In [4]:

```
# 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
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
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='
# 4th Convolutional Layer
model.add(Conv2D(filters=384, kernel_size=(3,3), activation='relu',strides=(1,1), padding='
# 5th Convolutional Layer
model.add(Conv2D(filters=256, kernel_size=(3,3), activation='relu',strides=(1,1), padding='
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"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 54, 54, 96)        34944

 max_pooling2d (MaxPooling2D  (None, 27, 27, 96)       0
 )

 conv2d_1 (Conv2D)           (None, 17, 17, 256)       2973952

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 256)        0
 2D)

 conv2d_2 (Conv2D)           (None, 6, 6, 384)         885120
```

```
 conv2d_3 (Conv2D)           (None, 4, 4, 384)         1327488

 conv2d_4 (Conv2D)           (None, 2, 2, 256)         884992

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 256)        0
 2D)

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 4096)              1052672

 dropout (Dropout)           (None, 4096)              0

 dense_1 (Dense)             (None, 4096)              16781312

 dropout_1 (Dropout)         (None, 4096)              0

 dense_2 (Dense)             (None, 1000)              4097000

 dropout_2 (Dropout)         (None, 1000)              0

 dense_3 (Dense)             (None, 17)                17017

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

In [5]:

```
# 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())
```

```
Model: "inception_v3"
__________________________________________________________________________________
_________________________
 Layer (type)                Output Shape              Param #     Connected
to
==================================================================================
=========================
 input_1 (InputLayer)        [(None, 299, 299, 3  0           []
                             )]

 conv2d_5 (Conv2D)           (None, 149, 149, 32  864         ['input_1
[0][0]']
                             )

 batch_normalization (BatchNorm  (None, 149, 149, 32  96         ['conv2d_
5[0][0]']
 alization)                  )

 activation (Activation)     (None, 149, 149, 32  0           ['batch_n
```

In [6]:

```
# A Python code that you can use to create a simple custom built deep learning neural netwo
# summary of the model. It  contains an input layer (28, 28, 1), a convolution layer, a max
# a dropout layer, a flatten layer, a dense layer, and an output layer. The dense
# layer is a neural network layer that each neuron in the dense layer receives
# input from all neurons of its previous layer. The flatten layer flattens the data
# into a one-dimensional array, which is typically used before the dense layer in
# convolutional neural networks.

from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
# Create model
model = Sequential()
model.add(Conv2D(32, (5, 5), input_shape=(28, 28, 1),
activation='relu'))
model.add(MaxPooling2D())
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(2, activation='softmax'))
# Compile model
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_99 (Conv2D)          (None, 24, 24, 32)        832

 max_pooling2d_7 (MaxPooling  (None, 12, 12, 32)       0
 2D)

 dropout_3 (Dropout)         (None, 12, 12, 32)        0

 flatten_1 (Flatten)         (None, 4608)              0

 dense_4 (Dense)             (None, 128)               589952

 dense_5 (Dense)             (None, 2)                 258

=================================================================
Total params: 591,042
Trainable params: 591,042
Non-trainable params: 0
_________________________________________________________________
None
```