

In [2]:

```
# a Keras implementation of the LeNet-5 deep Learning
# neural network and the output of the summary of the model from keras.models
# import Sequential
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, AveragePooling2D
from keras import optimizers
model = Sequential()
model.add(Conv2D(filters=6, kernel_size=(3, 3), activation='relu',
input_shape=(32,32,1)))
model.add(AveragePooling2D())
model.add(Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model.add(AveragePooling2D())
model.add(Flatten())
model.add(Dense(units=120, activation='relu'))
model.add(Dense(units=84, activation='relu'))
model.add(Dense(units=10, activation = 'softmax'))
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 30, 30, 6)	60
average_pooling2d (AveragePooling2D)	(None, 15, 15, 6)	0
conv2d_1 (Conv2D)	(None, 13, 13, 16)	880
average_pooling2d_1 (AveragePooling2D)	(None, 6, 6, 16)	0
flatten (Flatten)	(None, 576)	0
dense (Dense)	(None, 120)	69240
dense_1 (Dense)	(None, 84)	10164
dense_2 (Dense)	(None, 10)	850
=====		
Total params: 81,194		
Trainable params: 81,194		
Non-trainable params: 0		

In [7]:

```

# a Keras implementation of the AlexNet deep Learning
# neural network and the output of the summary of the model.

import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.layers import BatchNormalization
#Create the AlexNet model
model = Sequential()
# 1st Convolutional Layer
model.add(Conv2D(filters=96, input_shape=(224,224,3), kernel_size=(11,11), activation='relu',
                strides=(4,4), padding='valid'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2),
padding='valid'))
# 2nd Convolutional Layer
model.add(Conv2D(filters=256, kernel_size=(11,11), activation='relu',
                strides=(1,1), padding='valid'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2),
padding='valid'))
# 3rd Convolutional Layer
model.add(Conv2D(filters=384, kernel_size=(3,3), activation='relu',
                strides=(1,1), padding='valid'))
# 4th Convolutional Layer
model.add(Conv2D(filters=384, kernel_size=(3,3), activation='relu',
                strides=(1,1), padding='valid'))
# 5th Convolutional Layer
model.add(Conv2D(filters=256, kernel_size=(3,3), activation='relu',
                strides=(1,1), padding='valid'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
# 1st Fully Connected Layer
model.add(Flatten())
model.add(Dense(4096, activation='relu', input_shape=(224*224*3,)))
model.add(Dropout(0.4))
# 2nd Fully Connected Layer
model.add(Dense(4096,activation='relu'))
model.add(Dropout(0.4))
# 3rd Fully Connected Layer
model.add(Dense(1000,activation='relu'))
model.add(Dropout(0.4))
# Output Layer
model.add(Dense(17,activation='softmax'))
model.summary()
# Compile the model
model.compile(loss=keras.losses.categorical_crossentropy,
optimizer='adam', metrics=["accuracy"])
# Fit the model
#model.fit()
# Prediction with the model
#model.evaluate()

```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
=====		
conv2d_2 (Conv2D)	(None, 54, 54, 96)	34944
max_pooling2d (MaxPooling2D)	(None, 27, 27, 96)	0
)		

conv2d_3 (Conv2D)	(None, 17, 17, 256)	2973952
max_pooling2d_1 (MaxPooling 2D)	(None, 8, 8, 256)	0
conv2d_4 (Conv2D)	(None, 6, 6, 384)	885120
conv2d_5 (Conv2D)	(None, 4, 4, 384)	1327488
conv2d_6 (Conv2D)	(None, 2, 2, 256)	884992
max_pooling2d_2 (MaxPooling 2D)	(None, 1, 1, 256)	0
flatten_1 (Flatten)	(None, 256)	0
dense_3 (Dense)	(None, 4096)	1052672
dropout (Dropout)	(None, 4096)	0
dense_4 (Dense)	(None, 4096)	16781312
dropout_1 (Dropout)	(None, 4096)	0
dense_5 (Dense)	(None, 1000)	4097000
dropout_2 (Dropout)	(None, 1000)	0
dense_6 (Dense)	(None, 17)	17017

```

=====
Total params: 28,054,497
Trainable params: 28,054,497
Non-trainable params: 0

```

In [8]:

```
# a Python code that can load Google Inception V3 and
# show the summary of the models using the Keras built-in functions. It is just
# three lines of code
```

```
from tensorflow.keras.applications import inception_v3
# init the models
model = inception_v3.InceptionV3(weights='imagenet')
print(model.summary())
```

```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels.h5
(https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels.h5)
96112376/96112376 [=====] - 6s 0us/step
Model: "inception_v3"

```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 299, 299, 3)]	0	[]
conv2d_7 (Conv2D)	(None, 149, 149, 32)	864	['input_1[0][0]']
batch_normalization_7 (Batch Normalization)	(None, 149, 149, 32)	0	['conv2d_7']