# Nueral Network Assi

Pankajan T.

Libraries used

In [ ]:
```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Conv2D, BatchNormalization, Activation, Add, Av
from tensorflow.keras.regularizers import l2
from tensorflow.keras.models import Model
from tensorflow.keras.applications import VGG16
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras import optimizers
from tensorflow.keras.applications import Xception
```

## Dataset used

Here we use CIFAR10 of keras dataset that have extensive images of objects and animals containing 10 classes given below.

'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'

## Goal

Th main objective to develpo the model that can classify images that are each of the above given label. We design the model to be more accurate as possible.

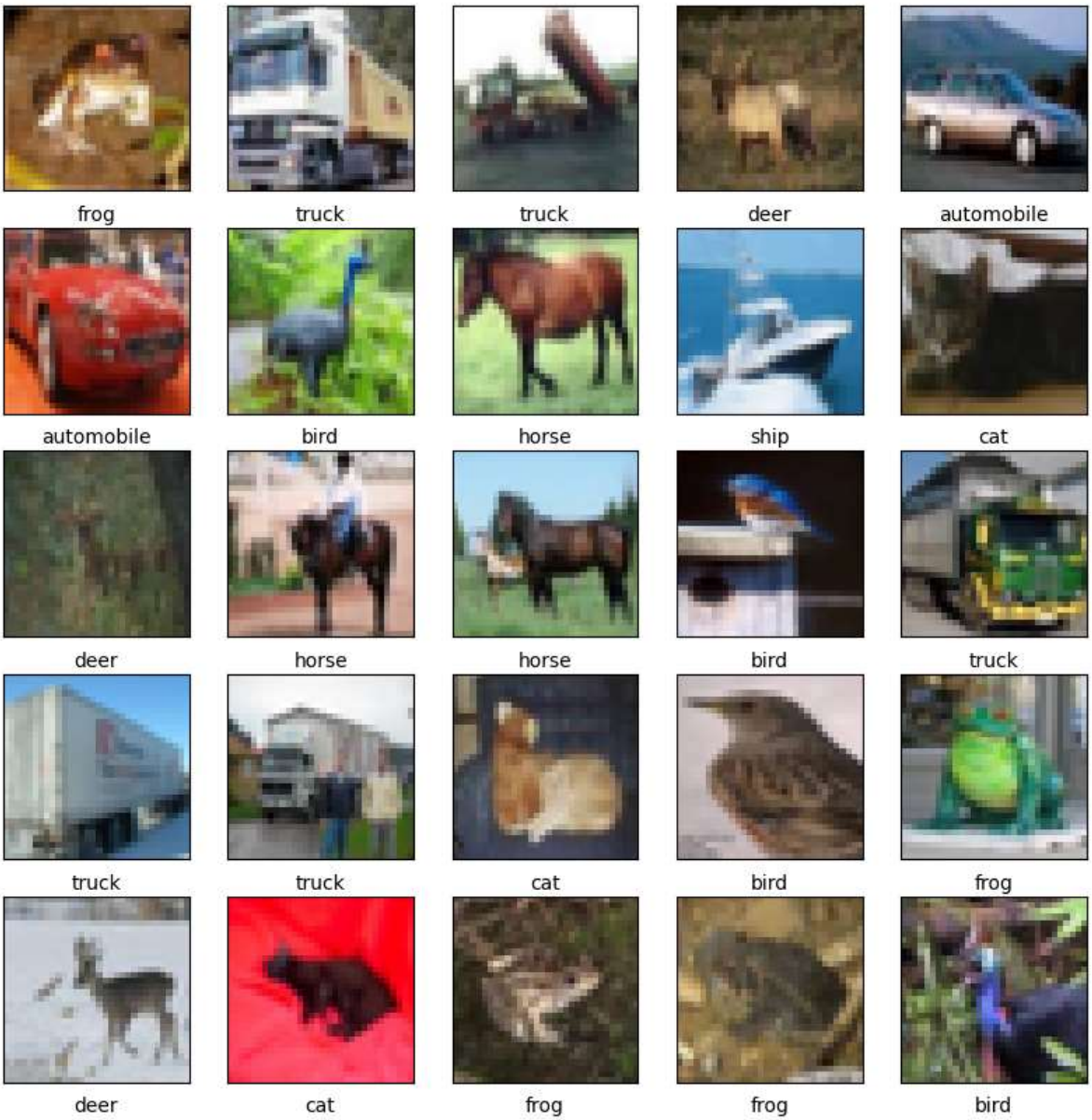
## Data preprocessing

In [ ]:
```
(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

In [ ]:
```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']
```

In [ ]:
```
plt.figure(figsize=(10,10))
for i in range(25):
```

```
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i])
    plt.xlabel(class_names[y_train[i][0]])
plt.show()
```




## Create the model architecture

In [ ]:
```
np.random.seed(42)
tf.random.set_seed(42)
```

## 1. CNN-1

In [ ]:
```
model = keras.models.Sequential()
model.add(keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3
```

```
model.add(keras.layers.MaxPooling2D((2, 2)))
model.add(keras.layers.Conv2D(64, (3, 3), activation='relu'))
model.add(keras.layers.MaxPooling2D((2, 2)))
model.add(keras.layers.Conv2D(64, (3, 3), activation='relu'))
```

In [ ]:
```
model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_11 (Conv2D)          (None, 30, 30, 32)        896

 max_pooling2d_7 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_12 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_8 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_13 (Conv2D)          (None, 4, 4, 64)          36928

=================================================================
Total params: 56,320
Trainable params: 56,320
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dense(10))
```

In [ ]:
```
model.summary()
```

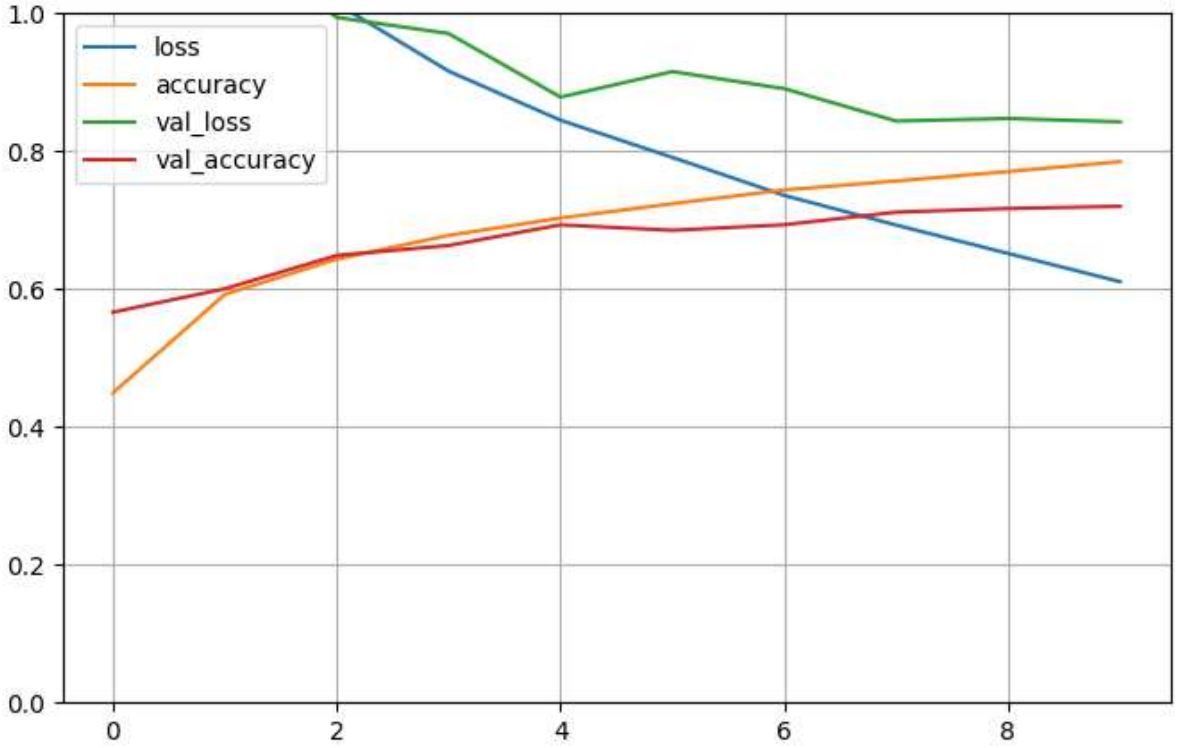
```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_11 (Conv2D)          (None, 30, 30, 32)        896

 max_pooling2d_7 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_12 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_8 (MaxPooling  (None, 6, 6, 64)         0
 2D)

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_11 (Conv2D)          (None, 30, 30, 32)        896

 max_pooling2d_7 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_12 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_8 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_13 (Conv2D)          (None, 4, 4, 64)          36928

 flatten_6 (Flatten)         (None, 1024)              0

 dense_17 (Dense)            (None, 64)                65600

 dense_18 (Dense)            (None, 10)                650

=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

In [ ]:
```
history = model.fit(x_train, y_train, epochs=10,
                    validation_data=(x_test, y_test))
```

```
Epoch 1/10
1563/1563 [==============================] - 64s 39ms/step - loss: 1.5172 - accura
cy: 0.4492 - val_loss: 1.2107 - val_accuracy: 0.5664
Epoch 2/10
1563/1563 [==============================] - 62s 40ms/step - loss: 1.1524 - accura
cy: 0.5923 - val_loss: 1.1139 - val_accuracy: 0.6004
Epoch 3/10
1563/1563 [==============================] - 52s 33ms/step - loss: 1.0142 - accura
cy: 0.6434 - val_loss: 0.9941 - val_accuracy: 0.6488
Epoch 4/10
1563/1563 [==============================] - 72s 46ms/step - loss: 0.9161 - accura
cy: 0.6779 - val_loss: 0.9707 - val_accuracy: 0.6632
Epoch 5/10
1563/1563 [==============================] - 100s 64ms/step - loss: 0.8449 - accur
acy: 0.7031 - val_loss: 0.8785 - val_accuracy: 0.6932
Epoch 6/10
1563/1563 [==============================] - 106s 68ms/step - loss: 0.7910 - accur
acy: 0.7238 - val_loss: 0.9155 - val_accuracy: 0.6852
Epoch 7/10
1563/1563 [==============================] - 119s 76ms/step - loss: 0.7358 - accur
acy: 0.7439 - val_loss: 0.8906 - val_accuracy: 0.6933
Epoch 8/10
1563/1563 [==============================] - 88s 56ms/step - loss: 0.6928 - accura
cy: 0.7568 - val_loss: 0.8437 - val_accuracy: 0.7116
Epoch 9/10
1563/1563 [==============================] - 79s 51ms/step - loss: 0.6517 - accura
cy: 0.7705 - val_loss: 0.8474 - val_accuracy: 0.7169
Epoch 10/10
1563/1563 [==============================] - 78s 50ms/step - loss: 0.6107 - accura
cy: 0.7848 - val_loss: 0.8426 - val_accuracy: 0.7200
```

In [ ]:
```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```




In [ ]:
```
ev = model.evaluate(x_test, y_test)
```

```
313/313 [==============================] - 6s 18ms/step - loss: 0.8426 - accuracy:
0.7200
```

In [ ]:
```
ev
```

Out[ ]:
```
[0.842645525932312, 0.7200000286102295]
```

In [ ]:
```
keras.backend.clear_session
```

Out[ ]:
```
<function keras.backend.clear_session()>
```

In [ ]:
```
del model
```

## 2. CNN with pooling

In [ ]:
```
model_a = keras.models.Sequential()
model_a.add(keras.layers.Conv2D(filters = 32, kernel_size = (3, 3), strides=1, padd
model_a.add(keras.layers.MaxPooling2D((2, 2)))
model_a.add(keras.layers.Flatten())
model_a.add(keras.layers.Dense(300, activation="relu"))
model_a.add(keras.layers.Dense(100, activation="relu"))
model_a.add(keras.layers.Dense(10, activation="softmax"))
```

In [ ]:
```
model_a.summary()
```

```
Model: "sequential_20"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_112 (Conv2D)         (None, 30, 30, 32)        896
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_112 (Conv2D)         (None, 30, 30, 32)        896

 max_pooling2d_11 (MaxPoolin  (None, 15, 15, 32)       0
 g2D)

 flatten_15 (Flatten)        (None, 7200)              0

 dense_40 (Dense)            (None, 300)               2160300

 dense_41 (Dense)            (None, 100)               30100

 dense_42 (Dense)            (None, 10)                1010

=================================================================
Total params: 2,192,306
Trainable params: 2,192,306
Non-trainable params: 0
_________________________________________________________________
```

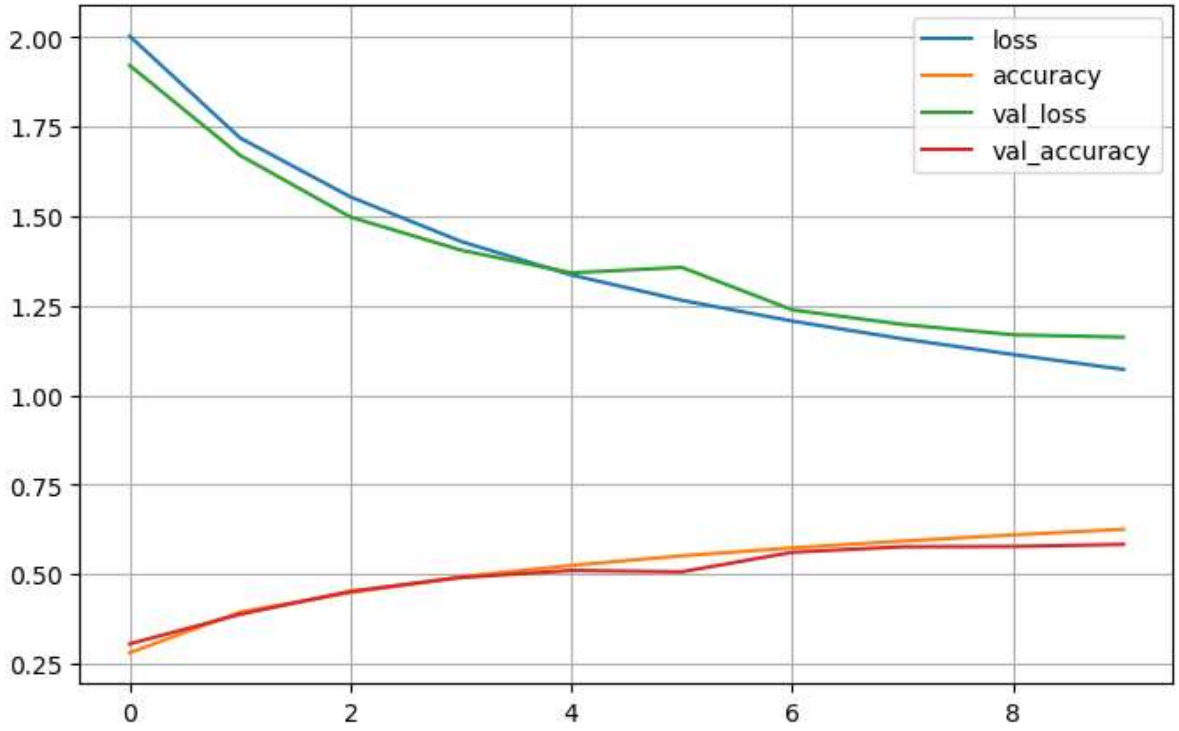
In [ ]:
```
model_a.compile(loss="sparse_categorical_crossentropy",
              optimizer="sgd",
              metrics=["accuracy"])
```

In [ ]:
```
model_history_a = model_a.fit(x_train,y_train, epochs=20,batch_size= 64,
                    validation_data=(x_test, y_test))
```

```
Epoch 1/20
782/782 [==============================] - 42s 48ms/step - loss: 0.8253 - accurac
y: 0.7142 - val_loss: 1.1542 - val_accuracy: 0.6026
Epoch 2/20
782/782 [==============================] - 37s 47ms/step - loss: 0.7874 - accurac
y: 0.7271 - val_loss: 1.1062 - val_accuracy: 0.6202
Epoch 3/20
782/782 [==============================] - 36s 45ms/step - loss: 0.7520 - accurac
y: 0.7380 - val_loss: 1.1140 - val_accuracy: 0.6140
Epoch 4/20
782/782 [==============================] - 32s 41ms/step - loss: 0.7165 - accurac
y: 0.7542 - val_loss: 1.0424 - val_accuracy: 0.6421
Epoch 5/20
782/782 [==============================] - 28s 36ms/step - loss: 0.6830 - accurac
y: 0.7620 - val_loss: 1.2674 - val_accuracy: 0.5852
Epoch 6/20
782/782 [==============================] - 30s 38ms/step - loss: 0.6450 - accurac
y: 0.7790 - val_loss: 1.2177 - val_accuracy: 0.6073
Epoch 7/20
782/782 [==============================] - 39s 49ms/step - loss: 0.6115 - accurac
y: 0.7902 - val_loss: 1.0817 - val_accuracy: 0.6402
Epoch 8/20
782/782 [==============================] - 35s 45ms/step - loss: 0.5781 - accurac
y: 0.8036 - val_loss: 1.0546 - val_accuracy: 0.6547
Epoch 9/20
782/782 [==============================] - 37s 48ms/step - loss: 0.5417 - accurac
y: 0.8149 - val_loss: 1.1382 - val_accuracy: 0.6339
Epoch 10/20
782/782 [==============================] - 43s 55ms/step - loss: 0.5070 - accurac
y: 0.8286 - val_loss: 1.2070 - val_accuracy: 0.6118
Epoch 11/20
782/782 [==============================] - 36s 46ms/step - loss: 0.4713 - accurac
y: 0.8427 - val_loss: 1.2342 - val_accuracy: 0.6220
Epoch 12/20
782/782 [==============================] - 34s 43ms/step - loss: 0.4375 - accurac
y: 0.8546 - val_loss: 1.3746 - val_accuracy: 0.5960
Epoch 13/20
782/782 [==============================] - 36s 46ms/step - loss: 0.4031 - accurac
y: 0.8684 - val_loss: 1.1688 - val_accuracy: 0.6453
Epoch 14/20
782/782 [==============================] - 33s 42ms/step - loss: 0.3661 - accurac
y: 0.8825 - val_loss: 1.4833 - val_accuracy: 0.5961
Epoch 15/20
782/782 [==============================] - 33s 42ms/step - loss: 0.3368 - accurac
y: 0.8938 - val_loss: 1.1873 - val_accuracy: 0.6524
Epoch 16/20
782/782 [==============================] - 31s 40ms/step - loss: 0.3049 - accurac
y: 0.9044 - val_loss: 2.8354 - val_accuracy: 0.4862
Epoch 17/20
782/782 [==============================] - 28s 36ms/step - loss: 0.2794 - accurac
y: 0.9147 - val_loss: 1.4298 - val_accuracy: 0.6235
Epoch 18/20
782/782 [==============================] - 32s 41ms/step - loss: 0.2463 - accurac
y: 0.9267 - val_loss: 1.2798 - val_accuracy: 0.6474
Epoch 19/20
782/782 [==============================] - 35s 45ms/step - loss: 0.2186 - accurac
```

```
y: 0.9370 - val_loss: 1.4434 - val_accuracy: 0.6375
Epoch 20/20
782/782 [==============================] - 34s 43ms/step - loss: 0.1933 - accurac
y: 0.9468 - val_loss: 1.4510 - val_accuracy: 0.6312
```

In [ ]:
```
pd.DataFrame(model_history_a.history).plot(figsize=(8, 5))
plt.grid(True)
plt.show()
```




## 3. VGG16

In [ ]:
```
conv_base = VGG16(weights='imagenet',
                  include_top=False,
                  input_shape=(32, 32, 3))
```

In [ ]:
```
conv_base.summary()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_5 (InputLayer)        [(None, 32, 32, 3)]       0

 block1_conv1 (Conv2D)       (None, 32, 32, 64)        1792

 block1_conv2 (Conv2D)       (None, 32, 32, 64)        36928

 block1_pool (MaxPooling2D)  (None, 16, 16, 64)        0

 block2_conv1 (Conv2D)       (None, 16, 16, 128)       73856

 block2_conv2 (Conv2D)       (None, 16, 16, 128)       147584

 block2_pool (MaxPooling2D)  (None, 8, 8, 128)         0

 block3_conv1 (Conv2D)       (None, 8, 8, 256)         295168

 block3_conv2 (Conv2D)       (None, 8, 8, 256)         590080

 block3_conv3 (Conv2D)       (None, 8, 8, 256)         590080

 block3_pool (MaxPooling2D)  (None, 4, 4, 256)         0

 block4_conv1 (Conv2D)       (None, 4, 4, 512)         1180160

 block4_conv2 (Conv2D)       (None, 4, 4, 512)         2359808

 block4_conv3 (Conv2D)       (None, 4, 4, 512)         2359808

 block4_pool (MaxPooling2D)  (None, 2, 2, 512)         0

 block5_conv1 (Conv2D)       (None, 2, 2, 512)         2359808

 block5_conv2 (Conv2D)       (None, 2, 2, 512)         2359808

 block5_conv3 (Conv2D)       (None, 2, 2, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 1, 1, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(10, activation='sigmoid'))
```

In [ ]:
```
model.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 1, 1, 512)         14714688

 flatten_12 (Flatten)        (None, 512)               0

 dense_27 (Dense)            (None, 256)               131328
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 1, 1, 512)         14714688

 flatten_12 (Flatten)        (None, 512)               0

 dense_27 (Dense)            (None, 256)               131328

 dense_28 (Dense)            (None, 10)                2570

=================================================================
Total params: 14,848,586
Trainable params: 133,898
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [ ]:
```
conv_base.trainable = False
```

In [ ]:
```
model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=2e-5),
              metrics=['acc'])
```

```
WARNING:absl:`lr` is deprecated, please use `learning_rate` instead, or use the le
gacy optimizer, e.g.,tf.keras.optimizers.legacy.RMSprop.
```

In [ ]:
```
checkpoint_cb = keras.callbacks.ModelCheckpoint("CNN_Project_Model-{epoch:02d}.h5")
```

In [ ]:
```
history = model.fit(x_train,y_train, epochs=50,batch_size= 64,
                    validation_data=(x_test, y_test),
                validation_steps=50,
                callbacks=[checkpoint_cb])
```

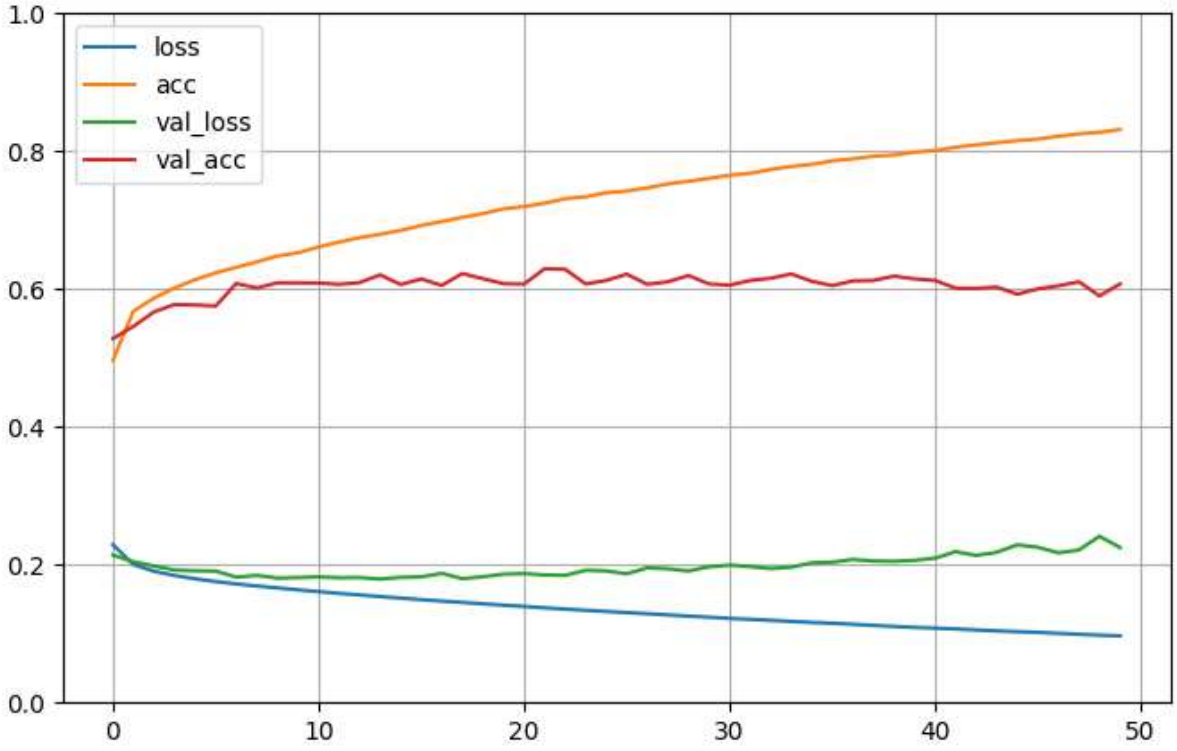
```
Epoch 1/50
782/782 [==============================] - 154s 192ms/step - loss: 0.2292 - acc:
0.4963 - val_loss: 0.2141 - val_acc: 0.5281
Epoch 2/50
782/782 [==============================] - 171s 219ms/step - loss: 0.2003 - acc:
0.5681 - val_loss: 0.2043 - val_acc: 0.5459
Epoch 3/50
782/782 [==============================] - 193s 246ms/step - loss: 0.1906 - acc:
0.5869 - val_loss: 0.1982 - val_acc: 0.5669
Epoch 4/50
782/782 [==============================] - 196s 250ms/step - loss: 0.1844 - acc:
0.6017 - val_loss: 0.1923 - val_acc: 0.5775
Epoch 5/50
782/782 [==============================] - 170s 218ms/step - loss: 0.1797 - acc:
0.6136 - val_loss: 0.1913 - val_acc: 0.5769
Epoch 6/50
782/782 [==============================] - 205s 262ms/step - loss: 0.1757 - acc:
0.6236 - val_loss: 0.1908 - val_acc: 0.5753
Epoch 7/50
782/782 [==============================] - 110s 141ms/step - loss: 0.1722 - acc:
0.6315 - val_loss: 0.1820 - val_acc: 0.6081
Epoch 8/50
782/782 [==============================] - 112s 143ms/step - loss: 0.1691 - acc:
0.6393 - val_loss: 0.1846 - val_acc: 0.6016
Epoch 9/50
782/782 [==============================] - 111s 142ms/step - loss: 0.1664 - acc:
0.6481 - val_loss: 0.1807 - val_acc: 0.6091
Epoch 10/50
782/782 [==============================] - 112s 143ms/step - loss: 0.1636 - acc:
0.6526 - val_loss: 0.1814 - val_acc: 0.6091
Epoch 11/50
782/782 [==============================] - 113s 144ms/step - loss: 0.1610 - acc:
0.6610 - val_loss: 0.1823 - val_acc: 0.6087
Epoch 12/50
782/782 [==============================] - 111s 143ms/step - loss: 0.1585 - acc:
0.6681 - val_loss: 0.1811 - val_acc: 0.6069
Epoch 13/50
782/782 [==============================] - 111s 142ms/step - loss: 0.1562 - acc:
0.6746 - val_loss: 0.1815 - val_acc: 0.6094
Epoch 14/50
782/782 [==============================] - 111s 142ms/step - loss: 0.1539 - acc:
0.6795 - val_loss: 0.1792 - val_acc: 0.6206
Epoch 15/50
782/782 [==============================] - 112s 143ms/step - loss: 0.1516 - acc:
0.6851 - val_loss: 0.1818 - val_acc: 0.6066
Epoch 16/50
782/782 [==============================] - 111s 142ms/step - loss: 0.1494 - acc:
0.6922 - val_loss: 0.1824 - val_acc: 0.6147
Epoch 17/50
782/782 [==============================] - 111s 142ms/step - loss: 0.1473 - acc:
0.6980 - val_loss: 0.1875 - val_acc: 0.6053
Epoch 18/50
782/782 [==============================] - 111s 142ms/step - loss: 0.1453 - acc:
0.7040 - val_loss: 0.1798 - val_acc: 0.6225
Epoch 19/50
782/782 [==============================] - 111s 142ms/step - loss: 0.1434 - acc:
```

```
0.7095 - val_loss: 0.1826 - val_acc: 0.6150
Epoch 20/50
782/782 [==============================] - 111s 142ms/step - loss: 0.1413 - acc:
0.7163 - val_loss: 0.1863 - val_acc: 0.6078
Epoch 21/50
782/782 [==============================] - 112s 143ms/step - loss: 0.1396 - acc:
0.7197 - val_loss: 0.1872 - val_acc: 0.6072
Epoch 22/50
782/782 [==============================] - 112s 143ms/step - loss: 0.1375 - acc:
0.7245 - val_loss: 0.1852 - val_acc: 0.6294
Epoch 23/50
782/782 [==============================] - 112s 143ms/step - loss: 0.1357 - acc:
0.7311 - val_loss: 0.1846 - val_acc: 0.6288
Epoch 24/50
782/782 [==============================] - 112s 143ms/step - loss: 0.1339 - acc:
0.7339 - val_loss: 0.1920 - val_acc: 0.6075
Epoch 25/50
782/782 [==============================] - 112s 143ms/step - loss: 0.1323 - acc:
0.7398 - val_loss: 0.1910 - val_acc: 0.6125
Epoch 26/50
782/782 [==============================] - 111s 142ms/step - loss: 0.1306 - acc:
0.7421 - val_loss: 0.1870 - val_acc: 0.6216
Epoch 27/50
782/782 [==============================] - 116s 148ms/step - loss: 0.1290 - acc:
0.7468 - val_loss: 0.1955 - val_acc: 0.6069
Epoch 28/50
782/782 [==============================] - 119s 152ms/step - loss: 0.1272 - acc:
0.7525 - val_loss: 0.1943 - val_acc: 0.6106
Epoch 29/50
782/782 [==============================] - 119s 152ms/step - loss: 0.1255 - acc:
0.7563 - val_loss: 0.1910 - val_acc: 0.6197
Epoch 30/50
782/782 [==============================] - 119s 152ms/step - loss: 0.1239 - acc:
0.7606 - val_loss: 0.1970 - val_acc: 0.6075
Epoch 31/50
782/782 [==============================] - 119s 152ms/step - loss: 0.1222 - acc:
0.7651 - val_loss: 0.1995 - val_acc: 0.6056
Epoch 32/50
782/782 [==============================] - 119s 152ms/step - loss: 0.1209 - acc:
0.7679 - val_loss: 0.1973 - val_acc: 0.6125
Epoch 33/50
782/782 [==============================] - 119s 152ms/step - loss: 0.1194 - acc:
0.7735 - val_loss: 0.1945 - val_acc: 0.6156
Epoch 34/50
782/782 [==============================] - 131s 168ms/step - loss: 0.1178 - acc:
0.7779 - val_loss: 0.1965 - val_acc: 0.6219
Epoch 35/50
782/782 [==============================] - 126s 161ms/step - loss: 0.1164 - acc:
0.7808 - val_loss: 0.2027 - val_acc: 0.6112
Epoch 36/50
782/782 [==============================] - 152s 195ms/step - loss: 0.1151 - acc:
0.7861 - val_loss: 0.2037 - val_acc: 0.6050
Epoch 37/50
782/782 [==============================] - 135s 173ms/step - loss: 0.1136 - acc:
0.7889 - val_loss: 0.2077 - val_acc: 0.6119
Epoch 38/50
```

```
782/782 [==============================] - 125s 160ms/step - loss: 0.1121 - acc:
0.7927 - val_loss: 0.2055 - val_acc: 0.6125
Epoch 39/50
782/782 [==============================] - 137s 176ms/step - loss: 0.1105 - acc:
0.7944 - val_loss: 0.2051 - val_acc: 0.6187
Epoch 40/50
782/782 [==============================] - 127s 162ms/step - loss: 0.1091 - acc:
0.7987 - val_loss: 0.2064 - val_acc: 0.6147
Epoch 41/50
782/782 [==============================] - 129s 166ms/step - loss: 0.1081 - acc:
0.8010 - val_loss: 0.2096 - val_acc: 0.6125
Epoch 42/50
782/782 [==============================] - 143s 183ms/step - loss: 0.1069 - acc:
0.8059 - val_loss: 0.2190 - val_acc: 0.6012
Epoch 43/50
782/782 [==============================] - 154s 197ms/step - loss: 0.1054 - acc:
0.8093 - val_loss: 0.2136 - val_acc: 0.6009
Epoch 44/50
782/782 [==============================] - 156s 200ms/step - loss: 0.1040 - acc:
0.8122 - val_loss: 0.2180 - val_acc: 0.6028
Epoch 45/50
782/782 [==============================] - 152s 194ms/step - loss: 0.1029 - acc:
0.8154 - val_loss: 0.2290 - val_acc: 0.5925
Epoch 46/50
782/782 [==============================] - 135s 173ms/step - loss: 0.1019 - acc:
0.8176 - val_loss: 0.2254 - val_acc: 0.6003
Epoch 47/50
782/782 [==============================] - 131s 167ms/step - loss: 0.1005 - acc:
0.8217 - val_loss: 0.2173 - val_acc: 0.6047
Epoch 48/50
782/782 [==============================] - 141s 180ms/step - loss: 0.0992 - acc:
0.8251 - val_loss: 0.2215 - val_acc: 0.6106
Epoch 49/50
782/782 [==============================] - 136s 174ms/step - loss: 0.0979 - acc:
0.8275 - val_loss: 0.2413 - val_acc: 0.5900
Epoch 50/50
782/782 [==============================] - 138s 177ms/step - loss: 0.0969 - acc:
0.8314 - val_loss: 0.2250 - val_acc: 0.6075
```

In [ ]:

```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```




In [ ]:
```
hist_df = pd.DataFrame(history.history)
```

In [ ]:
```
results = model.evaluate(x_test, y_test, batch_size=128)
```

```
79/79 [==============================] - 21s 265ms/step - loss: 0.2277 - acc: 0.61
17
```

## 4. Xception

In [ ]:
```
base_model = Xception(include_top=False, weights='imagenet', input_shape=(224,224,3
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-application
s/xception/xception_weights_tf_dim_ordering_tf_kernels_notop.h5
83683744/83683744 [==============================] - 74s 1us/step
```

In [ ]:
```
base_model.trainable = False
```

In [ ]:
```
from tensorflow.keras.layers import Input,Flatten,Dense,BatchNormalization,Activati
data_augmentation = Sequential(
    [RandomFlip("horizontal"),
     RandomRotation(0.1),
     RandomZoom(0.1)]
)
```

In [ ]:
```
inputs = tf.keras.Input(shape=(32, 32, 3))
x = tf.keras.layers.Lambda(lambda image: tf.image.resize(image, (224,224)))(inputs)
x = tf.keras.applications.xception.preprocess_input(x)
x = base_model(x, training=False)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dropout(0.3)(x)
```

```
outputs = tf.keras.layers.Dense(10, activation=('softmax'))(x)
model = tf.keras.Model(inputs, outputs)
```

In [ ]:
```
model.summary()
```

```
Model: "model_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_17 (InputLayer)       [(None, 32, 32, 3)]       0

 lambda_1 (Lambda)           (None, 224, 224, 3)       0

 tf.math.truediv_1 (TFOpLamb  (None, 224, 224, 3)      0
 da)

 tf.math.subtract_1 (TFOpLam  (None, 224, 224, 3)      0
 bda)

 xception (Functional)       (None, 7, 7, 2048)        20861480

 global_average_pooling2d_1   (None, 2048)             0
 (GlobalAveragePooling2D)

 dropout_1 (Dropout)         (None, 2048)              0

 dense_36 (Dense)            (None, 10)                20490

=================================================================
Total params: 20,881,970
Trainable params: 20,490
Non-trainable params: 20,861,480
_________________________________________________________________
```

In [ ]:
```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy
```

In [ ]:
```
epochs = 10
history = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=epoc
```

```
Epoch 1/10
1563/1563 [==============================] - 3975s 3s/step - loss: 2.2943 - accura
cy: 0.1226 - val_loss: 2.2764 - val_accuracy: 0.1372
Epoch 2/10
1563/1563 [==============================] - 3193s 2s/step - loss: 2.2697 - accura
cy: 0.1534 - val_loss: 2.2558 - val_accuracy: 0.2203
Epoch 3/10
 717/1563 [============>.................] - ETA: 9:23:26 - loss: 2.2555 - accurac
y: 0.1712
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[151], line 2
      1 epochs = 10
----> 2 history = model.fit(x_train, y_train, validation_data=(x_test, y_test), ep
ochs=epochs, verbose=1)

File e:\programes\python\lib\site-packages\keras\utils\traceback_utils.py:65, in f
ilter_traceback.<locals>.error_handler(*args, **kwargs)
     63 filtered_tb = None
     64 try:
---> 65     return fn(*args, **kwargs)
     66 except Exception as e:
     67     filtered_tb = _process_traceback_frames(e.__traceback__)

File e:\programes\python\lib\site-packages\keras\engine\training.py:1650, in Mode
l.fit(self, x, y, batch_size, epochs, verbose, callbacks, validation_split, valida
tion_data, shuffle, class_weight, sample_weight, initial_epoch, steps_per_epoch, v
alidation_steps, validation_batch_size, validation_freq, max_queue_size, workers,
use_multiprocessing)
   1642 with tf.profiler.experimental.Trace(
   1643     "train",
   1644     epoch_num=epoch,
   (...)
   1647     _r=1,
   1648 ):
   1649     callbacks.on_train_batch_begin(step)
-> 1650     tmp_logs = self.train_function(iterator)
   1651     if data_handler.should_sync:
   1652         context.async_wait()

File e:\programes\python\lib\site-packages\tensorflow\python\util\traceback_utils.
py:150, in filter_traceback.<locals>.error_handler(*args, **kwargs)
    148 filtered_tb = None
    149 try:
--> 150   return fn(*args, **kwargs)
    151 except Exception as e:
    152   filtered_tb = _process_traceback_frames(e.__traceback__)

File e:\programes\python\lib\site-packages\tensorflow\python\eager\polymorphic_fun
ction\polymorphic_function.py:880, in Function.__call__(self, *args, **kwds)
    877 compiler = "xla" if self._jit_compile else "nonXla"
    879 with OptionalXlaContext(self._jit_compile):
--> 880   result = self._call(*args, **kwds)
    882 new_tracing_count = self.experimental_get_tracing_count()
    883 without_tracing = (tracing_count == new_tracing_count)

File e:\programes\python\lib\site-packages\tensorflow\python\eager\polymorphic_fun
ction\polymorphic_function.py:912, in Function._call(self, *args, **kwds)
    909   self._lock.release()
    910   # In this case we have created variables on the first call, so we run th
e
    911   # defunned version which is guaranteed to never create variables.
--> 912   return self._no_variable_creation_fn(*args, **kwds)  # pylint: disable=n
ot-callable
    913 elif self._variable_creation_fn is not None:
```

```
    914   # Release the lock early so that multiple threads can perform the call
    915   # in parallel.
    916   self._lock.release()

File e:\programes\python\lib\site-packages\tensorflow\python\eager\polymorphic_fun
ction\tracing_compiler.py:134, in TracingCompiler.__call__(self, *args, **kwargs)
    131 with self._lock:
    132   (concrete_function,
    133    filtered_flat_args) = self._maybe_define_function(args, kwargs)
--> 134 return concrete_function._call_flat(
    135     filtered_flat_args, captured_inputs=concrete_function.captured_inputs)

File e:\programes\python\lib\site-packages\tensorflow\python\eager\polymorphic_fun
ction\monomorphic_function.py:1745, in ConcreteFunction._call_flat(self, args, cap
tured_inputs, cancellation_manager)
   1741 possible_gradient_type = gradients_util.PossibleTapeGradientTypes(args)
   1742 if (possible_gradient_type == gradients_util.POSSIBLE_GRADIENT_TYPES_NONE
   1743     and executing_eagerly):
   1744   # No tape is watching; skip to running the function.
-> 1745   return self._build_call_outputs(self._inference_function.call(
   1746       ctx, args, cancellation_manager=cancellation_manager))
   1747 forward_backward = self._select_forward_and_backward_functions(
   1748     args,
   1749     possible_gradient_type,
   1750     executing_eagerly)
   1751 forward_function, args_with_tangents = forward_backward.forward()

File e:\programes\python\lib\site-packages\tensorflow\python\eager\polymorphic_fun
ction\monomorphic_function.py:378, in _EagerDefinedFunction.call(self, ctx, args,
cancellation_manager)
    376 with _InterpolateFunctionError(self):
    377   if cancellation_manager is None:
--> 378     outputs = execute.execute(
    379         str(self.signature.name),
    380         num_outputs=self._num_outputs,
    381         inputs=args,
    382         attrs=attrs,
    383         ctx=ctx)
    384   else:
    385     outputs = execute.execute_with_cancellation(
    386         str(self.signature.name),
    387         num_outputs=self._num_outputs,
   (...)
    390         ctx=ctx,
    391         cancellation_manager=cancellation_manager)

File e:\programes\python\lib\site-packages\tensorflow\python\eager\execute.py:52,
in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     50 try:
     51   ctx.ensure_initialized()
---> 52   tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
     53                                       inputs, attrs, num_outputs)
     54 except core._NotOkStatusException as e:
     55   if name is not None:

KeyboardInterrupt:
```

### !! Due to running near 10 hours and not yet reaching a promissing accuracy level the model have to be stopped mid process

In [ ]:
```
results = model.evaluate(x_test, y_test, batch_size=128)
```

```
79/79 [==============================] - 618s 8s/step - loss: 2.2427 - accuracy:
0.2385
```

In [ ]:
```
results
```

Out[ ]:
```
[2.2427217960357666, 0.23849999904632568]
```

## Conclusion

Here the Normal Custom CNN that is implemented without the pooling have the best accuracy of 0.72 and CNN with pooling and VGG16 worked fine with the dataset but the Xception model is too slow, takes very long time and its accuracy is not improving.

## Insights

Here the Dataset works well with CNN models but Xception model is not recommended here. Here the CNN had not much of flaw VGG16 already hited saturation so not further training wont improve the situation.

## Sugestion

Dataset could be expanded to hold more labels and datas. Here VGG16 and Xception models doesn't need further training others could improve. Other complex models can be implemented here.