




# Seismic Event Classification Using Spectrograms and Deep Neural Nets

Aaron Salazar , Rodrigo Arroyo , Noel Pérez , and Diego S. Benítez  

Colegio de Ciencias e Ingenierías “El Politécnico”,  
Universidad San Francisco de Quito USFQ, Quito 170157, Ecuador  
{assalazar,rodrigo.arroyo}@alumni.usfq.edu.ec,  
{nperez,dbenitez}@usfq.edu.ec

**Abstract.** In this work, we proposed a new method to classify long-period and volcano-tectonic spectrogram images using eight different deep learning architectures. The developed method used three deep convolutional neural networks named DCNN1, DCNN2, and DCNN3, three deep convolutional neural networks combined with deep recurrent neural networks named DCNN-RNN1, DCNN-RNN2, and DCNN-RNN3, and two autoencoder neural networks named AE1 and AE2, to maximize the area under the curve of the receiver operating characteristic scores on a dataset of volcano seismic spectrogram images. The three deep recurrent neural network-based models reached the worst results due to the overfitting produced by the small number of samples in the training sets. The DCNN1 overcame the remaining models by obtaining an area under the curve of the receiver operating characteristic and accuracy scores of 0.98 and 95%, respectively. Although these values were not the highest values per metric, they did not represent statistical differences against other results obtained by more algorithmically complex models. The proposed DCNN1 model showed similar or superior performance compared to the majority of the state of the art methods in terms of accuracy. Therefore it can be considered a successful scheme to classify LP and VT seismic events based on their spectrogram images.

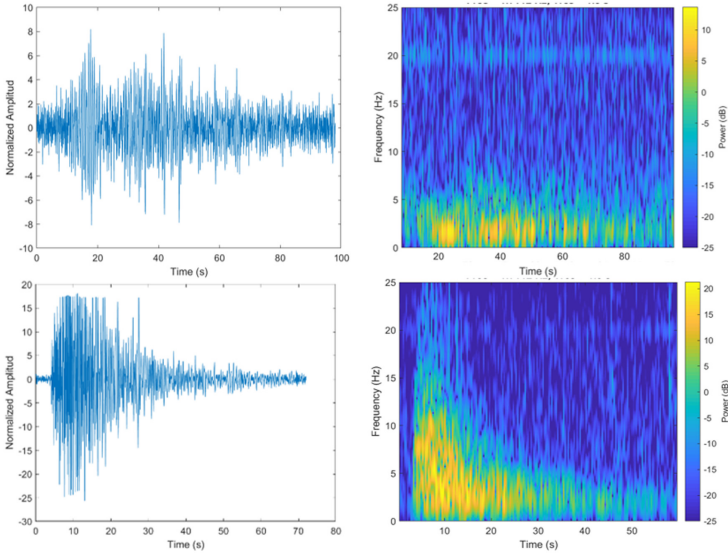
**Keywords:** Volcanic seismic event classification · Deep-learning models · Artificial intelligence · Spectrogram images

## 1 Introduction

Volcanic activity monitoring systems are essential to detect early signs of volcanic unrest and possible reawakening that can lead to eruptions [31]. Amongst the techniques used by scientists to estimate activity inside a volcano, the seismicity is one of the most effective tools for monitoring and forecasting eruptions [29]. In this regard, a wide variety of approaches have been used in recent years

---

Work funded by Universidad San Francisco de Quito (USFQ) through the Poli-Grants Program under Grants no. 10100, 12494 and 16916.



**Fig. 1.** An LP (top row) and VT (bottom row) seismic signals examples and their respective spectrogram. Taken from [27].

to address the problem of volcano seismic events classification, e.g., long-period (LP) and volcano-tectonic (VT) seismic events, as shown in Fig. 1. Machine learning classifiers (MLC) such as hidden Markov models (HMM) [1], boosting strategies [34], decision trees (DT) [16], random forest (RF) [26,28], Gaussian mixture models (GMM) [33], support vector machine (SVM) methods [7,26], and artificial neural networks (ANN) [2,5] were combined with classical time, frequency and scale domain features and non traditional features such as intensity statistic, shape and texture features extracted from the spectrogram images [26] to differentiate seismic events.

On the other hand, convolutional neural networks (CNN) are particular ANN architectures that are gaining more attention in image analysis contexts [30]. They avoid using intermediate, fully connected layers to employ pooling ones and thus optimizing the information pass-through from layer to layer. Lately, there is evidence of using deep learning techniques to analyze the seismic activity of volcanoes, e.g., deep neural networks to classify feature vectors computed from the time-domain signals [32], deep CNN models to classify spectrogram images [6]. Another well known deep learning architecture is the one based on recurrent neural network (RNN) where information flows sequentially, it is shared between layers and kept as a factor for decision making during the weight calculations [18]. The combination of both the CNN and RNN approach is quite possible, as demonstrated in [36], to classify different objects on individual images. However, the model growing is a must concern aspect. The more internal layers are included in the desired model, the more complex it will be [3].

There are other deep learning-based approaches for object detection [18,36] and classification [37]. But, the use of deep learning techniques in the context of volcano seismic event classification based on their spectrogram images is still limited. Therefore, in this work, we explore the use of eight different deep learning architectures to classify LP and VT spectrogram images to maximize the area under the curve (AUC) of the receiver operating characteristics curve on a dataset of volcano seismic spectrogram images from the Cotopaxi volcano, in Ecuador.

## 2 Materials and Methods

### 2.1 Spectrogram Images Dataset

This work considered the use of a public dataset (*MicSigV1*) from the ESeismic<sup>1</sup> repository, which contains several seismic event samples recorded at the Cotopaxi volcano [27]. It has a total of 1187 seismic records from two different seismic stations (VC1 and BREF) installed at the Cotopaxi volcano. This dataset contains samples distributed in five classes: LP, VT, regional (REG), hybrid (HB), and icequakes (ICE). Due to the small number of samples from REG, HB, and ICE events, we considered only the LP and VT events belonging to the same seismic station (BREF) to guarantee the same acquisition protocol and to avoid mixed signals. Therefore, the formed experimental dataset contains 668 spectrogram images (587 of LP and 81 of VT).

### 2.2 Deep-Learning Networks

Deep learning can enhance computational models by including multiple layers to process large amounts of data and to improve the learning process. Thus, severe problems regarding image classification and recognition in the past are presently easier to tackle. The deep CNN and RNN are two exclusive deep learning models [18,36], which are increasing their popularity on sequential data analysis and image labeling, respectively.

The deep CNN is a multilayered approach of conventional convolutional neural networks that include an input layer, a set of hidden layers (which could vary depending on the network architecture from two to hundreds of layers), and an output layer (fully connected layer). In deep CNN learning, each hidden layer is mainly composed of the CNN architecture core, consisting of at least the convolutional and max-pooling layers. Other configurations extend the basic scheme by adding dropout and flatten layers. This multi-layer structure enables the network to learn different data abstractions while transitioning from layer to layer until reaching the output result [6].

---

<sup>1</sup> ESeismic repository was provided by courtesy of the Instituto Geofísico of Escuela Politécnica Nacional (IGEPN) and collaborators, and it is available at [http://www.igepn.edu.ec/eseismic\\_web\\_site/index.php](http://www.igepn.edu.ec/eseismic_web_site/index.php). Please note that you must register and complete a disclaimer agreement to obtain the data.

The deep learning RNN is based on the classic feed-forward ANN architecture, but it includes an extra working piece called loops in connections. In contrast to the feed-forward ANN, the RNN architecture processes the inputs in a sequential way considering a recurrent hidden state in which the current activation is dependant on the previous step activation. The main drawback is related to long-term sequential data, where the gradients tend to vanish during the training. However, there is a more sophisticated approach to design recurrent units and to avoid vanishing problems known as long short-term memory [10]. It allows for recurrent units to learn long-term dependencies, which are a vital key when developing deep RNN models [24].

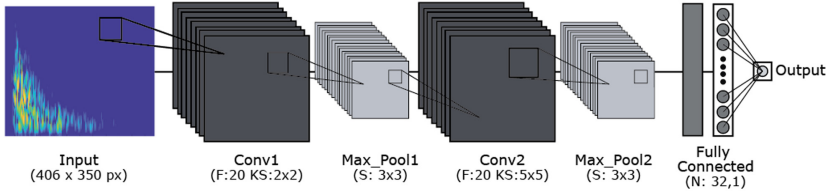
The deep learning autoencoder (AE) architecture is based on a classic AE artificial neural network. It efficiently learns compressed representations (encodings) of the data, typically for dimensionality reduction, by training the network to ignore the noise (signal). This type of architecture utilizes a bottleneck structure reducing the neurons in each layer as well as the volume of information that passes through the entire network reaching the latent space representation. Several variants to the basic AE model have been proven to be effective in learning representations for classification tasks [35], face recognition [9], and to extract the semantic meaning of words [19]. Thus, an adequate AE architecture will be able to recognize the useful features of the input data, while avoiding the redundant ones and the overfitting.

### 2.3 Proposed Method

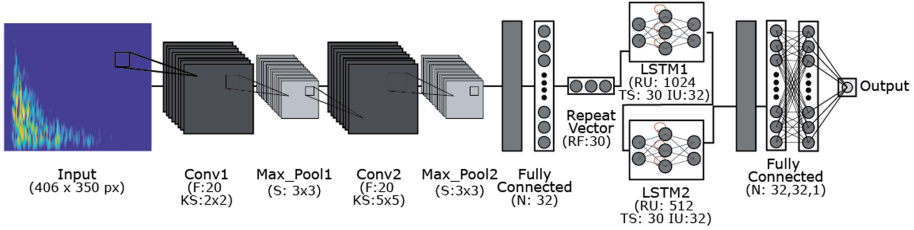
We adopted the deep CNN, RNN, and AE neural networks to build the proposed method, which extends these neural networks to eight different deep learning architectures: DCNN1, DCNN2, DCNN3, DCNN-RNN1, DCNN-RNN2, DCNN-RNN3, AE1, and AE2. For a better explanation of the proposed method, we focus our description in the DCNN1, DCNN-RNN1, and AE1 models.

The DCNN1 architecture is composed of several layers, as it is shown in Fig. 2. From this figure, it is possible to read that the spectrogram images are used to feed the first convolutional layer composed of 16 convolutional filters with a  $3 \times 3$  kernel size each. This layer aims to predict the class probabilities of the input sample by creating a feature map representation computed by the structure of the filters. Subsequently, the feature map enters the pooling layer with a  $4 \times 4$  kernel size each to reduce irrelevant features (information) while retaining the relevant ones. Then, the reduced feature space is used to feed another convolutional and pooling layer with the same configurations as the previous ones. This second convolutional module concentrated the most relevant (important) features to classify the input sample. Finally, the fully connected layer consists of two dense layers, the flatten to convert the reduced bi-dimensional input feature space into a single feature vector with its corresponding weights and the output layer, which provides the final classification of the feature vector using a sigmoid function.

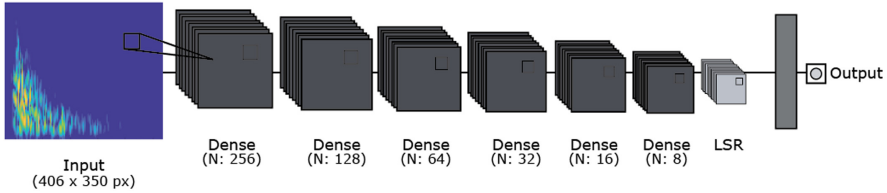
The DCNN-RNN1 architecture is a mixed model that combines a two convolutional layer based deep CNN architecture with some extra RNN layers, as



**Fig. 2.** The DCNN1 architecture of the proposed method; F - number of filters; KS - convolutional kernel size; S - max pool kernel size; N - number of neurons.



**Fig. 3.** The DCNN-RNN1 architecture of the proposed method; F - number of filters; KS - convolutional kernel size; S - max pool kernel size; N - number of neurons; RF - repeat factor; RU - recurrent units; TS - time stamp; IU - input units.



**Fig. 4.** The AE1 architecture of the proposed method; N - number of neurons; LSR - latent space representation

it is shown in Fig. 3. In such a sense, the first convolutional layer (Conv1.) used 32 convolutional filters with a  $2 \times 2$  kernel size each and a pooling layer (Max\_pool1) with a  $3 \times 3$  kernel size. The second convolutional unit (Conv2. and Max\_pool2 layers) used the same number of filters as in the Conv1., but it increased the convolutional kernel size to  $5 \times 5$ . The Max\_pool2 layer remained as equal as the Max\_pool1 in terms of configurations. Then, a flatten and dense layer transforms the bidimensional feature space into a single feature vector that is the input to a repeat vector layer. The later transforms the input feature vector into a data stream that is propagated with a repeat factor of 30 as input to the first long-short term memory (LSTM) unit. The LSTM1 unit is composed of 1024 recurrent units (RU) with input shape (IS) size of 30 and input units (IU) number of 32. Then, the LSTM1 layer output feeds the LSTM2 unit that is set to 512 RU and the same input shape size and number of input units as the

LSTM1 layer. After that, a fully connected layer containing a flatten and dense layers with sigmoid function provide the final classification.

On the other hand, the AE1 architecture is composed of multiple dense layers that are gradually reducing the number of neurons per layer, simulating a bottleneck workflow, as shown in Fig. 4. From this figure, it is possible to observe that the input image transits throughout this architecture, starting with a dense layer with 256 neurons. Subsequently, there are five more dense layers, in which the total number of neurons is reduced from layer to layer by a power factor of two until reaching a total of 8 neurons in the fifth layer. After that, it is possible to find the latent space representation (LSR) layer, which holds the compressed data that passed through all the layers and uses them to generate the prediction. Then, a flattened layer that is connected to the final dense layer with one neuron returns the final output. The deep-learning of this architecture is benefited from the data compression for better representation [20].

The remaining architectures, DCNN2, DCNN3, DCNN-RNN2, DCNN-RNN3, and AE2, follow the same base architecture (described here), varying the layers configurations and hyperparameters. The other architectures are summarized next:

The DCNN2 model contains three convolutional layers with 32, 64, and 128 filters with a kernel size of  $3 \times 3$  each. Three max-pooling layers (one by each convolutional layer) with a pool size of  $6 \times 6$  each, one flatten layer and a fully connected layer (output) composed of three dense layers (32, 32 and 1 neurons). The DCNN3 model uses two convolutional layers with 20 filters each, and a kernel size of  $2 \times 2$  and  $3 \times 3$ , respectively. Two max-pooling layers (one by each convolutional layer) with a pool size of  $3 \times 3$ , one flatten layer, and a fully connected layer (output), containing three dense layers (32, 32, and 1 neurons).

The DCNN-RNN2 model employs three convolutional layers with 20 filters each and kernel size of  $2 \times 2$ ,  $2 \times 2$ , and  $5 \times 5$ , respectively. Three max-pooling layers (one by each convolutional layer) with a pool size of  $3 \times 3$ , one flatten layer, one dense layer with 32 neurons, one repeat vector layer with a repetition factor of 30 units, two LSTM layers: the first one with 1024 recurrent units and input shape  $30 \times 32$ , and the second one with 512 recurrent units and a fully connected layer (output), composed of three dense layers (32, 32 and 1 neurons). The DCNN-RNN3 model involves three convolutional layers with 32 filters each and with a kernel size of  $2 \times 2$  each. Three max-pooling layers (one by each convolutional layer) with a pool size of  $3 \times 3$ , one flatten layer, one dense layer with 32 neurons, one repeat vector layer with a repetition factor of 30 units, two LSTM layers with 512 and 256 recurrent units, an input shape of  $30 \times 32$ , and a fully connected layer (output), containing three dense layers (32, 32 and 1 neurons).

The AE2 model used a similar architecture as the AE1 model, but with four dense layers in its composition. In this case, the input image enters the initial dense layer with 32 neurons and transits through three more layers with a reduced number of neurons each (by the power of two of the previous layer), until reaching the last dense layer with four neurons. Then, the LSR and the

flatten layers evaluate the final characteristics to obtain the final result in the output layer.

## 2.4 Experimental Setup

**Spectrogram Image Preprocessing:** All spectrogram images were down-scaled to 50% from their original size, thus decreasing the volume of information used to feed the learning models. The dataset provides spectrogram images without noise; therefore, the seismic event pattern presented on each image is invariant to the downscaling operation. This operation is use frequently in image analysis context with deep learning [6]. Besides, the pixels values of each spectrogram image were normalized using the min-max method [12] to bring them into the range of 0 to 1, thus, avoiding data dispersion. Besides, we used a data augmentation technique to increase and balance the number of samples per class. Thus, each spectrogram image underwent shearing, scaling, and rotation operations, as defined in [23]. Affinity transformations are widely used [6] and allowed us to reach a total of 1108 spectrogram images, which reinforces the models learning process by training them with more samples per class, helping to avoid overfitting.

**Training and Test Partitions:** The stratified 10-fold cross-validation method [21] was applied before the classification step to build disjoint training and test partitions and to ensure the sample ratio between both types of events for all folds. Thus, individual deep learning models were trained using different training sets, which enable it to learn from different input space representations. Testing on these different sets promotes trustworthy resulting variability in the classification of individual samples.

**Deep Architectures Configurations:** For all models, we configured three main hyperparameters to explore the proposed method limits. Thus, the number of iterations (epochs) was set from 50 to 150 with increment step of 50 units; the batch size was tuned to 16, 32 and 64 units, and the learning rate used the *adam* optimizer, which is based on adaptive estimation of lower-order moments [14]. This optimizer was designed to combine the advantages of the well-known optimizers *AdaGrad* and *RMSPProp* [15].

**Validation Metrics:** The classification performance of the proposed method was based on the AUC and accuracy (ACC) metrics. The statistical comparison among all classification schemes was conducted using the Wilcoxon test with a significance decision value of 5% ( $\alpha = 0.05$ ) for a two-tailed test [11]. This test ranks the differences in performances of two MLCs [8] and thus, allowed us to select the best classification model.



**Selection Criteria:** The best model was selected based on the following criteria: (1) the model with the statistically highest AUC score by architecture, (2) if there was a tie rating performance in the AUC scores, the one that has the lowest algorithm complexity is preferred, and (3) the statistically highest AUC score among all models selected according to the previous two rules. More than one model can be chosen per architecture if there is no significant AUC-based difference between them. This exception is only valid for intra-architectural analysis. Therefore, the proposed method provides only one classification model as a result.

All implementations were done in Python programming language version 3.7.4 using *scikit-learn (SKlearn)* [25], Keras [4] with *ImageDataGenerator* and TensorFlow backend, and *sciPy* for statistical analysis [13].

### 3 Results and Discussion

#### 3.1 Performance Evaluation of the Proposed Method

The DCNN1 architecture provided seven out of nine classification models using the first selection criterion. This set of classifiers did not represent statistical differences in terms of AUC performance when compared to each other. The AUC range of variation was above the 0.95, which is an outstanding classification threshold for any classification problem. Although the highest AUC score of 0.99 was reached by the model using a batch size of 32 units and 150 epochs (iterations), the remaining models performed similarly statistically. According to the second selection criterion, the selected classification model in this architecture is the one implementing a batch size of 32 units, 50 epochs, and AUC score of 0.98 (see Table 1, bold line).

Likewise, DCNN2 architecture was able to produce six out of nine classification models that were similar statistically in AUC performances. The range of AUC variation in this set was between 0.71 and 0.79, which are not good enough scores to tackle the problem at hand. The highest AUC score of 0.79 was reached by the model with 16 units of batch size and 100 epochs. But, the model composed of the same batch size and 50 epochs, which obtained an AUC value of 0.71 was selected as the best model from this architecture, taking into consideration the second selection criterion (see Table 1, bold line). Similarly, in the DCNN3 architecture, a total of five out of nine classifiers were highlighted as classification models without statistical difference among them. The AUC scores varied from 0.90 to 0.94, which are considered reasonable scores in the context of spectrogram images classifications. The highest AUC value of 0.94 was obtained by the model composed of a batch size of 64 units and 50 epochs. However, there was another model using the same number of epochs as the highest model, batch size of 16, and AUC score of 0.91, which was selected as the best model inside this architecture according to the second selection criterion (see Table 1, bold line).

The AE1 architecture produced six out of nine models using the first selection criterion. They did not present a statistically significant difference among them



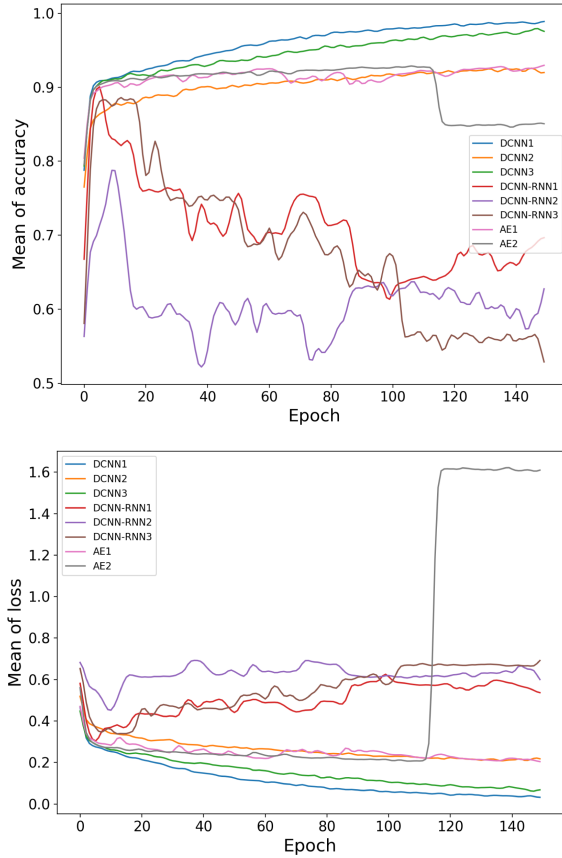
**Table 1.** Performance results of deep learning models selected by the first selection criterion.

| Architecture | AE Dense layer (n).       | Conv. layer (f) | Kernel size | Pool size per layer | FC layer (n) | Batch size | Epochs (u) | AUC         | Wilcoxon at $\alpha = 0.05$ (p value) | ACC (%)         |
|--------------|---------------------------|-----------------|-------------|---------------------|--------------|------------|------------|-------------|---------------------------------------|-----------------|
| DCNN1        | -                         | (16, 16)        | (3 × 3)     | (4 × 4)             | (32, 1)      | <b>32</b>  | <b>50</b>  | <b>0.98</b> | <b>0.19</b>                           | 95              |
|              |                           |                 |             |                     |              | 64         | 50         | 0.99        | 0.45                                  | 97              |
|              |                           |                 |             |                     |              | 16         | 100        | 0.99        | 0.41                                  | 96              |
|              |                           |                 |             |                     |              | 32         | 100        | 0.98        | 0.16                                  | 98              |
|              |                           |                 |             |                     |              | 64         | 100        | 0.99        | 0.50                                  | 99              |
|              |                           |                 |             |                     |              | 32         | 150        | <u>0.99</u> | -                                     | 99              |
|              |                           |                 |             |                     |              | 64         | 150        | 0.98        | 0.45                                  | 99              |
| DCNN2        | -                         | (32, 64, 128)   | (3 × 3)     | (6 × 6)             | (32, 32, 1)  | <b>16</b>  | <b>50</b>  | <b>0.71</b> | <b>0.15</b>                           | 90              |
|              |                           |                 |             |                     |              | 32         | 50         | 0.74        | 0.45                                  | 91              |
|              |                           |                 |             |                     |              | 64         | 50         | 0.73        | 0.33                                  | 92              |
|              |                           |                 |             |                     |              | 16         | 100        | <u>0.79</u> | -                                     | 92              |
|              |                           |                 |             |                     |              | 32         | 100        | 0.72        | 0.23                                  | 93              |
|              |                           |                 |             |                     |              | 64         | 100        | 0.72        | 0.33                                  | 94              |
|              |                           |                 |             |                     |              | DCNN3      | -          | (20, 20)    | (2 × 2)                               | (2 × 2) (3 × 3) |
| AE1          | (256, 128, 64, 32, 16, 8) | -               | -           | -                   | (1)          | <b>16</b>  | <b>50</b>  | <b>0.89</b> | <b>0.24</b>                           | 89              |
|              |                           |                 |             |                     |              | 64         | 50         | <u>0.93</u> | -                                     | 92              |
|              |                           |                 |             |                     |              | 32         | 100        | 0.89        | 0.50                                  | 89              |
|              |                           |                 |             |                     |              | 64         | 100        | 0.89        | 0.40                                  | 91              |
|              |                           |                 |             |                     |              | 32         | 150        | 0.87        | 0.40                                  | 87              |
| 64           | 150                       | 0.90            | 0.40        | 90                  |              |            |            |             |                                       |                 |
| AE2          | (32, 16, 8, 4)            | -               | -           | -                   | (1)          | <b>16</b>  | <b>50</b>  | <b>0.84</b> | -                                     | 88              |
|              |                           |                 |             |                     |              | 32         | 50         | 0.80        | 0.41                                  | 92              |
|              |                           |                 |             |                     |              | 64         | 50         | 0.73        | 0.30                                  | 95              |
|              |                           |                 |             |                     |              | 16         | 100        | 0.70        | 0.41                                  | 92              |
|              |                           |                 |             |                     |              | 64         | 100        | 0.70        | 0.41                                  | 92              |

Conv.- convolutional; f- number of filters per layer; n- number of neurons per layer; FC- fully; connected; u- units; AUC and ACC - mean of AUC and ACC metrics over ten folds; underlined AUC value is the Wilcoxon test pivot value; ACC - mean of accuracy.

in terms of AUC performances. The variation of AUC scores was in the range between 0.87 to 0.93, which is considered as a reasonable performance. Even though the highest score of 0.93 was obtained by the model with a batch size of 64 units and 50 epochs, the selected classification model in this architecture was the one using a batch size of 16 units, 50 epochs, and AUC score of 0.89 (see Table 1, bold line). Similarly, in the AE2 architecture, only four out of nine models did not produce AUC-based statistical differences among them. The AUC scores varied between 0.70 and 0.84. These results evidenced poor performances in the volcano activity context. In this architecture, the highest AUC performance (0.84) and the lowest algorithm complexity (batch size of 16 units and 50 epochs) were reached by the same classification model. Thus, it was selected as the best model on this architecture (see Table 1, bold line).

The combined classification models based on deep CNN and RNN architectures were the worst in terms of AUC performances. The three explored architectures provided AUC scores of 0.50 on all classification models, which means very poor schemes generalization. This effect is extremely linked to the number of samples employed during the models training. Despite using the data augmentation technique and the 10-fold cross-validation method on the experimental



**Fig. 5.** Performance of proposed deep learning models based on the mean of the accuracy (left) and loss function (right) over ten folds.

dataset before feeding the classifiers, they incur in a poor generalization power (see Fig. 5, top plot) and a weak learning (Fig. 5, bottom plot); both causes are symptoms of overfitting. It should be noted that the mean of the loss function never meets the established learning rate on these models, suggesting that more samples are required in the training process.

According to the first two selection criteria, the deep CNN-based classification models provided evidence of successful performance without incurring on overfitting. They performed over the 90% of the mean of ACC in the validation and the loss values converged to the learning rate across the defined epochs (see Fig. 5, top and bottom plots). Despite the good classification performances, the best selection model in the DCNN2 based architecture reached AUC and ACC scores of 0.71 and 90%, respectively. These values are statistically lower ( $p < 0.05$ ) when compared to the best model selection inside the DCNN3 architecture, which achieved AUC and ACC scores of 0.91 and 94%, respectively.

**Table 2.** Comparison based on the ACC between related previous works and the best selected model produced in this work.

| Method             | Number of samples | Computed features | Spectrogram images | ACC* (%)  |
|--------------------|-------------------|-------------------|--------------------|-----------|
| ANN [16]           | 914               | 6                 | No                 | 97        |
| DT [16]            | 914               | 3                 | No                 | 96        |
| ANN [26]           | 637               | 17                | No                 | 95        |
| RF [26]            | 637               | 17                | No                 | 93        |
| Linear SVM [17]    | 914               | 5                 | No                 | 97        |
| ANN [5]            | 1033              | 8                 | No                 | 94        |
| HMM [1]            | 512               | 39                | No                 | 90        |
| GMM [33]           | 667               | 2                 | No                 | 94        |
| CNN [6]            | 15895             |                   | Yes                | 97        |
| SVM [22]           | 105000            | 102               | Yes                | 92        |
| <b>DCNN1 model</b> | <b>1108</b>       |                   | <b>Yes</b>         | <b>95</b> |

ACC - accuracy; \*values rounded to the closest integer

The difference in performance is linked to the model complexity inherited from its architecture and the number of samples used to train it. The DCNN2 architecture is the most complex among all the developed deep CNN architectures. Thus, it is very reasonable to assume that this model needs more samples and epochs to learn the feature space properly (see Fig. 5, bottom plot).

Moreover, the selected classification model using the DCNN1 architecture provided the best performances on both validation metrics. It obtained scores of 0.98 and 95% for the AUC and ACC metrics, respectively. It statistically ( $p < 0.05$ ) overcomes the performance of the remaining models (see Table 1). This success is related to the DCNN1 architecture, which employed two convolutional layers with only 16 neurons (filters) per layer (lower than the DCNN3 architecture). Thus, it was able to learn from the provided features space satisfactorily (see Fig. 5, bottom plot). Regarding the third selection criterion, the selected classification model of the DCNN1 architecture constituted the proposed method output and the most appropriate classifier to face the problem of volcano spectrogram image classification.

### 3.2 State of the Art Based Comparison

Although it is not possible to make a direct statistical comparison against some previously developed state of the art methods such as those developed in [1, 5, 6, 22], because they used different experimental conditions, we aimed to carry out the comparison based on the ACC scores reported by them, as it is shown in Table 2. The majority of presented machine learning models reached ACC scores ranging from 90 to 97%, being the linear SVM and ANN the models which provided the higher classification performance. The proposed method has similar

and superior performance compared to several states of the art methods in terms of ACC scores. That was possible because deep learning-based approaches are able to learn data abstraction from layer to layer, using different mathematical functions. Meanwhile, machine learning methods, except for nonlinear models like ANN, attempt to fit the data with a single mathematical function, which limited the learning ability.

On the other hand, the method developed in [6], used a deep CNN model that achieved an ACC score of 97%. This result was superior when compared to the 95% obtained by the proposed method. However, they classified four types of seismic events instead of two, like in this work. Also, they made the training-test validation using an extensive dataset, which provided a decent number of samples during the model learning.

## 4 Conclusions and Future Work

We explored the use of eight different deep learning architectures based on deep CNN (DCNN1, DCNN2, and DCNN3), RNN (DCNN-RNN1, DCNN-RNN2, and DCNN-RNN3) and AE (AE1 and AE2) models to classify LP and VT seismic events on a dataset of seismic spectrogram images. The models based on the combination of deep CNN and RNN architectures reached the worst classification performances. The data augmentation operation helped to reinforce the learning of the DCNN1, DCNN2, DCNN3, AE1, and AE2 models. But it was not enough for the deep RNN based models, leading then to the overfitting anyway. The DCNN1 was the best model when compared with the other deep CNN based models, attaining AUC and ACC scores of 0.98 and 95%, respectively. Although these values were not the highest values per metric, they did not represent statistical differences against other results that were obtained by more algorithmically complex models. Furthermore, the proposed DCNN1 model showed similar or superior performance when compared to the majority of the state of the art methods in terms of the ACC metric. Therefore it can be considered as a successful scheme to classify LP and VT seismic events based on their spectrogram images.

As future work, we plan to increase the number of samples per class to experiment with more complex architectures like the deep CNN+RNN models and improve the hyperparameter configurations to explore the limits of the implemented models.

**Acknowledgment.** We thank the Applied Signal Processing and Machine Learning Research Group of UFSQ for providing the computing infrastructure (NVidia DGX workstation) to implement and execute the developed source code.

## References

1. Benitez, M.C., et al.: Continuous HMM-based seismic-event classification at deception island, Antarctica. *IEEE Trans. Geosci. Remote Sens.* **45**(1), 138–146 (2007). <https://doi.org/10.1109/TGRS.2006.882264>
2. Bueno, A., Benítez, C., De Angelis, S., Díaz Moreno, A., Ibáñez, J.M.: Volcano-seismic transfer learning and uncertainty quantification with bayesian neural networks. *IEEE Trans. Geosci. Remote Sens.* **58**(2), 892–902 (2020). <https://doi.org/10.1109/TGRS.2019.2941494>
3. Canziani, A., Paszke, A., Culurciello, E.: An analysis of deep neural network models for practical applications. arXiv e-prints [arXiv:1605.07678](https://arxiv.org/abs/1605.07678), May 2016
4. Chollet, F., et al.: Keras (2015). <https://keras.io>
5. Curilem, G., Vergara, J., Fuentealba, G., Acuña, G., Chacón, M.: Classification of seismic signals at villarrica volcano (chile) using neural networks and genetic algorithms. *J. Volcanol. Geoth. Res.* **180**(1), 1–8 (2009)
6. Curilem, M., Canário, J.P., Franco, L., Rios, R.A.: Using cnn to classify spectrograms of seismic events from llaima volcano (chile). In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8, July 2018. <https://doi.org/10.1109/IJCNN.2018.8489285>
7. Curilem, M., Vergara, J., San Martin, C., Fuentealba, G., Cardona, C., Huenupan, F., Chacón, M., Khan, M.S., Hussein, W., Yoma, N.B.: Pattern recognition applied to seismic signals of the llaima volcano (chile): an analysis of the events’ features. *J. Volcanol. Geoth. Res.* **282**, 134–147 (2014)
8. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **7**(Jan), 1–30 (2006)
9. Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. In: Honkela, T., Duch, W., Girolami, M., Kaski, S. (eds.) ICANN 2011. LNCS, vol. 6791, pp. 44–51. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-21735-7\\_6](https://doi.org/10.1007/978-3-642-21735-7_6)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
11. Hollander, M., Wolfe, D.A., Chicken, E.: *Nonparametric Statistical Methods*, vol. 751. John Wiley & Sons, Hoboken (2013)
12. Jain, Y.K., Bhandare, S.K.: Min max normalization based data perturbation method for privacy protection. *Int. J. Comput. Commun. Technol.* **2**(8), 45–50 (2011)
13. Jones, E., Oliphant, T., Peterson, P., et al.: *SciPy: open source scientific tools for Python* (2001). <http://www.scipy.org/>
14. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv e-prints [arXiv:1412.6980](https://arxiv.org/abs/1412.6980), December 2014
15. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014)
16. Lara-Cueva, R., Carrera, E.V., Morejon, J.F., Benítez, D.: Comparative analysis of automated classifiers applied to volcano event identification. In: 2016 IEEE Colombian Conference on Communications and Computing (COLCOM), pp. 1–6, April 2016. <https://doi.org/10.1109/ColComCon.2016.7516377>
17. Lara-Cueva, R.A., Benítez, D.S., Carrera, E.V., Ruiz, M., Rojo-Álvarez, J.L.: Automatic recognition of long period events from volcano tectonic earthquakes at cotopaxi volcano. *IEEE Trans. Geosci. Remote Sens.* **54**(9), 5247–5257 (2016). <https://doi.org/10.1109/TGRS.2016.2559440>

18. Lecun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**, 436–444 (2015). <https://doi.org/10.1038/nature14539>
19. Liou, C.Y., Cheng, W.C., Liou, J.W., Liou, D.R.: Autoencoder for words. *Neurocomputing* **139**, 84–96 (2014). <https://doi.org/10.1016/j.neucom.2013.09.055>
20. Liu, G., Yan, S.: Latent low-rank representation for subspace segmentation and feature extraction. In: 2011 International Conference on Computer Vision, pp. 1615–1622 (2011)
21. López, F.G., Torres, M.G., Batista, B.M., Pérez, J.A.M., Moreno-Vega, J.M.: Solving feature subset selection problem by a parallel scatter search. *Eur. J. Oper. Res.* **169**(2), 477–489 (2006)
22. Malfante, M., Dalla Mura, M., Métaixian, J.P., Mars, J.I., Macedo, O., Inza, A.: Machine learning for volcano-seismic signals: challenges and perspectives. *IEEE Signal Process. Mag.* **35**(2), 20–30 (2018)
23. Mikołajczyk, A., Grochowski, M.: Data augmentation for improving deep learning in image classification problem. In: 2018 International Interdisciplinary PhD Workshop (IIPhDW), pp. 117–122 (2018)
24. Mou, L., Ghamisi, P., Zhu, X.X.: Deep recurrent neural networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **55**(7), 3639–3655 (2017)
25. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
26. Pérez, N., Venegas, P., Benítez, D., Lara-Cueva, R., Ruiz, M.: A new volcanic seismic signal descriptor and its application to a data set from the cotopaxi volcano. *IEEE Trans. Geosci. Remote Sens.* **58**(9), 6493–6503 (2020). <https://doi.org/10.1109/TGRS.2020.2976896>
27. Pérez, N., Benítez, D., Grijalva, F., Lara-Cueva, R., Ruiz, M., Aguilar, J.: Eiseismic: towards an ecuadorian volcano seismic repository. *J. Volcanol. Geoth. Res.* **396**, 106855 (2020). <https://doi.org/10.1016/j.jvolgeores.2020.106855>
28. Rodgers, M., Smith, P., Pyle, D., Mather, T.: Waveform classification and statistical analysis of seismic precursors to the July 2008 vulcanian eruption of Soufrière Hills Volcano, Uontserratt. In: EGU General Assembly Conference Abstracts, vol. 18 (2016)
29. Schmincke, H.-U.: Volcanic hazards, volcanic catastrophes, and disaster mitigation. *Volcanism*, pp. 229–258. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-642-18952-4\\_13](https://doi.org/10.1007/978-3-642-18952-4_13)
30. Shin, H.C., et al.: Deep convolutional neural networks for computer-aided detection: cnn architectures, dataset characteristics and transfer learning. *IEEE Trans. Med. Imaging* **35**(5), 1285–1298 (2016)
31. Tilling, R.I.: Hazards and climatic impact of subduction-zone volcanism: a global and historical perspective. *Wash. DC Am. Geophys. Union Geophys. Monogr. Ser.* **96**, 331–335 (1996). <https://doi.org/10.1029/GM096p0331>
32. Titos, M., Bueno, A., García, L., Benítez, C.: A deep neural networks approach to automatic recognition systems for volcano-seismic events. *IEEE J. Sel. Topics Appl. Earth Observations Remote Sens.* **11**(5), 1533–1544 (2018). <https://doi.org/10.1109/JSTARS.2018.2803198>
33. Venegas, P., Pérez, N., Benítez, D., Lara-Cueva, R., Ruiz, M.: Combining filter-based feature selection methods and gaussian mixture model for the classification of seismic events from cotopaxi volcano. *IEEE J. Sel. Topics Appl. Earth Observations Remote Sens.* **12**(6), 1991–2003 (2019). <https://doi.org/10.1109/JSTARS.2019.2916045>

34. Venegas, P., Pèrez, N., Benítez, D.S., Lara-Cueva, R., Ruiz, M.: Building machine learning models for long-period and volcano-tectonic event classification. In: 2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), pp. 1–6. IEEE (2019)
35. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A., Bottou, L.: Stacked denoising autoencoders: learning useful representations in a deepnetwork with a local denoising criterion. *J. Mach. Learn. Res.* **11**(12) (2010)
36. Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., Xu, W.: CNN-RNN: a unified framework for multi-label image classification. arXiv e-prints [arXiv:1604.04573](https://arxiv.org/abs/1604.04573), April 2016
37. Yu, Q., Yang, Y., Song, Y.Z., Xiang, T., Hospedales, T.: Sketch-a-net that beats humans. arXiv e-prints [arXiv:1501.07873](https://arxiv.org/abs/1501.07873), January 2015