluation conducted in 2016 and the second was made on Twitter through the participation of students of the Universidad Politécnica de Aguascalientes. In this work we applied Support Vector Machines with three kernels: linear, radial and polynomial, to predict a classification of comments in positive, negative or neutral and we calculated evaluation measures.

Keywords: Reviews of teacher performance · Support Vector Machine Evaluation measures

1 Introduction

In the teaching process is crucial to evaluate the teaching performance [1]. This evaluation is one of the most complex processes in any university, since various factors and criteria like: planning of classes, schedules, delivery of evaluation evidence, attendance to courses for the improvement of teaching and teaching styles, among others, should be met to be concentrated in order to provide a final assessment for the professor. Teacher evaluation can be performed by an observation guide or a rubric. However, when teacher performance is evaluated by students, varied opinions are collected from the same established criteria. Therefore Education is one of the areas that in recent years has shown interest in analyzing the comments of students in order that teachers improve their teaching techniques promoting appropriate learning in students. This is possible by Sentiment Analysis [2] an application of natural language processing, text mining and computational linguistics, to identify information from the text.

C. Brito-Loeza and A. Espinosa-Romero (Eds.): ISICS 2018, CCIS 820, pp. 113–122, 2018.
https://doi.org/10.1007/978-3-319-76261-6_9

In his research, Binali [3] ensures that students represents their emotions in comments, so it is a way to learn about various aspects of the student. Wen [4] applies Sentiment Analysis on feedback from students about their teachers, enrolled in online courses in order to know their opinion and determine whether there is a connection between emotions and dropout rates. Students feedback on quality and standards of learning is considered as a strategy to improve the teaching process [4] and can be collected through a variety of social networks, blogs and surveys.

In this paper, we presented a model called SocialMining to support the Teacher Performance Assessment applying Support Vector Machines (SVM). We selected SVM as a classifier due to its high performance in classification applications [5, 6]. Further experiments with other machine learning algorithms will follow.

This paper is organized as follows. Section 2 presents related work. Section 3 shows the SocialMining model architecture. Section 4 describes data and methods used and experimental design. Section 5 includes the results. Finally, the conclusions of the work are presented in Sect. 6.

2 Related Work

The Table 1 shows an overview of some related work. All these works have obtained good results in their different combinations of methods and algorithms. This table is not exhaustive.

From Table 1 we can see that most of previous research has focused on particular aspects like: know the student emotional state, analyze the terms and phrases from opinions of students, detect the feelings of students on some topics and know the user opinions of the E-learning systems. In this work we proposed a model to evaluate teacher performance considering spanish reviews from students and applying machine learning algorithms to classify them as positive, negative and neutral. The results of this work may help to improve the classification process of comments and suggest courses to teachers.

Table 1. List of features to analyze

Reference	Description	Algorithms used	Dataset used
[7]	Development of an application to know the student emotional state	SVM, *Naïve Bayes, Complement Naïve Bayes*	Students reviews at the University of Portsmouth
[8]	Development of an application, called SentBuk to retrieve and identify sentiment polarity and emotional changes of users	Lexicon based, machine-learning and hybrid approaches	Around 1,000 comments in Spanish
[9]	Construction of a teaching evaluation sentiment lexicon, to analyze the terms and phrases from opinions of students	Lexicon in Thai, SVM, ID3 and Naïve Bayes	Reviews by students at Loei Raja hat University

(*continued*)

Table 1. (*continued*)

Reference	Description	Algorithms used	Dataset used
[10]	Design of an experimental study to predict teacher performance	Lexicon with 167 keywords positive and 108 keywords negative	1,148 feedbacks by students, obtained from RateMyProfessors.com
[11]	Proposed a method to detect the feeling of students on some topics and support the teacher to improve their teaching process	Latent Dirichlet Allocation, SVM, Naïve Bayes and, Maximum Entropy	Movie reviews dataset [12] and comments by students collected from Moodle
[13]	Proposed a system to help the developer and the educator to identify the most concentrated pages in E-learning portals	Bayesian classification, Naïve Bayesian and SVM	Use a collection of 100 reviews of users from the website Functionspace.org
[14]	Implement sentiment analysis in M-learning System, to know the user opinions of the M-learning system	*Naïve Bayes*, KNN and *Random Forest*	300 Reviews from www.market.android.com

3 SocialMining Model Architecture

The SocialMining model is composed of three phases: a comments extraction process (feedback from students about their teachers) and cleaning, a feature selection process, and classification of comments into positive, negative and neutral, applying SVM. The last phase includes an evaluation process of SVM results in kernels.

Phase 1: Comments Extraction and Cleaning Process. In this phase we extracted feedback from students about their teachers to generate a corpus of comments. Then we do a labeling process to classify the comments into positive and negative considering a numeric range. The numeric range varied from: −2 to −0.2 is used for negative comments, −2 value express very negative comments. Values between +0.2 to +2 apply to positive comments, +2 is used as a positive comments. Likewise, those comments labeled with the number 1 are considered as neutral (Fig. 1).

In this cleaning process, the stop words and nouns that appear in most of the comments are deleted (e.g. teacher, university, class, subject, school and others). In addition, punctuation marks are removed and capitalized words are converted to lowercase. The output in this phase is the corpus of comments.

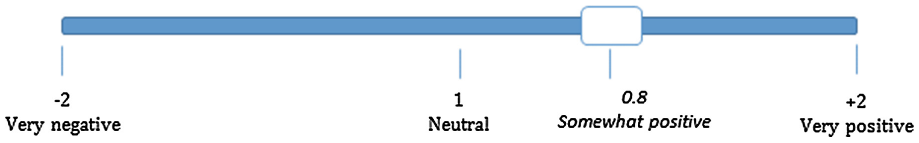


Fig. 1. Numeric range used to tagging process in comments

Phase 2: Feature Selection. Once finished the cleaning process, we performed a feature selection process, removing repetitive terms and applying some functions to select the required terms or features, this process is like a filtering. So the input in this phase is the corpus of comments and the output are the features.

A feature in Sentiment Analysis is a term or phrase that helps to express a positive or negative opinion. There are several methods used in feature selection, where some of them are based on the syntactic word position, based in information gain, using a importance variable calculated by genetic algorithms [15] and trees like the variable importance measures for random forest [16]. In this phase is necessary to know the importance of each feature, by their weight. So the Term Frequency - Inverse Document Frequency (TF-IDF) is applied (Fig. 2).

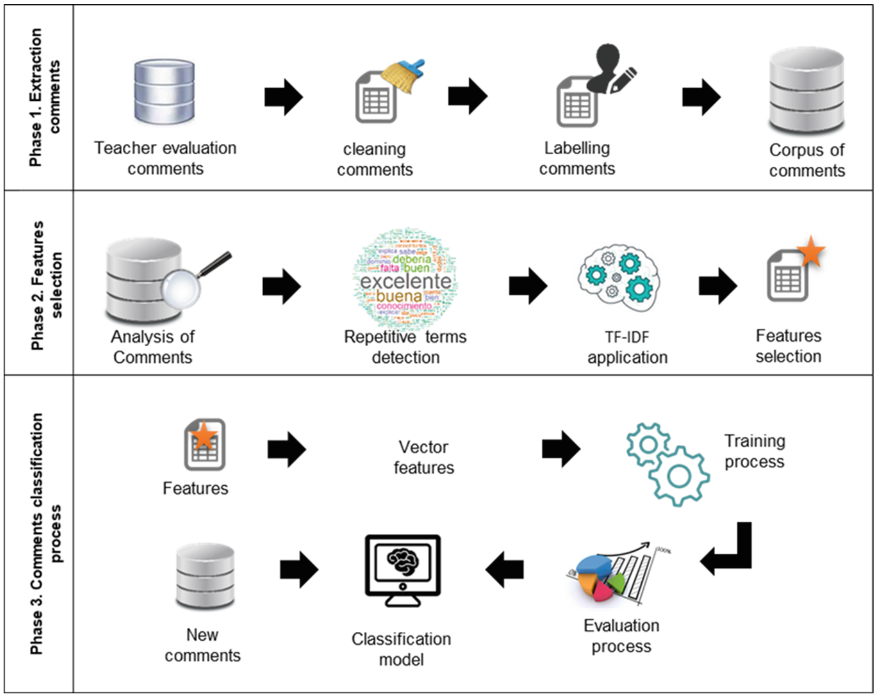


Fig. 2. SocialMining model architecture

Phase 3: Comments Classification Process. In this phase, the corpus of comments and features (matrixCF) is partitioned into two independent datasets. The first dataset is dedicated to training process (train) and is used in classification to find patterns or relationships among data; the second dataset is considered for the testing process (test) in order to adjust the model performance. In this work two thirds of the matrixCF are used for training dataset and one-third for test dataset. Then the cross-validation method of k iterations is applied to control the tuning and training of SVM. In this method matrixCF is divided into K subsets. One of the subsets is used as test data and the remaining (K−1) as training data. The cross-validation process is repeated for K iterations, with each of the possible subsets of test data, resulting in a confusion matrix with average values. Once the K iterations have been completed, cross validation accuracy is obtained. In this research, K is equal to 10.

The tuning process in SVM allows adjusting the parameters of each kernel (linear, radial basis and polynomial). Then is performed a training process, through which is identified whether the value of the parameters vary or remain constant.

Finally, the implementation of SVM is performed presenting as a result the confusion matrix and accuracy values as well as the metric Receiver operating characteristic (ROC) curve.

4 Materials and Methods

4.1 Data

The dataset used in this work comprises 1040 comments in Spanish of three groups of systems engineering students at Universidad Politécnica de Aguascalientes. They evaluated 21 teachers in the first scholar grade (2016). For this study we considered only those comments free from noise or spam (characterized in this study as texts with strange characters, empty spaces, no opinion or comments unrelated to teacher evaluation). In this work we identified a set of 99 features. An extract of the features are listed in Table 2.

Table 2. List of features

Feature (Spanish)	Feature	Polarity feature	Feature	Feature (Spanish)	Polarity feature
Atenta	Attentive	Positive	Debería	Should	Negative
Agrada	Likes	Positive	Debe	Must	Negative
Apoya	Supports	Positive	Impuntual	Impuntual	Negative
Aprender	Learn	Positive	Elitista	Elitist	Negative
Bien	Fine	Positive	Impaciente	Impatient	Negative
Imparcial	Impatial	Positive	Problemático	Problematic	Negative

4.2 Performance Measures

We used typical performance measures in machine learning such as:

- Accuracy, primary measure to evaluate the performance of a predictive model.
- Balanced accuracy, a better estimate of a classifier performance when a unequal distribution.
- Sensitivity, which measures the proportion of true positives.
- Specificity, measures the proportion of true negatives.
- ROC curve, measures the performance of a classifier through graphical representation [17, 18].

4.3 Classifiers

SVM is an algorithm introduced by Vapnik [19] for the classification of both linear and nonlinear data, it has been known for its quality in text classification [20]. There are kernels that can be used in SVM, such as: linear, polynomial, radial basis function (RBF) and sigmoid. Each of these kernels has particular parameters and they must be tuned in order to achieve the best performance. In this work we selected the first three kernels to classify comments; this is mainly because of their good performance in text classification [5, 6]. Table 3 shows the parameters of each kernel used in this study.

Table 3. Kernel parameters.

Kernel	Parameter
Linear	C
Radial basis	C, sigma
Polynomial	C, degree

- C is the parameter for the soft margin cost function, it determines a tradeoff between a wide margin and classifier error. A very small value of C cause a larger margin separating hyperplane and the model get fit tighter to data, however a large value of C reduce the margin and this may cause more error on the training set.
- Sigma determines the width for Gaussian distribution in Radial basis kernel.
- Degree control the flexibility of the resulting classifier in Polynomial Kernel.

4.4 Experimental Design

We created a dataset containing 1040 comments and 99 features associated with teacher performance assessment. We used train-test evaluation, two-thirds (2/3) for training, and (1/3) one-third for testing, then there were performed 30 runs applying SVM with polynomial, radial basis function (RBF) and linear kernel. For each run performance measures are computed. In each run we set a different seed to ensure different splits of training and testing sets, all kernels use the same seed at the same run.

Each kernel requires tuning different parameters (see Table 3). A simple and effective method of tuning parameters of SVM has been proposed by Hsu [21], the grid search. The C values used for the kernels, range from 0.1 to 2, the value of sigma (σ) varied from 0.01 to 2, the degree value parameter range from 2 to 10, and values between 0 and 1 are assigned for coef parameter. We performed 30 train-test runs using different seeds and calculated the accuracy and balanced accuracy for each run.

5 Discussion and Results

In this section, we present the results with three kernels in SVM. The first step is to determine the parameters of each kernel of SVM, so we first load the data and create a partition of corpus of comments, then divided it into training and testing datasets, then use a train control in R [22] to set the training method. We use the Hsu [21] methodology to specify the search space in each kernel parameter. ROC is the performance criterion used to select the optimal kernels parameters of SVM.

Setting the seed to 1 in the process of optimization parameters, we generated paired samples according to Hothorn [23] and compare models using a resampling technique. Table 4 shows the summary of resampling results using R [22], the performance metrics are: ROC, sensibility and specificity. In the Fig. 3 we can see the plot of summary resampling results, in this case, the polynomial kernel apparently has a better performance than linear and RBF (radial).

Table 4. Summary resampling results of parameters optimization

ROC						
Kernel	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Linear	0.7872	0.8315	0.8616	0.8579	0.8714	0.9080
Radial	0.7881	0.8218	0.8462	0.8462	0.8635	0.9176
Poly	0.8350	0.8608	0.8773	0.8755	0.8918	0.9321
Sensibility						
Kernel	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Linear	0.5857	0.6571	0.7	0.7006	0.7571	0.8143
Radial	0.6571	0.7286	0.7571	0.7663	0.8	0.9
Poly	0.3429	0.7	0.7571	0.7131	0.7714	0.8143
Specificity						
Kernel	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Linear	0.7971	0.8696	0.8971	0.889	0.9118	0.9565
Radial	0.6812	0.7681	0.7971	0.7925	0.8261	0.8696
Poly	0.8088	0.8529	0.8841	0.8849	0.913	0.9855

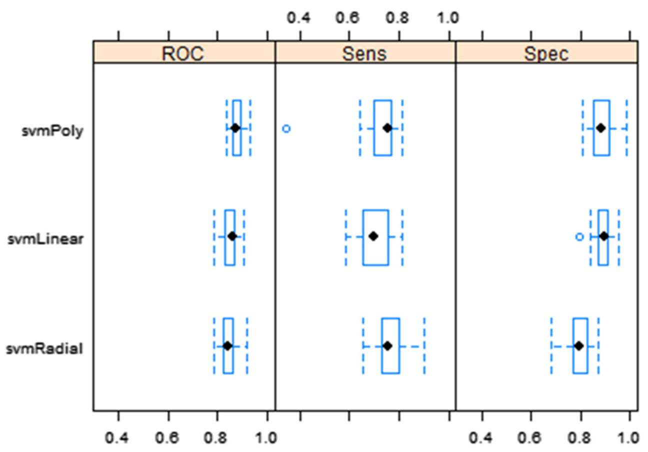


Fig. 3. Summary resampling results of kernels performance

Once obtained optimized parameters for each kernel, the execution of each SVM model is performed.

The Table 5 shows the average results of each kernel of SVM across 30 runs. Also the standard deviation of each metric is presented.

Table 5. Average results across 30 runs in three kernels of SVM

	Accuracy	Balanced accuracy	Sensitivity	Specificity
SVM Linear	**0.8038**	**0.8149**	**0.8936**	**0.7160**
	0.0153	0.0146	0.0277	0.0364
SVM Radial	**0.7850**	**0.7893**	**0.8242**	**0.7467**
	0.0190	0.0159	0.0422	0.0561
SVM Poly	**0.6779**	**0.7014**	**0.8661**	**0.4941**
	0.0363	0.1336	0.1649	0.1009

The linear kernel obtained a balanced accuracy above 0.80, this is an indicator that the classifier is feasible to use in comments classification. Values obtained in Sensitivity were much higher than those obtained in specificity in all kernels, which indicates that the classifier can detect the negative comments of the teachers. The kernel polynomial (SVM Poly) had the lowest performance in all metrics except in sensitivity. The three kernels resulted more sensitive than specific.

6 Conclusions

In computer science is attractive the use of this type of machine learnings models to automate processes, save time and contribute to decision-making. The SocialMining model supports the analysis of the behavior from unstructured data provided by students. The sentiment analysis is based on the analysis of texts and the SocialMining can provide a feasible solution to the problem of analysis of teacher evaluation comments. Further experiments will be conducted in this ongoing research project.

It is important to point out that is necessary reduce the number of features through a depth analysis to identify the most relevant features of teacher performance assessment, in order to improve the results of comments classification process. Also we considered important having a corpus of balanced comments (positive and negative comments in equal quantity) for testing and training process.

In addition to conduct a deeper analysis for relevant features selection, we considered necessary to implement other machine learning algorithms in order to measure the performance of each algorithm in the classification of comments and select the optimal with high accuracy results.

Based on the adequate results that have been obtained by the SocialMining model applying Naïve Bayes and a corpus of subjectivity [24], we considered that with the implementation of other algorithms of machine learning well-known for their good performance in classification process.

About how SocialMining model support the improvement of teaching in the first instance it allows a quicker analysis of student comments, identifying which teachers have mostly negative comments which allows interventions with the teacher in order to support it through teacher improvement courses. Each school period, courses are offered to teachers, however the comments of students are not considered among the criteria to recommend a certain course to the teacher. For this reason it is believed that the Model presented in this work will support the improvement of teaching.

References

1. Careaga, A.: La evaluación como herramienta de transformación de la práctica docente. Educere **5**(15), 345–352 (2001)
2. Bing, L.: Sentiment Analysis and Opinion Mining (Synthesis Lectures on Human Language Technologies). Morgan & Claypool Publishers, San Rafael (2012)
3. Binali, H.H., Wu, C., Potdar, V.: A new significant area: emotion detection in E-learning using opinion mining techniques. In: 3rd IEEE International Conference on Digital Ecosystems and Technologies (DEST 2009). IEEE (2009)
4. Wen, M., Yang, D., Penstein Rosé, C.: Sentiment analysis in MOOC discussion forums: what does it tell us? In: Proceedings of Educational Data Mining, pp. 1–8 (2014). http://goo.gl/fViyBH
5. Altrabsheh, N., Cocea, M., Fallahkhair, S.: Learning sentiment from students' feedback for real-time interventions in classrooms. In: Bouchachia, A. (ed.) Adaptive and Intelligent Systems. LNCS, vol. 8779, pp. 40–49. Springer, Cham (2014). http://dx.doi.org/10.1007/978-3-319-11298-5_5
6. Manning, C.D., Raghavan, P., Schütze, H.: Support vector machines and machine learning on documents. In: Introduction to Information Retrieval, pp. 319–348 (1998)
7. Sarkar, A., et al.: Text Classification using Support Vector Machine (2015)
8. Ortigosa, A., Martín, J., Carro, R.: Sentiment analysis in Facebook and its application to e-learning. Comput. Hum. Behav. **31**, 527–541 (2014). http://dx.doi.org/10.1016/j.chb.2013.05.024

9. Pong-Inwong, C.R., Rungworawut, W.S.: Teaching senti-lexicon for automated sentiment polarity definition in teaching evaluation. In: 10th International Conference on Semantics, Knowledge and Grids (SKG), Beijing, pp. 84–91. IEEE (2014)
10. Kaewyong, P., Sukprasert, A., Salim, N., Phang, A.: The possibility of students' comments automatic interpret using lexicon based sentiment analysis to teacher evaluation. In: The 3rd International Conference on Artificial Intelligence and Computer Science 2015, Penang, Malaysia (2015)
11. Francesco, C., de Santo, M., Greco, L.: SAFE: a sentiment analysis framework for e-learning. Int. J. Emerg. Technol. Learn. **9**(6), 37 (2014)
12. Pang, B., Lee, L.: Thumbs up? Sentiment classification using machine learning techniques. In: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, vol. 10, pp. 79–86 (2002). https://doi.org/10.3115/1118693.1118704
13. Bharathisindhu, P., Brunda, S.: Identifying e-learner's opinion using automated sentiment analysis in e-learning. Int. J. Res. Eng. Technol. **3**(1) (2014). http://dx.doi.org/10.15623/ijret.2014.0319086
14. Kirubakaran, E.: M-learning sentiment analysis with data mining techniques. Int. J. Comput. Sci. Telecommun. (2012)
15. Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques, 3rd edn. Morgan Kaufmann (2011)
16. Punch III, W.F., et al.: Further research on feature selection and classification using genetic algorithms. In: ICGA, pp. 557–564 (1993)
17. Strobl, C., et al.: Conditional variable importance for random forests. BMC Bioinform. **9**(1), 307 (2008)
18. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco (2012)
19. Vapnick, V.: Statistical Learning Theory. Wiley, New York (1998)
20. Esuli, A., Sebastiani, F., SENTIWORDNET: a publicly available lexical resource for opinion mining. In: Proceedings of the 5th Conference on Language Resources and Evaluation (LREC 2006), Genoa, Italy, pp. 417–422 (2006)
21. Hsu, C.-W., Chang, C.-C., Lin, C.-J.: A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University (2003)
22. R-Core-Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2013). http://goo.gl/e40yiU
23. Hothorn, T., Leisch, F., Hornik, K., Zeileis, A.: The design and analysis of benchmark experiments. J. Comput. Graph. Stat. **14**(3), 675–699 (2005)
24. Gutiérrez, G., Padilla, A., Canul-Reich, J., De-Luna, A., Ponce, J.: Proposal of a sentiment analysis model in tweets for improvement of the teaching - learning process in the classroom using a corpus of subjectivity. Int. J. Comb. Optim. Probl. Inform. **7**(2), 22–34 (2016)

A Selective Segmentation Model for Inhomogeneous Images

Lavdie Rada[1](✉), Haider Ali[2], and Haroon Niaz Ali Khan[2]

[1] Department of Biomedical Engineering, BAU, Istanbul, Turkey
lavdie.rada@eng.bau.edu.tr

[2] Department of Mathematics, University of Peshawar, Peshawar, Pakistan
{dr.haider,haroonniaz}@uop.edu.pk

Abstract. Automatic accurate detection and segmentation of all of the boundaries in a given image has been of an interest in the last decades. In contrast with global segmentation, where all the object boundaries in a given image are detected, for a large class of image segmentation tasks, only one object is required to be extracted. To successfully segment one single object, interactive/selective segmentation techniques has been delivered. However, the existing interactive/selective models cannot cope with images that have intensity inhomogeneity and presence of noise. In this paper, we propose a new variational level-set model which can deal with intensity inhomogeneity and presence of noise for the selective segmentation task. To evaluate the performance of our new model, we compare our results with the latest and state of the art models. Comparison of the proposed model with the latest and state of the art models show same efficiency and reliability on detecting objects/regions for homogeneous intensity images and outperforms when applied to inhomogeneous images with or without noise.

Keywords: Image processing · Image selective segmentation
Total variation · Level set function · Edge detection
2D image segmentation

1 Introduction

As the number of photography and imaging instruments increased in the last decades it provided a lot of valuable information in different fields. On the other hand, it increased the requirements on developing automatic post processes to save human labor hours. Segmentation, which separates the objects from their surroundings, is the most common post-processing task. Region based variational segmentation techniques [1–3] prove to be very efficient for extracting homogeneous areas and segment the boundaries of the object in a given image compared with other models such as histogram analysis and thresholding [4–6], region growing [7,8], edge detection and active contours [9,10], etc. On the other hand, statistical methods [11,12], etc., are known for their ability in segmenting inhomogeneous images. Both categories of segmentation models mentioned

C. Brito-Loeza and A. Espinosa-Romero (Eds.): ISICS 2018, CCIS 820, pp. 123–136, 2018.
https://doi.org/10.1007/978-3-319-76261-6_10

above do segment the boundaries of all the objects, in other words a global segmentation, where all features are to be segmented.

On the other hand, in many segmentation problems, we need to segment a particular object of interest in the image and not all objects in it. This is a task of selective segmentation in which an aimed target object of interest is detected, given some additional information of location or geometric constraints. The hardest case for a selective segmentation would be if the objects are near, the intensity difference is small, or the object is inhomogeneous with background and foreground intensities varying in the same grey-level. To solve the selective segmentation problem different techniques such as random walks [13], geodesics [14], graph cut [15], have been exploited. Those technique use the distributions probability, edge based function or graph cut theory, respectively. Lastly, Nguyen et al. [16] introduced a continuous-domain convex active contour model. Geometric points outside and inside the objects directs the geometrical constraints for a successful segmentation in a combination with Split Bregman method [17] for a fast convergence. This method competes with all the previous methods. Even though the Nguyen et al. [16] method is a state of art, it cannot handle transparent or semi-transparent boundaries and the method works properly under the assumption that the object is smooth and can be well described by the weighed shortest boundary length. An alternative to the above mentioned methods are variational based methods. The first variational based method was introduced in the work of Gout-Guyader-Vese [18,19]. The method is a variational edge based method. This method was further improved in a combined edge based and region information introduced by [20]. Lately, in the work of [21], a variation model improves the models [18–20] and computes the state of art introduced by Nguyen et al. [16].

In this paper we design a variational selective segmentation model, which ensures same performance with state of arts [16,21] for homogeneous intensity images and is capable to cope with inhomogeneous and noisy images. The new method is based on (1) an adaptive parameter edge detection function (2) a fidelity term which helps with for inhomogeneous images with or without noise (3) an area-based minimization fitting term considered to enhance the model's reliability (area fitting term serves as a constraint rather than precise area preserving) (4) a distance function from the given geometrical points which can be used only if needed when it is feasible to give an accurate estimate of the object of interest.

This paper is organized in the following way. Section 2 contains a review of some related works. In Sect. 3 we present our proposed new model of minimization and derive the Euler-Lagrange equations. We describe the discretization of the method and develop an additive operator splitting (AOS) algorithm for solving the PDE, which is very efficient for this kind of problem. In Sect. 4 we present some experimental results on different data-sets. To demonstrate the effectiveness of our proposed method a comparison with the states of art method introduced by Nguyen et al. [16] and with the variational model proposed by Rada-Chen [21] has been shown. We conclude the paper in Sect. 5.

2 Review of Existing Variational Selective Segmentation Models

The segmentation problem, is a problem of partitioning a digital image $u_0(\mathbf{x})$ defined on a rectangular domain Ω into multiple segments. If the aim of the segmentation is to segment all the objects in a given image we call it a global segmentation, and we will denote the contour of this segmentation as Γ_G. In the case of selective segmentation, a Γ contour will separate an aimed object from the surrounding. One of the most well known global segmentation variational model was introduced by Chan-Vese (CV) [1]. The CV model restricts the piecewise constant Mumford-Shah model [3] to only two phases, representing the foreground and the background of the image. As the CV model is not a gradient based model, it can detect contours with and without gradients, and deals satisfactory with or without noise images. The energy minimization functional for the CV active contour model is given by:

$$\min_{c_1,c_2,\Gamma} F(\Gamma_G,c_1,c_2) = \min_{c_1,c_2,\Gamma_G} \Big\{\mu \text{length}(\Gamma_G) + \lambda_1 \int_{\text{in}(\Gamma_G)} |u_0(\mathbf{x}) - c_1|^2 d\mathbf{x} + \lambda_2 \int_{\text{out}(\Gamma_G)} |u_0(\mathbf{x}) - c_2|^2 d\mathbf{x}\Big\} \quad (1)$$

where c_1 and c_2 are the average values of $u_0(\mathbf{x})$ inside and outside of the variable contour Γ_G, denoted as $\text{in}(\Gamma_G)$ and $\text{out}(\Gamma_G)$, respectively. Also, μ, λ_1 and λ_2 are non-negative fixed parameters. To solve the above minimization problem a level set function $\phi(\mathbf{x})$ has been introduced [22,23], and the above Eq. (1) is rewritten as:

$$\min_{\phi(\mathbf{x}),c_1,c_2} F(\phi(\mathbf{x}),c_1,c_2) = min_{\phi(\mathbf{x}),c_1,c_2}\Big\{\mu \int_\Omega |\nabla H(\phi(\mathbf{x}))| d\mathbf{x} + \lambda_1 \int_\Omega |u_0(\mathbf{x}) - c_1|^2 H(\phi(\mathbf{x})) d\mathbf{x} + \lambda_2 \int_\Omega |u_0(\mathbf{x}) - c_2|^2 (1 - H(\phi(\mathbf{x}))) d\mathbf{x}\Big\}, \quad (2)$$

where H is the Heaviside function. Since the Heaviside function is not differentiable at the origin, the Heaviside function H and its corresponding delta function $\delta = H'$, is replaced with a regularized function denoted by H_ϵ and δ_ϵ, respectively. Different regularized Heaviside functions can be considered [1,24], e.g. $H_\epsilon = \frac{1}{2}(1 + \frac{2}{\pi}\arctan(\frac{z}{\epsilon}))$; $\delta_\epsilon(z) = \epsilon/\{\pi\Big(\epsilon^2 + z^2\Big)\}$. Rewriting (2) in terms H_ϵ we have:

$$\min_{\phi(\mathbf{x}),c_1,c_2} F_\epsilon(\phi(\mathbf{x}),c_1,c_2) = \min_{\phi(\mathbf{x}),c_1,c_2} \Big\{\mu \int_\Omega \delta_\epsilon(\phi(\mathbf{x})) |\nabla \phi(\mathbf{x})| d\mathbf{x} + \lambda_1 \int_\Omega |u_0(\mathbf{x}) - c_1|^2 H_\epsilon(\phi(\mathbf{x})) d\mathbf{x} + \lambda_2 \int_\Omega |u_0(\mathbf{x}) - c_2|^2 (1 - H_\epsilon(\phi(\mathbf{x}))) d\mathbf{x}\Big\}. \quad (3)$$

Keeping $\phi(\mathbf{x})$ fixed and minimizing with respect to c_1 and c_2, one gets

$$c_1 = \frac{\int_\Omega u_0(\mathbf{x}) H_\epsilon(\phi(\mathbf{x})) d\mathbf{x}}{\int_\Omega H_\epsilon(\phi(\mathbf{x})) d\mathbf{x}}; c_2 = \frac{\int_\Omega u_0(\mathbf{x})(1 - H_\epsilon(\phi(\mathbf{x}))) d\mathbf{x}}{\int_\Omega (1 - H_\epsilon(\phi(\mathbf{x}))) d\mathbf{x}} \quad (4)$$

if $\int_\Omega H_\epsilon(\phi(\mathbf{x}))d\mathbf{x} > 0$ and $\int_\Omega(1 - H_\epsilon(\phi(\mathbf{x})))d\mathbf{x} > 0$. Finally keeping c_1 and c_2 fixed, one can minimize (3) with respect to $\phi(\mathbf{x})$. Thus we have the following Euler-Lagrange equation for $\phi(\mathbf{x})$

$$\delta_\epsilon(\phi(\mathbf{x}))\Big[\mu\nabla\cdot\Big(\frac{\nabla\phi(\mathbf{x})}{|\nabla\phi(\mathbf{x})|}\Big) - \Big(\lambda_1(u_0(\mathbf{x}) - c_1)^2 - \lambda_2(u_0(\mathbf{x}) - c_2)^2\Big)\Big] = 0,$$

in Ω, with boundary condition $\frac{\partial\phi}{\partial \boldsymbol{n}}\Big|_{\partial\Omega} = 0$.

To improve the performance of CV model for inhomogeneous images several models have been introduced in the last years. In the work of Li et al. [25], a local binary fitting (LBF) functional has been introduced:

$$F^{LBF}(c, f_1, f_2) = \int_\Omega |\nabla H(\phi(\mathbf{x}))|d(\mathbf{x}) + \lambda_1\int_\Omega K_\sigma(\mathbf{x}-\mathbf{y})|u_0(\mathbf{y}) - f_1(\mathbf{x})|^2 H(\phi(\mathbf{y}))d\mathbf{y}$$
$$+ \lambda_2\int_\Omega K_\sigma(\mathbf{x}-\mathbf{y})|u_0(\mathbf{y}) - f_2(\mathbf{x})|^2(1 - H(\phi(\mathbf{x})))d\mathbf{y}, \qquad (5)$$

where $K_\sigma(\mathbf{x})$ is a Gaussian Kernel of the form $K_\sigma(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}\sigma^2}\cdot e^{-|\mathbf{x}|^2/2\sigma^2}$, with a scale parameter $\sigma > 0$. It should be emphasized that the functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$, that minimize the energy (5), vary with the center point $\mathbf{x}$, giving in this way essential local information, which makes the LBF model different from the CV model. Minimizing Eq. (5) respect $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ we have as follows:

$$f_1(\mathbf{x}) = \frac{K_\sigma(\mathbf{x}) * H_\varepsilon(\phi(\mathbf{x}))u_0(\mathbf{x})}{K_\sigma(\mathbf{x}) * H_\varepsilon(\phi(\mathbf{x}))}; \qquad f_2(\mathbf{x}) = \frac{K_\sigma(\mathbf{x}) * (1 - H_\varepsilon(\phi(\mathbf{x})))u_0(\mathbf{x})}{K_\sigma(\mathbf{x}) * (1 - H_\varepsilon(\phi(\mathbf{x})))}. \qquad (6)$$

Keeping $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ fixed and minimizing the above energy functional with respect to $\phi(\mathbf{x})$, we get the following equation,

$$\frac{\partial\phi(\mathbf{x})}{\partial t} = -\delta_\epsilon(\phi(\mathbf{x}))\left\{(\lambda_1 e_1(\mathbf{x}) - \lambda_2 e_2(\mathbf{x})) + \mu\nabla\cdot(\frac{\nabla\phi(\mathbf{x})}{|\nabla\phi(\mathbf{x})|})\right\}$$

where e_1 and e_2 are given by:

$$e_1(\mathbf{x}) = \int_\Omega K_\sigma(\mathbf{x}-\mathbf{y})|u_0(\mathbf{x}) - f_1|^2 d\mathbf{y}; \qquad e_2(\mathbf{x}) = \int_\Omega K_\sigma(\mathbf{x}-\mathbf{y})|u_0(\mathbf{x}) - f_2|^2 d\mathbf{y}.$$

Even though, the fitting term empowers LBF model for intensity inhomogeneity images, the model struggles in presence of noise. In contrast with global segmentation, the aim of the selective segmentation is to find an optimal contour $\Gamma \subset \Omega$ near the given set $\mathcal{A}$, which represents an object with minimal geometric distance from the set $\mathcal{A}$, and Γ a subset of Γ_G. Lastly, in the paper introduced by Rada-Chen [21], a combination of an edge based function, distance weight and area fitting term, improves previous selective segmentation models of Gut-Guyader-Vese [18,19] and Badshah-Chen [20], as well as the highly effective and outperforms state-of-the-art interactive image segmentation algorithm introduced by Nguyen et al. [16].

Given image $u_0(\mathbf{x})$, defined on the rectangular domain Ω, the aim of selective segmentation is to detect an image feature/object that is close to the geometrical domain of the points $\mathcal{A} = \{\mathbf{x}_i^* \in \Omega, \quad 1 \le i \le n_1\} \subset \Omega$, which are placed inside the object or in the boundaries. The given set of points indicates the level set initialization and at the same time an indicator of the relative size of the object in the area term. Denoting c_1 the mean intensity of the polygon constructed with the set $\mathcal{A} =$ the energy minimization functional considered at [21] is given as follows:

$$\min_{\Gamma, c_2} F(\Gamma, c_2) = \min_{\Gamma, c_2} \Big\{\mu \int_\Gamma d(\mathbf{x}) g(|\nabla u_0(\mathbf{x})|) d\mathbf{x} + \lambda_1 \int_{in(\Gamma)} |u_0(\mathbf{x}) - c_1|^2 d\mathbf{x}$$
$$+ \lambda_2 \int_{out(\Gamma)} |u_0(\mathbf{x}) - c_2|^2 d\mathbf{x} + \nu \Big[\Big(\int_{in(\Gamma)} d\mathbf{x} - A_1\Big)^2 + \Big(\int_{out(\Gamma)} d\mathbf{x} - A_2\Big)^2\Big]\Big\}, \tag{7}$$

where $\lambda_1, \lambda_2, \mu, \nu$ and are empirical weights, g is the edge detector function $g(|\nabla u_0(\mathbf{x})|) = \frac{1}{1+k|\nabla u_0(\mathbf{x})|^2}$, with k a positive constant, $d(\mathbf{x}) =$ distance$((\mathbf{x}), \mathcal{A}) = \prod_{i=1}^{n_1} \left(1 - e^{-\dfrac{(\mathbf{x}-\mathbf{x}_i^*)^2}{2\tau^2}}\right)$ a distance function which helps to stop the evolving curve when approaching the points from set $\mathcal{A}$, c_1 is the known mean of the polygon constructed with the given markers (with the supposition that the markers are placed inside the object or not too far the boundaries), c_2 is a region term defined below in Eq. (9) representing the mean intensity outside the target object. In the above energy equation we can notice that the first term (weighted by μ) is the regularizer of the inverse problem and expresses the weighted geodesic length of the contour, the second and the third term (weighted by λ_1, λ_2) are region fitting terms to the mean intensity inside the object and outside, respectively, while the forth and fifth term (weighted by ν) are a priori terms stating that the volume area of each object remains close to a reference area (or volume for 3-D) $A_i, \ i = 1, 2$. Here A_i is the area of the polygon inside and outside the given markers. The distance function d acts *locally* and will be approximately 0 in the neighborhood of points of $\mathcal{A}$. To have a good influence from this function we need to increase the number of points in the set $\mathcal{A}$ and for more they have to be near the boundaries, which is not practically convenient. For this reason the distance function is applied only if needed when it is feasible to give an accurate estimate of the object of interest. In this way we do not fail in cases where inaccurate geometrical information is given.

Rewriting the above equation in terms of the level-set function with a regularized Heaviside function, similar to [1,9], we have:

$$\min_{\phi(\mathbf{x}), c_2} F_\epsilon(\phi(\mathbf{x}), c_2) = min\Big\{\mu \int_\Omega d(\mathbf{x}) g(|\nabla u_0(\mathbf{x})|) \delta_\epsilon(\phi(\mathbf{x})) |\nabla(\phi(\mathbf{x}))| d\mathbf{x}$$
$$+ \lambda_1 \int_\Omega |u_0(\mathbf{x}) - c_1|^2 H_\epsilon(\phi(\mathbf{x})) d\mathbf{x} \ + \lambda_2 \int_\Omega |u_0(\mathbf{x}) - c_2|^2 (1 - H_\epsilon(\phi(\mathbf{x}))) d\mathbf{x} \tag{8}$$
$$+ \nu \Big[\Big(\int_\Omega H_\epsilon(\phi(\mathbf{x})) d\mathbf{x} - A_1\Big)^2 + \Big(\int_\Omega (1 - H_\epsilon(\phi(\mathbf{x})) d\mathbf{x}) - A_2\Big)^2\Big] d\mathbf{x}\Big\},$$

where $\delta_\epsilon(\phi(\mathbf{x}))$ is a regularized Delta function corresponding to the Heaviside function introduced above. Keeping $\phi(\mathbf{x})$ fixed and minimizing with respect to the unknown intensity outside the object, one gets the following equations for computing c_2:

$$c_2(\phi(x,y)) = \frac{\int_\Omega u_0(x,y)(1 - H_\epsilon(\phi(x,y)))dxdy}{\int_\Omega (1 - H_\epsilon(\phi(x,y)))dxdy} \tag{9}$$

if $\int_\Omega (1 - H_\epsilon(\phi(x,y)))dxdy > 0$ (i.e. if the curve is nonempty in Ω). In case of a reliable non-noisy homogeneous image c_1 can be updated

$$c_1(\phi(\mathbf{x})) = \frac{\int_\Omega u_0(\mathbf{x})H_\epsilon(\phi(\mathbf{x}))d\mathbf{x}}{\int_\Omega H_\epsilon(\phi(\mathbf{x}))d\mathbf{x}} \tag{10}$$

given $\int_\Omega H_\epsilon(\phi(\mathbf{x}))d\mathbf{x} > 0$. In case of unreliability, due to noise or non-homogeneity, the mean intensity inside the polygon is a more reliable quantity. Keeping c_1 and c_2 fixed and denoting $W = dg(|\nabla u_0|)$, we minimize (8) with respect to $\phi(x,y)$ and get the following Euler-Lagrange equation:

$$\begin{aligned} &\delta_\epsilon(\phi)\Big\{\mu\nabla\cdot\left(W\frac{\nabla\phi}{|\nabla\phi|}\right) - \left[\lambda_1(u_0(x,y)-c_1)^2 - \lambda_2(u_0(x,y)-c_2)^2\right] \\ &-\nu\Big[\Big(\int_\Omega H dxdy - A_1\Big) - \Big(\int_\Omega (1-H)dxdy - A_2\Big)\Big]\Big\} = 0 \quad \text{in } \Omega, \end{aligned} \tag{11}$$

where $\left.\frac{\partial\phi}{\partial\boldsymbol{n}}\right|_{\partial\Omega} = 0$.

3 Proposed Method

In this section, we shall present details of our proposed model and its numerical implementation. The energy functional of the proposed model consists of three parts: global term to drive the evolving curve(s) towards the true boundaries, local regularization term which consist of an intensity and surface (volume) driven information, and a locally computed denoising constrained surface and a denoising fidelity term to overcome the inhomogeneous intensity distribution. This combination provides more satisfying selective segmentation result for inhomogeneous images in comparison with [21] or [16]. The last term contributing to the minimisation functional has been motivated by the work of [26], originally designed for image denoising. The Aubert et al. [26] introduces a statistical model for recovering a clean image $u(\mathbf{x})$ given a corrupted image $u_0(\mathbf{x}) = u(\mathbf{x})\eta(\mathbf{x})$, where $\eta(\mathbf{x})$ is some multiplicative noise. The model minimizes the following energy function

$$F^{AA}(u(\mathbf{x})) = \int_\Omega |\nabla u(\mathbf{x})|d\mathbf{x} + \lambda\int_\Omega \Big(\log(u(\mathbf{x})) + \frac{u_0(\mathbf{x})}{u(\mathbf{x})}\Big)d\mathbf{x}, \tag{12}$$

and is known for his ability to compete all the previous denoising models. We design our variational model by recalling Eq. (8) and enriching this equation with a fidelity term which helps for inhomogeneous images with or without noise. Given an image $u_0(\mathbf{x})$ over a domain Ω we can consider its statistically locally computed approximated image $u(\mathbf{x})$ in the following form:

$$u(\mathbf{x}) = f_1(\mathbf{x}) H_\epsilon(\phi(\mathbf{x})) + f_2(\mathbf{x})\Big(1 - H_\epsilon(\phi(\mathbf{x}))\Big), \tag{13}$$

where $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are spatially varying averages, given in Eq. (6), used to preserve image local information. This representation of the given image $u_0(\mathbf{x})$ will reflect more local information in comparison with piecewise constant and piecewise smooth approximations models [2,25,27]. In order to distinguish between noise and valuable image fine detail, we will add a coupled denoising term. This constraint will ensure accurate recovery of $u(\mathbf{x})$ in additive noise, multiplicative noise and mix of them. Thus we do minimise the following function:

$$\begin{aligned}\min_{\Gamma, c_2} F(\Gamma, c_2) = \min_{\Gamma, c_2} \Big\{ \mu \int_\Gamma W(\mathbf{x}) d\mathbf{x} + \lambda_1 \int_{in(\Gamma)} |u_0(\mathbf{x}) - c_1|^2 d\mathbf{x} \\ + \lambda_2 \int_{out(\Gamma)} |u_0(\mathbf{x}) - c_2|^2 d\mathbf{x} + \nu \Big[\Big(\int_{in(\Gamma)} d\mathbf{x} - A_1 \Big)^2 + \Big(\int_{out(\Gamma)} d\mathbf{x} - A_2 \Big)^2 \Big] \\ + (1-\theta) \int_\Omega \Big(u(\mathbf{x}) - u_0(\mathbf{x}) \Big)^2 d\mathbf{x} + \theta \int_\Omega \Big(\log u(\mathbf{x}) + \frac{u_0(\mathbf{x})}{u(\mathbf{x})} \Big) d\mathbf{x} \Big\},\end{aligned} \tag{14}$$

The parameters μ, λ_1, λ_2, ν, and θ are fixed constants where as A_1, A_2, c_1, c_2, are computed as in (8). Minimizing the above energy functional with respect to $\phi(\mathbf{x})$, we get the following gradient descent flow:

$$\begin{aligned}\delta_\epsilon(\phi) \Big\{ \mu \nabla \cdot \Big(W(\mathbf{x}) \frac{\nabla \phi}{|\nabla \phi|} \Big) - \Big[\lambda_1 (u_0(\mathbf{x}) - c_1)^2 - \lambda_2 (u_0(\mathbf{x}) - c_2)^2 \Big] \\ - \nu \Big[\Big(\int_\Omega H d\mathbf{x} - A_1 \Big) - \Big(\int_\Omega (1 - H) d\mathbf{x} - A_2 \Big) \Big] \\ - D \Big(2(1-\theta)(u(\mathbf{x}) - u_0(\mathbf{x})) + \theta \frac{u(\mathbf{x}) - u_0(\mathbf{x})}{u(\mathbf{x})^2} \Big) \Big] \Big\} = 0.\end{aligned} \tag{15}$$

where $D = (f_1(\mathbf{x}) - f_2(\mathbf{x}))$. In order to solving Eq. (15), a fast and a low computational cost method is required. The additive operator splitting (AOS) method, proposed by Tai et al. [28] and Weickert [29] is widely applied to a diffusion equation, see [19,20,30]. This method allows the decomposition of the two-dimensional problem into two one-dimensional ones. To implement this algorithm in our method we have to make the discretization of the Eq. (15), form a semi-implicit linear system and develop the iterative approximation scheme which solves a diagonally dominant tridiagonal linear system.

We first recall the evolution equation satisfied by:

$$\begin{aligned}\frac{\partial \phi}{\partial t} = \mu \delta_\epsilon(\phi) \nabla \cdot \Big(W(\mathbf{x}) \frac{\nabla \phi}{|\nabla \phi|} \Big) - \delta_\epsilon(\phi) \Big\{ - \Big[\lambda_1 (u_0(\mathbf{x}) - c_1)^2 - \lambda_2 (u_0(\mathbf{x}) - c_2)^2 \Big] \\ + \nu \Big[\Big(\int_\Omega H d\mathbf{x} - A_1 \Big) - \Big(\int_\Omega (1 - H) d\mathbf{x} - A_2 \Big) \Big] + D \Big[2(1-\theta)(u(\mathbf{x}) - u_0(\mathbf{x})) + \theta \frac{u(\mathbf{x}) - u_0(\mathbf{x})}{u^2(\mathbf{x})} \Big] \Big\},\end{aligned} \tag{16}$$

where $\mathbf{x} = (x, y)$, $\phi(\mathbf{x}, 0) = \phi^0(\mathbf{x})$ and $\frac{\partial\phi}{\partial\boldsymbol{n}}\Big|_{\partial\Omega} = 0$. To avoid singularities we can replace the term $|\nabla\phi|$ with $|\nabla\phi|_\beta = \sqrt{\phi_x^2 + \phi_y^2 + \beta}$, for a small β. Denoting

$$F(x,y) = -\delta_\epsilon(\phi)\Big\{ -\Big[\lambda_1(u_0(\mathbf{x}) - c_1)^2 - \lambda_2(u_0(\mathbf{x}) - c_2)^2\Big] + \nu\Big[\Big(\int_\Omega H d\mathbf{x} - A_1\Big) \\ - \Big(\int_\Omega (1-H)d\mathbf{x} - A_2\Big)\Big] + D\Big[2(1-\theta)(u(x,y) - u_0(x,y)) + \theta\frac{u(x,y) - u_0(x,y)}{u^2(x,y)}\Big]\Big\}, \tag{17}$$

and $E = \frac{W}{|\nabla\phi|_\beta}$, Eq. (16) can be written in the compact form:

$$\begin{cases} \dfrac{\partial\phi}{\partial t} = \mu\delta_\epsilon(\phi)\nabla\cdot(E\nabla\phi) + F(x,y) \\ \qquad = \mu\delta_\epsilon(\phi)(\partial_x(E\partial_x\phi) + \partial_y(E\partial_y\phi)) + F(x,y). \end{cases} \tag{18}$$

Discretizing in the spatial step, the Eq. (16) can be rewritten in the matrix-vector form:

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = \sum_{l=1}^{2} A_l(\phi^n)\phi^{n+1} + F(x,y)$$

where Δt is the time step size, n denotes the n^{th} iteration and A_l is the diffusion quantity in the l direction ($l = 1$ and $l = 2$ respectively for x and y direction for the two dimensional case). We can rewrite the above equation in the semi-implicit form:

$$\phi^{n+1} = (I - \Delta t\sum_{l=1}^{2} A_l(\phi^n))^{-1}\ \hat{\phi}^n \text{ for } l = 1, 2$$

and $\hat{\phi}^n = \phi^n + \Delta t F(x,y)$
which, by employing the AOS scheme, can be split additively as shown below to define the AOS solution

$$\phi^{n+1} = \frac{1}{2}\sum_{l=1}^{2}(I - 2\Delta t A_l(\phi^n))^{-1}\hat{\phi}^n \tag{19}$$

Here the matrices A_l, for $l = 1, 2$, are tridiagonal matrices derived using finite differences:

$$(A_1(\phi^n)\phi^{n+1})_{i,j} = \mu\delta_\epsilon(\phi^n)\bigg(\partial_x(E\partial_x\phi^{n+1}\bigg)_{i,j}$$
$$= \mu\delta_\epsilon(\phi^n)\frac{E^n_{i+1/2,j}(\partial_x\phi^{n+1})_{i+1/2,j} - E^n_{i-1/2,j}(\partial_x\phi^{n+1})_{i-1/2,j}}{h_x}$$
$$= \mu\delta_\epsilon(\phi^n)\frac{E^n_{i+1,j} + E^n_{i,j}}{2h_x^2}(\phi^{n+1}_{i+1,j} - \phi^{n+1}_{i,j}) - \mu\delta_\epsilon(\phi^n)\frac{E^n_{i,j} + E^n_{i-1,j}}{2h_x^2}(\phi^{n+1}_{i,j} - \phi^{n+1}_{i-1,j}),$$

and similarly

$$
\begin{aligned}
&(A_2(\phi^n)\phi^{n+1})_{i,j} = \mu\delta_\epsilon(\phi^n)\Big(\partial_y(E\partial_y\phi^{n+1}\Big)_{i,j} \\
&= \mu\delta_\epsilon(\phi^n)\frac{E^n_{i,j+1/2}(\partial_y\phi^{n+1})_{i,j+1/2} - E^n_{i,j-1/2}(\partial_y\phi^{n+1})_{i,j-1/2}}{h_y} \\
&= \mu\delta_\epsilon(\phi^n)\frac{E^n_{i,j+1}+E^n_{i,j}}{2h_y^2}(\phi^{n+1}_{i,j+1}-\phi^{n+1}_{i,j}) - \mu\delta_\epsilon(\phi^n)\frac{E^n_{i,j}+E^n_{i,j-1}}{2h_y^2}(\phi^{n+1}_{i,j}-\phi^{n+1}_{i,j-1}).
\end{aligned}
$$

As $A_l, l = 1, 2$ are tridiagonal matrices allows us to profit from the Thomas algorithm for a fast solution to the system.

4 Experimental Results

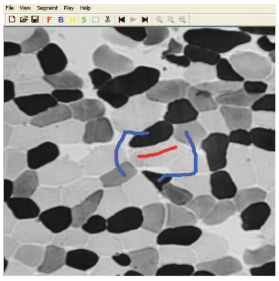
In order to illustrate the performance and the accuracy of the proposed method, experiments were carried out on homogeneous and inhomogeneous type of images. For all the shown experiments the distance function was not considered, in other words $d(x, y) = 1$. We like to emphasize that in all the results shown below the new method performs same speed as Rada-Chen method [21]. Two different type of experimental results are shown:

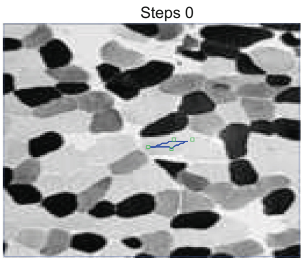
– Test Set 1 consisting of homogeneous images: Comparison with the Nguyen et al. model [16], which is a state of the art method and is known for its robustness, accurate boundary with a small amount of user interaction, as well as a comparison with the work Rada-Chen model [21], which shows good performance in segmenting hard homogeneous cases, where the objects have the same intensity and the boundaries are vague.
– Test Set 2 consisting of inhomogeneous images: Comparison with the Rada-Chen model [21].

In the following experiments the parameters $\mu = 100$, $\nu = 0.1$, $k = 1$, Δt, λ_1, λ_2, α, h (step size), ϵ and β, have been fixed at $\Delta t = 1$, $\lambda_1 = \lambda_2 = 1$, $h = 1$, $\epsilon = 1$, $\beta = 10^{-6}$, $\theta = 1$, respectively. As initial level set we consider a sign distant function of the given polygon constructed with the markers.

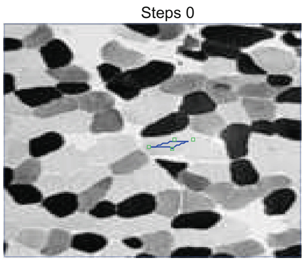
Test Set 1 — Robustness and Accuracy of the New Model for Homogeneous Images. Comparison with Nguyen et al. [16] and Rada-Chen [21] Models

In the first test set, we demonstrate the ability of recognizing objects in homogeneous images which have little difference in intensity. This type of images are a real challenge for selective segmentation. Figure 1(e) and (f) shows that the proposed model works satisfactorily and gives the same results with Rada-Chen [21] selective segmentation model. On the other side, we can see from Fig. 1(d) that the state of arts introduced at Nguyen et al. [16] fails. The Nguyen et al. [16]

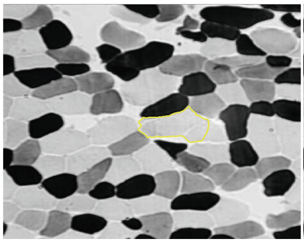

(a) Two types of strokes has been labeling some foreground and background pixels

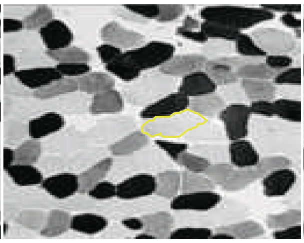


(b) In green are 4 given markers and in blue the first level set



(c) In green are 4 given markers and in blue the first level set


(d) Unsuccessful result of the cell by Nguyen-Cai-Zhang-Zheng model [16]


(e) Successful segmentation of the cell with Rada-Chen [21] model

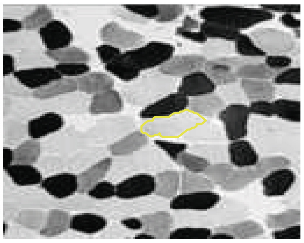
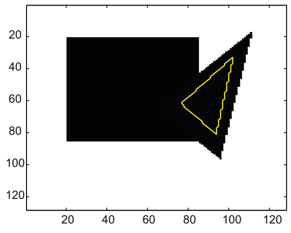
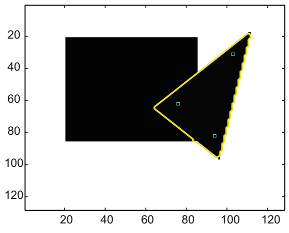

(f) Successful segmentation of the cell with the new model

Fig. 1. Test Set 1 – Comparison with Nguyen et al. [16] and Rada-Chen [21] models. Successful result by Rada-Chen [21] model and the new proposed model for the case of two cells with same intensity and semi-transparent boundaries (e–f). (Color figure online)


(a)


(b)

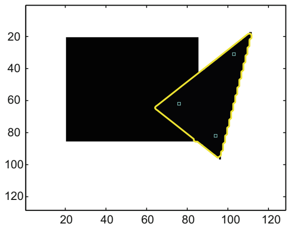

(c)

Fig. 2. Test Set 1 – Comparison with Rada-Chen [21] model for the case of two object with small intensity difference. (a) First level set; (b–c) Successful result by Rada-Chen [21] and new model.

method cannot handle transparent or semi-transparent boundaries; as shown in Fig. 1. Although the performance of Nguyen et al. model [16] is much better than other methods its results still contain some artifacts of fail due to similar objects appearance. A challenging selective segmentation presented in Rada-Chen [21] paper was the segmentation of a non empty intersection of a rectangle and triangle with small intensity difference of those two objects. Figure 2(b) and (c) presents satisfactory results of capturing the triangle object separately, by both Rada-Chen [21] and new proposed model.

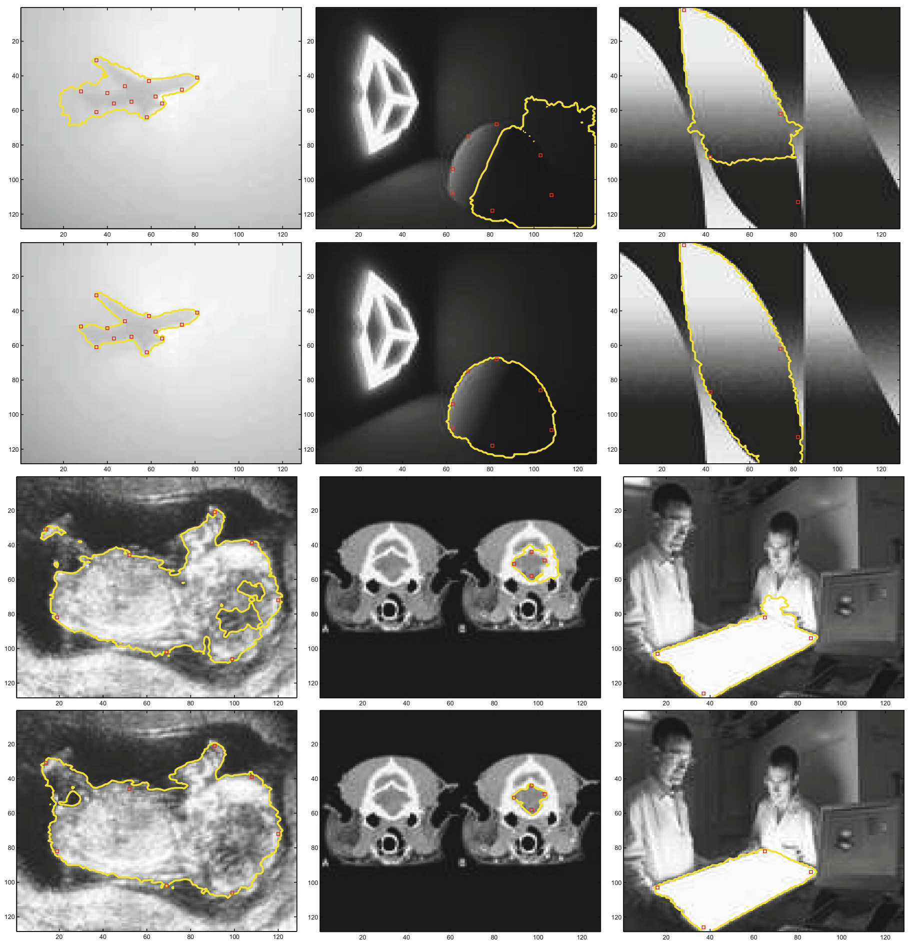


Fig. 3. Test Set 2 – Comparison with Rada-Chen [21] model: First and third row sets show unsuccessful result by Rada-Chen [21] model for the case inhomogeneous images; Second and forth row show successful result by the new model for the case of inhomogeneous images with small intensity difference in presence of noise.

Test Set 2 — Comparison with Model [21] for Inhomogeneous Images

We continue our experiments by giving more examples and compare our model with the Rada-Chen method [21]. Here we show 6 more different images, which are inhomogeneous images with or without presence of noise. The first and the second rows of the Fig. 3 consist of inhomogeneous artificial images post processed with Rada-Chen model [21] and the proposed model, respectively. From the results we conclude that the proposed model works satisfactorily for cases where the features are nearby, with different shapes, and in presence of noise, while the Rada-Chen model [21] fails. In the third and fourth rows of Fig. 3 we show a comparison for medical and real life images which are considered as harder inhomogeneous cases due to the low-quality data. In the fourth row we see an accurate segmentation of those images with the proposed method while Rada-Chen model, as shown in the third row, partially or totally fails. This test set shows that the proposed method is found to be successful for inhomogeneous images.

5 Conclusions

We have proposed a new variational selective segmentation model suitable for segmenting a range of images that have intensity inhomogeneity, noise and both of them together. We compared our proposed model with previous works that can handle problems with certain degree of intensity inhomogeneity including states of arts by Nguyen et al. [16] and Rada-Chen variational selective segmentation model [21]. For images with same intensity, semi-transparent boundaries or a certain amount of intensity inhomogeneity the Nguyen et al. [16] model will fail where as Rada-Chen [21] and our proposed models give equally successful results. Moreover, in cases strong inhomogeneity with our without noise the comparisons with Rada-Chen [21] model shows that our new proposed selective segmentation model is the only one that can deliver the expected segmentation quality. The introduced method can be widely used in different fields such as medical imaging where inhomogeneity and high level of noise is present.

References

1. Chan, T.F., Vese, L.A.: Active contours without edges. CAM Report, 98-53, UCLA (1998)
2. Chan, T.F., Vese, L.A.: Active contours without edges. IEEE Trans. Image Process. **10**(2), 266–277 (2001)
3. Mumford, D., Shah, J.: Optimal approximation by piecewise smooth functions and associated variational problems. Commun. Pure Appl. Math. **42**, 577–685 (1989)
4. Malik, J., Leung, T., Shi, J.: Contour and texture analysis for image segmentation. Int. J. Comput. Vis. **43**, 7–27 (2001)
5. Sen, D., Pal, S.K.: Histogram thresholding using fuzzy and rough measures of association error. IEEE Trans. Image Process. **18**(4), 879–888 (2009)

6. Vincent, L., Soille, P.: Watersheds in digital spaces: an efficient algorithm based on immersion simulations. IEEE Trans. Pattern Anal. Mach. Intell. **13**, 583–598 (1991)
7. Adams, R., Bischof, L.: Seeded region growing. IEEE Trans. Pattern Anal. Mach. Intell. **16**(6), 641–647 (1994)
8. Zucker, S.W.: Region growing: childhood and adolescence. Comput. Graph. Image Process. **5**, 382–399 (1976)
9. Aubert, G., Kornprobst, P.: Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations. Springer, New York (2001)
10. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: active contour models. Int. J. Comput. Vis. **1**(4), 321–331 (1988)
11. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. **24**, 603–619 (2002)
12. Geman, S., Geman, D.: Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell. **6**, 721–741 (1984)
13. Grady, L.: Random walks for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **28**, 1768–1783 (2006)
14. Bai, X., Sapiro, G.: A geodesic framework for fast interactive image and video segmentation and matting. In: IEEE ICCV, pp. 1–8 (2007)
15. Rother, C., Kolmogorov, V., Blake, A.: Grabcut: interactive foreground extraction using iterated graph cuts. ACM SIGGRAPH, **23**(3), 309–314 (2004)
16. Nguyen, T.N.A., Cai, J., Zhang, J., Zheng, J.: Robust interactive image segmentation using convex active contours. IEEE Trans. Image Process. **21**, 3734–3743 (2012)
17. Goldstein, T., Bresson, X., Osher, S.: Geometric applications of the split Bregman method: segmentation and surface reconstruction. J. Sci. Comput. **45**, 272–293 (2010)
18. Gout, C., Guyader, C.L., Vese, L.A.: Segmentation under geometrical consitions with geodesic active contour and interpolation using level set methods. Numer. Algorithms **39**, 155–173 (2005)
19. Guyader, C.L., Gout, C.: Geodesic active contour under geometrical conditions theory and 3D applications. Numer. Algorithms **48**, 105–133 (2008)
20. Badshah, N., Chen, K.: Image selective segmentation under geometrical constraints using an active contour approach. Commun. Comput. Phys. **7**(4), 759–778 (2009)
21. Rada, L., Chen, K.: Improved selective segmentation model using one level-set. J. Algorithms Comput. Technol. **7**, 506–540 (2013)
22. Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: algorithms based on HJ formulations. J. Comput. Phys. **79**, 12–49 (1988)
23. Sethian, J.A.: Level Set Methods and Fast Marching Methods. Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision and Material Science. Cambridge University Press, Cambridge (1999)
24. Zhao, H.K., Merriman, B., Osher, S.: A variational level set approach to multiphase motion. J. Comput. Phys. **127**, 79–195 (1996)
25. Li, C., Kao, C.Y., Gore, C.J., Ding, Z.: Implicit active contours driven by local binary fitting energy. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 42, pp. 1–7 (2007)
26. Aubert, G., Aujol, J.: A variational approach to remove multiplicative noise. SIAM J. Appl. Math. **68**, 925–946 (2008)
27. Wang, X.F., Huang, D.S., Xu, H.: An efficient local Chan-Vese model for image segmentation. Pattern Recogn. **43**, 603–618 (2010)

28. Lu, T., Neittaanmaki, P., Tai, X.C.: A parallel splitting-up method for partial differential equations and its application to Navier-Stokes equations. RAIRO Math. Model. Numer. Anal. **26**(6), 673–708 (1992)
29. Weickert, J., Romeny, B., Viergever, M.: Efficient and reliable schemes for nonlinear diffusion filtering. IEEE Trans. Image Process. **7**(3), 398–410 (1998)
30. Weickert, J., Kühne, G.: Fast methods for implicit active contour. In: Osher, S., Paragios, N. (eds.) Geometric Level Set Methods in Imaging Vision and Graphics, pp. 43–57. Springer, New York (1995)

A Review of SAR Hybrid De-Speckling Methods

Memoona Malik[1], Muhammad Haris[1](✉), Aamir Hanif Dar[1], Asad Ali Safi[2], and Mahmood Ashraf[3]

[1] COMSATS Institute of Information Technology, Islamabad, Pakistan
mharispak@gmail.com
[2] University of Islamabad, Islamabad, Pakistan
[3] Federal Urdu University of Arts, Science and Technology, Islamabad, Pakistan

Abstract. Significance of high resolution SAR imagery is irrefutable in Earth monitoring applications as it provides valuable information to be analyzed. To examine this information, post image processing techniques such as speckle noise removal, segmentation, edge and object detection are necessary to per-form. Despeckling among them is the fundamental process that makes images visually appealing and comprehensible. This survey paper thus focuses on this primary process and investigates about the recent two-step hybrid despeckling techniques that suppress speckle noise in SAR images. The paper briefly talks about the importance of SAR imaging system, the speckle noise that distorts the information contained in them and the despeckling techniques that exist in literature to eliminate the speckle noise. The review, however, studies the hybrid despeckling techniques in detail by discussing advantages and limitations of models these techniques are following. In the end, it will provide suggestions on how to improve these models.

Keywords: SAR imagery · Despeckling · SAR-BM3D · PCA

1 Introduction

For many years, synthetic aperture radar (SAR) imaging systems have been extensively used for monitoring Earth's resources rather than optical imaging systems. SAR imaging systems can capture and provide high resolution images even in the presence of natural obstacles such as absence of light, presence of clouds, dust, snow and drizzle [1, 2]. In contrast, optical imaging systems do not capture clear images in the presence of natural blockages. This all-weather and whole-day acquisition capability of SAR imaging systems makes it more reliable than the early optical imaging systems which get affected by weather conditions and have dependency on daylight [1, 3]. Independence from daylight is achieved by an artificial illumination source installed in the SAR imaging systems. The larger wavelength of microwaves is responsible for all-weather acquisition capability as this wavelength can penetrate through any obstacles present in the atmosphere [1, 3, 4]. Thus, SAR becomes a highly beneficial and reliable source for acquiring Earth's images. Another noticeable characteristic of SAR imaging systems is the high-resolution imagery which is acquired through synthetic aperture. Synthetic aperture is obtained by positioning the antenna on moving platforms where antenna continuously sends and receives microwave pulses and computes their aggregate while platform is in motion hovering over the target. High resolution images are achieved through the aggregate computed by SAR imaging system [2, 3].

Figure 1 illustrates the transmitted microwave pulse and its backscattering to comprehend the image formation process of SAR imaging system [3]. All the backscattered microwaves pulses either do not reach the receiving antenna or their phases get interfered with other waves which distorts the information contained in them. This distortion is described as speckle noise which affects the quality of SAR images. Reasons of the speckle noise in SAR images include scattering mechanism and interference of the electromagnetic waves [5]. Researchers categorized the speckle noise as multiplicative noise which follows Rayleigh distribution, however it can also be described as a degradation function which degrades the quality of SAR imagery [6].

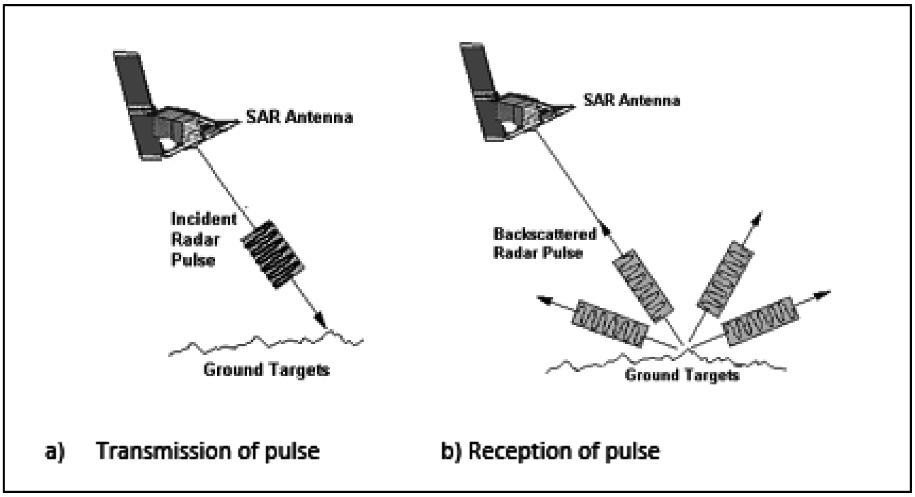


Fig. 1. Transmission and reception of microwave pulses by SAR antenna

Severity of this noise can be observed in Fig. 2 which shows an outdoor noise free and noisy image and clearly demonstrates the effect of noise [7]. Due to the speckle noise, extracting useful information from SAR images becomes difficult which restricts their use in a wide range of applications [1–6]. To restore their useful information such as edges and textures, despeckling becomes necessary as it reduces the amount of speckle and make image contents visible and useful for further processing. Despeckling process is thus necessary to enhance SAR images to be used efficiently and effectively in various applications such as geosciences, forestry, monitoring of climate changes, resource monitoring, oceanology and change detection. To accomplish this task, researchers have devised numerous despeckling techniques in varied domains. As hybrid techniques are the focus of this paper so only these techniques will be discussed in depth in the literature section. However, interested readers are suggested to view the comprehensive surveys on despeckling techniques provided by [5, 8] for better understanding of the despeckling literature. Despeckling filters can be grouped into local [9–11], non-local [12–18], transform based [19–25] and hybrid ones. Local filters despeckle images by alternating the selected pixel's value based on some statistical criterion which is calculated through its neighboring pixels.

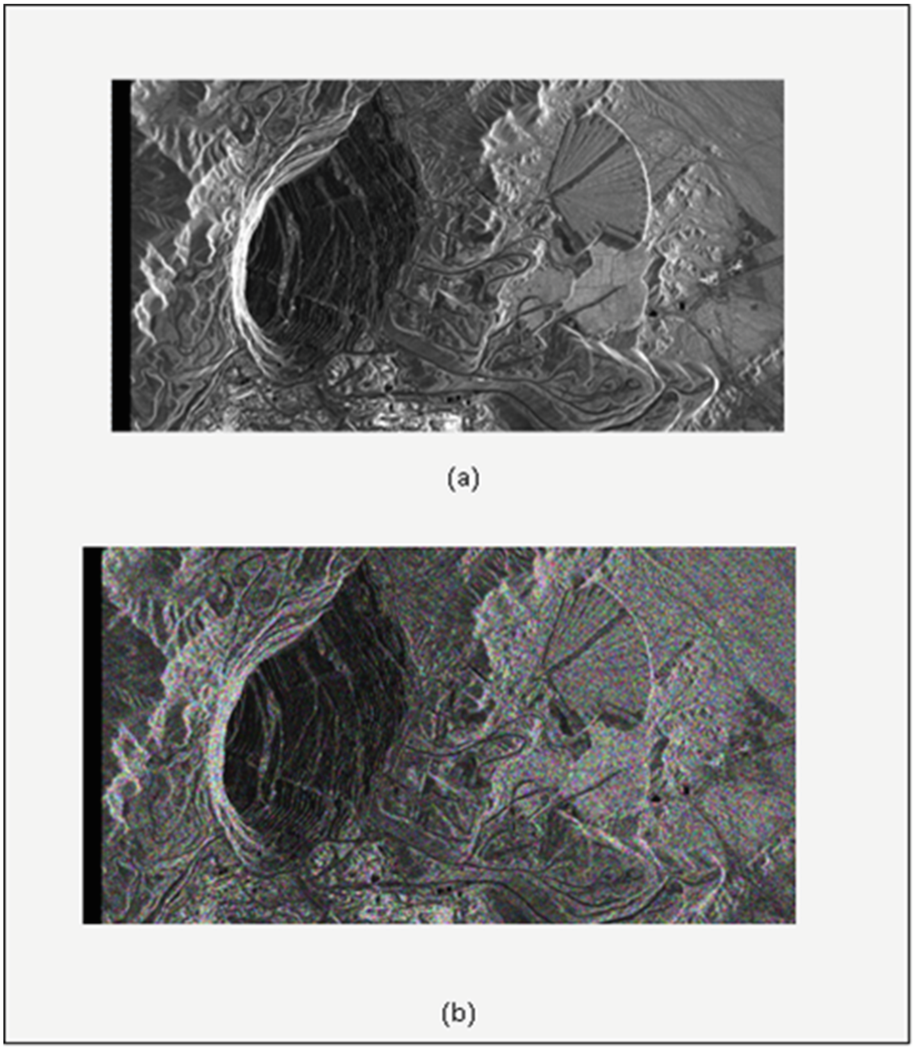


Fig. 2. (a) Noise free satellite image (b) Noisy satellite image

Their concept assumes that neighborhood pixels are of similar statistical nature as that of the selected pixel. This is true in case of homogeneous regions however false for heterogeneous ones. Hence local filters are effective at uniform regions but are responsible for over smoothening fine structures in non-uniform regions [5]. To over-come this problem, researchers have devised non-local filters in past few years. Unlike local filters, these filters look up for a similar pixel or a block anywhere in the image which exhibits the same statistical nature as that of the selected pixel thus increasing the search space. However, these filters perform better at preserving fine structures in heterogeneous regions than local filters [5]. The major limitation of these filters is increased computational cost due to the expanded search space. The transform based approaches are also good at despeckling, but accurate threshold computation and how it should be applied on the image are some serious concerns regarding transforms which put limits to their use [26]. Hybrid approaches in literature combined pixel grouping strategies such as block matching and clustering with various transforms i.e. wavelets and principal component analysis, to despeckle the SAR imagery. These approaches are computationally inefficient but outperform all previous techniques [27, 28]. In this paper existing hybrid approaches will be discussed in Sect. 2. Section 3 provides a discussion on pros and cons of hybrid techniques and their models. And finally, Sect. 4 concludes the paper.

2 Hybrid Filters

The hybrid filter that used pixel grouping strategy and transforms was first proposed by authors to remove additive Gaussian noise [29]. The approach was later modified for removing the speckle noise [27]. This new hybrid filter is dubbed as SAR-BM3D which performed block matching and employed local linear minimum mean square error (LLMMSE) shrinkage transformation instead of hard thresholding as used in the former approach. Although SAR-BM3D performed better than all local, non-local and transform based despeckling techniques, still it is not a cost-effective solution due to its block matching phase. Other problems associated with SAR-BM3D includes introduction of artificial edges into the denoised image and incapability of handling heterogeneity due to usage of lossy transforms i.e. undecimated wavelet and discrete cosine transforms [13, 30]. To overcome the computational complexity of SAR-BM3D, a fast non-local despeckling filter was proposed which replaced the Wiener filter of SAR-BM3D with soft and hard wavelet thresholding for flat and active homogenous blocks respectively [31]. For reducing the complexity of block-matching phase, it used a variable sized search area and a probabilistic early termination criterion. Results revealed that it took less time than SAR-BM3D and showed better despeckling performance than other techniques except SAR-BM3D. The limitations of FANS include over-smoothened edges and inability to preserve textures. Another version of SAR-BM3D known as classification based SAR-BM3D (C-SAR-BM3D) was proposed in [32]. First this filter classified all the pixels as homogeneous or non-homogeneous. Secondly, it filtered the homogeneous regions with simple non-local means filter whereas its strategy remained same for non-homogeneous regions as of SAR-BM3Ds. However, its despeckling results did not out-perform SAR-BM3D due to misclassification of pixels [32].

To improve the despeckling accuracy of SAR-BM3D, another attempt has been made in 2014 [28]. The proposed technique transformed the image into principal component analysis (PCA) domain and get signal PCs for applying k-means clustering.

The clusters were again transformed to PCA domain for applying LMMSE shrinkage. This method showed satisfactory results on homogeneous regions as compared to SAR-BM3D and minimized computational cost. However, it did not perform well on heterogeneous regions. Moreover, k-means clustering produced fixed clusters which is not an optimal solution for grouping of high dimensional data like SAR [33]. An almost similar research was carried out in 2015 [34] which used linear discriminant analysis (LDA) instead of PCA while keeping the remaining strategy same as of [28]. This new research did not provide an optimal solution as well due to k-means clustering [34]. Later, a framework was suggested in [35] which applied SAR-BM3D and homomorphic version of learned simultaneous sparse coding (H-LSSC) on an image to obtain two different estimates. In this method, a soft classifier classified the image content into various classes and computes a despeckled image based on its classification and the two calculated estimates. It did not preserve edges and textures although it performed better despeckling on homogenous regions. In 2015, another attempt has been made to despeckle SAR imagery that utilized the concept of sparse representation [36]. This algorithm does not completely belong to this last category as it did not use any transform. It uses non-local sparse model with iterative regularization technique and demonstrated promising results at some images, however computationally complex when compared with SAR-BM3D. Drawback associated with this technique includes induction of new artifacts in the denoised image.

In 2016, an approach which computed digital elevation model for natural scenes was developed [37]. The designed algorithm performed quite well on homogenous natural scenes as it can describe the scattering mechanism on these regions. But it is incapable of describing the scattering mechanism at man-made structures and non-topographic edges. Sensitivity analysis of SB-SARBM3D has been done afterwards to analyze the influence of scattering model, surface parameters errors, DEM resolution and errors in co-registration step on this filter [38]. Furthermore, the sensitivity analysis of scattering based non-local means despeckling algorithm has been carried out in [39]. Advantages and limitations of proposed hybrid approaches are provided in Table 1.

Table 1. Advantages and limitations of Hybrid approaches

	Technique name	Advantages	Limitations
1.	SAR-BM3D [27]	Preserves image details in both homogeneous and heterogeneous regions and outperforms previous approaches	Computationally inefficient and introduces artifacts in homogeneous regions
2.	FANS [31]	Computationally efficient than SARBM3D	Overall despeckling performance is not superior than SARBM3D due to miss-classification of flat and active regions
3.	C-SARBM3D [32]	No artifacts in homogeneous regions	Manual parameter tuning for each class and poor performance on unstructured texture

(*continued*)

Table 1. (*continued*)

	Technique name	Advantages	Limitations
4.	Clustering Based PCA [28]	Outperforms previous techniques	Could not accurately group pixels as it used k-means for clustering
5.	LDA-Based Solution [34]	Showed satisfactory performance	Clustering accuracy is affected due to k-means
6.	Soft-Classification [35]	Performance is comparable with other approaches	Due to low classification accuracy, results get affected.
7.	SB-SARBM3D [37]	Shows better performance on homogeneous regions than SAR-BM3D	Can only denoise natural scenes as it needs to calculate the digital elevation model

3 Discussion

Hybrid approaches perform better despeckling and are computationally complex than local, non-local and transform based filters. The hybrid techniques can be broadly classified according to the two despeckling models i.e. SAR-BM3D and clustering-based PCA illustrated in Figs. 3 and 4. Techniques that work on the principle of SAR-BM3D used block matching to group similar pixels which is based on the concept of non-local filters. These techniques thus exhibit the limitations of NLM filter such as issue in selecting appropriate similarity measure for comparing blocks and increased computational cost. Increased computational cost makes block matching unsuitable for high dimensional SAR data. FANS, an extension to SAR-BM3D, attempted to reduce the computational cost of SAR-BM3D by introducing the concept of variable search size. This method significantly minimized the computational cost, however, there seems a tradeoff between accuracy and time complexity as FANS do not surpass SAR-BM3D in preserving textures. Some techniques applied classification methods to group the pixels which requires training data set. Absence of data set caused issues of misclassification which leads to poor despeckling performance.

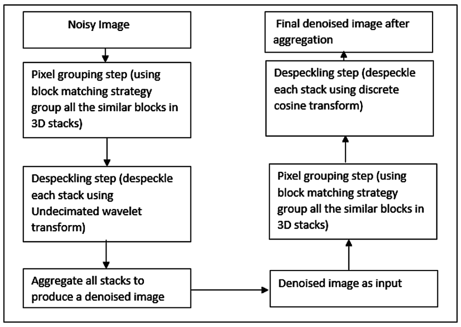

Noisy Image
Pixel grouping step (using block matching strategy group all the similar blocks in 3D stacks)
Despeckling step (despeckle each stack using Undecimated wavelet transform)
Aggregate all stacks to produce a denoised image
Denoised image as input
Pixel grouping step (using block matching strategy group all the similar blocks in 3D stacks)
Despeckling step (despeckle each stack using discrete cosine transform)
Final denoised image after aggregation

Fig. 3. SAR-BM3D model

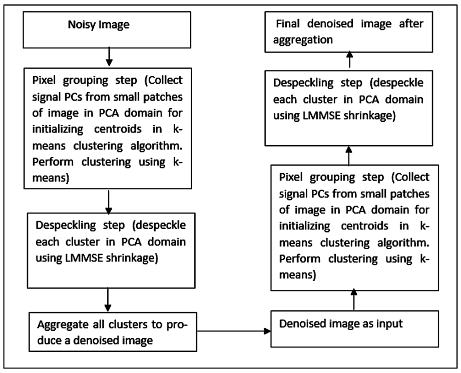

Noisy Image
Pixel grouping step (Collect signal PCs from small patches of image in PCA domain for initializing centroids in k-means clustering algorithm. Perform clustering using k-means)
Despeckling step (despeckle each cluster in PCA domain using LMMSE shrinkage)
Aggregate all clusters to produce a denoised image
Denoised image as input
Pixel grouping step (Collect signal PCs from small patches of image in PCA domain for initializing centroids in k-means clustering algorithm. Perform clustering using k-means)
Despeckling step (despeckle each cluster in PCA domain using LMMSE shrinkage)
Final denoised image after aggregation

Fig. 4. Clustering-based PCA model

Techniques that follow the second model group similar pixels using k-means clustering. K-means clustering algorithm does not cater high dimensionality and produce fixed number of clusters in both homogeneous and heterogeneous regions. Thus, it may not exactly map the real image contents. However, its lower computational cost makes it time efficient than block matching. Advantages and limitations of pixel grouping strategies are highlighted in Table 2. For removing noise from the groups of similar pixels, both these models used transforms like wavelets and PCA which are inherently lossy and thus affect the preservation of important image contents such as edges and textures.

Table 2. Advantages and limitations of the pre-processing step used in hybrid approaches

Sr. #	Technique name	Advantages	Limitations
1	SAR-BM3D [27]	Achieved better accuracy rates than local, non-local and transform based filters	Computationally inefficient, rare-block problem, hard to find similarity measure, difficult to choose an optimal block size
2	FANS [31]	Computationally efficient than SARBM3D	Still have issues of similarity measure and block size
4	Clustering-Based PCA [28]	Computationally efficient than FANS	Could not accurately group pixels as it used k-means for clustering
3	LDA-Based Solution [34]	Same as Clustering Based PCA	Clustering accuracy is affected due to k-means
5	SB-SARBM3D [37]	Same as SAR-BM3D	Same as SAR-BM3D

Despeckling performance of hybrid techniques depend on how appropriately they group the SAR images into similar regions. For grouping, hybrid techniques used block matching and k-means clustering. These grouping strategies do not provide an optimal solution for segmenting high dimensional SAR data which leaves an open margin for finding an appropriate strategy to improve the despeckling performance. The capability of denoising strategies used by hybrid techniques also affects their despeckling performance. The existing denoising strategies are inherently lossy which arises a need to find the denoising strategy that will better preserve image contents of SAR images.

4 Conclusion

This survey paper provided a brief introduction to SAR imaging systems; how they acquire high resolution images and get corrupted with speckle noise. The review paper has discussed hybrid despeckling techniques in detail. These techniques are categorized in this paper according to the two despeckling models i.e. SAR-BM3D and Clustering-based PCA. These models first group the similar pixels of SAR images via block matching or k-means clustering and afterwards apply transforms i.e. wavelets and PCA to denoise. Limitations of grouping and denoising strategies have also been discussed. From their limitations, it can be concluded that clustering and transforms that remain unexplored in this context should be considered and compared with the existing to find a more appropriate solution towards despeckling of SAR imagery.

References

1. Moreira, A., Prats-Iraola, P., Younis, M., Krieger, G., Hajnsek, I., Papathanassiou, K.: A tutorial on synthetic aperture radar. IEEE Geosci. Remote Sens. Mag. **1**(1), 6–43 (2013)
2. European space agency, "Synthetic Aperture Radar Land Applications Tutorial", European space agency. https://earth.esa.int/documents/10174/2700124/sar_land_apps_1_theory.pdf\
3. Centre for remote imaging, sensing and processing, "Principles of remote sensing", Centre for Remote Imaging, Sensing and Processing. https://crisp.nus.edu.sg/~research/tutorial/mw.htm
4. National Aeronautics and Space Administration, Science Mission Directorate. "Microwaves", NASA Science website. https://science.nasa.gov/ems/06_microwaves. Accessed 25 July 2017
5. Argenti, F., Lapini, A., Bianchi, T., Alparone, L.: A tutorial on speckle reduction in synthetic aperture radar images. IEEE Geosci. Remote Sens. Mag. **1**(3), 6–35 (2013)
6. Gupta, A.: Multiplicative Noise Removal Techniques - A Study. https://doi.org/10.13140/rg.2.1.2651.7844/1. November 2015
7. http://www.geo-airbusds.com/satellite-image-gallery
8. Devapal, D., Kumar, S., Jojy, C.: Comprehensive survey on SAR image despeckling techniques. Indian J. Sci. Technol. **8**(24) (2015)
9. Manth, N., Kalra, N., Virmani, J., et al.: Despeckle filtering: performance evaluation for malignant focal hepatic lesions. In: International Conference on Computing for Sustainable Global Development (INDIACom), May 2015
10. Molina, D.E., Gleich, D., Datcu, M.: Evaluation of Bayesian despeckling and texture extraction methods based on gauss–mark and auto-binomial gibbs random fields: application to TerraSAR-X data. IEEE Trans. Geosci. Remote Sens. **50**(5), 2001–2025 (2012)
11. Lee, J., Wen, J., Ainsworth, T.L., et al.: Improved sigma filter for speckle filtering of SAR imagery. IEEE Trans. Geosci. Remote Sens. **47**(1), 202–213 (2009)

12. Deledalle, C., Denis, L., Tupin, F.: Iterative weighted maximum likelihood denoising with probabilistic patch-based weights. IEEE Trans. Image Process. **18**(12), 2661–2672 (2009)
13. Deledalle, C., Denis, L., Tupin, F., Reigber, A., Jager, M.: NL-SAR: a unified non-local framework for resolution-preserving (Pol)(In)SAR denoising. IEEE Trans. Geosci. Remote Sens. **53**(4), 2021–2038 (2015)
14. Feng, W., Lei, H.: Combination of geometric clustering and nonlocal means for SAR image despeckling. Electron. Lett. **50**(5), 395–396 (2014)
15. Zhong, H., Li, Y., Jiao, L.: SAR image despeckling using Bayesian nonlocal means filter with sigma preselection. IEEE Geosci. Remote Sens. Lett. **8**(4), 809–813 (2011)
16. Gomez, L., Munteanu, C., Buemi, M., et al.: Supervised constrained optimization of Bayesian nonlocal means filter with sigma preselection for despeckling SAR Images. IEEE Trans. Geosci. Remote Sens. **51**(8), 4563–4575 (2013)
17. Ni, W., Gao, X.: Despeckling of SAR image using generalized guided filter with Bayesian nonlocal means. IEEE Trans. Geosci. Remote Sens. **54**(1) (2016)
18. Verdoliva, L., Gaetano, R., Ruello, G., Poggi, G.: Optical-driven nonlocal SAR despeckling. IEEE Geosci. Remote Sens. Lett. **12**(2), 314–318 (2015)
19. Zhang, W., Liu, F., Jiao, L., Hou, B., Wang, S., Shang, R.: SAR image despeckling using edge detection and feature clustering in bandelet domain. IEEE Geosci. Remote Sens. Lett. **7**(1), 131–135 (2010)
20. Li, Y., Gong, H., Feng, D., Zhang, Y.: An adaptive method of speckle reduction and feature enhancement for SAR images based on curvelet transform and particle swarm optimization. IEEE Trans. Geosci. Remote Sens. **49**(8), 3105–3116 (2011)
21. Shanthi, I., Valarmathi, M.L.: Contourlet transform with modified particle swarm optimisation for despeckling and feature enhancement of SAR images. Int. J. Comput. Appl. Technol. **51**(1), 31–42 (2015)
22. Huang, S., Huang, W., Zhang, T., Xu, C.: A statistical and Wiener filtering algorithm based on the curvelet transform for SAR images. Int. J. Remote Sens. **37**(23), 5581–5604 (2016)
23. Argenti, F., Bianchi, T., Scarfizzi, G., Alparone, L.: LMMSE and MAP estimators for reduction of multiplicative noise in the nonsubsampled contourlet domain. Signal Process. **89**, 1891–1901 (2009)
24. Xia, C., Licheng, J., Fan, L., Yuheng, S.: SAR image despeckling using scale mixtures of Gaussians in the nonsubsampled contourlet domain. Chin. J. Electron. **24**(1), 205–211 (2015)
25. Gao, F., Xue, X., Sun, J., Wang, J., Zhang, Y.: A SAR image despeckling method based on two-dimensional S transform shrinkage. IEEE Trans. Geosci. Remote Sens. **54**(5), 1–10 (2016)
26. Malik, M., Ahsan, F., Mohsin, S.: Adaptive image denoising using cuckoo algorithm. Soft. Comput. **20**(3), 925–938 (2014)
27. Parrilli, S., Poderico, M., Angelino, C., Verdoliva, L.: A nonlocal SAR image denoising algorithm based on LLMMSE wavelet shrinkage. IEEE Trans. Geosci. Remote Sens. **50**(2), 606–616 (2012)
28. Xu, L., Li, J., Shu, Y., Peng, J.: SAR image denoising via clustering based principal component analysis. IEEE Trans. Geosci. Remote Sens. **52**(11), 6858–6869 (2014)
29. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising by sparse 3-D transform-domain collaborative filtering. IEEE Trans. Image Process. **16**(8), 2080–2095 (2007)
30. Deledalle, C., Denis, L., Poggi, G., Tupin, F., Verdoliva, L.: Exploiting patch similarity for SAR image processing: the nonlocal paradigm. IEEE Signal Process. Mag. **31**(4), 69–78 (2014)

31. Cozzolino, D., Parrilli, S., Scarpa, G., Poggi, G., Verdoliva, L.: Fast adaptive nonlocal SAR despeckling. IEEE Geosci. Remote Sens. Lett. **11**(2), 524–528 (2014)
32. Gragnaniello, D., Poggi, G., Verdoliva, L.: Classification based nonlocal SAR despeckling. In: Tyrrhenian Workshop on Advances in Radar and Remote Sensing, pp. 121–125 (2012)
33. Xu, D., Tian, Y.: A comprehensive survey of clustering algorithms. Ann. Data Sci. **2**(2), 165–193 (2015)
34. Adharshini, R., Raja, B.: SAR image denoising via clustering based linear discriminant analysis. In: IEEE Sponsored 2nd International Conference on Innovations in Information Embedded and Communication Systems (2015)
35. Gragnaniello, D., Poggi, G., Scarpa, G., Verdoliva, L.: SAR image despeckling by soft classification. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **9**(6), 2118–2130 (2016)
36. Xu, B., Cui, Y., Li, Z., Yang, J.: An iterative SAR image filtering method using non-local sparse model. IEEE Geosci. Remote Sens. Lett. **12**(8), 1635–1639 (2015)
37. Di Martino, G., Di Simone, A., Iodice, A., Poggi, G., Riccio, D., Verdoliva, L.: Scattering-based SARBM3D. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **9**(6), 2131–2144 (2016)
38. Di Martino, G., Di Simone, A., Iodice, A., Riccio, D.: Scattering-based non-local means SAR despeckling. IEEE Trans. Geosci. Remote Sens. **54**(6), 3574–3588 (2016)
39. Di Simone, A., Di Martino, G., Iodice, A., Riccio, D.: Sensitivity analysis of a scattering-based nonlocal means Despeckling Algorithm. Eur. J. Remote Sens. **50**, 87–97 (2017)

Zero-Shot Learning with Partial Attributes

Matías Molina[1,2](✉) and Jorge Sánchez[2,3]

[1] Facultad de Matemática, Astronomía, Física y Computación, Universidad Nacional de Córdoba, Córdoba, Argentine
matias.molina@unc.edu.ar
[2] CONICET, Córdoba, Argentine
[3] CIEM-CONICET, Córdoba, Argentine

Abstract. Zero-shot classification, i.e. the task of learning predictors for classes with no training samples, requires to resort to subsidiary information in order to overcome the lack of annotated data. In the literature, one of the most popular approaches is to represent the class information by a set of visual attributes and to learn a visual-semantic embedding that allow us to transfer the information from those classes with plenty of annotated samples to those with no data available a training time. One mayor limitation of the attribute-based approach is that adding a new class requires a non-negligible annotation effort. This has motivated the search for alternative sources of semantic information. Here, the use of word embeddings learned from raw text appears as an appealing and scalable choice. In this paper, we consider a middleground scenario in which attribute vectors are only available for the training categories. We propose a deterministic approach to infer the attributes for the testing classes which, despite its simplicity, shows competitive results. We also propose two simple improvements to the structured embedding formulation of Akata *et al.*, leading to significant improvements on attribute-only classification. Experiments on the *Animals With Attributes* and *Caltech-UCSD Birds* datasets show competitive performance.

Keywords: Zero-shot learning · Visual attributes
Semantic embeddings

1 Introduction

Image classification is one of the most well studied problems in the computer vision literature. In the past decade, models used to tackle this classical problem have evolved from combinations of hand-crafted features and classifiers to larger deep-learning based models trained end-to-end. This paradigm shift has lead to astonishing results on different benchmarks [1–3]. However, training such models requires large quantities of manually annotated data. Although such datasets exist for a large number of common object classes, extending those models to less known and more specific categories (e.g. fine-grained problems) requires a great annotation effort. This has motivated the interest on methods that can be trained with just a few training samples [4,5]. Zero-shot classification corresponds to an extreme case in which, for some of the classes, no training samples are provided. In this case, one must rely on subsidiary information as a way of bypassing the lack of image-level annotations as well as gathering knowledge about the visual concepts we aim to learn. One popular approach along this line of work is to rely on visual attributes [6], a characterization of the classes based on visually distinguishable properties of the objects, e.g. its color, shape, presence/absence parts, etc. Here, the attributes-based approach to zero-shot learning (ZSL) consists on training a model on the images and classes for which we have actual annotations (image-level labels and class-level attributes, respectively) and to transfer this knowledge to the classes with no training samples based on the information gathered about the visual appearance of the objects in the training set and how this information relates to the visual attributes encoding the different categories.

We can distinguish two cases: the standard ZSL setting [6], where training and evaluation (testing) categories are disjoint sets, and the generalized ZSL setting (GZSL) [7] where it is allowed for some of the training categories to be present during evaluation. Beyond the advantages of using class attributes as an alternative to image-level annotations, generating such descriptions for new images still requires a non-negligible annotation effort compared to other approaches, i.e. the use of word-vector embeddings [8,9], class hierarchies [10], etc. These alternatives, however, have shown inferior performance compared to attribute-based models [11].

In this work, we focus on a variation of the standard zero-shot classification problem in which class attributes are only available during training. We term this problem as *ZSL with partial attributes*. To tackle this problem, we propose a model that leverages pre-trained word embeddings in order to propagate the attribute information from the training to the testing classes. A model for ZSL is then learned on the set inferred attributes. Experiments on two popular datasets for ZSL show the benefits of our approach.

Our contributions are the following: *(1)* We propose a model for attribute-based classification when attribute annotations are not available at test time; *(2)* We propose a series of simple modifications to a known ZSL model that lead to significant improvements on performance, and *(3)* we achieve state-of-the-art results on two popular dataset.

The paper is organized as follows: in Sect. 2 we discuss related work. In Sect. 3 we present our model. In Sect. 4 we present experimental results. Finally, in Sect. 5 we draw some conclusions.

2 Related Work

Different methods have been proposed in the literature to address the ZSL problem [6,10,12–15]. Most of them assume that both images and class labels can be encoded as points in some vector space. For the case of the images, the usual choice is to represent them by feature vectors extracted from an intermediate layer of a pre-trained convolutional neural network, e.g. using the activation scores of some layer of the VGG [16] or ResNet [17] models. For the labels, different works have explored the use of visual attributes [6], class hierarchies [10] or more generic text-based embeddings models like word2vec [8] and GloVe [9]. In the ZSL literature, the use of visual attributes has lead to consistently better performance compared to other output encoding techniques [12]. However, given a new class, generation of its attribute descriptor is rather more involved[1] than, e.g., a word-vector embedding based on its name. This poses a limitation on the use of visual attributes for problems where a subset of testing cases are not known at training time. Now, having defined representations for the images (inputs) and class labels (outputs), the next step is to seek for a model that allow us to relate the elements from both spaces, e.g. to compute distances or similarities between image features and class labels. For this task, most methods in the literature [11] try to find projections onto an intermediate common space on which the comparison takes place [10,19] or to directly model a compatibility score between image and class label features [12,14,15]. Here, a popular approach is the generation of a bilinear compatibility mechanism. In this case, the scoring process can be seen as projection from the input (output) to the output (input) space followed by a dot-product similarity on the target space. There are also other cases where this projection is performed explicitly, e.g. the ConSE [20] model employs class posteriors for the known classes to build a representation for an unknown class. These models, which are more closely related to our approach, will be discussed next.

Before going into the details, let us first introduce some notation. We denote as $\mathcal{X}^{tr}$ and $\mathcal{Y}^{tr}$ ($\mathcal{X}^{ts}$ and $\mathcal{Y}^{ts}$) the set of images and labels available at training (testing) time, respectively. In this work, we focus on the standard ZSL classification problem, i.e. the case where $\mathcal{Y}^{tr} \cap \mathcal{Y}^{ts} = \emptyset$.

Direct Attribute Prediction (DAP) [6]. DAP is perhaps the earliest approach that considered the use of visual attributes for classification. In [6], the authors propose to learn a set of attribute predictors (classifiers) using the data available for training and to use this information as a proxy for transferring knowledge acquired on $\mathcal{Y}^{ts}$, to $\mathcal{Y}^{tr}$. More specifically, the method consists on learning classifiers $p(a_i|x)$, $i = 1, \ldots, E$, using the image-attribute pairs from the training set. Given an image $x \in \mathcal{X}^{ts}$, the model predict a class $\hat{y} \in \mathcal{Y}^{ts}$ according to the MAP rule:

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{Y}^{ts}} \prod_{i=1}^{E} \frac{p(a_{y,i}|x)}{p(a_{y,i})}, \tag{1}$$

where $a_{y,i}$ denotes the presence or absence of attribute i on class y and $p(a_{y,i})$ is estimated from statistics on the training set.

Convex Combination of Semantic Embeddings (ConSE) [20]. Whereas previous approaches treated the ZSL problem as a regression from the image feature space

[1] Attribute-based representations are generally obtained by using crowd sourcing techniques or by relying on expert human knowledge [18].

to the label space, in ConSE, the authors followed a more classical approach and propose to directly learn classifiers from $\mathcal{X}^{tr}$ to $\mathcal{Y}^{tr}$ and transfer the predictions made by these models to $\mathcal{Y}^{ts}$. The method is based on learning a set of classifiers $p(y|x)$. Then, for a new input sample x, the model first predict a semantic embedding $\alpha(x)$ for x:

$$\alpha(x) = \frac{1}{Z} \sum_{t=1}^{T} p(\hat{y}_t|x)\psi(\hat{y}_t), \tag{2}$$

and predicts an output category based on the nearest neighbor rule $\hat{y}(x) = \text{argmax}_{y\in\mathcal{Y}^{ts}} \cos(\hat{a}(x), a_y)$, with $\cos(\cdot,\cdot)$ the cosine similarity function. In Eq. (2), $p(\hat{y}_t|x)$ denotes the top-t prediction made by the model, $\psi(\hat{y}_t)$ is the output embedding for $\hat{y}_t$, e.g. its attributes vector, T is an hyperparameter and Z a normalizer that makes the sum convex.

Structure Joint Embedding (SJE) [12]. The model defines a bilinear compatibility score between inputs and outputs of the form:

$$F(x, y; W) = \phi(x)^T W \psi(y), \tag{3}$$

where $\phi(x) \in \mathbb{R}^D$ and $\psi(y) \in \mathbb{R}^E$ correspond to the vector embeddings of x and y, respectively. Learning of the matrix W is posed as a max-marging structured output learning problem [21] and carried out via stochastic gradient descent (SGD) with a random initialization strategy. Predictions are made according to the argmax rule $\hat{y}(x) = \text{argmax}_{y\in\mathcal{Y}^{ts}} F(x, y; W)$. Note that in Eq. (3), we can see $W^T\phi(x)$ as a projection from the input to the output space, in which case the prediction rule corresponds to a dot-product similarity between output embeddings. The same analysis applies to $W\psi(y) \in \mathbb{R}^D$. The recently proposed SAE [14] algorithm is based in this idea, i.e. finding an compatibility matrix W that minimizes the projection error from input to output as well as from output to input.

A similar model was considered also in [15] but learning of the parameter matrix in Eq. (3) is posed as a regularized regression problem. A particular choice of the regularizers and its parameters allowed the authors to derive a closed-form solution that competes with SJE in terms of classification performance.

3 Attribute Inference for ZSL

In the standard as well as in the generalized ZSL settings, it is assumed we have access to a representation of the classes seen during training as well as for those exposed to the system during the testing phase. Such representations range from human generated attribute descriptors [6] to word embeddings learned from raw text [8,9]. In the later case, adding new classes is not an issue as long as the vocabulary on which the word embeddings have been learned is rich enough. However, if the classes are represented by means of visual attributes, each new category need to be characterized in terms of the existing set of visual attributes. If such a set was build to encode subtle differences in the visual domain, e.g. as in the fine-grained recognition case [18], adding new classes is more involved and it might require expert human knowledge and a non-negligible annotation effort. Nevertheless, attribute-based representations are very appealing as they have shown significantly better performance on ZSL problems compared to other approaches [11]. Based on the above, we consider the problem of learning ZSL scenario where the attributes for the testing set are not provided.

3.1 Model Description

To address the above problem we first introduce an attribute prediction function based on a notion of semantic similarity between the classes in the training and testing sets. The intuition behind this approach is that categories which are close in the semantic space should be characterized by a similar set of visual attributes. Figure 1 illustrates the idea behind our approach.

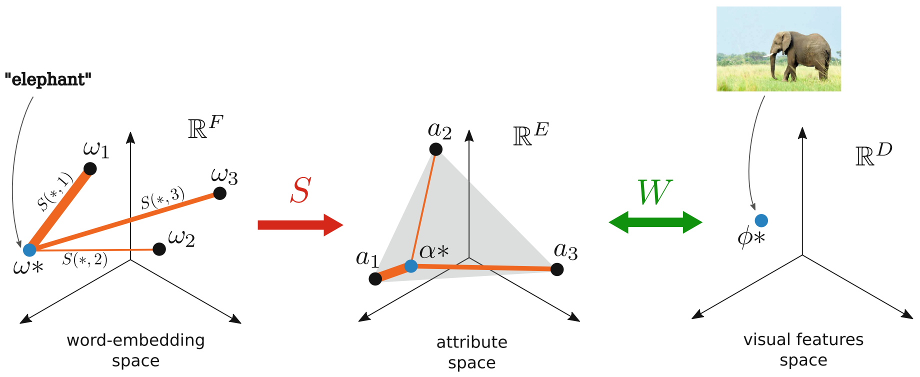


Fig. 1. Intuition behind our approach. We leverage word-vector embeddings to infer the attribute vector for a class. A bilinear map W relates the visual and inferred attribute representations.

To encode the notion of semantic relatedness between classes, we use as a proxy a similarity score between the word embeddings of the respective class names. Our prediction function is defined as follows:

$$\alpha(y) \equiv \alpha_y = \frac{1}{Z(y)} \sum_{y' \in \mathcal{Y}^{tr}} S(y, y') a_{y'}. \tag{4}$$

Here, $a_z \in \mathbb{R}^E$ is the attribute descriptor associated to class $z \in \mathcal{Y}^{tr}$, S denotes a semantic similarity measure defined between words in a vocabulary and $Z(y)$ is a normalizer. In our case, we define S through a similarity between word-vectors:

$$S(y, y') := s(\omega_y, \omega_{y'}) = \exp\{-\tau ||\omega_y - \omega_{y'}||^2\} \tag{5}$$

with $\omega_z \in \mathcal{R}^F$ the word embedding of class z and $\tau \in \mathbb{R}_+$ a free parameter. Note that by setting $Z(y) = 1$ and without imposing any constraint on $S(\cdot,\cdot)$, the α's generated by (4) lie in $\texttt{span}\{a_{y_1}, \cdots, a_{y_M}\}$ while if we set $Z(y) = \sum_{y' \in \mathcal{Y}^{tr}} S(y, y')$ and constrain $S(\cdot,\cdot)$ to $\mathbb{R}_+$, the transformed representations is restricted to lie within the convex hull of the set $\{a_{y_1}, \cdots, a_{y_M}\}$ (the gray region in the middle part of Fig. 1 illustrates this idea). We adopt the later definition in our formulation.

For each $y \in \mathcal{Y}^{tr} \cup \mathcal{Y}^{ts}$, we compute a new (transformed) attribute descriptor $\alpha(y)$ using Eq.(4). Note that this representation depend on the attributes for the training classes only. Next, following the SJE formulation of [12], we define a bilinear compatibility score between a class label y and an image x as follows:

$$f(x, y) = \phi(x)^T W \alpha(y) \tag{6}$$

with W a $D \times E$-matrix that has to be learned from data. Prediction for a new image x is performed as:

$$\hat{y} = \underset{y \in \mathcal{Y}^{ts}}{\operatorname{argmax}} f(x, y). \tag{7}$$

For learning, we adopt the formulation of [12] and fit W by minimizing of the following objective:

$$\frac{1}{M} \sum_{n=1}^{M} \max_{y \in \mathcal{Y}^{tr}} \left\{0, \Delta(y, y_n) + \phi(x_n)^T W (\alpha_y - \alpha_{y_n})\right\}, \tag{8}$$

with $M = |\mathcal{Y}^{tr}|$ and $\Delta(y, y') = 1$ if $y = y'$ and 0 otherwise. We optimize Eq. (8) using Stochastic Gradient Descent (SGD) and a fixed learning rate η.

Later, in Sect. 4.3, we will show that by introducing two simple modifications to the SJE processing pipeline, we can improve ZSL performance by a large margin. Such modifications include a PCA decorrelation and L2-normalization of the inputs and outputs and the use of the closed form solution found by the ESZSL method of [15] as initialization for the weight matrix W when learning the objective (8).

Differences with ConSE. The definition of the attribute prediction function (4) and the similarity score (5) resembles the embedding function of the ConSE [20, Eq. (2)] algorithm. There are, however, important difference that need to be highlighted. First, in ConSE, it is assumed that class embeddings (e.g. attributes) are available for *both* training and testing categories. In our case, we relay in an auxiliary and easy to compute representation of the classes to infer a hopefully more robust signature. Second, learning in ConSE takes place at the attribute generation step by means of standard classifiers trained to predict the classes in the training set. Zero-shot prediction follows a simple nearest neighbor rule on the semantic space. However, in our approach, we learn a mapping between the image feature space and the semantic space using the attributes inferred from an auxiliary representation.

3.2 Attribute Inference as a Multilinear Embedding Model

We can think of Eq. (4) as being the product of the attribute matrix $A_{tr} = \left(a_{y_1} \cdots a_{y_M}\right)$ and the vector $\nu(z) = \frac{1}{Z}\left(S(y_1, z), \cdots, S(y_M, z)\right)^T$, with $y_i \in \mathcal{Y}^{tr}$. In this case $\nu(z)$ can be regarded as a representation for the class z that encodes its similarity to the other classes known to the system during training. Based on this observation, Eqs. (4)–(6) can be condensed into a trilinear map, $g : \mathbb{R}^D \times \mathbb{R}^E \times \mathbb{R}^M \to \mathbb{R}$ of the form:

$$g(\phi(x), \alpha(y), \nu(y)) = \sum_{i,j,k} \phi_i(x)\alpha_j(y)\nu_k(y)\tilde{W}_{ijk}. \tag{9}$$

In our case, $\tilde{W}_{ijk} := W_{ij}$, $\forall k$. Equation (9) provides an explicit mechanism for handling data with multiple modalities. Nevertheless, a comprehensive evaluation of such an approach is beyond the scope of the paper and it will be the subject of future research.

4 Experiments

In this section we provide an experimental evaluation regarding different aspects of our work and compare the proposed approach against other methods in the literature.

4.1 Datasets and Evaluation Protocols

For evaluation, we use two datasets commonly used in the literature: the Caltech UCSD Birds (CUB) [18] and Animals-with-Attributes (AWA) [6]. CUB is a fine grained dataset containing 11788 images of 200 different bird species. AWA contains 30475 images of 50 different animals. For CUB, we use the same zero-shot splits of [10] with 150 classes for training and validation and the remaining (disjoint) 50 classes for testing. For AWA, we use the predefined split with 40 classes for training+validation and 10 for testing. Table 1 show summary statistics for these datasets. ZSL performace is measured on both cases as the average accuracy on the test set.

Table 1. Summary statistics for the CUB and AWA datasets.

	AWA	CUB
Attributes	85	312
Training classes	40	150
Test classes	10	50
Instances	30475	11788

4.2 Input and Output Embeddings

To encode the images we rely on features extracted from a pre-trained deep convolutional neural network (CNN). In our experiments, we use features from three different models: GoogLeNet [22], VGG19 [16] and ResNet101 [17]. For AWA we only use the VGG19 features provided by the authors of the dataset[2] since images are not longer available due to copyright restrictions. As output encodings, we use use the 85- and 312-dimensional continuous attribute vectors provided with AWA and CUB, respectively. In some of the experiments, we also use word2vec [8] and GloVe [9] vectors provided by [10] as a proxy for attribute inference. Unless stated otherwise, both input and output features are normalized to have unit Euclidean norm.

4.3 An Improved Variant of SJE

Our model, as introduced in Sect. 3.1, is a two-stages approach consisting of an attribute inference step (Eq. (4)) followed by a model learning step (Eqs. (6)–(8)), with the later following the SJE formulation of [12]. In this subsection, we show experimentally that by introducing two simple modifications to the SJE pipeline, we can improve ZSL performance by a large margin. Such modifications are the following:

- *PCA decorrelation and L_2 normalization.* We apply a PCA projection step followed by an L_2-normalization on the inputs and the outputs. Inputs (CNN features) are projected onto a space of the same dimensionality. For the outputs (attributes and word-vectors), since we have a single sample per class to learn the projections, we project the data onto a space of $|\mathcal{Y}|^{tr}$ dimensions. In this case, to compute the projection matrix, we regularize the output sample covariance by adding a small constant to its diagonal in order to avoid numerical instabilities.
- *SGD initialization by ESZSL* [15]. We use the closed form solution found by the ESZSL method of [15] as initialization for the weight matrix W when learning the objective (8).

We name the improved SJE variant as SJE++. We show in Table 2 the performance gain brought by SJE++ compared to the original results of [12] (SJE(O)) and those obtained with an own implementation of the method (SJE(R)). Parameters of our model (*tau* in Eq. (5)) where chosen by cross-validation on the training set. ZSL performance is shown for different combinations of input and output embeddings. From the table, we observe that performance is fairly consistent between the original (O) and reproduced (R) implementations. The small differences observed can be attributed to difference in the random initialization as well as differences in the SGD iterations. We observe also that, compared to the standard SJE algorithm, the improved version of the algorithm exhibits a significant improvements on most configurations.

[2] https://cvml.ist.ac.at/AWA/.

Although not shown in the table, we observed that most of the improvement comes from the PCA + normalization step, with the ESZSL initialization bringing at most 1 point of absolute improvement for both AWA and CUB. In this case, the improvement brought by the feature transformation step can be explained by considering the granularity of the visual concepts (classes) being modeled, i.e. fine-grained bird species for CUB and coarse-grained animal categories for AWA. In the case of CUB, the PCA projection step helps in disentangling the subtle differences between the representations on both the visual and semantic spaces, preconditioning the optimization problem given by Eq. (8).

Table 2. Comparison of the standard and improved SJE formulations on the CUB and AWA datasets for different combinations of input and output embeddings (attr.: attributes, w2v: word2vec).

		SJE(O)		SJE(R)		SJE++		Improvement	
		attr.	w2v	attr.	w2v	attr.	w2v	attr.	w2v
CUB	GoogLeNet	50.10	28.40	49.81	28.40	54.69	34.47	+4.88	+6.07
	ResNet	-	-	56.02	30.96	59.94	36.82	+3.92	+5.86
	VGG19	-	-	49.13	25.43	49.98	34.47	+0.85	+9.04
AWA	GoogLeNet	66.70	51.20	69.03	49.24	65.66	54.14	−3.37	+4.90
	VGG19	-	-	81.32	61.62	81.02	68.40	−0.30	+6.78

In Table 3, we compare the performance of the SJE++ model to other methods proposed in the literature, for different combinations of input and output features on the AWA and CUB datasets. Results for all the methods other than SJE++ correspond to those reported in the survey of Xian *et al.* [11]. From the table, it can be seen that the improved algorithm outperforms most of the methods by a considerable margin.

4.4 Attribute Inference Evaluation

In this subsection, we evaluate the performance of our approach on the challenging problem of zero-shot classification with partial attributes, i.e. when visual attribute are only available for the training classes. We report results using the same split as in [12]. Table 4 compare the performance of our attribute inference approach (α-attr.) discussed in Sect. 3 with the standard and improved SJE models trained on word2vec data. We can see from the table that most of the gain comes from the use of the SJE++ model while the improvements brought by the α-attr. is mostly marginal. The only case when there is a clear improvement on performance is with GloVe vectors, where we can observe an improvement of almost 5 points with respect to the SJE++ baseline.

Table 3. Comparison of the proposed method against the state-of-the-art methods for attribute-based zero-shot classification on the AWA and CUB datasets.

	ResNet		VGG19		GoogLeNet	
	AWA	CUB	AWA	CUB	AWA	CUB
DAP [6]	57.1	37.5	57.23	-	-	-
SSE [13]	68.8	43.3	76.33	30.41	-	-
LATEM [23]	74.8	49.4	-	-	-	-
SYNC [24]	72.2	54.1	-	-	-	-
ConSe [20]	63.6	36.7	-	-	-	-
ALE [10]	78.6	53.2	-	-	-	-
SJE [12]	76.2	55.3	-	-	66.7	50.1
ESZSL [15]	74.7	55.1	75.32	-	-	-
SJE++	-	59.94	81.02	49.98	65.66	54.69

Table 4. ZSL with partial attributes. Performance of our attribute inference approach (α-attr.) compared to the SJE and SJE++ models trained on word2vec and GloVe vectors.

	CUB			AWA				
	SJE	SJE++	α-attr.	SJE	SJE++	α-attr.	SJE++	α-attr.
	w2v	w2v	ω =w2v	w2v	w2v	ω=w2v	GloVe	ω =GloVe
GoogLeNet	28.40	34.47	34.57	49.24	54.14	51.17	65.05	64.92
ResNet	30.96	36.82	36.75	-	-	-	-	-
VGG19	25.43	34.47	33.99	61.62	68.82	64.11	68.38	73.33

5 Conclusion

In this work, we considered a variation of the attribute-based zero-shot classification problem where the class attributes are not available at test time. We proposed to leverage additional sources of semantic information (word embeddings) as a proxy for attribute inference. The rationale behind this approach is that, while attribute might be costly to define for a new class, other embeddings as word2vec or GloVe can be computed more easily. We proposed a two stages approach consisting of an attribute inference step, followed by a input-output similarity computation. For the later, we base our approach on the SJE model of Akata *et al.* and propose two simple modifications that lead to a significant boost in classification performance. Experimental results on the AWA and CUB datasets show the benefits of the approach.

References

1. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vision **115**(3), 211–252 (2015)
2. Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
3. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: large-scale scene recognition from abbey to zoo. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3485–3492. IEEE (2010)
4. Mensink, T., Verbeek, J., Perronnin, F., Csurka, G.: Distance-based image classification: generalizing to new classes at near-zero cost. IEEE Trans. Pattern Anal. Mach. Intell. **35**(11), 2624–2637 (2013)
5. Hariharan, B., Girshick, R.: Low-shot visual object recognition. arXiv preprint arXiv:1606.02819 (2016)
6. Lampert, C.H., Nickisch, H., Harmeling, S.: Learning to detect unseen object classes by between-class attribute transfer. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp. 951–958. IEEE (2009)
7. Chao, W.-L., Changpinyo, S., Gong, B., Sha, F.: An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 52–68. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_4
8. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR abs/1301.3781 (2013)
9. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
10. Akata, Z., Perronnin, F., Harchaoui, Z., Schmid, C.: Label-embedding for attribute-based classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 819–826 (2013)
11. Xian, Y., Schiele, B., Akata, Z.: Zero-shot learning-the good, the bad and the ugly. arXiv preprint arXiv:1703.04394 (2017)
12. Akata, Z., Reed, S., Walter, D., Lee, H., Schiele, B.: Evaluation of output embeddings for fine-grained image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2927–2936 (2015)
13. Zhang, Z., Saligrama, V.: Zero-shot learning via semantic similarity embedding. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4166–4174 (2015)
14. Kodirov, E., Xiang, T., Gong, S.: Semantic autoencoder for zero-shot learning. In: IEEE CVPR 2017 (2017)
15. Romera-Paredes, B., Torr, P.: An embarrassingly simple approach to zero-shot learning. In: International Conference on Machine Learning, pp. 2152–2161 (2015)
16. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

18. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset (2011)
19. Weston, J., Bengio, S., Usunier, N.: Wsabie: scaling up to large vocabulary image annotation. IJCAI **11**, 2764–2770 (2011)
20. Norouzi, M., Mikolov, T., Bengio, S., Singer, Y., Shlens, J., Frome, A., Corrado, G.S., Dean, J.: Zero-shot learning by convex combination of semantic embeddings. In: International Conference on Learning Representations (ICLR) (2014)
21. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. J. Mach. Learn. Res. **6**, 1453–1484 (2005)
22. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
23. Xian, Y., Akata, Z., Sharma, G., Nguyen, Q., Hein, M., Schiele, B.: Latent embeddings for zero-shot classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 69–77 (2016)
24. Changpinyo, S., Chao, W.L., Gong, B., Sha, F.: Synthesized classifiers for zero-shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5327–5336 (2016)

Author Index