Name - Atharva Gondkar

Class - IS2

Roll Number - 2176032

Enrollment Number - MITU17BTMA0013

EXPERIMENT #3

DATE: 11-09-2020

TITLE : ML II ASSIGNMENT 3

AIM

Perform Image Classification using CatsVsDogs dataset.

OBJECTIVE

1. Implement CNN for Image Classification using CatsVsDogs dataset.

DRIVE LINK - *https://drive.google.com/drive/u/0/folders/11s0LnDfN-u8BMit5F-ZcLMOLsEQNV22L*

**Notebook, code, pdf, output snapshots have been stored on the above given drive link.*

```
!pip3 install patool
!pip3 install pyunpack
```

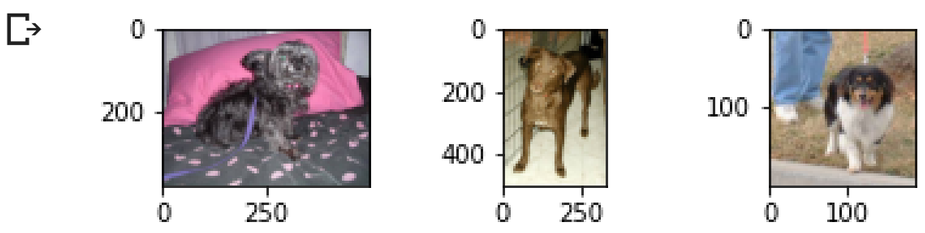
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
from pyunpack import Archive
Archive("/content/gdrive/My Drive/ML_LAB_ASSIGNMENT_2020/ASSIGNMENT_3/CatvsDogs.rar").extractall(".")
```

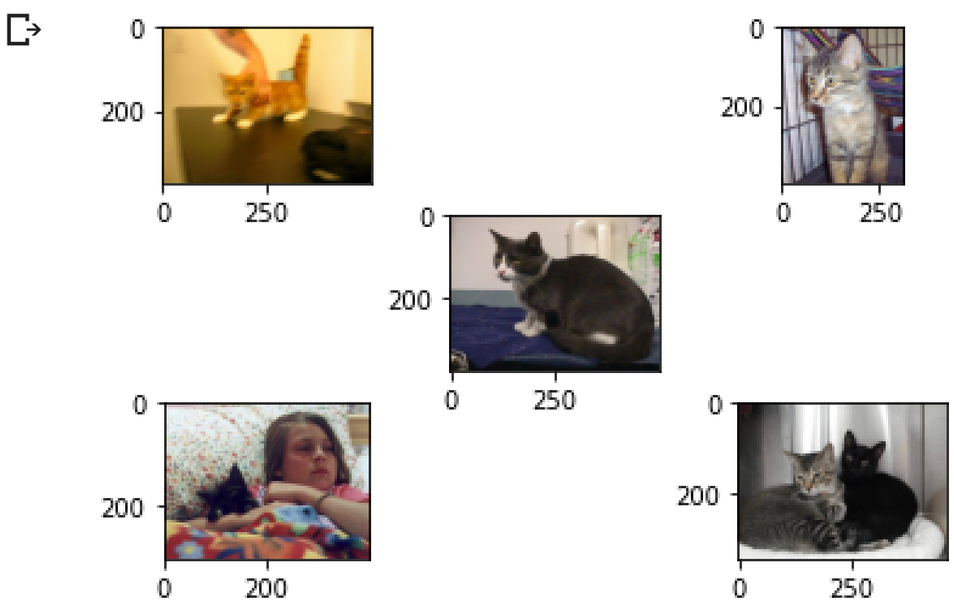
```
!mkdir dataset
!mkdir ./dataset/cat
!mkdir ./dataset/dog
!mkdir validation
!mkdir ./validation/cat
!mkdir ./validation/dog

from os import makedirs
from os import listdir
from shutil import copyfile
from random import seed
from random import random

train_src_directory = './train'
cnt_cat = 0
cnt_dog = 0
for file in listdir(train_src_directory):
  src = train_src_directory + '/' + file
  if file.startswith('cat.'):
    if cnt_cat < 10000:
      dst = 'dataset/cat/'+ file
    else:
      dst = 'validation/cat/'+ file
    copyfile(src, dst)
    cnt_cat+=1

  elif file.startswith('dog.'):
    if cnt_dog < 10000:
      dst = 'dataset/dog/'+ file
    else:
      dst = 'validation/dog/'+ file
    copyfile(src, dst)
    cnt_dog+=1
```

```
from matplotlib import pyplot as plt
from matplotlib.image import imread
```

```
folder = './dataset/dog/'
for i in range(3):
  plt.subplot(330 + 1 + i)
  filename = folder + 'dog.' + str(i) + '.jpg'
  image = imread(filename)
  plt.imshow(image)
plt.show()
```




```
folder = './dataset/cat/'
for i in range(0,10,2):
  plt.subplot(330 + 1 + i)
  filename = folder + 'cat.' + str(i) + '.jpg'
  image = imread(filename)
  plt.imshow(image)
plt.show()
```




```
from matplotlib import pyplot
import numpy as np
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.optimizers import SGD
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
```

IMAGE AUGMENTATON

```
train_datagen = ImageDataGenerator(rescale = 1.0/255.0,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True,
                                   )

training_set = train_datagen.flow_from_directory('./dataset',
                                                 target_size = (48, 48),
                                                 batch_size = 64,
                                                 class_mode = 'binary')
```

```
validation_datagen = ImageDataGenerator(rescale = 1.0/255.0,)
validation_set = validation_datagen.flow_from_directory('./validation',
                                                        target_size = (48, 48),
                                                        batch_size = 64,
                                                        class_mode = 'binary')
```

```
Found 20000 images belonging to 2 classes.
Found 5000 images belonging to 2 classes.
```

MODEL 1 - OPTIMIZER : NADAM

```
model = Sequential()
model.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(48, 48, 3)))
model.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2), strides=(2,2)))

model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2), strides=(2,2)))

model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2), strides=(2,2)))

model.add(Flatten())

model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))

model.add(Dense(1, activation='sigmoid'))

opt = tf.keras.optimizers.Nadam(
   learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07)
model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
```

```
model.summary()
```

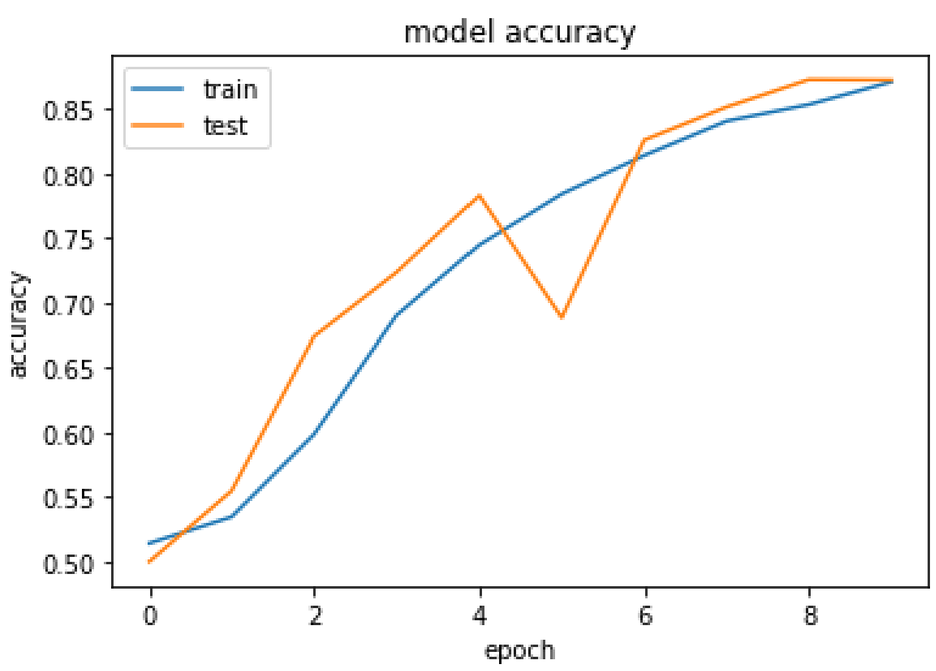
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 48, 48, 256)       7168
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 48, 48, 256)       590080
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 24, 24, 256)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 128)       295040
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 128)       147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 64)        73792
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0
_________________________________________________________________
dense (Dense)                (None, 128)               295040
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 1,519,617
Trainable params: 1,519,617
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint(filepath = 'best_model_NADAM.h5',save_best_only = True,verbose=1)
```

```
history_nadam = model.fit(training_set,
                  epochs=10,
                  callbacks=[checkpoint],
                  validation_data=validation_set)
```

```
Epoch 1/10
313/313 [==============================] - ETA: 0s - loss: 0.7196 - accuracy: 0.5142
Epoch 00001: val_loss improved from inf to 0.69263, saving model to best_model_NADAM.h5
313/313 [==============================] - 84s 269ms/step - loss: 0.7196 - accuracy: 0.5142 - val_loss: 0.6926 - val_accuracy: 0.
Epoch 2/10
313/313 [==============================] - ETA: 0s - loss: 0.6893 - accuracy: 0.5347
Epoch 00002: val_loss improved from 0.69263 to 0.68071, saving model to best_model_NADAM.h5
313/313 [==============================] - 82s 261ms/step - loss: 0.6893 - accuracy: 0.5347 - val_loss: 0.6807 - val_accuracy: 0.
Epoch 3/10
313/313 [==============================] - ETA: 0s - loss: 0.6702 - accuracy: 0.5985
Epoch 00003: val_loss improved from 0.68071 to 0.60553, saving model to best_model_NADAM.h5
313/313 [==============================] - 80s 257ms/step - loss: 0.6702 - accuracy: 0.5985 - val_loss: 0.6055 - val_accuracy: 0.
Epoch 4/10
313/313 [==============================] - ETA: 0s - loss: 0.5851 - accuracy: 0.6899
Epoch 00004: val_loss improved from 0.60553 to 0.54122, saving model to best_model_NADAM.h5
313/313 [==============================] - 80s 254ms/step - loss: 0.5851 - accuracy: 0.6899 - val_loss: 0.5412 - val_accuracy: 0.
Epoch 5/10
313/313 [==============================] - ETA: 0s - loss: 0.5181 - accuracy: 0.7444
Epoch 00005: val_loss improved from 0.54122 to 0.46390, saving model to best_model_NADAM.h5
313/313 [==============================] - 80s 255ms/step - loss: 0.5181 - accuracy: 0.7444 - val_loss: 0.4639 - val_accuracy: 0.
Epoch 6/10
313/313 [==============================] - ETA: 0s - loss: 0.4558 - accuracy: 0.7839
Epoch 00006: val_loss did not improve from 0.46390
313/313 [==============================] - 79s 252ms/step - loss: 0.4558 - accuracy: 0.7839 - val_loss: 0.6407 - val_accuracy: 0.
Epoch 7/10
313/313 [==============================] - ETA: 0s - loss: 0.4069 - accuracy: 0.8137
Epoch 00007: val_loss improved from 0.46390 to 0.38349, saving model to best_model_NADAM.h5
313/313 [==============================] - 79s 251ms/step - loss: 0.4069 - accuracy: 0.8137 - val_loss: 0.3835 - val_accuracy: 0.
Epoch 8/10
313/313 [==============================] - ETA: 0s - loss: 0.3555 - accuracy: 0.8403
Epoch 00008: val_loss improved from 0.38349 to 0.34450, saving model to best_model_NADAM.h5
313/313 [==============================] - 78s 250ms/step - loss: 0.3555 - accuracy: 0.8403 - val_loss: 0.3445 - val_accuracy: 0.
Epoch 9/10
313/313 [==============================] - ETA: 0s - loss: 0.3301 - accuracy: 0.8531
Epoch 00009: val_loss improved from 0.34450 to 0.29266, saving model to best_model_NADAM.h5
313/313 [==============================] - 78s 251ms/step - loss: 0.3301 - accuracy: 0.8531 - val_loss: 0.2927 - val_accuracy: 0.
Epoch 10/10
313/313 [==============================] - ETA: 0s - loss: 0.2985 - accuracy: 0.8706
Epoch 00010: val_loss did not improve from 0.29266
313/313 [==============================] - 78s 250ms/step - loss: 0.2985 - accuracy: 0.8706 - val_loss: 0.2938 - val_accuracy: 0.
```

```
plt.plot(history_nadam.history['accuracy'])
plt.plot(history_nadam.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




```
model.save_weights('Atharva_Nadam_weights_48.h5')
np.save('Atharva_Nadam_48.npy',history_nadam.history)
```

MODEL 2 - OPTIMIZER : SGD

```
model_sgd = Sequential()
model_sgd.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(48, 48, 3)))
model_sgd.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_sgd.add(MaxPooling2D((2, 2), strides=(2,2)))
```

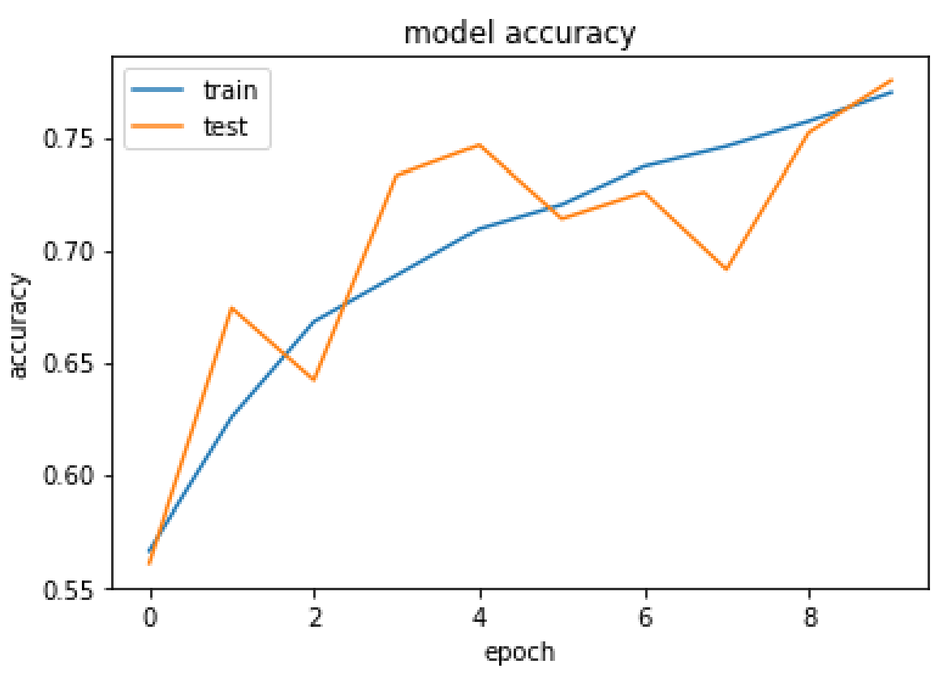
```
model_sgd.add(MaxPooling2D((2, 2), strides=(2,2)))

model_sgd.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_sgd.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_sgd.add(MaxPooling2D((2, 2), strides=(2,2)))

model_sgd.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_sgd.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_sgd.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_sgd.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_sgd.add(MaxPooling2D((2, 2), strides=(2,2)))

model_sgd.add(Flatten())

model_sgd.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))

model_sgd.add(Dense(1, activation='sigmoid'))

opt = tf.keras.optimizers.SGD(
    learning_rate=0.01, momentum=0.0, nesterov=False)
model_sgd.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])

model_sgd.summary()
```

```
Model: "sequential_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_104 (Conv2D)          (None, 48, 48, 256)       7168
_________________________________________________________________
conv2d_105 (Conv2D)          (None, 48, 48, 256)       590080
_________________________________________________________________
max_pooling2d_39 (MaxPooling (None, 24, 24, 256)       0
_________________________________________________________________
conv2d_106 (Conv2D)          (None, 24, 24, 128)       295040
_________________________________________________________________
conv2d_107 (Conv2D)          (None, 24, 24, 128)       147584
_________________________________________________________________
max_pooling2d_40 (MaxPooling (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_108 (Conv2D)          (None, 12, 12, 64)        73792
_________________________________________________________________
conv2d_109 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_110 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_111 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_41 (MaxPooling (None, 6, 6, 64)          0
_________________________________________________________________
flatten_13 (Flatten)         (None, 2304)              0
_________________________________________________________________
dense_26 (Dense)             (None, 128)               295040
_________________________________________________________________
dense_27 (Dense)             (None, 1)                 129
=================================================================
Total params: 1,519,617
Trainable params: 1,519,617
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint(filepath = 'best_model_SGD.h5',save_best_only = True,verbose=1)

history_sgd = model_sgd.fit(training_set,
                    epochs=10,
                    callbacks=[checkpoint],
                    validation_data=validation_set)
```

```
Epoch 1/10
313/313 [==============================] - ETA: 0s - loss: 0.6796 - accuracy: 0.5662
Epoch 00001: val_loss improved from inf to 0.68032, saving model to best_model_SGD.h5
313/313 [==============================] - 79s 251ms/step - loss: 0.6796 - accuracy: 0.5662 - val_loss: 0.6803 - val_accuracy: 0.
Epoch 2/10
313/313 [==============================] - ETA: 0s - loss: 0.6421 - accuracy: 0.6255
Epoch 00002: val_loss improved from 0.68032 to 0.61192, saving model to best_model_SGD.h5
313/313 [==============================] - 79s 251ms/step - loss: 0.6421 - accuracy: 0.6255 - val_loss: 0.6119 - val_accuracy: 0.
Epoch 3/10
313/313 [==============================] - ETA: 0s - loss: 0.6091 - accuracy: 0.6681
Epoch 00003: val_loss did not improve from 0.61192
313/313 [==============================] - 79s 252ms/step - loss: 0.6091 - accuracy: 0.6681 - val_loss: 0.6181 - val_accuracy: 0.
Epoch 4/10
313/313 [==============================] - ETA: 0s - loss: 0.5835 - accuracy: 0.6888
Epoch 00004: val_loss improved from 0.61192 to 0.53752, saving model to best_model_SGD.h5
313/313 [==============================] - 78s 250ms/step - loss: 0.5835 - accuracy: 0.6888 - val_loss: 0.5375 - val_accuracy: 0.
Epoch 5/10
313/313 [==============================] - ETA: 0s - loss: 0.5612 - accuracy: 0.7095
Epoch 00005: val_loss improved from 0.53752 to 0.51241, saving model to best_model_SGD.h5
313/313 [==============================] - 78s 249ms/step - loss: 0.5612 - accuracy: 0.7095 - val_loss: 0.5124 - val_accuracy: 0.
Epoch 6/10
313/313 [==============================] - ETA: 0s - loss: 0.5478 - accuracy: 0.7203
Epoch 00006: val_loss did not improve from 0.51241
313/313 [==============================] - 78s 249ms/step - loss: 0.5478 - accuracy: 0.7203 - val_loss: 0.5565 - val_accuracy: 0.
Epoch 7/10
313/313 [==============================] - ETA: 0s - loss: 0.5281 - accuracy: 0.7372
```

```
plt.plot(history_sgd.history['accuracy'])
plt.plot(history_sgd.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




```
model_sgd.save_weights('Atharva_SGD_48.h5')
np.save('Atharva_SGD_48.npy',history_sgd.history)
```

MODEL 3 - OPTIMIZER : SGD + NESTROV

```
model_sgdn = Sequential()
model_sgdn.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(48, 48, 3)))
model_sgdn.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_sgdn.add(MaxPooling2D((2, 2), strides=(2,2)))

model_sgdn.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_sgdn.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_sgdn.add(MaxPooling2D((2, 2), strides=(2,2)))

model_sgdn.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_sgdn.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_sgdn.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_sgdn.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_sgdn.add(MaxPooling2D((2, 2), strides=(2,2)))

model_sgdn.add(Flatten())

model_sgdn.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))

model_sgdn.add(Dense(1, activation='sigmoid'))

opt = tf.keras.optimizers.SGD(
```

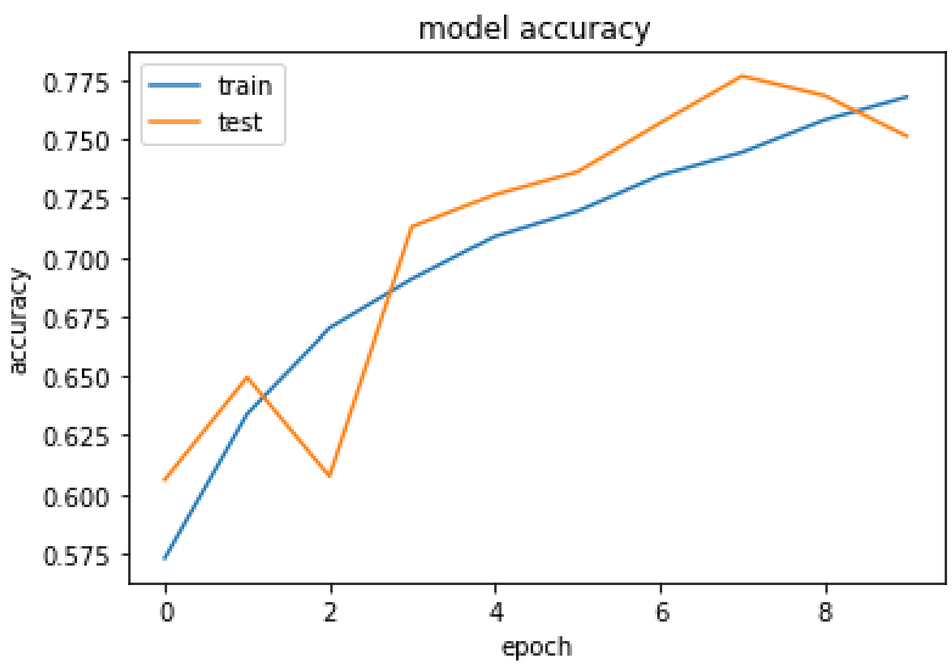
```
    learning_rate=0.01, momentum=0.0, nesterov=True)
model_sgdn.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])

model_sgdn.summary()
```

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_112 (Conv2D)          (None, 48, 48, 256)       7168
_________________________________________________________________
conv2d_113 (Conv2D)          (None, 48, 48, 256)       590080
_________________________________________________________________
max_pooling2d_42 (MaxPooling (None, 24, 24, 256)       0
_________________________________________________________________
conv2d_114 (Conv2D)          (None, 24, 24, 128)       295040
_________________________________________________________________
conv2d_115 (Conv2D)          (None, 24, 24, 128)       147584
_________________________________________________________________
max_pooling2d_43 (MaxPooling (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_116 (Conv2D)          (None, 12, 12, 64)        73792
_________________________________________________________________
conv2d_117 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_118 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_119 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_44 (MaxPooling (None, 6, 6, 64)          0
_________________________________________________________________
flatten_14 (Flatten)         (None, 2304)              0
_________________________________________________________________
dense_28 (Dense)             (None, 128)               295040
_________________________________________________________________
dense_29 (Dense)             (None, 1)                 129
=================================================================
Total params: 1,519,617
Trainable params: 1,519,617
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint(filepath = 'best_model_SGDN.h5',save_best_only = True,verbose=1)

history_sgdn = model_sgdn.fit(training_set,
                    epochs=10,
                    callbacks=[checkpoint],
                    validation_data=validation_set)
```

```
Epoch 1/10
313/313 [==============================] - ETA: 0s - loss: 0.6764 - accuracy: 0.5730
Epoch 00001: val_loss improved from inf to 0.65042, saving model to best_model_SGDN.h5
313/313 [==============================] - 79s 252ms/step - loss: 0.6764 - accuracy: 0.5730 - val_loss: 0.6504 - val_accuracy: 0.
Epoch 2/10
```

```
plt.plot(history_sgdn.history['accuracy'])
plt.plot(history_sgdn.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




```
Epoch 9/10
```

```
model_sgdn.save_weights('Atharva_SGDN_48.h5')
np.save('Atharva_SGDN_48.npy',history_sgdn.history)
```

```
Epoch 10/10
```

MODEL 4 - OPTIMIZER : RMS

```
model_rms = Sequential()
model_rms.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(48, 48, 3)))
model_rms.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_rms.add(MaxPooling2D((2, 2), strides=(2,2)))

model_rms.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_rms.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_rms.add(MaxPooling2D((2, 2), strides=(2,2)))

model_rms.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_rms.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_rms.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_rms.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_rms.add(MaxPooling2D((2, 2), strides=(2,2)))

model_rms.add(Flatten())

model_rms.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))

model_rms.add(Dense(1, activation='sigmoid'))
```

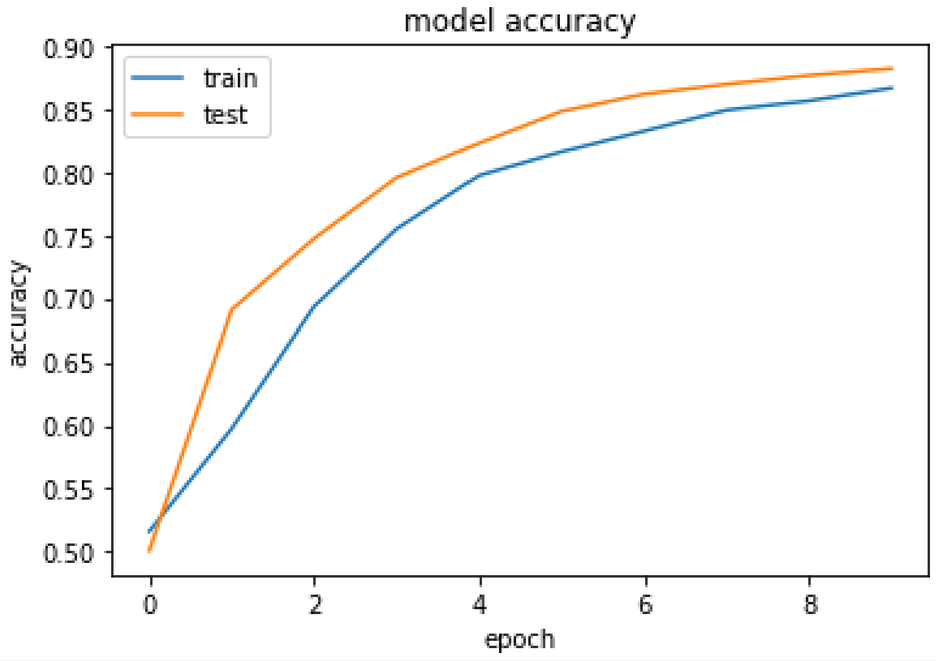
```
opt = tf.keras.optimizers.RMSprop(
    learning_rate=0.001,
    rho=0.9,
    momentum=0.0,
    epsilon=1e-07,
    centered=False,)
model_rms.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])

model_rms.summary()
```

```
Model: "sequential_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_120 (Conv2D)          (None, 48, 48, 256)       7168
_________________________________________________________________
conv2d_121 (Conv2D)          (None, 48, 48, 256)       590080
_________________________________________________________________
max_pooling2d_45 (MaxPooling (None, 24, 24, 256)       0
_________________________________________________________________
conv2d_122 (Conv2D)          (None, 24, 24, 128)       295040
_________________________________________________________________
conv2d_123 (Conv2D)          (None, 24, 24, 128)       147584
_________________________________________________________________
max_pooling2d_46 (MaxPooling (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_124 (Conv2D)          (None, 12, 12, 64)        73792
_________________________________________________________________
conv2d_125 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_126 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_127 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_47 (MaxPooling (None, 6, 6, 64)          0
```

```
from keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint(filepath = 'best_model_RMS.h5',save_best_only = True,verbose=1)

history_rms = model_rms.fit(training_set,
                    epochs=10,
                    callbacks=[checkpoint],
                    validation_data=validation_set)
```

```
Epoch 1/10
313/313 [==============================] - ETA: 0s - loss: 1.1355 - accuracy: 0.5162
Epoch 00001: val_loss improved from inf to 0.69124, saving model to best_model_RMS.h5
313/313 [==============================] - 78s 248ms/step - loss: 1.1355 - accuracy: 0.5162 - val_loss: 0.6912 - val_accuracy: 0.
Epoch 2/10
313/313 [==============================] - ETA: 0s - loss: 0.6784 - accuracy: 0.5974
Epoch 00002: val_loss improved from 0.69124 to 0.59583, saving model to best_model_RMS.h5
313/313 [==============================] - 77s 246ms/step - loss: 0.6784 - accuracy: 0.5974 - val_loss: 0.5958 - val_accuracy: 0.
Epoch 3/10
313/313 [==============================] - ETA: 0s - loss: 0.5916 - accuracy: 0.6934
Epoch 00003: val_loss improved from 0.59583 to 0.51424, saving model to best_model_RMS.h5
313/313 [==============================] - 78s 248ms/step - loss: 0.5916 - accuracy: 0.6934 - val_loss: 0.5142 - val_accuracy: 0.
Epoch 4/10
313/313 [==============================] - ETA: 0s - loss: 0.5089 - accuracy: 0.7548
Epoch 00004: val_loss improved from 0.51424 to 0.47275, saving model to best_model_RMS.h5
313/313 [==============================] - 78s 248ms/step - loss: 0.5089 - accuracy: 0.7548 - val_loss: 0.4728 - val_accuracy: 0.
Epoch 5/10
313/313 [==============================] - ETA: 0s - loss: 0.4463 - accuracy: 0.7972
Epoch 00005: val_loss improved from 0.47275 to 0.37714, saving model to best_model_RMS.h5
313/313 [==============================] - 77s 246ms/step - loss: 0.4463 - accuracy: 0.7972 - val_loss: 0.3771 - val_accuracy: 0.
Epoch 6/10
313/313 [==============================] - ETA: 0s - loss: 0.4009 - accuracy: 0.8160
Epoch 00006: val_loss improved from 0.37714 to 0.34803, saving model to best_model_RMS.h5
313/313 [==============================] - 77s 246ms/step - loss: 0.4009 - accuracy: 0.8160 - val_loss: 0.3480 - val_accuracy: 0.
Epoch 7/10
313/313 [==============================] - ETA: 0s - loss: 0.3708 - accuracy: 0.8323
Epoch 00007: val_loss improved from 0.34803 to 0.31253, saving model to best_model_RMS.h5
313/313 [==============================] - 77s 246ms/step - loss: 0.3708 - accuracy: 0.8323 - val_loss: 0.3125 - val_accuracy: 0.
Epoch 8/10
313/313 [==============================] - ETA: 0s - loss: 0.3437 - accuracy: 0.8490
Epoch 00008: val_loss improved from 0.31253 to 0.31074, saving model to best_model_RMS.h5
313/313 [==============================] - 77s 245ms/step - loss: 0.3437 - accuracy: 0.8490 - val_loss: 0.3107 - val_accuracy: 0.
Epoch 9/10
313/313 [==============================] - ETA: 0s - loss: 0.3290 - accuracy: 0.8562
Epoch 00009: val_loss did not improve from 0.31074
313/313 [==============================] - 77s 245ms/step - loss: 0.3290 - accuracy: 0.8562 - val_loss: 0.3168 - val_accuracy: 0.
Epoch 10/10
313/313 [==============================] - ETA: 0s - loss: 0.3126 - accuracy: 0.8663
Epoch 00010: val_loss improved from 0.31074 to 0.28536, saving model to best_model_RMS.h5
313/313 [==============================] - 77s 245ms/step - loss: 0.3126 - accuracy: 0.8663 - val_loss: 0.2854 - val_accuracy: 0.
```

```
plt.plot(history_rms.history['accuracy'])
plt.plot(history_rms.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




```
model_rms.save_weights('Atharva_RMS_4850.h5')
np.save('Atharva_RMS_4850.npy',history_rms.history)
```

MODEL 5 - OPTIMIZER : ADAM

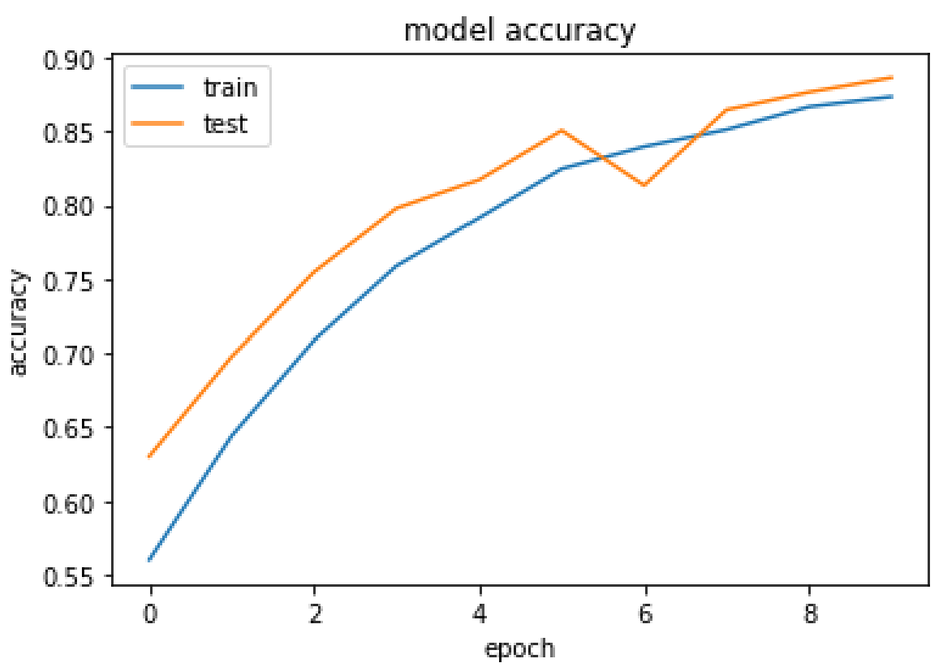
```
model_adam = Sequential()
model_adam.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(48, 48, 3)))
model_adam.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adam.add(MaxPooling2D((2, 2), strides=(2,2)))

model_adam.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adam.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adam.add(MaxPooling2D((2, 2), strides=(2,2)))

model_adam.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adam.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adam.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adam.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adam.add(MaxPooling2D((2, 2), strides=(2,2)))

model_adam.add(Flatten())

model_adam.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))

model_adam.add(Dense(1, activation='sigmoid'))

opt = tf.keras.optimizers.Adam(
    learning_rate=0.001,
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-07,
    amsgrad=False,)
model_adam.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])

model_adam.summary()
```

```
Model: "sequential_16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_128 (Conv2D)          (None, 48, 48, 256)       7168
_________________________________________________________________
conv2d_129 (Conv2D)          (None, 48, 48, 256)       590080
_________________________________________________________________
max_pooling2d_48 (MaxPooling (None, 24, 24, 256)       0
```

```
from keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint(filepath = 'best_model_ADAM.h5',save_best_only = True,verbose=1)

history_adam = model_adam.fit(training_set,
                    epochs=10,
                    callbacks=[checkpoint],
                    validation_data=validation_set)
```

```
Epoch 1/10
313/313 [==============================] - ETA: 0s - loss: 0.6936 - accuracy: 0.5601
Epoch 00001: val_loss improved from inf to 0.65515, saving model to best_model_ADAM.h5
313/313 [==============================] - 77s 247ms/step - loss: 0.6936 - accuracy: 0.5601 - val_loss: 0.6551 - val_accuracy: 0.
Epoch 2/10
313/313 [==============================] - ETA: 0s - loss: 0.6354 - accuracy: 0.6440
Epoch 00002: val_loss improved from 0.65515 to 0.58563, saving model to best_model_ADAM.h5
313/313 [==============================] - 77s 245ms/step - loss: 0.6354 - accuracy: 0.6440 - val_loss: 0.5856 - val_accuracy: 0.
Epoch 3/10
313/313 [==============================] - ETA: 0s - loss: 0.5680 - accuracy: 0.7085
Epoch 00003: val_loss improved from 0.58563 to 0.50897, saving model to best_model_ADAM.h5
313/313 [==============================] - 77s 245ms/step - loss: 0.5680 - accuracy: 0.7085 - val_loss: 0.5090 - val_accuracy: 0.
Epoch 4/10
313/313 [==============================] - ETA: 0s - loss: 0.4974 - accuracy: 0.7588
Epoch 00004: val_loss improved from 0.50897 to 0.44578, saving model to best_model_ADAM.h5
313/313 [==============================] - 77s 245ms/step - loss: 0.4974 - accuracy: 0.7588 - val_loss: 0.4458 - val_accuracy: 0.
Epoch 5/10
313/313 [==============================] - ETA: 0s - loss: 0.4427 - accuracy: 0.7914
Epoch 00005: val_loss improved from 0.44578 to 0.39582, saving model to best_model_ADAM.h5
313/313 [==============================] - 77s 246ms/step - loss: 0.4427 - accuracy: 0.7914 - val_loss: 0.3958 - val_accuracy: 0.
Epoch 6/10
313/313 [==============================] - ETA: 0s - loss: 0.3863 - accuracy: 0.8246
Epoch 00006: val_loss improved from 0.39582 to 0.34291, saving model to best_model_ADAM.h5
313/313 [==============================] - 77s 245ms/step - loss: 0.3863 - accuracy: 0.8246 - val_loss: 0.3429 - val_accuracy: 0.
Epoch 7/10
313/313 [==============================] - ETA: 0s - loss: 0.3628 - accuracy: 0.8396
Epoch 00007: val_loss did not improve from 0.34291
313/313 [==============================] - 77s 245ms/step - loss: 0.3628 - accuracy: 0.8396 - val_loss: 0.3883 - val_accuracy: 0.
Epoch 8/10
313/313 [==============================] - ETA: 0s - loss: 0.3288 - accuracy: 0.8512
Epoch 00008: val_loss improved from 0.34291 to 0.29846, saving model to best_model_ADAM.h5
313/313 [==============================] - 77s 246ms/step - loss: 0.3288 - accuracy: 0.8512 - val_loss: 0.2985 - val_accuracy: 0.
Epoch 9/10
313/313 [==============================] - ETA: 0s - loss: 0.3052 - accuracy: 0.8667
Epoch 00009: val_loss improved from 0.29846 to 0.28250, saving model to best_model_ADAM.h5
313/313 [==============================] - 77s 247ms/step - loss: 0.3052 - accuracy: 0.8667 - val_loss: 0.2825 - val_accuracy: 0.
Epoch 10/10
313/313 [==============================] - ETA: 0s - loss: 0.2934 - accuracy: 0.8735
Epoch 00010: val_loss improved from 0.28250 to 0.26183, saving model to best_model_ADAM.h5
313/313 [==============================] - 78s 248ms/step - loss: 0.2934 - accuracy: 0.8735 - val_loss: 0.2618 - val_accuracy: 0.
```

```
plt.plot(history_adam.history['accuracy'])
plt.plot(history_adam.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



```
model_adam.save_weights('Atharva_ADAM_48.h5')
np.save('Atharva_ADAM_48.npy',history_adam.history)
```

MODEL 6 - OPTIMIZER : ADADelta

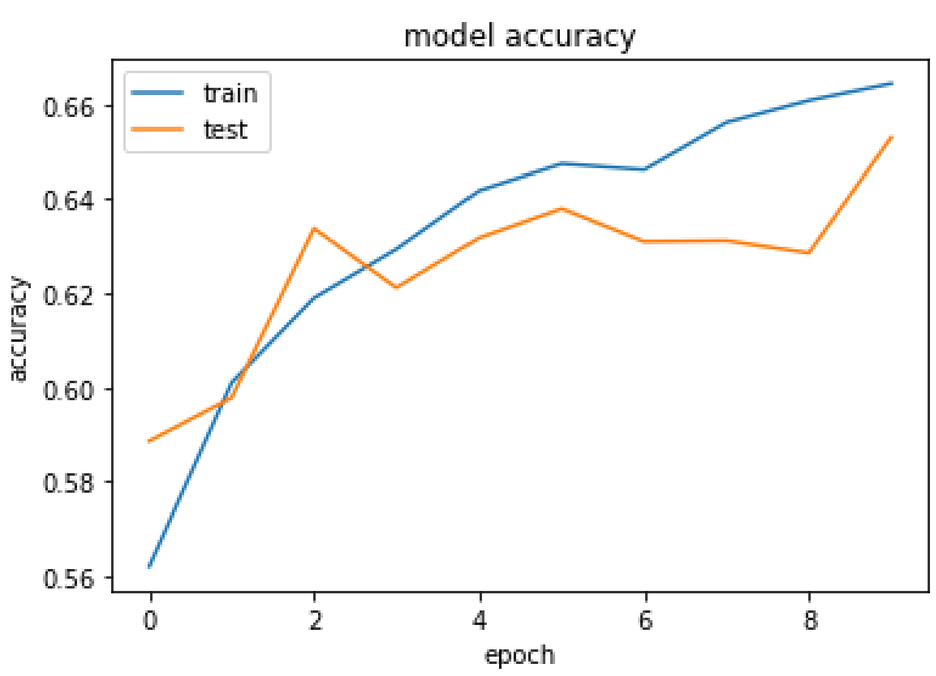
```
model_adad = Sequential()
model_adad.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(48, 48, 3)))
model_adad.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adad.add(MaxPooling2D((2, 2), strides=(2,2)))

model_adad.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adad.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adad.add(MaxPooling2D((2, 2), strides=(2,2)))

model_adad.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adad.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adad.add(MaxPooling2D((2, 2), strides=(2,2)))

model_adad.add(Flatten())

model_adad.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))

model_adad.add(Dense(1, activation='sigmoid'))

opt = tf.keras.optimizers.Adadelta(
    learning_rate=0.001, rho=0.95, epsilon=1e-07, name="Adadelta", )
model_adad.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])

model_adad.summary()
```

```
Model: "sequential_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_136 (Conv2D)          (None, 48, 48, 256)       7168
_________________________________________________________________
conv2d_137 (Conv2D)          (None, 48, 48, 256)       590080
_________________________________________________________________
max_pooling2d_51 (MaxPooling (None, 24, 24, 256)       0
_________________________________________________________________
conv2d_138 (Conv2D)          (None, 24, 24, 128)       295040
_________________________________________________________________
conv2d_139 (Conv2D)          (None, 24, 24, 128)       147584
_________________________________________________________________
max_pooling2d_52 (MaxPooling (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_140 (Conv2D)          (None, 12, 12, 64)        73792
_________________________________________________________________
conv2d_141 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_142 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_143 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_53 (MaxPooling (None, 6, 6, 64)          0
_________________________________________________________________
flatten_17 (Flatten)         (None, 2304)              0
_________________________________________________________________
dense_34 (Dense)             (None, 128)               295040
_________________________________________________________________
dense_35 (Dense)             (None, 1)                 129
=================================================================
Total params: 1,519,617
Trainable params: 1,519,617
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint(filepath = 'best_model_ADADELTA.h5',save_best_only = True,verbose=1)

history_adad = model_adad.fit(training_set,
                    epochs=10,
                    callbacks=[checkpoint],
                    validation_data=validation_set)
```

```
Epoch 1/10
313/313 [==============================] - ETA: 0s - loss: 0.6824 - accuracy: 0.5621
Epoch 00001: val_loss improved from inf to 0.67433, saving model to best_model_ADADELTA.h5
313/313 [==============================] - 79s 251ms/step - loss: 0.6824 - accuracy: 0.5621 - val_loss: 0.6743 - val_accuracy: 0.
Epoch 2/10
313/313 [==============================] - ETA: 0s - loss: 0.6702 - accuracy: 0.6010
Epoch 00002: val_loss improved from 0.67433 to 0.66563, saving model to best_model_ADADELTA.h5
313/313 [==============================] - 79s 252ms/step - loss: 0.6702 - accuracy: 0.6010 - val_loss: 0.6656 - val_accuracy: 0.
Epoch 3/10
313/313 [==============================] - ETA: 0s - loss: 0.6615 - accuracy: 0.6190
Epoch 00003: val_loss improved from 0.66563 to 0.65408, saving model to best_model_ADADELTA.h5
313/313 [==============================] - 79s 253ms/step - loss: 0.6615 - accuracy: 0.6190 - val_loss: 0.6541 - val_accuracy: 0.
Epoch 4/10
313/313 [==============================] - ETA: 0s - loss: 0.6528 - accuracy: 0.6294
Epoch 00004: val_loss improved from 0.65408 to 0.65017, saving model to best_model_ADADELTA.h5
313/313 [==============================] - 79s 253ms/step - loss: 0.6528 - accuracy: 0.6294 - val_loss: 0.6502 - val_accuracy: 0.
Epoch 5/10
313/313 [==============================] - ETA: 0s - loss: 0.6444 - accuracy: 0.6418
Epoch 00005: val_loss improved from 0.65017 to 0.64251, saving model to best_model_ADADELTA.h5
313/313 [==============================] - 80s 255ms/step - loss: 0.6444 - accuracy: 0.6418 - val_loss: 0.6425 - val_accuracy: 0.
Epoch 6/10
313/313 [==============================] - ETA: 0s - loss: 0.6373 - accuracy: 0.6474
Epoch 00006: val_loss improved from 0.64251 to 0.63577, saving model to best_model_ADADELTA.h5
313/313 [==============================] - 79s 253ms/step - loss: 0.6373 - accuracy: 0.6474 - val_loss: 0.6358 - val_accuracy: 0.
Epoch 7/10
313/313 [==============================] - ETA: 0s - loss: 0.6325 - accuracy: 0.6461
Epoch 00007: val_loss improved from 0.63577 to 0.63574, saving model to best_model_ADADELTA.h5
313/313 [==============================] - 80s 254ms/step - loss: 0.6325 - accuracy: 0.6461 - val_loss: 0.6357 - val_accuracy: 0.
Epoch 8/10
313/313 [==============================] - ETA: 0s - loss: 0.6269 - accuracy: 0.6562
Epoch 00008: val_loss improved from 0.63574 to 0.63433, saving model to best_model_ADADELTA.h5
313/313 [==============================] - 79s 253ms/step - loss: 0.6269 - accuracy: 0.6562 - val_loss: 0.6343 - val_accuracy: 0.
Epoch 9/10
313/313 [==============================] - ETA: 0s - loss: 0.6213 - accuracy: 0.6608
Epoch 00009: val_loss did not improve from 0.63433
313/313 [==============================] - 79s 253ms/step - loss: 0.6213 - accuracy: 0.6608 - val_loss: 0.6356 - val_accuracy: 0.
Epoch 10/10
313/313 [==============================] - ETA: 0s - loss: 0.6180 - accuracy: 0.6644
Epoch 00010: val_loss improved from 0.63433 to 0.61984, saving model to best_model_ADADELTA.h5
313/313 [==============================] - 79s 252ms/step - loss: 0.6180 - accuracy: 0.6644 - val_loss: 0.6198 - val_accuracy: 0.
```

```
plt.plot(history_adad.history['accuracy'])
plt.plot(history_adad.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




```
model_adad.save_weights('Atharva_ADADELTA_48.h5')
np.save('Atharva_ADADELTA_48.npy',history_adad.history)
```

MODEL 7 - OPTIMIZER : ADAGrad

```
model_adag = Sequential()
model_adag.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(48, 48, 3)))
model_adag.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adag.add(MaxPooling2D((2, 2), strides=(2,2)))

model_adag.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adag.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adag.add(MaxPooling2D((2, 2), strides=(2,2)))

model_adag.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
```

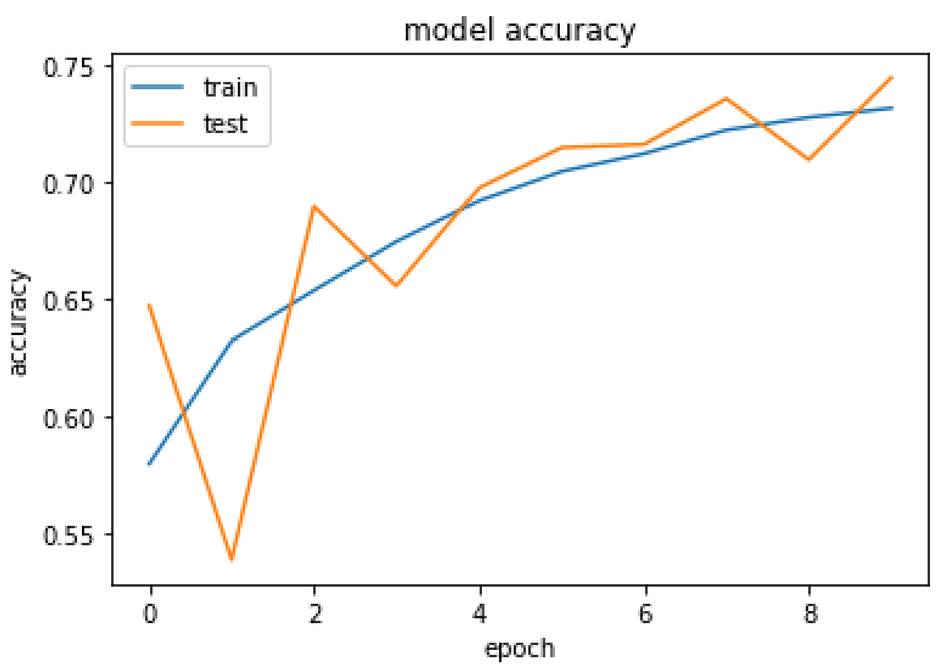
```
model_adag.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adag.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adag.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adag.add(MaxPooling2D((2, 2), strides=(2,2)))

model_adag.add(Flatten())

model_adag.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))

model_adag.add(Dense(1, activation='sigmoid'))

opt = tf.keras.optimizers.Adagrad(
    learning_rate=0.001,
    initial_accumulator_value=0.1,
    epsilon=1e-07,)
model_adag.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])

model_adag.summary()
```

```
Model: "sequential_18"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_144 (Conv2D)          (None, 48, 48, 256)       7168
_________________________________________________________________
conv2d_145 (Conv2D)          (None, 48, 48, 256)       590080
_________________________________________________________________
max_pooling2d_54 (MaxPooling (None, 24, 24, 256)       0
_________________________________________________________________
conv2d_146 (Conv2D)          (None, 24, 24, 128)       295040
_________________________________________________________________
conv2d_147 (Conv2D)          (None, 24, 24, 128)       147584
_________________________________________________________________
max_pooling2d_55 (MaxPooling (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_148 (Conv2D)          (None, 12, 12, 64)        73792
_________________________________________________________________
conv2d_149 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_150 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_151 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_56 (MaxPooling (None, 6, 6, 64)          0
_________________________________________________________________
flatten_18 (Flatten)         (None, 2304)              0
_________________________________________________________________
dense_36 (Dense)             (None, 128)               295040
_________________________________________________________________
dense_37 (Dense)             (None, 1)                 129
=================================================================
Total params: 1,519,617
Trainable params: 1,519,617
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint(filepath = 'best_model_ADAGRAD.h5',save_best_only = True,verbose=1)

history_adag = model_adag.fit(training_set,
                  epochs=10,
                  callbacks=[checkpoint],
                  validation_data=validation_set)
```

```
Epoch 1/10
313/313 [==============================] - ETA: 0s - loss: 0.6747 - accuracy: 0.5795
Epoch 00001: val_loss improved from inf to 0.64739, saving model to best_model_ADAGRAD.h5
313/313 [==============================] - 78s 250ms/step - loss: 0.6747 - accuracy: 0.5795 - val_loss: 0.6474 - val_accuracy: 0.
Epoch 2/10
313/313 [==============================] - ETA: 0s - loss: 0.6404 - accuracy: 0.6321
Epoch 00002: val_loss did not improve from 0.64739
313/313 [==============================] - 78s 250ms/step - loss: 0.6404 - accuracy: 0.6321 - val_loss: 0.7419 - val_accuracy: 0.
Epoch 3/10
313/313 [==============================] - ETA: 0s - loss: 0.6191 - accuracy: 0.6536
Epoch 00003: val_loss improved from 0.64739 to 0.58838, saving model to best_model_ADAGRAD.h5
313/313 [==============================] - 78s 248ms/step - loss: 0.6191 - accuracy: 0.6536 - val_loss: 0.5884 - val_accuracy: 0.
Epoch 4/10
313/313 [==============================] - ETA: 0s - loss: 0.6012 - accuracy: 0.6745
Epoch 00004: val_loss did not improve from 0.58838
313/313 [==============================] - 78s 250ms/step - loss: 0.6012 - accuracy: 0.6745 - val_loss: 0.6177 - val_accuracy: 0.
Epoch 5/10
313/313 [==============================] - ETA: 0s - loss: 0.5848 - accuracy: 0.6920
Epoch 00005: val_loss improved from 0.58838 to 0.57214, saving model to best_model_ADAGRAD.h5
313/313 [==============================] - 79s 252ms/step - loss: 0.5848 - accuracy: 0.6920 - val_loss: 0.5721 - val_accuracy: 0.
Epoch 6/10
```

```
plt.plot(history_adag.history['accuracy'])
plt.plot(history_adag.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




```
model_adag.save_weights('Atharva_ADAG_48.h5')
np.save('Atharva_ADAG_48.npy',history_adag.history)
```

MODEL 8 - OPTIMIZER : ADAMax

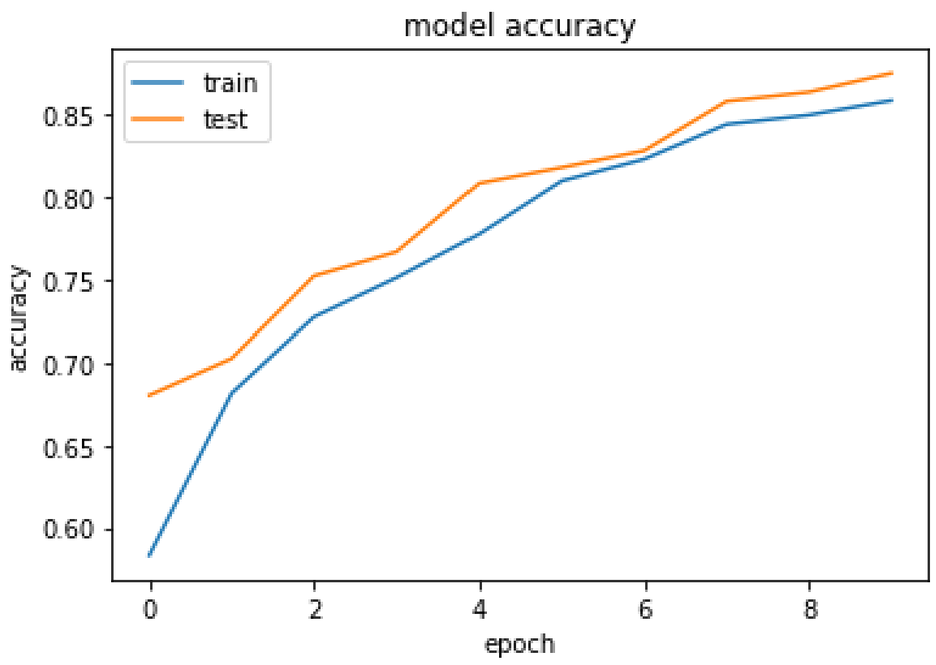
```
model_adamax = Sequential()
model_adamax.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(48, 48, 3)))
model_adamax.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adamax.add(MaxPooling2D((2, 2), strides=(2,2)))

model_adamax.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adamax.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adamax.add(MaxPooling2D((2, 2), strides=(2,2)))

model_adamax.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adamax.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adamax.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adamax.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adamax.add(MaxPooling2D((2, 2), strides=(2,2)))

model_adamax.add(Flatten())

model_adamax.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))

model_adamax.add(Dense(1, activation='sigmoid'))

opt = tf.keras.optimizers.Adamax(
    learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07,)
model_adamax.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])

model_adamax.summary()
```

```
Model: "sequential_19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_152 (Conv2D)          (None, 48, 48, 256)       7168
_________________________________________________________________
conv2d_153 (Conv2D)          (None, 48, 48, 256)       590080
_________________________________________________________________
max_pooling2d_57 (MaxPooling (None, 24, 24, 256)       0
_________________________________________________________________
conv2d_154 (Conv2D)          (None, 24, 24, 128)       295040
_________________________________________________________________
conv2d_155 (Conv2D)          (None, 24, 24, 128)       147584
_________________________________________________________________
max_pooling2d_58 (MaxPooling (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_156 (Conv2D)          (None, 12, 12, 64)        73792
_________________________________________________________________
conv2d_157 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_158 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_159 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_59 (MaxPooling (None, 6, 6, 64)          0
_________________________________________________________________
flatten_19 (Flatten)         (None, 2304)              0
_________________________________________________________________
dense_38 (Dense)             (None, 128)               295040
_________________________________________________________________
dense_39 (Dense)             (None, 1)                 129
=================================================================
Total params: 1,519,617
Trainable params: 1,519,617
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint(filepath = 'best_model_ADAMAX.h5',save_best_only = True,verbose=1)

history_adamax = model_adamax.fit(training_set,
                  epochs=10,
                  callbacks=[checkpoint],
                  validation_data=validation_set)
```

Epoch 1/10

```
plt.plot(history_adamax.history['accuracy'])
plt.plot(history_adamax.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



313/313 [==============================] - 78s 250ms/step - loss: 0.3897 - accuracy: 0.8232 - val_loss: 0.3813 - val_accuracy: 0.

```
model_adamax.save_weights('Atharva_ADAMAX_48.h5')
np.save('Atharva_ADAMAX_48.npy',history_adamax.history)
```

313/313 [==============================] - 77s 247ms/step - loss: 0.3548 - accuracy: 0.8444 - val_loss: 0.3268 - val_accuracy: 0.

Testing the Model against custom test data

We will validate the test accuracy using the model generated using the NADAM optimizer as it has the highest degree of performance

Epoch 00010: val_loss improved from 0.31083 to 0.28348, saving model to best_model_ADAMAX.h5

```
!mkdir augmentedtest
```

```
test_datagen = ImageDataGenerator(rescale = 1.0/255.0,
                                  shear_range = 0.2,
                                  zoom_range = 0.2,
                                  horizontal_flip = True,
                                  )

test_set = test_datagen.flow_from_directory('./test/',
                                            target_size = (48, 48),
                                            batch_size = 1,
                                            class_mode = 'binary',
                                        save_to_dir='./augmentedtest',
         save_prefix='aug',
        save_format='jpg')
```

Found 40 images belonging to 1 classes.

```
import cv2
predict_files = glob.glob("/content/augmentedtest/*.jpg*")
predictor, image_id = [], []
for file in predict_files:
  image = cv2.imread(file)
  img = cv2.resize(image, (48,48))
  img = np.expand_dims(img, 0)

  prediction = model.predict(img)

  predict = str(model.predict_classes(img)[0][0])

  predictor.extend(list(predict))
```

```
predictions =[]
c = 0
w = 0
classified = ['cat','dog']
for i in range(len(predictor)):
  if predictor[i] == 0:
    predictions.append(classified[0])
  else:
    predictions.append(classified[1])
```

```
predict_files = glob.glob("/content/augmentedtest*.jpg*")

for file in predict_files:

  data.append(os.path.basename(file)[:3])

  data = pd.DataFrame(data, columns=['FileName'])

  data['Predictions'] = predictions
```

Last half images are of dogs - so by using data.tail we can see the predicted results

```
data.tail(10)
```

	FileName
150	cat
151	dog
152	dog
153	dog
154	cat
155	dog
156	dog
157	cat
158	dog
159	cat

First half images are of cats - so by using data.head we can see the predicted results

```
data.head(10)
```

	FileName
0	dog
1	cat
2	cat
3	dog
4	dog
5	cat
6	cat
7	cat
8	cat
9	dog

Regularized Model 1 - Optimizer : NADAM

Since Nadam and RMSProp performed the best in unregularised models, we will only try regularisation on them.

```
model_nadam_r = Sequential()
model_nadam_r.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(48, 48, 3)))
model_nadam_r.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_nadam_r.add(MaxPooling2D((2, 2), strides=(2,2)))
model_nadam_r.add(Dropout(0.1))
model_nadam_r.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_nadam_r.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_nadam_r.add(MaxPooling2D((2, 2), strides=(2,2)))
model_nadam_r.add(Dropout(0.1))
model_nadam_r.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_nadam_r.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_nadam_r.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_nadam_r.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
```

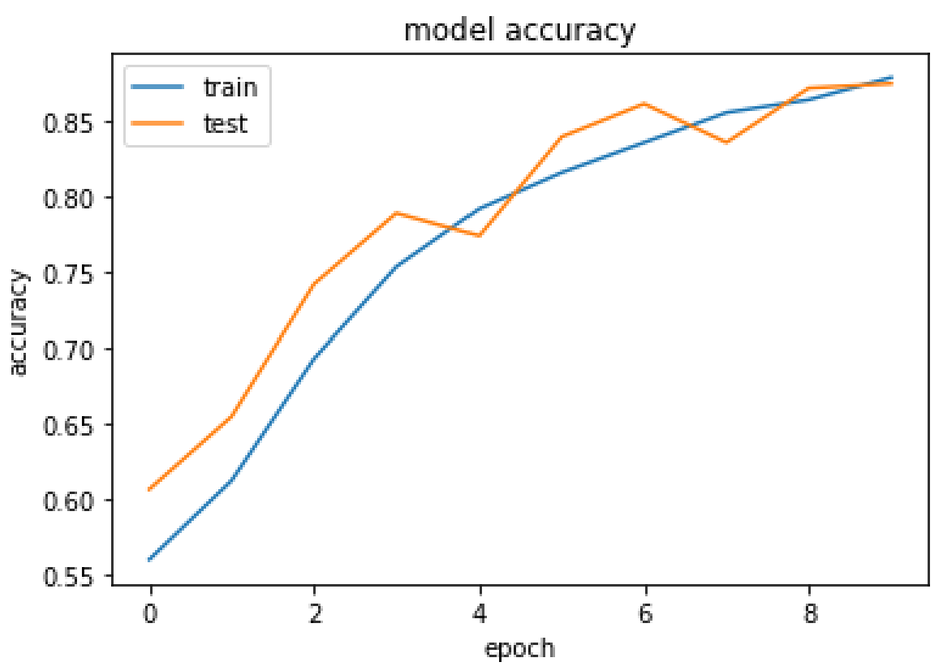
```
model_nadam_r.add(MaxPooling2D((2, 2), strides=(2,2)))
model_nadam_r.add(Dropout(0.2))
model_nadam_r.add(Flatten())
model_nadam_r.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model_nadam_r.add(Dense(1, activation='sigmoid'))

opt = tf.keras.optimizers.Nadam(
   learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07)
model_nadam_r.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])

model_nadam_r.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 48, 48, 256)       7168
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 48, 48, 256)       590080
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 24, 24, 256)       0
_________________________________________________________________
dropout (Dropout)            (None, 24, 24, 256)       0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 24, 24, 128)       295040
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 24, 24, 128)       147584
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 12, 12, 64)        73792
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               295040
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 129
=================================================================
Total params: 1,519,617
Trainable params: 1,519,617
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint(filepath = 'best_model_NADAM_Regularized.h5',save_best_only = True,verbose=1)

history_nadam_r = model.fit(training_set,
                    epochs=10,
                    callbacks=[checkpoint],
                    validation_data=validation_set)
```

```
Epoch 1/10
313/313 [==============================] - ETA: 0s - loss: 0.6989 - accuracy: 0.5597
Epoch 00001: val_loss improved from inf to 0.65994, saving model to best_model_NADAM_Regularized.h5
313/313 [==============================] - 71s 228ms/step - loss: 0.6989 - accuracy: 0.5597 - val_loss: 0.6599 - val_accuracy: 0.
Epoch 2/10
313/313 [==============================] - ETA: 0s - loss: 0.6613 - accuracy: 0.6120
Epoch 00002: val_loss improved from 0.65994 to 0.61558, saving model to best_model_NADAM_Regularized.h5
313/313 [==============================] - 71s 227ms/step - loss: 0.6613 - accuracy: 0.6120 - val_loss: 0.6156 - val_accuracy: 0.
Epoch 3/10
313/313 [==============================] - ETA: 0s - loss: 0.5822 - accuracy: 0.6925
Epoch 00003: val_loss improved from 0.61558 to 0.51408, saving model to best_model_NADAM_Regularized.h5
313/313 [==============================] - 71s 227ms/step - loss: 0.5822 - accuracy: 0.6925 - val_loss: 0.5141 - val_accuracy: 0.
Epoch 4/10
313/313 [==============================] - ETA: 0s - loss: 0.5053 - accuracy: 0.7531
Epoch 00004: val_loss improved from 0.51408 to 0.45279, saving model to best_model_NADAM_Regularized.h5
313/313 [==============================] - 71s 227ms/step - loss: 0.5053 - accuracy: 0.7531 - val_loss: 0.4528 - val_accuracy: 0.
Epoch 5/10
313/313 [==============================] - ETA: 0s - loss: 0.4461 - accuracy: 0.7915
Epoch 00005: val loss did not improve from 0.45279
```

```
plt.plot(history_nadam_r.history['accuracy'])
plt.plot(history_nadam_r.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




```
model_nadam_r.save_weights('Atharva_NADAM_Regularized_48.h5')
np.save('Atharva_NADAM_Regularized_48.npy',history_nadam_r.history)
```

Regularized Model 2 - Optimizer : RMSProp

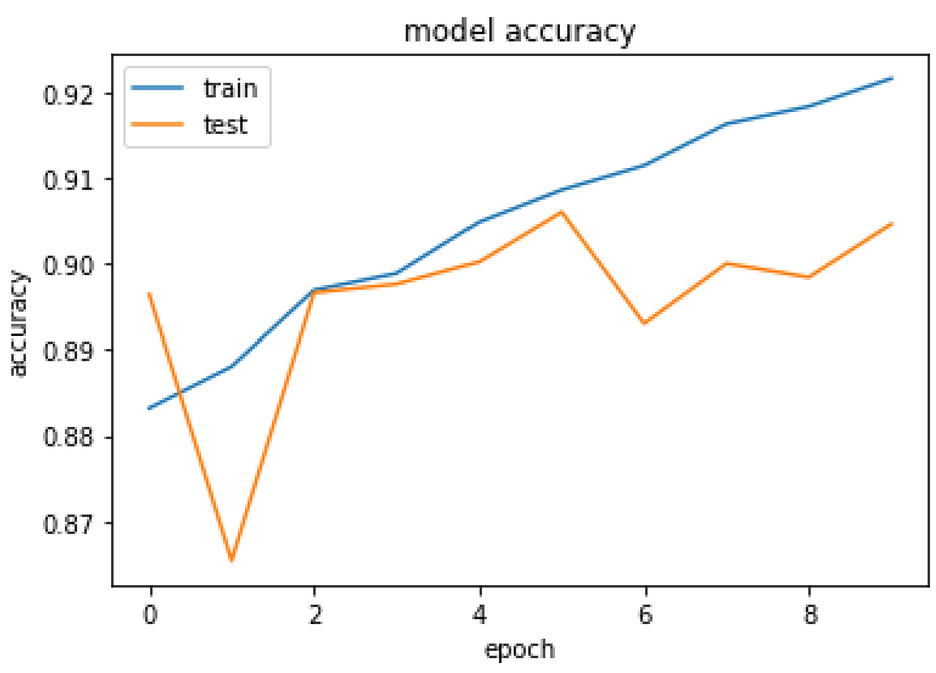
Since Nadam and RMSProp performed the best in unregularised models, we will only try regularisation on them.

```
model_rms_r = Sequential()
model_rms_r.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(48, 48, 3)))
model_rms_r.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_rms_r.add(MaxPooling2D((2, 2), strides=(2,2)))
model_rms_r.add(Dropout(0.1))
model_rms_r.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_rms_r.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_rms_r.add(MaxPooling2D((2, 2), strides=(2,2)))
model_rms_r.add(Dropout(0.1))
model_rms_r.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_rms_r.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_rms_r.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_rms_r.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_rms_r.add(MaxPooling2D((2, 2), strides=(2,2)))
model_rms_r.add(Dropout(0.2))
model_rms_r.add(Flatten())
model_rms_r.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model_rms_r.add(Dense(1, activation='sigmoid'))

opt = tf.keras.optimizers.RMSprop(
    learning_rate=0.001,
    rho=0.9,
    momentum=0.0,
    epsilon=1e-07,
    centered=False,)
model_rms_r.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
```

```
checkpoint = ModelCheckpoint(filepath = 'best_model_RMS_Regularized.h5',save_best_only = True,verbose=1)
history_rms_r = model.fit(training_set,
                  epochs=10,
                  callbacks=[checkpoint],
                  validation_data=validation_set)
```

```
Epoch 1/10
313/313 [==============================] - ETA: 0s - loss: 0.2744 - accuracy: 0.8832
Epoch 00001: val_loss improved from inf to 0.25109, saving model to best_model_RMS_Regularized.h5
313/313 [==============================] - 71s 228ms/step - loss: 0.2744 - accuracy: 0.8832 - val_loss: 0.2511 - val_accuracy: 0.
Epoch 2/10
313/313 [==============================] - ETA: 0s - loss: 0.2594 - accuracy: 0.8881
Epoch 00002: val_loss did not improve from 0.25109
313/313 [==============================] - 71s 226ms/step - loss: 0.2594 - accuracy: 0.8881 - val_loss: 0.3113 - val_accuracy: 0.
Epoch 3/10
313/313 [==============================] - ETA: 0s - loss: 0.2435 - accuracy: 0.8969
Epoch 00003: val_loss did not improve from 0.25109
313/313 [==============================] - 71s 225ms/step - loss: 0.2435 - accuracy: 0.8969 - val_loss: 0.2599 - val_accuracy: 0.
Epoch 4/10
313/313 [==============================] - ETA: 0s - loss: 0.2358 - accuracy: 0.8989
Epoch 00004: val_loss improved from 0.25109 to 0.23949, saving model to best_model_RMS_Regularized.h5
313/313 [==============================] - 71s 226ms/step - loss: 0.2358 - accuracy: 0.8989 - val_loss: 0.2395 - val_accuracy: 0.
Epoch 5/10
313/313 [==============================] - ETA: 0s - loss: 0.2252 - accuracy: 0.9049
Epoch 00005: val_loss did not improve from 0.23949
313/313 [==============================] - 71s 226ms/step - loss: 0.2252 - accuracy: 0.9049 - val_loss: 0.2398 - val_accuracy: 0.
Epoch 6/10
313/313 [==============================] - ETA: 0s - loss: 0.2121 - accuracy: 0.9086
Epoch 00006: val_loss improved from 0.23949 to 0.23455, saving model to best_model_RMS_Regularized.h5
313/313 [==============================] - 71s 227ms/step - loss: 0.2121 - accuracy: 0.9086 - val_loss: 0.2345 - val_accuracy: 0.
Epoch 7/10
313/313 [==============================] - ETA: 0s - loss: 0.2076 - accuracy: 0.9115
Epoch 00007: val_loss did not improve from 0.23455
313/313 [==============================] - 71s 226ms/step - loss: 0.2076 - accuracy: 0.9115 - val_loss: 0.2557 - val_accuracy: 0.
Epoch 8/10
313/313 [==============================] - ETA: 0s - loss: 0.2012 - accuracy: 0.9163
Epoch 00008: val_loss did not improve from 0.23455
313/313 [==============================] - 71s 225ms/step - loss: 0.2012 - accuracy: 0.9163 - val_loss: 0.2431 - val_accuracy: 0.
Epoch 9/10
313/313 [==============================] - ETA: 0s - loss: 0.1953 - accuracy: 0.9183
Epoch 00009: val_loss did not improve from 0.23455
313/313 [==============================] - 70s 225ms/step - loss: 0.1953 - accuracy: 0.9183 - val_loss: 0.2637 - val_accuracy: 0.
Epoch 10/10
313/313 [==============================] - ETA: 0s - loss: 0.1875 - accuracy: 0.9216
Epoch 00010: val_loss improved from 0.23455 to 0.23139, saving model to best_model_RMS_Regularized.h5
313/313 [==============================] - 71s 227ms/step - loss: 0.1875 - accuracy: 0.9216 - val_loss: 0.2314 - val_accuracy: 0.
```

```
plt.plot(history_rms_r.history['accuracy'])
plt.plot(history_rms_r.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




```
model_rms_r.save_weights('Atharva_RMS_Regularized_48.h5')
np.save('Atharva_RMS_Regularized_48.npy',history_rms_r.history)
```

CONCLUSION

Thus, we successfully implemented Image Classification using CatsvsDogs Dataset