Faculty of Computers and Artificial Intelligence

CS396: Selected Topics CS-2

Helwan University

Team ID: 33

	Student Name	Student ID	Level	Diss. Grade
1	عبدالرحمن خالد طاهر السيد	201900416	3	
2	اسلام رضا محمد عبد القادر	201900140	3	
3	عبدالرحمن نبيل محمد محمود	201900442	3	
4	عبدالكريم أنور أحمد محمد	201900451	3	
5	حازم محمد عبدالتواب محمد	201900254	3	
6	محمود عماد عبدالموجود إسماعيل	201900778	3	
7	عبدالرحمن عماد عبدالله السيد	201900425	3	

Paper details:

Name: Arabic Handwritten Characters Recognition Using Convolutional Neural Network

AUTHORS: Ahmed El-Sawy, Mohamed Loey, Hazem EL-Bakry

Publisher: WSEAS Transactions on Computer Research

Link:

https://www.wseas.org/multimedia/journals/computerresearch/2017/a045818-075.pdf

ABSTRACT: Handwritten Arabic character recognition systems face several challenges, including the unlimited variation in human handwriting and large public databases. In this work, we model a deep learning architecture that can be effectively apply to recognizing Arabic handwritten characters. A Convolutional Neural Network (CNN) is a special type of feed-forward multilayer trained in supervised mode. The CNN trained and tested our database that contain 16800 of handwritten Arabic characters. In this paper, the optimization methods implemented to increase the performance of CNN. Common machine learning methods usually apply a combination of feature extractor and trainable classifier. The use of CNN leads to significant improvements across different machine-learning classification algorithms. Our proposed CNN is giving an average 5.1% misclassification error on testing data.

Year of publication: 2017

Dataset used: Arabic Handwritten Characters Dataset

Link of used dataset:

https://www.kaggle.com/datasets/mloey1/ahcd1?datasetId=1453&sortBy=voteCount

Contents of the dataset:

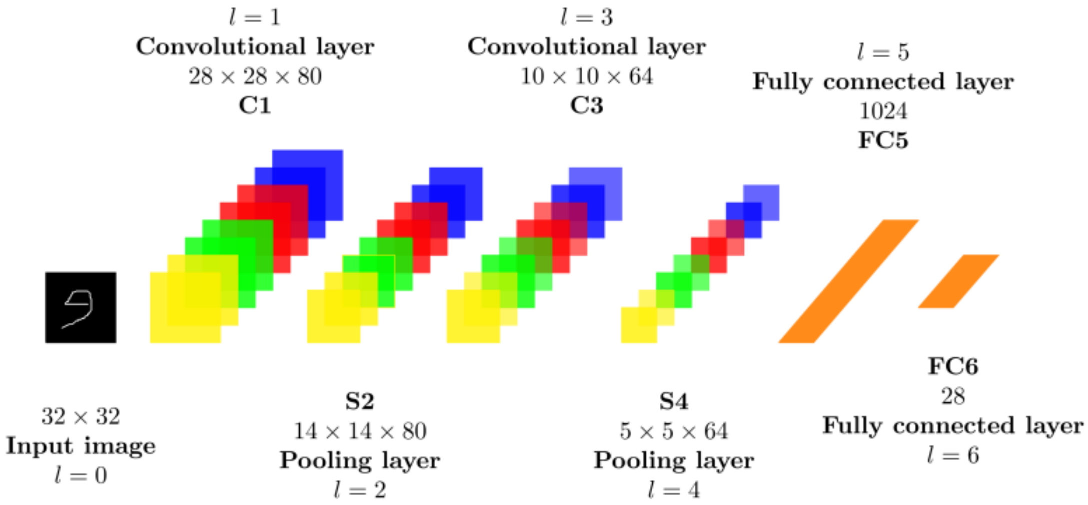
16800 images

13440 Training images

3360 Testing images

Implementation algorithms:

- This is a sequential Convolutional Neural Network for Arabic Handwritten Characters.



- The network consists of

  2 convolutional layers

  2 pooling layers

  2 fully connected layers

- The activation function used is: rectified linear units (RELU)

((increase the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer))

- The pooling layer used is: 2*2 max pooling

((reduces the resolution of the image that reduce the precision of the translation effect by taking the maximum of the block that they are pooling))

- Use stochastic gradient descent optimizer (SGD)
- Use 30 epochs
- Mini-batch size 256
- Use SoftMax classifier

Result: evaluating the performance of CNN on Arabic charters, incremental training approach was used on the proposed approach with the mini-batch mode. As shown the miss-classification rate for training data has reach 0% on epochs 25 to 30. Here our approach is trained for 30 epochs, but from epoch 25, the CNN shows a low miss-classification error.

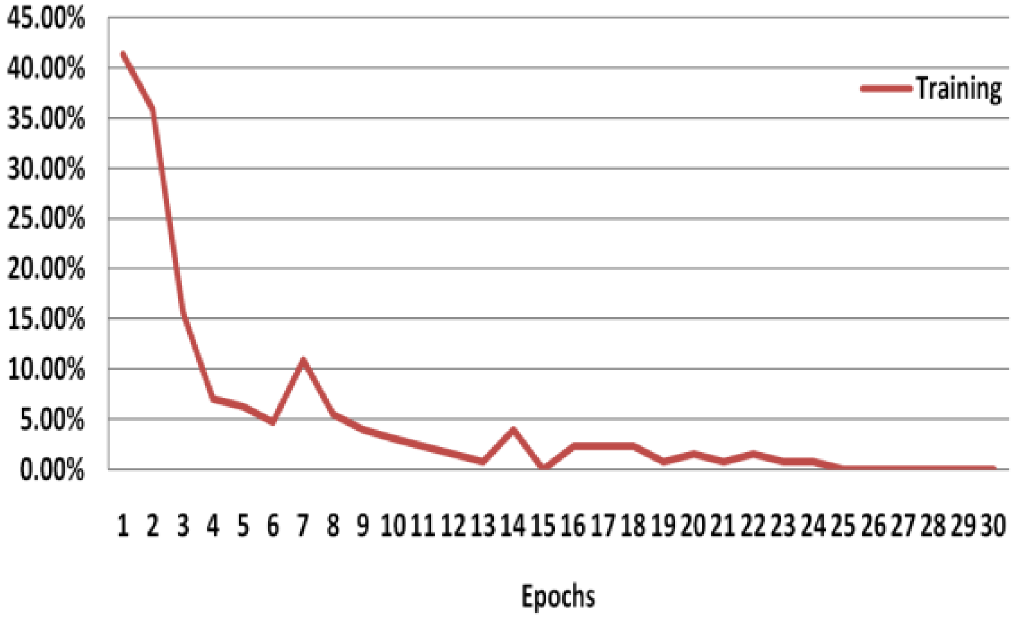
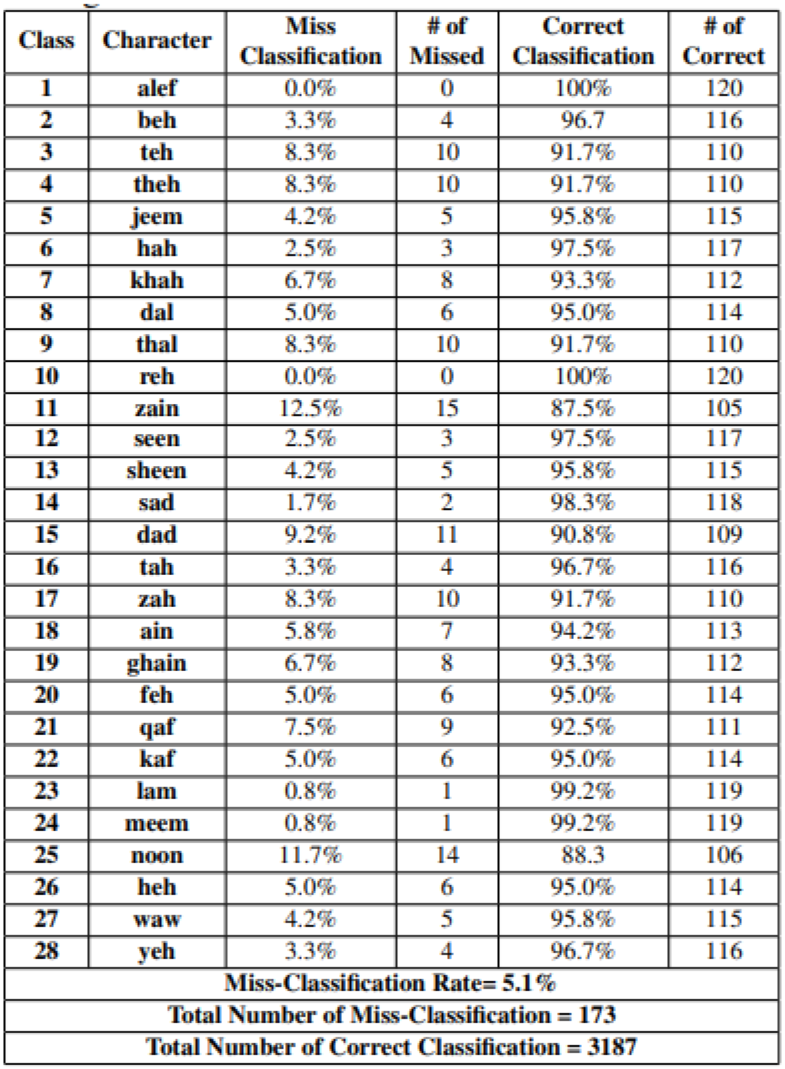


Table: Miss-Classification & correct-Classification rate and number of wrong and correct recognition on testing data



Class	Character	Miss Classification	# of Missed	Correct Classification	# of Correct
1	alef	0.0%	0	100%	120
2	beh	3.3%	4	96.7	116
3	teh	8.3%	10	91.7%	110
4	theh	8.3%	10	91.7%	110
5	jeem	4.2%	5	95.8%	115
6	hah	2.5%	3	97.5%	117
7	khah	6.7%	8	93.3%	112
8	dal	5.0%	6	95.0%	114
9	thal	8.3%	10	91.7%	110
10	reh	0.0%	0	100%	120
11	zain	12.5%	15	87.5%	105
12	seen	2.5%	3	97.5%	117
13	sheen	4.2%	5	95.8%	115
14	sad	1.7%	2	98.3%	118
15	dad	9.2%	11	90.8%	109
16	tah	3.3%	4	96.7%	116
17	zah	8.3%	10	91.7%	110
18	ain	5.8%	7	94.2%	113
19	ghain	6.7%	8	93.3%	112
20	feh	5.0%	6	95.0%	114
21	qaf	7.5%	9	92.5%	111
22	kaf	5.0%	6	95.0%	114
23	lam	0.8%	1	99.2%	119
24	meem	0.8%	1	99.2%	119
25	noon	11.7%	14	88.3	106
26	heh	5.0%	6	95.0%	114
27	waw	4.2%	5	95.8%	115
28	yeh	3.3%	4	96.7%	116
Miss-Classification Rate= 5.1%					
Total Number of Miss-Classification = 173					
Total Number of Correct Classification = 3187					

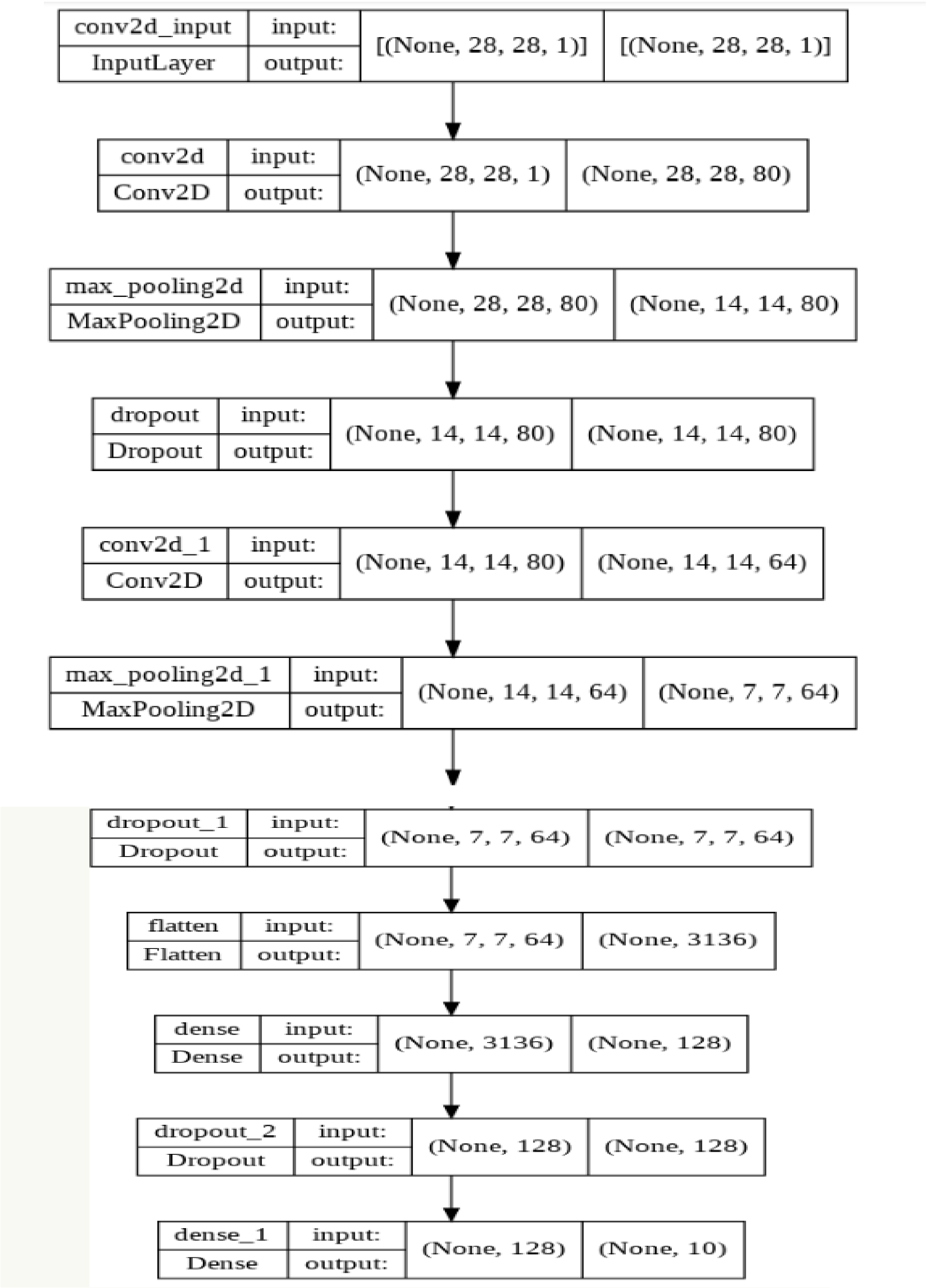
Misclassification Error: 94.9%

Project description Document

1. General information on the selected dataset
    - Name of the dataset: Arabic Handwritten Digits Recognizer
    - Link of the dataset: https://www.kaggle.com/code/abdelwahed43/arabic-handwritten-digits-recognizer
    - Total number of the samples in the dataset: 70000
    - Dimension of images: 28 * 28
    - Number of classes: 10
    - Number of labels: 10
    - The labels: (0,1,2,3,4,5,6,7,8,9)

2. Implementation details
    - Ratio of training: 85.7%
    - Ratio of validation: 1.42%
    - Ration of testing: 14.3%
    - Number of training samples: 60000
    - Number of validation samples: 1000
    - Number of testing samples: 10000

- Block diagram of implemented model



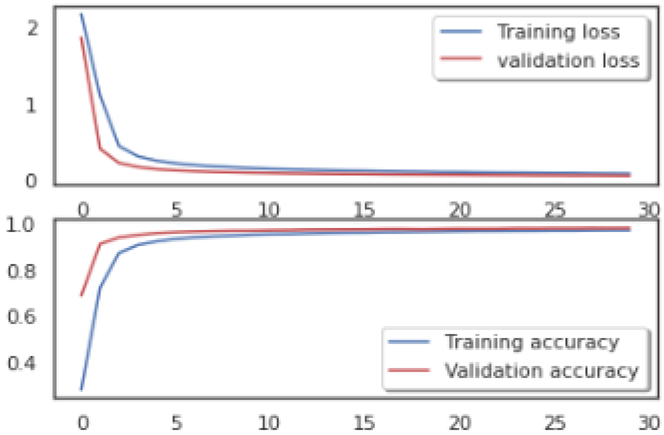
conv2d_input InputLayer	input: output:	[(None, 28, 28, 1)]	[(None, 28, 28, 1)]
conv2d Conv2D	input: output:	(None, 28, 28, 1)	(None, 28, 28, 80)
max_pooling2d MaxPooling2D	input: output:	(None, 28, 28, 80)	(None, 14, 14, 80)
dropout Dropout	input: output:	(None, 14, 14, 80)	(None, 14, 14, 80)
conv2d_1 Conv2D	input: output:	(None, 14, 14, 80)	(None, 14, 14, 64)
max_pooling2d_1 MaxPooling2D	input: output:	(None, 14, 14, 64)	(None, 7, 7, 64)
dropout_1 Dropout	input: output:	(None, 7, 7, 64)	(None, 7, 7, 64)
flatten Flatten	input: output:	(None, 7, 7, 64)	(None, 3136)
dense Dense	input: output:	(None, 3136)	(None, 128)
dropout_2 Dropout	input: output:	(None, 128)	(None, 128)
dense_1 Dense	input: output:	(None, 128)	(None, 10)

3. Hyperparameters

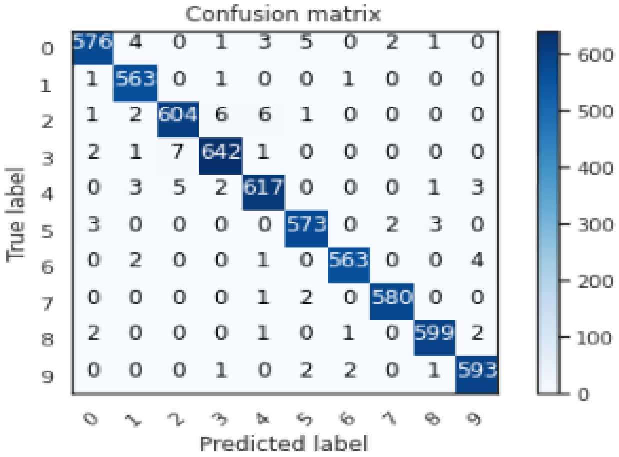
1 test_size = 0.1
2 random state = 4
3 filters in first convolutional layer = 80
4 filters in second convolutional layer = 64
5 kernel_size = 5 * 5
6 pool_size = 2 * 2
7 Dropout = 0.25
8 nodes in dense = 128
9 learning rate (Lr) = 0.001
10 momentum = 0.3

Result details

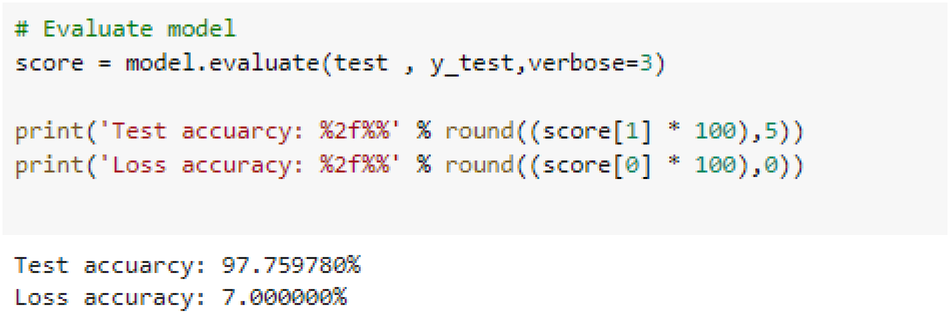
- Training validation curve (graph that describe the loss and accuracy of the training set and the validation set)



- Confusion matrix (graph that describe the predicted values of each class in the validation data)



- Evaluation of the model

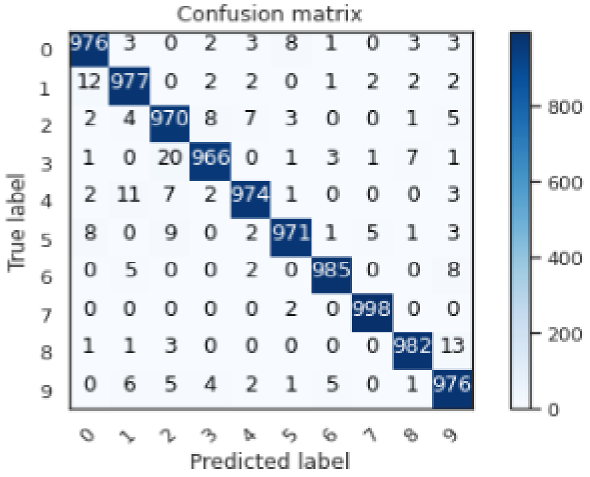


```
# Evaluate model
score = model.evaluate(test , y_test,verbose=3)

print('Test accuarcy: %2f%%' % round((score[1] * 100),5))
print('Loss accuracy: %2f%%' % round((score[0] * 100),0))
```

```
Test accuarcy: 97.759780%
Loss accuracy: 7.000000%
```

- Confusion matrix (graph that describe the predicted values of each class in the testing data)


Confusion matrix
True label
Predicted label