Deep Learning Architecture for Off-Line Recognition of Handwritten Math Symbols

Kawther Khazri Ayeb, Yosra Meguebli, and Afef Kacem Echi(✉)

ENSIT-LaTICE, University of Tunis, Tunis, Tunisia
afef.kacem@ensit.rnu.tn
http://www.latice.rnu.tn/francais/presentation.htm

Abstract. Real scientific challenge, handwritten math formula recognition is an attractive field of pattern recognition leading to practical applications. Hundreds of alphanumeric and math symbols need to be recognized, many are so similar in appearance that some use of context is necessary for disambiguation. Analysis of the spatial relationships between symbols is challenging. In this work, we focus on handwritten math symbols and propose to recognize them by a deep learning approach. The symbol images, used for train, validation, and test are generated from Competition on Recognition of Online Handwritten Mathematical Expressions dataset (CROHME) 2019's online patterns of mathematical symbols. As the large dataset is crucial for the performance of the deep learning model and it is labor-intensive to obtain a large amount of labeled data in real applications, we first augmented the database. Standing on the transfer learning technique, we then tested and compared several pre-trained Convolutional Neural networks (CNNs) like VGGNet, SqueezeNet, DenseNet, and Xception network and we tuned them to better fit our data. An accurate classification of 91.88% (train), 88.82% (validation), and 83.68% (test) for 101 classes is achieved, using only off-line features of the symbols.

Keywords: Handwriting math symbol recognition · Data augmentation · Transfer learning · Deep learning · CNN · Xception · VGGNet · SqueezeNet · DenseNet

1 Introduction

Handwritten math formula recognition is attracting interest due to its practical applications for consumers and academics in many areas such as education, office automation, etc. It offers an easy and direct way to input math formulas into computers, and therefore improves productivity for scientific writers. However, it is a challenging field due to the variety of writing styles and math formulas structures. Hundreds of alphanumeric and math symbols need to be recognized,

C. Djeddi et al. (Eds.): MedPRAI 2020, CCIS 1322, pp. 200–214, 2021.
https://doi.org/10.1007/978-3-030-71804-6_15

and also the two-dimensional structures, specifically the relationships between a pair of symbols, for example, superscript and subscript, both of them increase the difficulty of this recognition problem.

Many on-line techniques have been studied for handwritten math formula recognition. But when the recognition is carried out from a document image, therefore off-line techniques must be considered. In the last decade, most of the research has focused on typeset formulas but little research is published. The main difference between on-line and off-line recognition of math formulas is the temporal information that conveys the former problem that is lost in the latter problem. Math formula recognition can be divided into main steps: symbol recognition and structure analysis. Note that an accurate math formula recognition system greatly depends on an efficient symbol recognizer. Recently, deep learning marks the state of the art for math symbol classification problems, especially those including multiple layers of nonlinear information processing that automatically solve problems without using any prior knowledge. The results of recent researchers studies, summarized in Table 1 prove that Deep neural networks enhance mathematical recognition symbols [5] comparing to previous methods like Modified Quadratic Discriminant Functions (MQDFs) [4] with Bidirectional Long Short-term Memory (BLSTM) and Hidden Markov Models (HMM) [9].

The focus in this work is on handwritten math symbol recognition. It is one of the application in pattern classification: the task of labeling the symbol candidates to assign each of them a symbol class. It is as well a difficult task because the number of classes are quite important, more than one hundred different symbols including digits, alphabet, operators, Greek letters and some special math symbols (see Fig. 1).

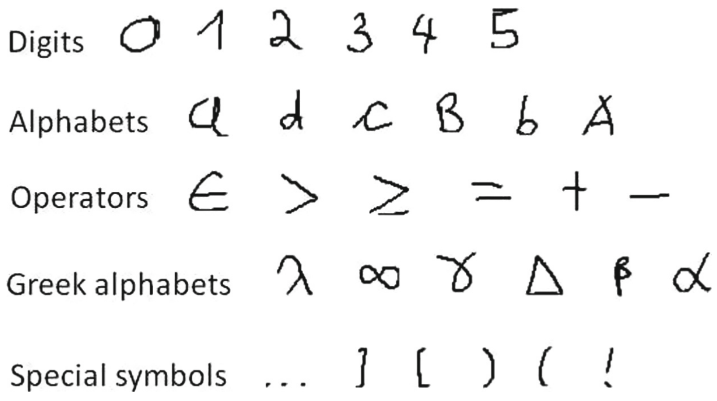

Fig. 1. Samples of math symbols.

It exists an overlapping between some symbol classes (inter-class variability): for instance, some symbols belonging to different classes might look about the same when considering different handwritten samples. There is also a high intra-class variability because each writer has his writing style (see Fig. 2). Besides, the challenges coming with the off-line. Many results showed that on-line recognition reached higher accuracy than off-line recognition [9], because of the absence of tracking coordinate of the symbol from start to stop, used to recognize it properly. So, it is important to design robust and efficient classifiers and to use a representative training data set. Nowadays, most of the proposed solutions use machine learning algorithms such as artificial neural networks or support vector machines.

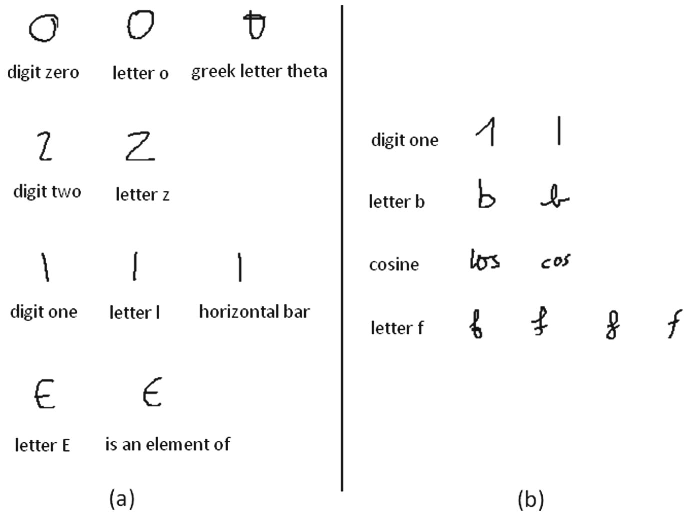

Fig. 2. Inter-class (a) and intra-class (b) variability.

This paper is organized as follows. In Sect. 2, we present a brief review in the field of math symbol recognition. In Sect. 3, we detail architectures of the proposed deep learning models and the used data augmentation and learning transfer mechanisms. In Sect. 4, we discuss the obtained results and compare the proposed models and some related works. In Sect. 5, we give some conclusions and prospects.

2 Related Works

In this section, we briefly discuss all advances in the area of math symbol recognition, as summarized in Table 1. The focus is on deep learning-based models.

In [1], authors define their proper CNN with a simple architecture composed of two convolutional layers, two pooling layers, a fully connected and a softmax layer to classify off-line handwritten math symbols. To improve their results, they tuned the performance of CNN by changing the number of feature maps

Table 1. Comparison between some related works.

Ref.	System	Model	Classes	Database	Accuracy (%)
[1]	Off-line Handwritten Math Symbols Classification	proper CNN	102	CROHME 2014 - train: 85781 - test: 10061	 - 93.27 - 87.72
[2]	Off-line Handwritten Math Symbols Classification	HMS-VGGNet	- 101: CROHME - 369: HASYv2	- train: 403729 CROHME 2016 + HASYv2 (part) - val: 6081 CROHME 2013 test - test: 10061 CROHME 2014 test - test: 10019 CROHME 2016 test - train: 366566 ± 1356 HASYv2 train - test: 16827 ± 166 HASYv2 test	 - 88.46 - 91.82 - 92.42 - 85.05
[3]	Off-line Handwritten Math Symbols Classification	- modified LeNet - pretrained SqueezeNet	- 87 - 101	- MNIST: 6000 - Set of Handwritten Digit Images: 2000	- 90 (modified LeNet) - 90 (SqueezeNet)
[4]	Online handwritten Math Expression Recognition	Online: Markov Random Field(MRF) Off-line: modified quadratic discriminant function (MQDF)	101	- Hands-Math dataset - Crohme 2016 [6]	- 88.24 - 86.05
[5]	Online Handwritten Math Symbols Detection	Inception v2, Resnet 50, Resnet 101, Inception Resnet v2	- 7 - 101	- flowchart dataset [7] - Crohme 2016	Inception Resnet v2: - 99 (val) 97 (test) - 89.7 (val) 86.8 (test)
[9]	Online Handwritten Math Expression Recognition	- structure recognition: 2D stochastic context-free grammars - symbol recognition: HMM (online and offline features)	- 100 - 37 - 57	- MathBrush [8] - CROHME 2011 Part1 - CROHME 2011 Part2	- 86.49 (W. dep.) 84.11 (W. Ind.) - 88.07 - 87.82

in convolutional layers, the number of nodes in a fully-connected layer, and the size of the input image. The authors used CROHME 2014 for training and evaluation. Authors declare obtaining 93.27% as train accuracy and 87.72% as a test, but they didn't show any accuracy or loss curve, which is necessary to add more credibility to their work. In [2], authors proposed a CNN, called HMS-VGGNet, for off-line recognition of handwritten math symbols. It is inspired by VGGNet, with smaller image sizes and additional batch normalization layers. The authors also used global average pooling layers to replace the fully connected layers. To prevent the lack of off-line data, the authors used elastic distortion to enrich the training set. Their proposed CNN uses only off-line features of the symbols and achieved an accuracy of 92.42% using CROHME 2016 test set. In [3], authors described an approach for off-line recognition of handwritten math symbols. They used Simple Linear Iterative Clustering for symbol segmentation and different methods: k-Nearest Neighbors (k-NN), LeNet, and SqueezeNet for symbol classification. The best-obtained accuracy using k-NN is 84% with 66 classes of symbols. Using modified LeNet, they achieved an accuracy of 90% with 87 classes. Finally, they reached 90% with a pre-trained SqueezeNet for 101 classes. The authors mentioned that they used the 6000 MNIST images from the CROHME dataset and 2000 images from the set of Handwritten Digit Images published by Computer Vision Group of the University of Sao Paulo, but they did not give details about the number of used instances for train, validation, and test and the cited accuracies. Recently, researchers used the off-line features extracted from the symbol images in combination with the online features to recognize the online math symbols and got great achievements. MyScript [6], the winner of CROHME 2016 extracted both online and off-line features and processed with a combination of Deep Multilayer Perceptron and Recurrent Neural Networks. MyScript achieved 92.81% in CROHME 2016 test set.

From our study of the state-of-the-art, we noted that the combination of online and off-line features betters the symbol recognition task performance and that the off-line recognition of math symbol should be more considered if we aim to reach the best performance. We also noted that many classification techniques are previously used, but there are very few works that compare different classification techniques on the same database and with the same experimental conditions.

3 Proposed System

We explored different architectures of CNNs, trained, and tested them using CROHME 2019 dataset. The objective is to find out the appropriate model for the off-line math symbol recognition. For that, we followed some steps, as described below.

3.1 Data Generation and Tuning

In CROHME 2019, there are 101 different classes of math symbols. The online data is given in Ink Markup Language (InkML) where each symbol is presented by an InkML file. This latter contains the set of symbol traces, knowing that a trace consists of a set of timing sampling points, and each point records its position. When generating symbol images from online data, we connected the points of the same trace with a single line. The generation of symbol images from InkML files is performed by a tool provided with the CROHME 2019 dataset. We then made some changes to ensure the automatic distribution of images on folders named with the class names. To train the proposed CNN models, we generated 30993 symbol images of size 38×38. To built the train and validation dataset, we automatically split the images dataset to have 24758 images for the train and 6235 for the validation. For the test, we generated and created ground truth of 15483 images from the 15483 CROHME test dataset. As it is known, improving the performance of a deep learning model depends either on tuning the applied model or tuning the used data. Since training deep learning models need several hours, we thought about normalizing the generated symbol dataset by binarizing and inverting them.

3.2 Model Tuning

To classify math symbols using a deep learning model, we have to choose one of these three alternatives: 1) to define a new model and train it on our data. 2) to use a predefined model, tune and train it on our data, or 3) to reuse a pre-trained model on other data and train it on our data. Based on the first tests and limited by the available data and computational resources, we have chosen the third alternative. One of the most common problems that we encountered while training these deep networks is overfitting. Recall that overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. To prevent such a problem, mainly due to overly complex models with too many parameters, we added a global average pooling layer where all the parameters in one feature map are averaged as a result. As deep networks need to be trained on large scale datasets and it is labor-intensive to obtain a large amount of labeled data in real applications, we first augmented the database. Standing on the transfer learning technique, we then tested and compared several pre-trained CNNs. This is will be dealt with in more detail in the next subsections.

Data Augmentation. Having a large dataset is crucial for the performance of the deep learning model. However, we can improve the performance of the model by augmenting the data we already have. Deep learning frameworks usually have built-in data augmentation utilities. Accordingly, to perform augmentation on a dataset of handwritten math symbols, it must be considered that it does not change the symbol meaning, for example, $<$when is vertically flipped, it is converted to$>$. Therefore, some augmentation techniques cannot be run on all symbols. In this work, we applied rotation with a random angle in the range of $[-15, 15]$ and horizontal and vertical shift augmentation techniques which are almost safe for handwritten math symbol recognition. Figure 3 shows several symbol image samples generated by the used augmentation techniques.

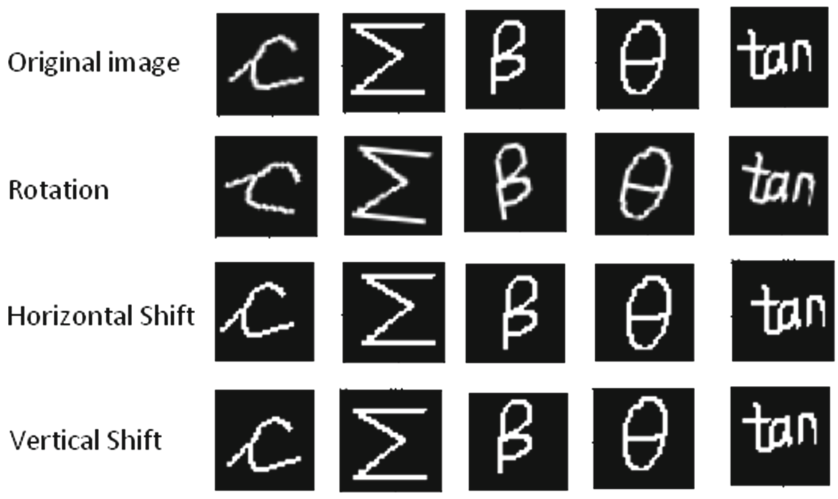

Fig. 3. Samples of symbol images result of data augmentation.

Used Models. Deep learning models need high computational resources with a huge dataset to obtain good results. One of the solutions to use deep learning-based models for symbol classification is to reuse pre-trained models and to test them with different parameters to improve accuracy. In our work, we have tried four pre-trained CNN models: VGGNet, SqueezeNet, Xception Network, and DenseNet. To these deep networks, we added two layers: 1) an average pooling layer to overcome the problem of over-fitting by averaging the parameters, and 2) a dense layer with regularization for math symbol class prediction. We started our tests from the smallest to the deeper network:

- Squeezenet: is a CNN with 18 layers deep, it is characterized by its compressed architecture design based on fire modules. A fire module is a combination of squeeze layers (1×1 convolution filters) and expand layers (a mix of 1×1 and 3×3 convolution filters)
- VGGNet19 [10]: is a CNN with 19 layers deep, it is composed of 16 convolutional layers and 3 fully connected layers
- Xception [11]: is a CNN with 71 layers deep it is based entirely on depthwise separable convolution layers.
- Densenet121 [12]: is a CNN with 121 layers deep. Recent work has shown that CNN can be deeper, more accurate, and efficient to train if they contain shorter connections between layers and this is what characterises in fact the Densenet. Each layer in Densenet is connected to every other layer in a feed-forward fashion. Whereas traditional CNN with L layers have L connections, one between each layer and its subsequent layer, Densenet has $L(L + 1)/2$ direct connections. DenseNets have several advantages: they strengthen feature propagation, encourage feature reuse, and reduce the number of

parameters. The efficiency of this model is proven by the tests that we did for the recognition of math symbols.

Table 2 and Fig. 4 show the architectures of the different models.

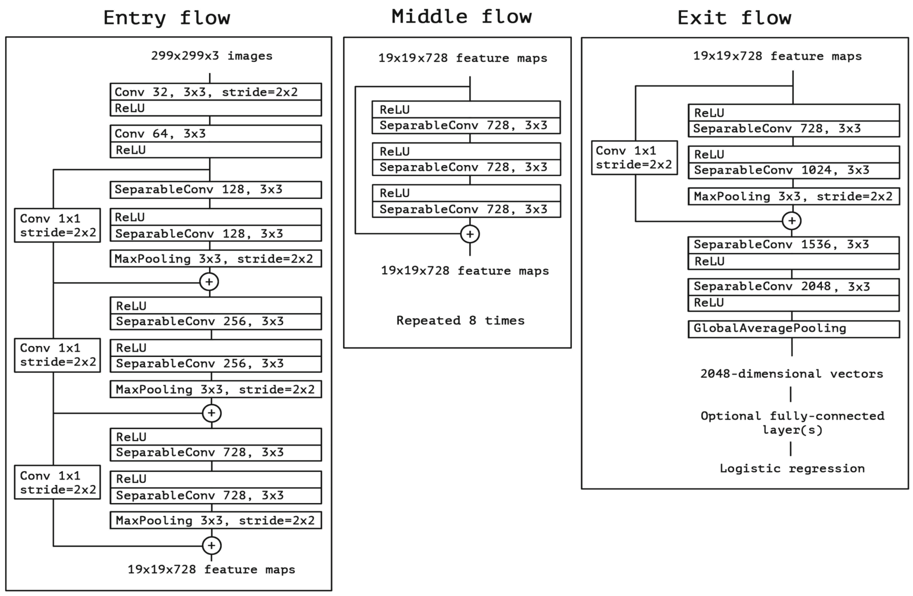

Fig. 4. Architecture of the Xception network.

Transfer Learning. In computer vision, transfer learning is expressed through the use of pre-trained models. A pre-trained model is a model that was trained on a large dataset to solve a problem similar to the one that we want to solve. It allows us to build accurate models in a timesaving way [13]. To well apply transfer learning and reuse some pre-trained model, we first have to correctly classify the treated problem, considering the size of the dataset and its similarity to the used dataset to train the pre-trained model. Figure 5 shows the size-similarity matrix that controls the choice of the model and guides us to fine-tune it to get successful results.

Table 2. Architecture of SqueezeNet, VGGNet and DenseNet

SqueezeNet	VGGNet19	DenseNet121
18 weight layers	19 weight layers	121 weight layers
Input		
conv1-64 Relu	conv3-64 conv3-64	con1-3
maxpool		
fire2-128 fire3-128 fire4-256	conv3-128 conv3-128	DenseBlock-64 (6×ConvBlock)
maxpool		TransitionLayer (conv1+Average pool)
fire5-256 fire6-384 fire7-384 fire8-512	conv3-256 conv3-256 conv3-256 conv3-256	DenseBlock-128 (12×Convblock)
maxpool		TransitionLayer
fire9-512	conv3-512 conv3-512 conv3-512 conv3-512	DenseBlock-256 (24×ConvBlock)
maxpool		TransitionLayer
con1-1000	conv3-512 conv3-512 conv3-512 conv3-512	DenseBlock-512 (16×ConvBlock)
average pool	maxpool	average pool
	FC-4096	
	FC-4096	
	FC-1000	
Softmax		

Fine-Tuning. Having situated our problem according to the size-similarity matrix, we can choose the adequate fine-tuning alternatives. Figure 6 represents a CNN model as a succession of two blocks: a convolutional base for feature extraction in the top and a classifier in the bottom. Following the size-similarity matrix, four fine-tuning decisions can be taken.

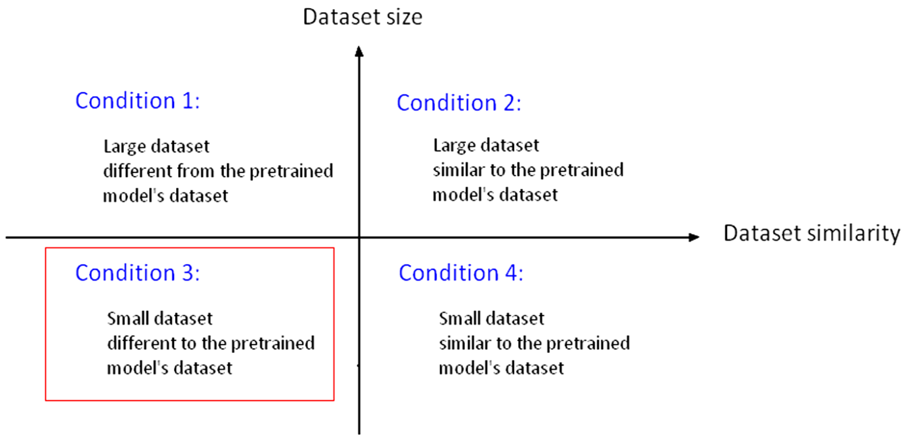
Dataset size
Dataset similarity
Condition 1:
Large dataset different from the pretrained model's dataset
Condition 2:
Large dataset similar to the pretrained model's dataset
Condition 3:
Small dataset different to the pretrained model's dataset
Condition 4:
Small dataset similar to the pretrained model's dataset

Fig. 5. Size similarity matrix.

4 Experimental Results

4.1 CROHME Dataset

Since the datasets of off-line handwritten mathematical symbols are rare, we used the online data of CROHME 2019 to generate symbol images for off-line symbol recognition. The number of symbol classes in the CROHME dataset is 102, including a junk class for erroneous symbols. To evaluate the proposed CNN models, we generated 30993 symbol images. To built the train and validation datasets, we automatically split the images dataset to have 24758 images for the train and 6235 for the validation. For the tests, we generated and created ground truth of 6820 images from CROHME 2019 test dataset.

4.2 Experimental Setup

Our experiments were performed on an Intel(R)Core (TM) with a CPU of 2.5 GHz and a memory of 8 GB. We trained our system using pre-trained deep learning models from the Tensorflow library, trained over the ImageNet dataset. Although our generated images are different from the natural images of the ImageNet dataset, we found that training using the pre-trained models allows for much faster convergence than training from scratch, especially with the presence of a small dataset. Regarding the size-similarity matrix presented in Fig. 5, we found that our classification problem satisfied the third condition (small dataset and different from the pre-trained model's dataset, that is why we fine-tuned

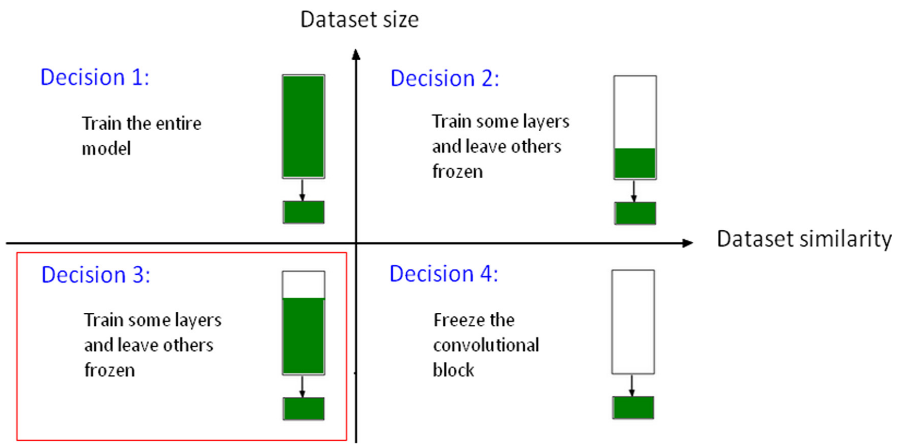
Dataset size
Dataset similarity
Decision 1:
Train the entire model
Decision 2:
Train some layers and leave others frozen
Decision 3:
Train some layers and leave others frozen
Decision 4:
Freeze the convolutional block

Fig. 6. Decision map for fine-tuning pre-trained models.

our model by freezing the ten first layers of the convolutional block responsible of the extraction of generic features and train the rest on our data. The initial learning rate was 0.001, the Batch Size was set to 32. We initialized the number of the epoch at 200 and we implemented early stopping callbacks.

4.3 Results and Discussion

We trained different CNN models with various parameters on our dataset. Figure 7 shows the accuracy and loss curves of the different CNN models. Our best obtained experimental results are shown in Table 3.

We can see that DenseNet121 achieves the state-of-the-art and outperforms the other models. This can be explained as follows: 1) This network is remarkably deeper than the others (121 layers), and 2) Adding more instances to the dataset enhances the capabilities of the model (using a dataset of 24758 train images and 6235 validation images improves the accuracy from 91.73% to 91.88% for the training and from 84.07% to 88.82% for the validation). Evaluating our model on the test dataset, we obtained an accuracy of 83.68%. Table 4 shows the performance measures: Precision, Recall and F1-score of the different classes and the overall system.

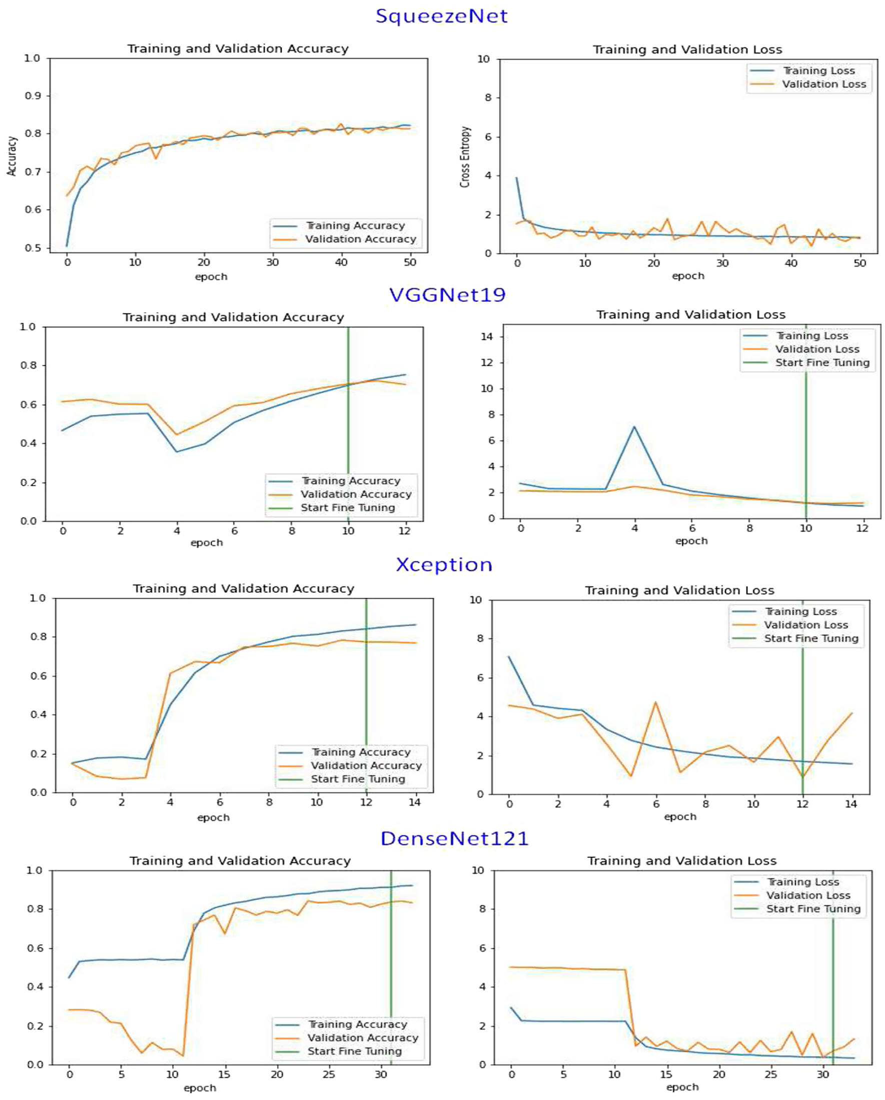

Fig. 7. Accuracy and loss curves.

Table 3. Accuracy and loss results.

Models	Dataset	Accuracy (%)	Loss
SqueezeNet	- train: 21610 - val: 5462	- 82.25 - 82.6	- 0.80 - 0.35
VGGNet19	- train: 21610 - val: 5462	- 72 - 78	- 1.01 - 1.02
Xception	- train: 15698 - val: 7012	- 86.02 - 78.15	- 1.55 - 0.85
DenseNet121	- train: 22794 - val: 5745	- 91.73 - 84.07	- 0.33 - 0.36
DenseNet121	**- train: 24758** **- val: 6235** **- test: 6820**	**- 91.88** **- 88.82** **- 83.68**	**- 0.28** **- 0.20** **- 0.06**

Comparing our work to others, we noted that obtained results are promising but still less than some systems of online symbol recognition or those utilizing online and offline features to classify symbols, and this because online data has the tracing information while off-line data does not. Online data has advantages when classifying symbols having similar shapes and different writing styles, such as 5 and s. Our networks only use offline features so it is hard for it to correctly classify those symbols.

Although the symbol recognition achieved good accuracy, that does not prevent it from making mistakes to predict some symbol classes. Analyzing the confusion matrix, we found that miss recognitions are mainly due to that certain distinct symbols are in close resemblance, such as the capital letter X and math symbol $\times$, the symbol division / and the comma sign, the letter O and the Greek letter Θ, the capital letter S and the digit 5, the digit 9 and letter g, etc.

Observing the event of confusion, we noted that confused symbols have roughly similar morphologies that make them difficult to be distinguished even for a human. We considered some of the misrecognition cases to be too difficult for any classifier to resolve without considering symbol context. That is why we keep resolving some of these confusion cases for future works dealing with the entire math formula recognition.

Table 4. Model performance evaluation.

Class	Prec.	Recall	F1-score	Nb.	Class	Prec.	Recall	F1-score	Nb.
!	0.67	1.00	0.80	2	forall	1.00	1.00	1.00	1
(	0.95	0.88	0.92	190	g	0.22	1.00	0.36	6
)	0.96	0.83	0.89	236	gamma	0.86	0.50	0.63	12
+	0.98	0.95	0.97	601	geq	0.88	1.00	0.93	7
−	1.00	0.92	0.96	723	gt	1.00	1.00	11.00	5
0	0.97	0.96	0.96	118	h	0.60	0.60	0.60	5
1	0.53	0.96	0.68	215	i	0.72	0.96	0.82	24
2	0.81	0.98	0.89	326	in	0.75	1.00	0.86	3
3	0.94	0.99	0.96	137	infty	0.92	0.92	0.92	12
4	0.90	0.86	0.88	81	int	0.89	0.64	0.74	25
5	0.92	0.80	0.85	95	j	0.84	0.67	0.74	24
6	0.94	0.83	0.88	41	junk	0.86	0.90	0.88	1350
7	0.90	0.90	0.90	31	k	0.79	0.96	0.87	24
8	0.90	0.90	0.90	31	l	0.33	0.17	0.22	6
9	0.79	0.39	0.53	79	lambda	0.50	1.00	0.67	1
=	0.98	0.95	0.96	273	ldots	0.77	1.00	0.87	10
A	1.00	0.75	0.86	16	leq	1.00	1.00	1.00	16
B	1.00	1.00	1.00	9	lim	0.41	0.88	0.56	8
comma	0.28	0.38	0.32	53	log	0.67	0.92	0.77	13
C	1.00	0.60	0.75	62	lt	0.67	1.00	0.80	2
Delta	1.00	1.00	1.00	1	m	0.77	0.71	0.74	28
E	1.00	1.00	1.00	11	mu	0.00	0.00	0.00	4
F	0.71	0.83	0.77	6	n	0.79	0.94	0.86	81
G	0.00	0.00	0.00	2	neq	0.67	0.33	0.44	6
H	0.57	1.00	0.73	4	o	0.00	0.00	0.00	8
I	1.00	1.00	1.00	2	p	0.11	0.33	0.17	3
L	1.00	0.60	0.75	20	phi	0.00	0.00	0.00	1
M	0.75	0.75	0.75	4	pi	0.79	0.96	0.86	23
N	0.43	0.50	0.46	6	pm	0.00	0.00	0.00	4
P	0.78	0.52	0.62	27	point	0.87	1.00	0.93	13
R	1.00	1.00	1.00	15	prime	0.00	0.00	0.00	4
S	0.53	0.82	0.65	28	q	0.20	0.29	0.24	17
T	0.67	1.00	0.80	4	r	0.30	0.58	0.40	12
V	1.00	0.70	0.82	10	rightarrow	0.55	0.86	0.67	7
X	0.06	0.30	0.10	27	s	1.00	1.00	1.00	1
Y	0.62	0.93	0.74	14	sigma	0.12	1.00	0.22	1
[	0.92	0.82	0.87	28	sin	0.86	0.86	0.86	22
]	0.94	0.60	0.73	25	sqrt	0.95	0.78	0.86	96
a	0.87	0.97	0.92	100	sum	1.00	0.88	0.94	25
alpha	0.91	0.86	0.89	36	t	0.80	0.77	0.79	31
b	0.89	1.00	0.94	59	tan	0.88	0.54	0.67	13
bar	0.00	0.00	0.00	131	theta	0.62	0.91	0.74	11
beta	1.00	0.93	0.97	15	times	1.00	0.08	0.15	173
c	0.05	1.00	0.10	1	u	0.48	1.00	0.65	12
cos	0.92	0.83	0.87	29	v	1.00	1.00	1.00	3
d	0.96	0.96	0.96	94	w	0.83	1.00	0.91	5
div	0.89	0.89	0.89	9	x	0.83	0.98	0.90	339
division	0.15	0.06	0.09	31	y	0.81	0.92	0.86	62
e	0.96	1.00	0.98	25	z	0.96	0.56	0.70	144
exists	1.00	1.00	1.00	1	{	1.00	0.50	0.67	4
f	0.91	0.40	0.56	25	}	1.00	1.00	1.00	4

5 Conclusion and Future Work

In this paper, we addressed the problem of offline recognition of handwritten mathematical symbols. We used a deep learning recognition method based on the Densenet model to which we did some modification. Our symbol recognition system has shown its efficiency on a reasonable number of handwritten symbols from Crohme 2019 dataset with an accuracy rate of 83.71%. In further works, we plan to improve the performance of the model by augmenting the data we already have. We will also work out on treating the case of junk symbol by making the focus on finding why they are considered junk, and how to treat them based on cause analysis.

References

1. Ramadhan, I., Purnama, B., Al Faraby, S.: Convolutional neural networks applied to handwritten mathematical symbols classification. In: Fourth International Conference on Information and Communication Technologies (ICoICT) (2016)
2. Dong, L., Liu, H.: Recognition of offline handwritten mathematical symbols using convolutional neural networks. In: Zhao, Y., Kong, X., Taubman, D. (eds.) ICIG 2017. LNCS, vol. 10666, pp. 149–161. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71607-7_14
3. Nazemi, A., Tavakolian, N., Fitzpatrick, D., Fernando, C., Suen, C.Y.: Offline handwritten mathematical symbol recognition utilising deep learning, cs.CV (2019)
4. Phan, K.M., Le, A.D., Indurkhya, B., Nakagawa, M.: Augmented incremental recognition of online handwritten mathematical expressions. Int. J. Doc. Anal. Recogn. (IJDAR) **21**(4), 253–268 (2018). https://doi.org/10.1007/s10032-018-0306-1
5. Julca-Aguilar, F.D., Hirata, N.S.T.: Symbol detection in online handwritten graphics using Faster R-CNN. In: International Workshop on Document Analysis Systems (DAS) (2018)
6. Mouchere, H., Viard-Gaudin, C., Zanibbi, R., Garain, U.: ICFHR2016 CROHME: competition on recognition of online handwritten mathematical expressions (2016)
7. Awal, A.M., Feng, G., Mouchere, H., Viard-Gaudin, C.: First experiments on a new online handwritten flowchart database. In: Document Recognition and Retrieval XVIII (2011)
8. MacLean, S., Labahn, G., Lank, E., Marzouk, M., Tausky, D.: Grammar-based techniques for creating ground-truthed sketch corpora. Int. J. Doc. Anal. Recogn. **14**, 65–74 (2011). https://doi.org/10.1007/s10032-010-0118-4
9. Alvaro, F., Sanchez, J.A., Benedi, J.M.: Recognition of on-line handwritten mathematical expressions using 2D stochastic context-free grammars and hidden Markov models. Pattern Recogn. Lett. **35**, 58–67 (2014)
10. Simonyan, K., Zisserman, A.: Very deep convolutional neural networks for large-scale image recognition. In: ICLR (2015)
11. Chollet, F.: Xception: deep learning with depthwise separable convolutions, cs.CV (2017)
12. Huang, G., Liu, Z., Maaten, L., Weinberger, K.: Densely connected convolutional networks, cs.CV (2018)
13. Rawat, W., Wang, Z.: Deep convolutional neural networks for image classification: a comprehensive review. Neural Comput. **29**(9), 2352–2449 (2017)