In [1]:

```
'''Name: Apeksha Chavan
BE COMPS
UID:2017130013
FCI Exp 2: Experiment on studying different CNN architectures

What is Convolutional Neural Network?
A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign in
(learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the othe
The pre-processing required in a ConvNet is much lower as compared to other classification algorithms.
While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn thes
The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was
Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field.
A collection of such fields overlap to cover the entire visual area.

Basic Architecture
There are two main parts to a CNN architecture
A convolution tool that separates and identifies the various features of the image for analysis in a process called a
A fully connected layer that utilizes the output from the convolution process and predicts the class of the image bas

Convolution Layers
There are three types of layers that make up the CNN which are the convolutional layers, pooling layers,
and fully-connected (FC) layers. When these layers are stacked, a CNN architecture will be formed.
In addition to these three layers, there are two more important parameters which are the dropout layer and the activa

1. Convolutional Layer
This layer is the first layer that is used to extract the various features from the input images.
In this layer, the mathematical operation of convolution is performed between the input image and a filter of a parti
By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input in

The output is termed as the Feature map which gives us information about the image such as the corners and edges.
Later, this feature map is fed to other layers to learn several other features of the input image.

2. Pooling Layer
In most cases, a Convolutional Layer is followed by a Pooling Layer.
The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs
This is performed by decreasing the connections between layers and independently operates on each feature map.
Depending upon method used, there are several types of Pooling operations.

In Max Pooling, the largest element is taken from feature map.
Average Pooling calculates the average of the elements in a predefined sized Image section.
```

```
The total sum of the elements in the predefined section is computed in Sum Pooling.
The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer

3. Fully Connected Layer
The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the n
These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

In this, the input image from the previous layers are flattened and fed to the FC layer.
The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place
In this stage, the classification process begins to take place.

4. Dropout
Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset.
Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model

To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network durin
On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.

5. Activation Functions
Finally, one of the most important parameters of the CNN model is the activation function.
They are used to learn and approximate any kind of continuous and complex relationship between variables of the netwo
In simple words, it decides which information of the model should fire in the forward direction and which ones should

It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Softmax,
Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and softmax functions a
```

```
Out[1]: 'Name: Apeksha Chavan\nBE COMPS\nUID:2017130013\nFCI Exp 2: Experiment on studying different CNN architectures\n\nWha
t is Convolutional Neural Network?\nA Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which c
an take in an input image, assign importance \n(learnable weights and biases) to various aspects/objects in the image
and be able to differentiate one from the other. \nThe pre-processing required in a ConvNet is much lower as compared
to other classification algorithms. \nWhile in primitive methods filters are hand-engineered, with enough training, C
onvNets have the ability to learn these filters/characteristics. \nThe architecture of a ConvNet is analogous to that
of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex.
\nIndividual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field.
\nA collection of such fields overlap to cover the entire visual area.\n\nBasic Architecture\nThere are two main part
s to a CNN architecture\nA convolution tool that separates and identifies the various features of the image for analy
sis in a process called as Feature Extraction\nA fully connected layer that utilizes the output from the convolution
process and predicts the class of the image based on the features extracted in previous stages.\n\nConvolution Layers
\nThere are three types of layers that make up the CNN which are the convolutional layers, pooling layers, \nand full
y-connected (FC) layers. When these layers are stacked, a CNN architecture will be formed.\nIn addition to these thre
e layers, there are two more important parameters which are the dropout layer and the activation function which are d
efined below.\n\n1. Convolutional Layer\nThis layer is the first layer that is used to extract the various features f
rom the input images. \nIn this layer, the mathematical operation of convolution is performed between the input image
```

```
and a filter of a particular size MxM. \nBy sliding the filter over the input image, the dot product is taken between
the filter and the parts of the input image with respect to the size of the filter (MxM).\n\nThe output is termed as
the Feature map which gives us information about the image such as the corners and edges. \nLater, this feature map i
s fed to other layers to learn several other features of the input image.\n\n2. Pooling Layer\nIn most cases, a Convo
lutional Layer is followed by a Pooling Layer. \nThe primary aim of this layer is to decrease the size of the convolv
ed feature map to reduce the computational costs. \nThis is performed by decreasing the connections between layers an
d independently operates on each feature map. \nDepending upon method used, there are several types of Pooling operat
ions.\n\nIn Max Pooling, the largest element is taken from feature map. \nAverage Pooling calculates the average of t
he elements in a predefined sized Image section. \nThe total sum of the elements in the predefined section is compute
d in Sum Pooling. \nThe Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer\n\n
3. Fully Connected Layer\nThe Fully Connected (FC) layer consists of the weights and biases along with the neurons an
d is used to connect the neurons between two different layers. \nThese layers are usually placed before the output la
yer and form the last few layers of a CNN Architecture.\n\nIn this, the input image from the previous layers are flat
tened and fed to the FC layer. \nThe flattened vector then undergoes few more FC layers where the mathematical functi
ons operations usually take place. \nIn this stage, the classification process begins to take place.\n\n4. Dropout\nU
sually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. \nOver
fitting occurs when a particular model works so well on the training data causing a negative impact in the model’s pe
rformance when used on a new data.\n\nTo overcome this problem, a dropout layer is utilised wherein a few neurons are
dropped from the neural network during training process resulting in reduced size of the model. \nOn passing a dropou
t of 0.3, 30% of the nodes are dropped out randomly from the neural network.\n\n5. Activation Functions\nFinally, one
of the most important parameters of the CNN model is the activation function. \nThey are used to learn and approximat
e any kind of continuous and complex relationship between variables of the network. \nIn simple words, it decides whi
ch information of the model should fire in the forward direction and which ones should not at the end of the networ
k.\n\nIt adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, So
ftmax, tanH and the Sigmoid functions. \nEach of these functions have a specific usage. For a binary classification C
NN model, sigmoid and softmax functions are preferred an for a multi-class classification, generally softmax us use
d.'
```

In [2]:

```python
import tensorflow as tf
from tensorflow import keras
import numpy as np
import time
import os
```

In [3]:

```python
(train_x, train_y), (test_x, test_y) = keras.datasets.mnist.load_data()
train_x = train_x / 255.0
test_x = test_x / 255.0

train_x = tf.expand_dims(train_x, 3)
test_x = tf.expand_dims(test_x, 3)
```

```
val_x = train_x[:5000]
val_y = train_y[:5000]
```

In [4]:

```
lenet_5_model = keras.models.Sequential([
    keras.layers.Conv2D(6, kernel_size=5, strides=1,  activation='tanh', input_shape=train_x[0].shape, padding='same'
    keras.layers.AveragePooling2D(), #S2    #activation function = Sigmoid
    keras.layers.Conv2D(16, kernel_size=5, strides=1, activation='tanh', padding='valid'), #C3
    keras.layers.AveragePooling2D(), #S4    #activation function = Sigmoid
    keras.layers.Flatten(), #Flatten
    keras.layers.Dense(120, activation='tanh'), #C5
    keras.layers.Dense(84, activation='tanh'), #F6
    keras.layers.Dense(10, activation='softmax') #Output layer  (In total 7 layers(1 Flatten layer) + 1 Output layer,
])
```

In [5]:

```
#Now we can compile and build the model
lenet_5_model.compile(optimizer='adam', loss=keras.losses.sparse_categorical_crossentropy, metrics=['accuracy'])
```

In [6]:

```
root_logdir = os.path.join(os.curdir, "logs\\fit\\")

def get_run_logdir():
    run_id = time.strftime("run_%Y_%m_%d-%H_%M_%S")
    return os.path.join(root_logdir, run_id)

run_logdir = get_run_logdir()
tensorboard_cb = keras.callbacks.TensorBoard(run_logdir)
```

In [7]:

```
lenet_5_model.fit(train_x, train_y, epochs=5, validation_data=(val_x, val_y), callbacks=[tensorboard_cb])
```

```
Epoch 1/5
1875/1875 [==============================] - 32s 17ms/step - loss: 0.2289 - accuracy: 0.9314 - val_loss: 0.0874 - val
_accuracy: 0.9730
Epoch 2/5
1875/1875 [==============================] - 30s 16ms/step - loss: 0.0857 - accuracy: 0.9736 - val_loss: 0.0596 - val
_accuracy: 0.9816
Epoch 3/5
1875/1875 [==============================] - 26s 14ms/step - loss: 0.0570 - accuracy: 0.9830 - val_loss: 0.0355 - val
_accuracy: 0.9896
```

```
Epoch 4/5
1875/1875 [==============================] - 26s 14ms/step - loss: 0.0447 - accuracy: 0.9860 - val_loss: 0.0329 - val
_accuracy: 0.9902
Epoch 5/5
1875/1875 [==============================] - 28s 15ms/step - loss: 0.0355 - accuracy: 0.9888 - val_loss: 0.0259 - val
_accuracy: 0.9922
```

Out[7]:
```
<tensorflow.python.keras.callbacks.History at 0x22c216ff070>
```

In [8]:
```
lenet_5_model.evaluate(test_x, test_y)
```

```
313/313 [==============================] - 2s 6ms/step - loss: 0.0493 - accuracy: 0.9844
```

Out[8]:
```
[0.0492842011153698, 0.9843999743461609]
```

In [2]:
```
'''CONCLUSION:
In this experiment, I have used the LENET-5 architecture, trained and tested the MNIST dataset,
Keras presents a Sequential API for stacking layers of the neural network on top of each other, so for the CNN archi
I have used  the Keras tools required to implement the classification model.
After training, the model achieves a validation accuracy of over 90%.
After training my model, I was able to achieve 98% accuracy on the test dataset, which is quite useful for such a si
```

Out[2]:
```
'CONCLUSION:\nIn this experiment, I have used the LENET-5 architecture, trained and tested the MNIST dataset,\nKeras
presents a Sequential API for stacking layers of the neural network on top of each other, so for the CNN architecture
layers\nI have used  the Keras tools required to implement the classification model.\nAfter training, the model achie
ves a validation accuracy of over 90%.\nAfter training my model, I was able to achieve 98% accuracy on the test datas
et, which is quite useful for such a simple network.'
```

In [ ]:
```

```