ASSIGNMENT NO.6

TITLE: CONVOLUTIONAL NEURAL NETWORK (CNN)

Problem Statement :

Use MNIST Fashion Dataset and create a classifier to classify fashion clothing into categories.

Objective :

To understand and implement:

1. CNN
2. Create a classifier to classify fashion clothing into categories.

Theory:

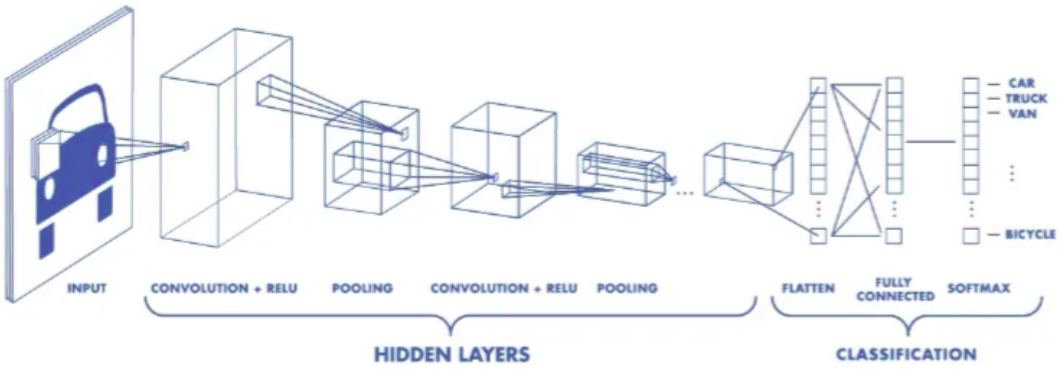
What is CNN?

A CNN, or Convolutional Neural Network, is a type of neural network commonly used in deep learning for image and video recognition tasks. It is made up of multiple layers of filters that are applied to an input image to extract features and patterns. These filters, called convolutional layers, help to reduce the dimensions of the image while preserving important features. The output of the convolutional layers is then fed into fully connected layers, which perform the classification task. CNNs have shown to be very effective in tasks such as image classification, object detection and facial recognition.

Convolutional Neural Network Architecture :

A CNN typically has three layers:

1. a convolutional layer,
2. a pooling layer and
3. a fully connected layer



Convolution Layer :

The convolution layer is the most important part of a CNN and carries most of the computational load. It uses a kernel matrix to perform a dot product with a restricted portion of the input image called the receptive field. The kernel is smaller than the image but has the same depth as the image's channels. During the forward pass, the kernel slides over the image and produces an activation map that shows the kernel's response at each spatial position of the image. The stride is the size of the kernel's sliding step.

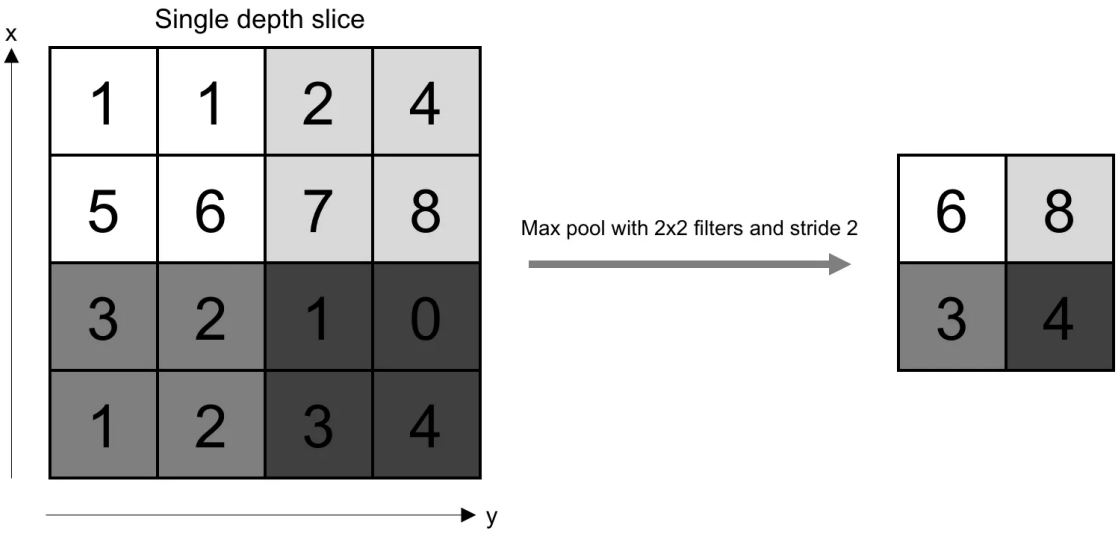
If we have an input of size W x W x D and Dout number of kernels with a spatial size of F with stride S and amount of padding P, then the size of output volume can be determined by the following formula:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

Formula for Convolution Layer

Pooling Layer :

In a CNN, the pooling layer reduces the spatial size of the input representation by summarising nearby outputs with a pooling function, such as max pooling. This decreases the computation and weights required in the network. The pooling operation is done individually on every slice of the input representation.



If we have an activation map of size W x W x D, a pooling kernel of spatial size F, and stride S, then the size of output volume can be determined by the following formula:

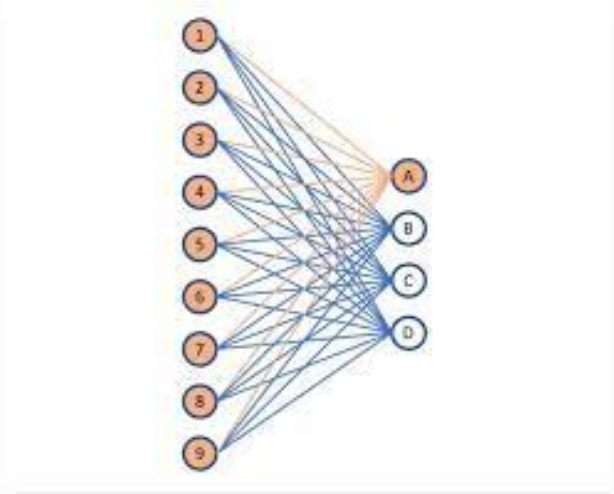
$$W_{out} = \frac{W - F}{S} + 1$$

Formula for Pooling Layer

Fully Connected Layer:

In a CNN, the fully connected layer is the last layer of the network that performs the classification task. It takes the output of the preceding convolutional and pooling layers and flattens it into a one-dimensional vector. This vector is then fed into a traditional neural network with fully connected layers, which learns to

classify the input into one or more classes. The weights in the fully connected layers are learned through backpropagation during the training process.



Steps to create a CNN Model :

Step 1: Import the Library

```
import numpy as np
from keras.utils import to_categorical
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.layers import BatchNormalization
```

Step 2 : Loading the Dataset

```
from keras.datasets import fashion_mnist
(train_X,train_Y), (test_X,test_Y) = fashion_mnist.load_data()
```

The keras.datasets module provides a collection of benchmark datasets for machine learning. In this case, we are importing the fashion_mnist dataset using the load_data() function. The function returns two tuples, the first containing the training data and labels (train_X and train_Y), and the second containing the test data and labels (test_X and test_Y).

Step 3 : Creating model and adding layers

```
fashion_model = Sequential()
```

Once the model is created, we can add layers to it using various Keras layer classes, such as Dense, Conv2D, and MaxPooling2D. These layers define the structure of the neural network and how it processes the input data.

```
fashion_model.add(Dropout(0.4))
fashion_model.add(Flatten())
fashion_model.add(Dense(128, activation='linear'))
fashion_model.add(LeakyReLU(alpha=0.1))
fashion_model.add(Dropout(0.3))
fashion_model.add(Dense(num_classes, activation='softmax'))
```

After adding the layers to the model, we can compile it using the compile() method, which sets the optimizer, loss function, and evaluation metrics for the model. Finally, we can train the model on the training data using the fit() method and evaluate its performance on the test data using the evaluate() method.

Step 4 : Compiling the model

```
fashion_model.compile(loss=keras.losses.categorical_crossentropy,
optimizer=keras.optimizers.Adam(),metrics=['accuracy'])
```

loss: This parameter specifies the loss function to use during training. In this case, the categorical cross-entropy loss is used, which is a common loss function for multi-class classification problems.
optimizer: This parameter specifies the optimizer algorithm to use during training. In this case, the Adam optimizer is used, which is a popular optimizer for deep learning models due to its adaptive learning rate and momentum.

metrics: This parameter specifies the evaluation metrics to use during training and testing. In this case, we are using the 'accuracy' metric to evaluate the performance of the model.

Step 5 : Fitting the model

```
fashion_train= fashion_model.fit(train_X, train_label,
batch_size=batch_size,epochs=epochs,verbose=1,validation_data=(valid_X,
valid_label))
```

train_X: This parameter represents the input training data.
train_label: This parameter represents the corresponding training labels.
batch_size: This parameter specifies the number of samples to use in each batch of training.
epochs: This parameter specifies the number of epochs (iterations over the entire training dataset) to train the model for.
verbose: This parameter specifies the verbosity mode during training. In this case, it is set to 1, which means progress updates will be printed to the console during training.
validation_data: This parameter specifies the validation data to use during training. In this case, it is set to (valid_X, valid_label), which represents the input validation data and corresponding validation labels.

Step 6 : Evaluating the model

```
from sklearn.metrics import classification_report
target_names = ["Class {}".format(i) for i in range(num_classes)]
print(classification_report(test_Y, predicted_classes, target_names=target_names))
```

test_Y: This parameter represents the true class labels for the test set.
predicted_classes: This parameter represents the predicted class labels generated by the model on the test set.
target_names: This parameter specifies the names of the classes in the dataset.
The classification_report() function computes and prints several classification metrics, such as precision, recall, and F1-score, for each class in the dataset, as well as

the overall accuracy of the model. The target_names parameter is used to label each class in the report.

Conclusion:

Using the MNIST Fashion Dataset, we have trained a CNN model to classify fashion clothing into categories.We have successfully trained a deep learning model that can classify fashion clothing with high accuracy. This can be useful in a variety of applications, such as online fashion retail, visual search, and recommendation systems.