Apparel Images Classification Using CNN

Deep Learning & Artificial Intelligence

Author:

Mohammad Azizul Huq

RUC

INDEX

1. INTRODUCTION

The deep learning method is part of a broader family of machine learning methods based on the layers used in artificial neural networks. The neural network is another technique to process the object like a human brain. It is a technique of extracting, analysing and understanding of useful information from its input. In this project, we are using the Convolution Neural network for image classification. CNN is a class of deep learning, and it is an important method in analysing images.

This kind of applications is using in E-commerce, Social media and in the law sector to identify suspects with their clothes.

2.CONVOLUTIONAL NEURAL NETWORK

Convolution Neural Network is a deep learning algorithm which is used for recognising images. In Convolution neural networks, convolutions over the input layer are used to compute the output. These results in local connections, where each region of the input is connected to a neuron in the output. Each layer applies different filters and combines their results.

2.1 Convolution Operation:

A convolution extracts images of the input feature map and applies filters to them to compute new features, producing an output feature map.

1st its extract the size of the images typically (3x3) or (5x5) pixels.

2nd depth of the output feature map which corresponds to the number of filters that are applied.

During a convolution, the filters a matrix is the same size of the image which effectively slides over the input feature maps. It extracts one pixel at a time on the corresponding image.

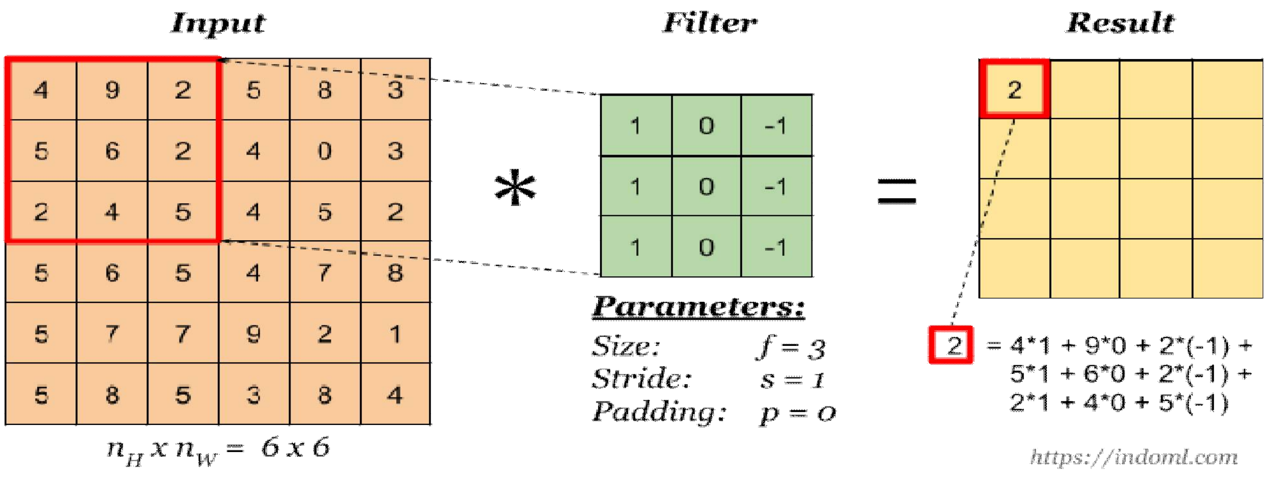


Fig: Convolvolution Operation from https://indoml.com

2.2 <u>ReLU Layer</u>:

Rectified Linear Unit(ReLU) transform function only activates a node if the input is above a certain quantity. While the input is below zero, then the output is zero. If the input rises above a certain threshold, it has a linear relationship with the dependent variable. It is applied into feature maps to increase non-linearity in the network.

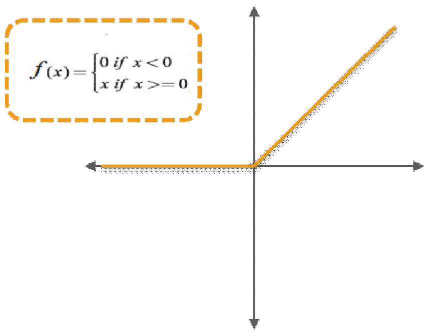


Fig: ReLU from internet

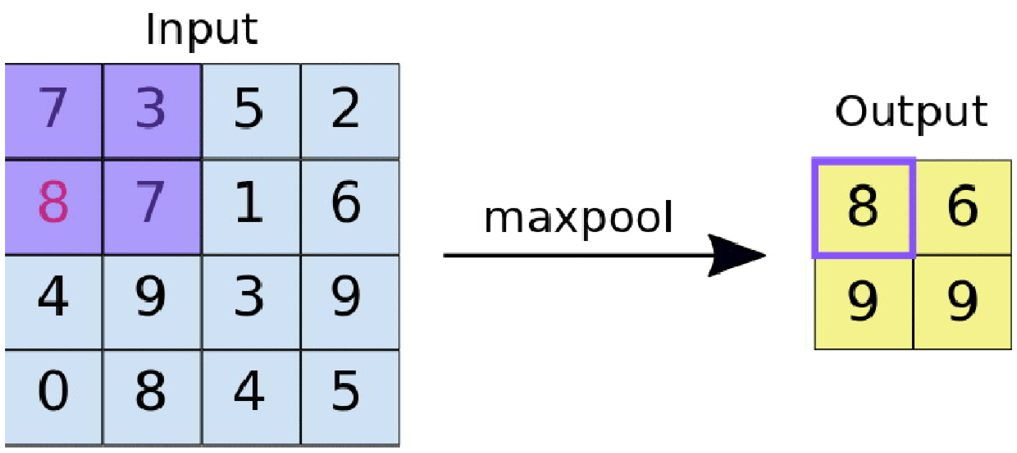
2.3 <u>Max Pooling</u>:

When ReLU comes into the pooling step, The CNN downsamples the convolved the feature reducing the number of dimensions of the feature map. Max pooling operates a similar fashion to convolution. It is grabbing the maximum value at each spot in the image and create a new feature map.

=>Size: The Max-pooling filter size typically 2x2 pixels

=>Stride: The distance in pixels, separating each extracted tile.stride is determine the location where each tile extracted 2x2 filter, a stride 2 specifies that the max pooling operation will extract all no overlapping 2x2 tiles from the feature map.

Max pooling helps to reduce the amount of processing time as well as required parameters, and it also helps to control overfitting.



2.4 **Flattening**:

In this step, the pooled feature map keep into a sequential column of numbers which allows the information to become the input layer of an Neural Network.

2.5 **Full Connection Layer:**

Lastly, the CNN are one or more fully connected with the artificial neural network, when two layers are 'fully connected', every node in the first layer is connected to every node in the second layer. The job is to perform classification based on features extracted by the convolution.

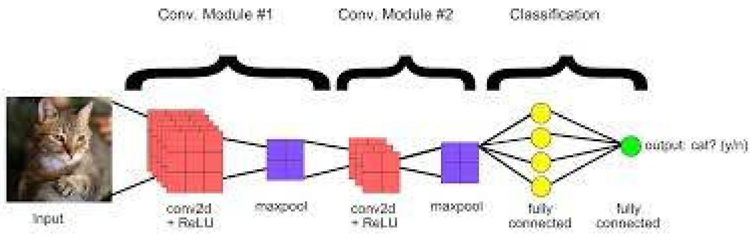


Figure: Convolution Neural Network from developers.google.com

3. DATA SET

The dataset is taken from the Kaggle website to analyse the data. The dataset contains70,000 images divide into a training set of 60,000 images and a test set of 10,000 images. Each image contains 28 pixels in height and 28 pixels in width and a total of 28*28=784 pixels. It is associated with a label from 10 classes.

The 10 classes are given below:

0 => T-shirt/top 1 => Trouser 2 => Pullover 3 => Dress 4 => Coat 5 => Sandal 6 => Shirt 7 => Sneaker 8 => Bag 9 => Ankle boot

3.1 DATA EXPLORATION

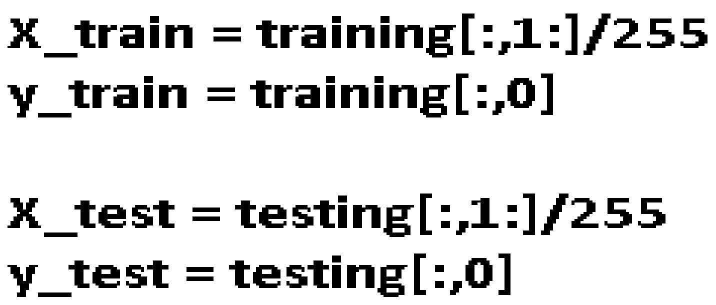
Before training the model, here we analyse how images in the dataset look like. After checked our dataset, we found that the train data has a shape of (60000x28x28) and test data has a shape of (10000x28x28) Here are also ten output classes which ranges are 0 to 9.And each row is representing one image, and each column representing the pixel whose values starts from 1 to 784.

3.2 DATA PRE-PROCESSING

We know that the machine can understand only numbers, so image perception by machine is also a number. The binary 2^8 (255) which represent 256 units of colour.32 bits of each colour with eight levels of colour grading gives a full scale of 256 units of colour definition.

Though these images are grayscale and images have a dimension of 28*28, and it is representing 0 for Black and 255 for white. Grayscale images have only one channel, but the RGB scale contains three channel.

To reduce the effect of illumination difference, we will normalise our datasets which will also contribute to work CNN faster. So before feeding to the neural network, we divide the values by 255, and It will scale the values to a range 0 to 1.



```
X_train = training[:,1:]/255
y_train = training[:,0]

X_test = testing[:,1:]/255
y_test = testing[:,0]
```

Later on, we use some code to visualise some images to confirm the data is an incorrect format.

3.3 Data Partitioning:

We split our data set 20 % for test and 80% for training randomly from our dataset. Here we also reshaped our Train and test dataset 28x28x1. So now we have 48000 samples, and each one of them is 28x28x1 We also have 1200 samples to validate, and in another file, we have 10000 samples for testing. It is a significant part which will be accepted by our convolution neural network.

4. DATA MODELING

Now we have to configure the layer of the model for the neural network. We loaded the necessary library file for the model. Here we use the Keras and TensorFlow in backend for our convolution neural network.

```python
# Import train_test_split from scikit library
# Import Keras
import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
from keras.optimizers import Adam
from keras.callbacks import TensorBoard
```

```python
cnn_model = Sequential()

# Try 32 fliters first then 64
cnn_model.add(Conv2D(64,3, 3, input_shape = (28,28,1), activation='relu'))
cnn_model.add(MaxPooling2D(pool_size = (2, 2)))

cnn_model.add(Dropout(0.25))

# cnn_model.add(Conv2D(32,3, 3, activation='relu'))
# cnn_model.add(MaxPooling2D(pool_size = (2, 2)))

cnn_model.add(Flatten())
cnn_model.add(Dense(output_dim = 32, activation = 'relu'))
cnn_model.add(Dense(output_dim = 10, activation = 'sigmoid'))
```

```python
cnn_model.compile(loss ='sparse_categorical_crossentropy', optimizer=Adam(lr=0.001),metrics =['accuracy'])
```

```python
epochs = 50

history = cnn_model.fit(X_train,
                        y_train,
                        batch_size = 512,
                        nb_epoch = epochs,
                        verbose = 1,
                        validation_data = (X_validate, y_validate))
```

First, we imported the Keras and TensorFlow backend and next step we imported our sequential, convolutional layers, MaxPooling, Dense, Flatten and Dropout. We also imported Adam for optimisation. Also, we are using TensoBoard just for calling.

Here convolution uses a kernel matrix to scan a given image and apply a filter to obtain a certain effect. An image kernel is a matrix used to apply effects such as blurring and sharpening. It is used in machine learning for feature extraction to select the essential pixels of an image.

In shortly specified, here our kernels 32x3x3 and input shape 28x28x1 which is the size of our image. Moreover, then we specify our activation function(ReLU) and the next steps we added Max pulling layer.so we specify our pulling size (2x2)

We also added flatten to our model, flatten will make our feature into one single array. It will feed or fully dense to fully connected neural network in Artificial Neural Network. In the next step our function Dense output dimension equal to 32 and specify the activation ReLU.

So, Finally, our target to get Ten classes, so we specified i, and for the outputs, we changed the activation layer in Sigmoid function.

4.1 <u>Data Optimize</u>

To optimise the output we just categorised by Adam optimiser.

4.1.1 Epoch and Batch Size :

In order to compile our model, we need to specify how many epoch we want to run. Here we choose epoch will execute 50 times for presenting our dataset and updating the weights.

4.1.2 Fit the model:

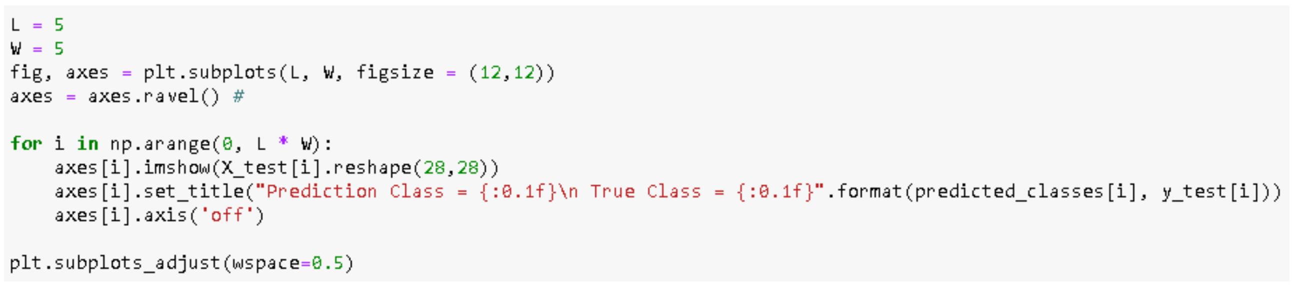
Here we apply the fit method to our CNN model, and then we used our training data and validation data. We also specified the memory size and number of epochs in this process we are performing the training.

4.1.3 Evaluate the model

So here we evaluate our testing data which provided the accuracy from our testing data set. So it seems the accuracy is 91 %.

ETA: 0sTest Accuracy : 0.910

4.1.4 Prediction the Test Data:

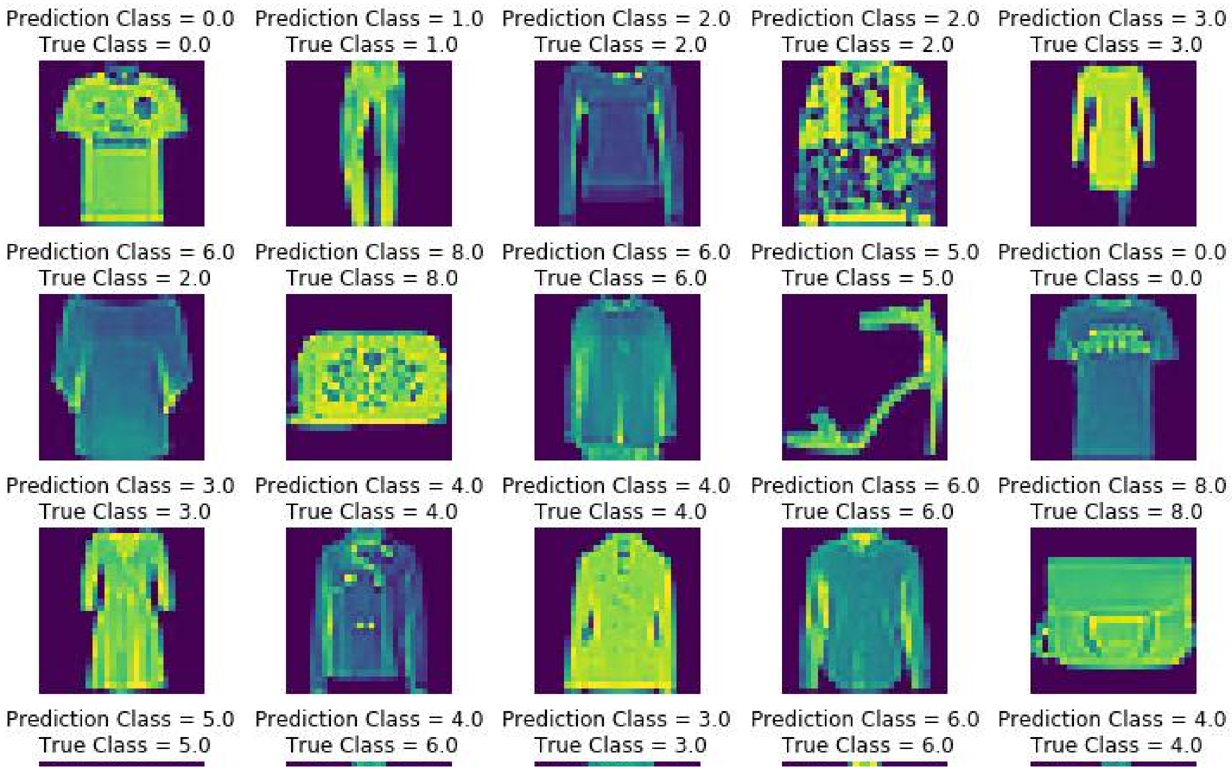


```
L = 5
W = 5
fig, axes = plt.subplots(L, W, figsize = (12,12))
axes = axes.ravel() #

for i in np.arange(0, L * W):
    axes[i].imshow(X_test[i].reshape(28,28))
    axes[i].set_title("Prediction Class = {:0.1f}\n True Class = {:0.1f}".format(predicted_classes[i], y_test[i]))
    axes[i].axis('off')

plt.subplots_adjust(wspace=0.5)
```

Here we flatten our axes array and then next step we used a loop from 0 to 25 which is the overall number of sample images and each image reshaped in a 28x28 format.



5. CONCLUSION

It was a good learning experience in deep learning and Neural network. Convolutional Neural Network applied to classify the image in this mini project. The accuracy of the prediction was 91 %.

6. REFERENCES

1.https://www.kaggle.com/

2.https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks

3. https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/neural_networks.html