Faculty of Computers and Artificial Intelligence

Computer Science Department

CS 396 Selected Topics in CS-2

Research Project

Team No. 41

	ID	Name	Grade
•	201900457	عبدالله جمال ابراهيم بسطاوي	
•	201900027	احمد خالد جابر احمد	
•	201900006	ابراهيم خالد ابراهيم عطية	
•	201900121	اسامه عبدالرسول ابراهيم صابر	
•	201900081	احمد مجدي عبدلله السيد	
•	201900402	عبدالرحمن احمد علي حسن غزالي	
•	201900009	ابراهيم عبدالعزيز ابراهيم	
•	201900123	اسامة علي حمزة	

1-Paper details:

Authors name: Lakshmi Prasanna

Paper name: MULTI LABEL CLASSIFICATION FOR AN IMAGE USING CONVOLUTIONAL NEURAL NETWORKS

Publisher name: International Journal of Computer Science and Mobile Computing

Year of publication: 7, July- 2021

Dataset name: Image dataset

Algorithm: CNN

Accuracy: 55%

2-General Information on the selected dataset:

Dataset name: MNIST digits classification dataset

Dataset source: https://keras.io/api/datasets/mnist/

Number of Classes: 10

Dataset labels: (0-1-2-3-4-5-6-7-8-9)

Dataset description: The MNIST dataset has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

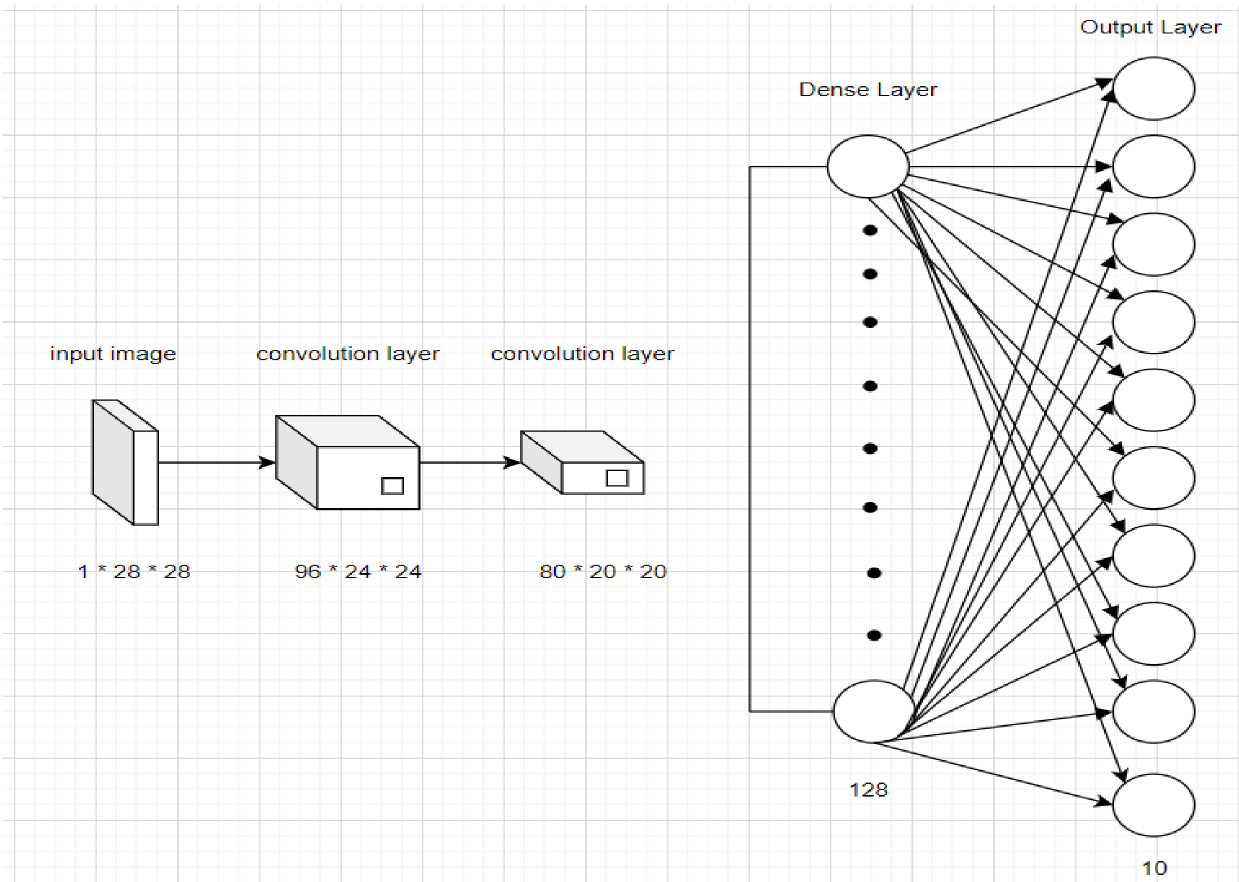
3-Implementation details:

Training set ratio: 68% (48000)

Validation set ratio: 18% (12000)

Testing set ratio: 14% (10000)

Block diagram:



Hyperparameters: we use 2 convolution layers and 2 dense layers

Conv1: filters=96, kernel_size=(5,5), strides=(1, 1), padding="valid", data_format=None, dilation_rate=(1, 1), groups=1, activation='rule', use_bias=True, kernel_initializer="glorot_uniform", bias_initializer="zeros", kernel_regularizer=None, bias_regularizer=None, activity_regularizer=None, kernel_constraint=None, bias_constraint=None, **kwargs

Conv2: filters=80, kernel_size=(5,5), strides=(1, 1), padding="valid", data_format=None, dilation_rate=(1, 1), groups=1, activation='rule', use_bias=True, kernel_initializer="glorot_uniform", bias_initializer="zeros", kernel_regularizer=None, bias_regularizer=None, activity_regularizer=None, kernel_constraint=None, bias_constraint=None, **kwargs

Dense1: units=128, activation='relu', use_bias=True, kernel_initializer="glorot_uniform", bias_initializer="zeros", kernel_regularizer=None ,bias_regularizer=None , activity_regularizer=None , kernel_constraint=None, bias_constraint=None, **kwargs

Dense2: units=10, activation='softmax', use_bias=True, kernel_initializer="glorot_uniform", bias_initializer="zeros", kernel_regularizer=None ,bias_regularizer=None , activity_regularizer=None , kernel_constraint=None, bias_constraint=None, **kwargs

Model Complie: optimizer='adam', loss='categorical_crossentropy', metrics= ['accuracy'], loss_weights=None, weighted_metrics=None, run_eagerly=None, steps_per_execution=None, jit_compile=None, **kwargs

Model fit: x=x_train, y=y_train, batch_size=64, epochs=10, verbose="auto", callbacks=None, validation_split=0.0, validation_data= (x_val, y_val), shuffle=True, class_weight=None, sample_weight=None, initial_epoch=0, steps_per_epoch=None, validation_steps=None, validation_batch_size=64, validation_freq=1, max_queue_size=10, workers=1, use_multiprocessing=False

4-Results details before optimization

```
model = Sequential()
model.add(Conv2D(32,(3,3),activation='relu', input_shape=(28,28,1)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64,(3,3),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(128,(3,3),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Dropout(0.25))
model.add(Flatten())

model.add(Dense(256,activation='relu'))
model.add(Dense(512,activation='relu'))
model.add(Dense(128,activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(10,activation='softmax'))
```

```
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
history = model.fit(x_train, y_train, epochs=10, batch_size= 128, validation_data = (x_val, y_val) ,validation_batch_size= 128)
```

```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print('\nTest accuracy:', test_acc)
```

```
313/313 - 1s - loss: 0.0515 - accuracy: 0.9879 - 1s/epoch - 4ms/step

Test accuracy: 0.9879000186920166
```

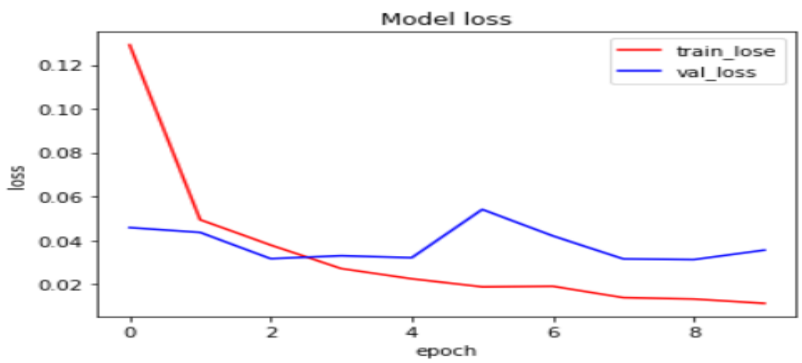
5-Results details after optimization

```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print('\nTest accuracy:', test_acc)
```

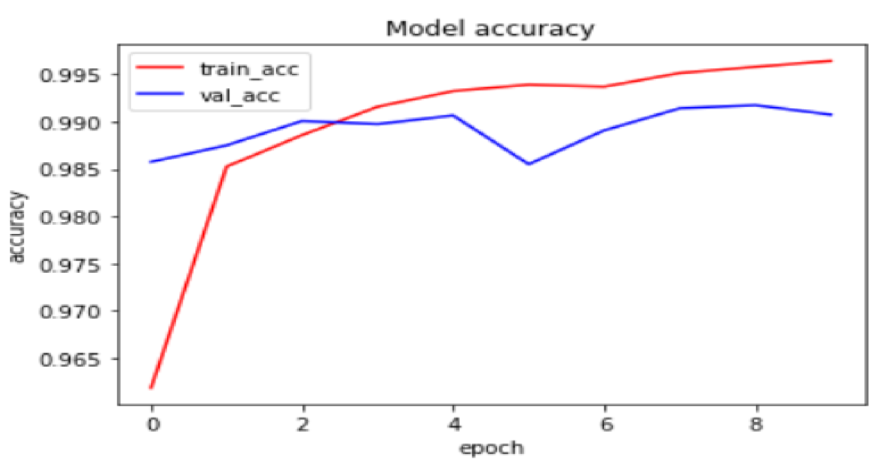
```
313/313 - 2s - loss: 0.0305 - accuracy: 0.9913 - 2s/epoch - 7ms/step

Test accuracy: 0.9912999868392944
```

```
plt.plot(history.history['loss'], color='r')
plt.plot(history.history['val_loss'], color ='b')
plt.title('Model loss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['train_lose', 'val_loss'], loc='upper right')
plt.show()
```




```
plt.plot(history.history['accuracy'], color='r')
plt.plot(history.history['val_accuracy'], color='b')
plt.title('Model accuracy')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.legend(['train_acc','val_acc'], loc='upper left')
plt.show()
```



```python
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[ 975    1    0    0    0    1    0    0    3    0]
 [   2 1122    2    2    1    2    2    0    2    0]
 [   0    0 1030    0    0    0    0    1    1    0]
 [   0    0    3 1001    0    2    0    1    3    0]
 [   0    0    0    0  979    0    2    0    0    1]
 [   1    0    0    4    0  884    3    0    0    0]
 [   1    1    0    0    2    1  951    0    2    0]
 [   0    2    9    0    1    0    0 1013    1    2]
 [   0    0    2    0    0    0    0    0  972    0]
 [   1    0    0    1   12    3    0    3    3  986]]
0.9913
```