A3-1102

Page No.
Date

Assignment

Title: Convolutional Neural Network

Problem Statement: Use MNIST fashion dataset and create a classifier to classify fashion clothing into categories.

Objectives:
i) To implement convolutional neural network.
ii) Build classifier to classify MNIST fashion data.

Outcomes:
Implementation of multi class classifier one convolutional neural network.

Software and Hardware Requirements
4 GB RAM, 500 GB HDD, PC, Intel I-5
Python programming, Ubuntu.

Theory:

A Convolutional Neural network (CNN) is type of deep learning architecture commonly used for image classification and recognition tasks.
It consists of multiple layer including Convolutional, pooling and fully connected layer.

The Convolutional layer applies filter to input image to extract features, pooling downsamples the image to reduce computation and fully connected layer makes prediction.

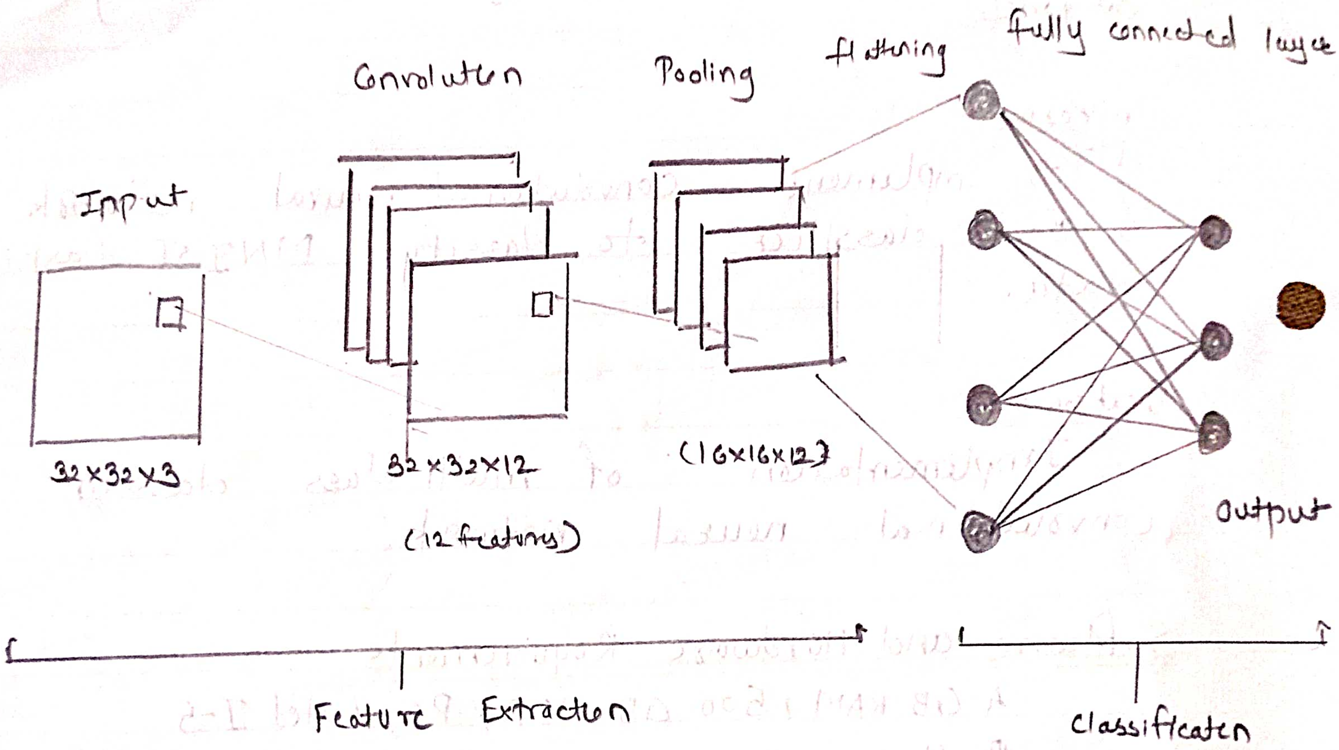


fig: CNN Architecture

layers used to build CNN:

A CNN is sequence of layers and every layer transforms one volume to another through differentiable function.

let us understand with image of dimension (32x32x3)

i) Input layer:
This layer holds the raw input of image with width 32, height 32, depth 3

ii) Convolutional layer:
This layer computes output volume by computing dot product between all filters and image patches. Suppose we have 12 features, then output volume will be 32x32x12

iii) Activation function layer: This layer will apply an element wise activation to output. Some common functions are Relu, sigmoid, etc.

iv) pooling layer:
This layer is periodically inserted in the convnets and its main function is to reduce the size of volume which makes computation fast, reduces overfitting and saves memory as well.

Two common types are:
1) Max pooling
2) Average pooling.

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

max pool (2x2 filter) → stride = 2

6	8
3	4

fig: max-pooling

If we use max pool of 2x2 with stride 2, the resultant volume will be 16x16x12

5. fully connected layer
This layer is regular network layer, that takes flattened input from previous layer and computes class scores and output 1-D array of size equal to number of classes.

In the context of the assignment, multi-class classifier can be modelled using CNN for MNIST dataset using keras and scikit learn ~~mod~~ libraries.

Flattening layer
flattening is nothing but converting 3D/2D matrix into 1D input and connecting it to fully connected dense layer for classification.

Dataset:
i) fashion MNIST consist of 60,000 images for training and 10,000 for testing
ii) Each image of (28x28), and grayscale
iii) There are in total 10 labels.

The Dataset can be trained and tested over CNN using keras, pandas, sklearn libraries.

Date

Conclusion:

Thus we have successfully implemented CNN model (convolutional neural network) by for MNIST fashion dataset.

```python
import tensorflow.keras as keras
from tensorflow.keras import layers
import tensorflow
(X_train, y_train), (X_test, y_test) =
keras.datasets.fashion_mnist.load_data()
```

```python
print("Training data shape: ", X_train.shape)
print("Training labels shape: ", y_train.shape)
print("Test data shape: ", X_test.shape)
print("Test labels shape: ", y_test.shape)
```

```
Training data shape:  (60000, 28, 28)
Training labels shape:  (60000,)
Test data shape:  (10000, 28, 28)
Test labels shape:  (10000,)
```

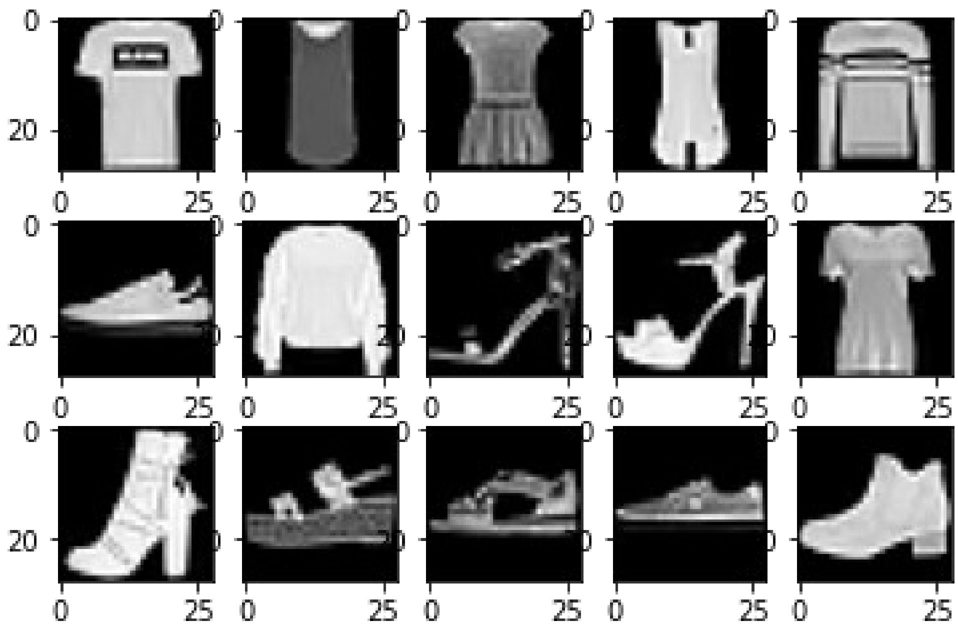
```python
import matplotlib.pyplot as plt

for i in range(1, 16):

    plt.subplot(3, 5, i)

    plt.imshow(X_train[i], cmap=plt.get_cmap('gray'))

# Display the entire plot
plt.show()
```



```python
# Reshape the data to add a channel dimension for convolutional layers
X_train = X_train.reshape(-1, 28, 28, 1)
X_test = X_test.reshape(-1, 28, 28, 1)
```

```python
X_train = X_train.astype("float32") / 255
X_test = X_test.astype("float32") / 255


model = tensorflow.keras.Sequential([
  layers.Conv2D(32, (3,3), activation='relu', input_shape=(28, 28,
1)),
  layers.MaxPooling2D((2,2)),
  layers.Conv2D(64, (3,3), activation='relu'),
  layers.MaxPooling2D((2,2)),
  layers.Conv2D(64, (3,3), activation='relu'),
  layers.Flatten(),
  layers.Dense(64, activation='relu'),
  layers.Dense(10, activation='softmax')
])


# Compile the model
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```python
# Train the model
model.fit(X_train, y_train, epochs=10, validation_data=(X_test,
y_test))
```

```
Epoch 1/10
1875/1875 [==============================] - 22s 11ms/step - loss:
0.4960 - accuracy: 0.8196 - val_loss: 0.3576 - val_accuracy: 0.8702
Epoch 2/10
1875/1875 [==============================] - 22s 12ms/step - loss:
0.3194 - accuracy: 0.8839 - val_loss: 0.3091 - val_accuracy: 0.8873
Epoch 3/10
1875/1875 [==============================] - 21s 11ms/step - loss:
0.2755 - accuracy: 0.8998 - val_loss: 0.2938 - val_accuracy: 0.8932
Epoch 4/10
1875/1875 [==============================] - 21s 11ms/step - loss:
0.2453 - accuracy: 0.9104 - val_loss: 0.2705 - val_accuracy: 0.9032
Epoch 5/10
1875/1875 [==============================] - 21s 11ms/step - loss:
```

```
0.2219 - accuracy: 0.9177 - val_loss: 0.2650 - val_accuracy: 0.9040
Epoch 6/10
1875/1875 [==============================] - 21s 11ms/step - loss: 
0.2021 - accuracy: 0.9249 - val_loss: 0.2733 - val_accuracy: 0.9037
Epoch 7/10
1875/1875 [==============================] - 21s 11ms/step - loss: 
0.1856 - accuracy: 0.9302 - val_loss: 0.2652 - val_accuracy: 0.9043
Epoch 8/10
1875/1875 [==============================] - 22s 12ms/step - loss: 
0.1692 - accuracy: 0.9361 - val_loss: 0.2708 - val_accuracy: 0.9077
Epoch 9/10
1875/1875 [==============================] - 21s 11ms/step - loss: 
0.1558 - accuracy: 0.9430 - val_loss: 0.2557 - val_accuracy: 0.9109
Epoch 10/10
1875/1875 [==============================] - 22s 11ms/step - loss: 
0.1407 - accuracy: 0.9475 - val_loss: 0.2763 - val_accuracy: 0.9096

<keras.callbacks.History at 0x27c42d20160>
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928     
                                                                 
 flatten (Flatten)           (None, 576)               0         
                                                                 
 dense (Dense)               (None, 64)                36928     
                                                                 
 dense_1 (Dense)             (None, 10)                650       
                                                                 
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

```python
from sklearn.metrics import confusion_matrix
import numpy as np

y_pred =model.predict(X_test)
Y_pred_classes = np.argmax(y_pred,axis = 1)
Y_true = y_test
acc = accuracy_score(Y_true,Y_pred_classes)
class_report = classification_report(Y_true,Y_pred_classes)



print('The accuracy: {}'.format(acc))
print('The Classification Report:\n {}'.format(class_report))
```

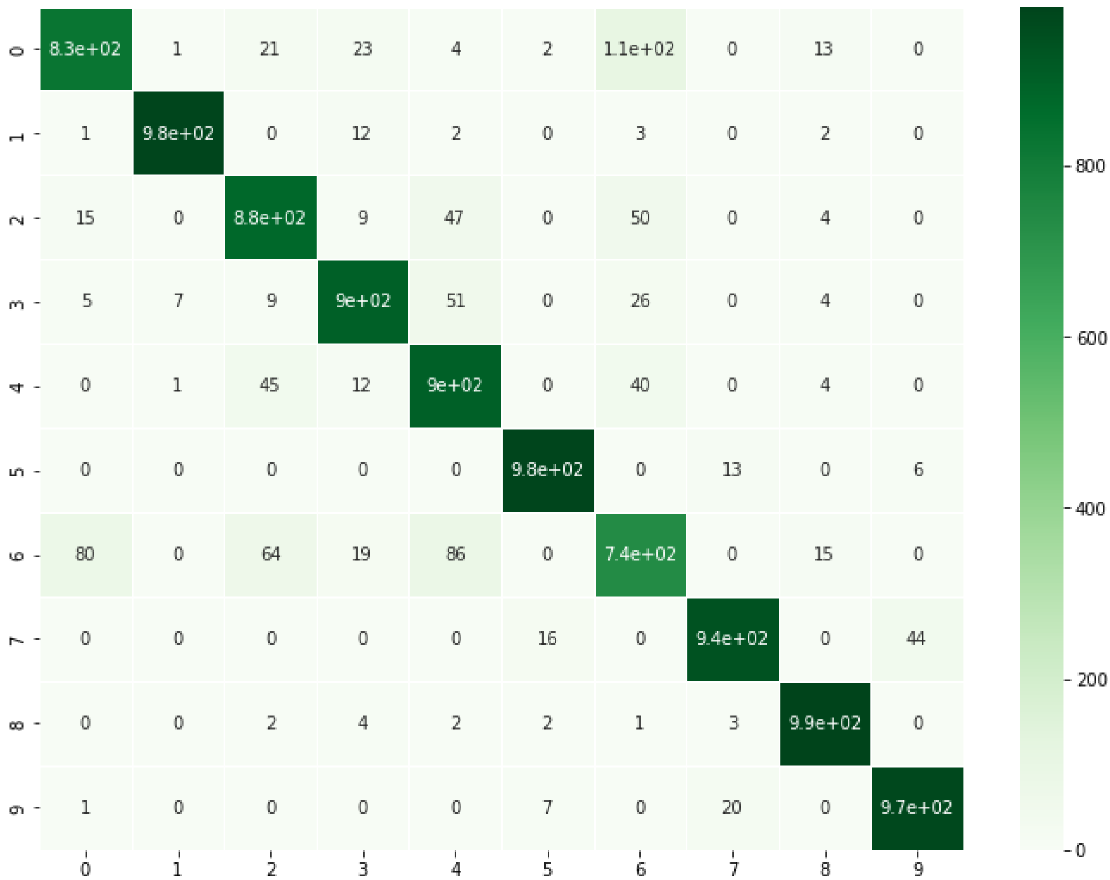
```
The accuracy: 0.9096
The Classification Report:
               precision    recall  f1-score   support

           0       0.89      0.83      0.86      1000
           1       0.99      0.98      0.99      1000
           2       0.86      0.88      0.87      1000
           3       0.92      0.90      0.91      1000
           4       0.82      0.90      0.86      1000
           5       0.97      0.98      0.98      1000
           6       0.77      0.74      0.75      1000
           7       0.96      0.94      0.95      1000
           8       0.96      0.99      0.97      1000
           9       0.95      0.97      0.96      1000

    accuracy                           0.91     10000
   macro avg       0.91      0.91      0.91     10000
weighted avg       0.91      0.91      0.91     10000
```

```python
import seaborn as sns
cm = confusion_matrix(Y_true, Y_pred_classes)
plt.figure(figsize = (12,9))
sns.heatmap(cm, annot=True,cmap="Greens",linewidths=.5)
```

```
<AxesSubplot:>
```



8.3e+02 1 21 23 4 2 1.1e+02 0 13 0
1 9.8e+02 0 12 2 0 3 0 2 0
15 0 8.8e+02 9 47 0 50 0 4 0
5 7 9 9e+02 51 0 26 0 4 0
0 1 45 12 9e+02 0 40 0 4 0
0 0 0 0 0 9.8e+02 0 13 0 6
80 0 64 19 86 0 7.4e+02 0 15 0
0 0 0 0 0 16 0 9.4e+02 0 44
0 0 2 4 2 2 1 3 9.9e+02 0
1 0 0 0 0 7 0 20 0 9.7e+02
0 1 2 3 4 5 6 7 8 9
800
600
400
200
0