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
```

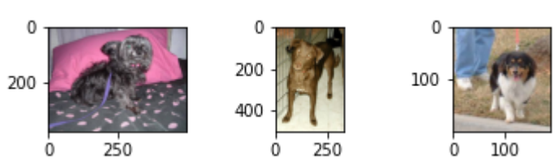
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3

Enter your authorization code:
4/4AFvIZgUD2m6K8S5oeA34L51WccKpWJbfWyTC_iCbSPlNhm6JmZSnKU
Mounted at /content/gdrive
```

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

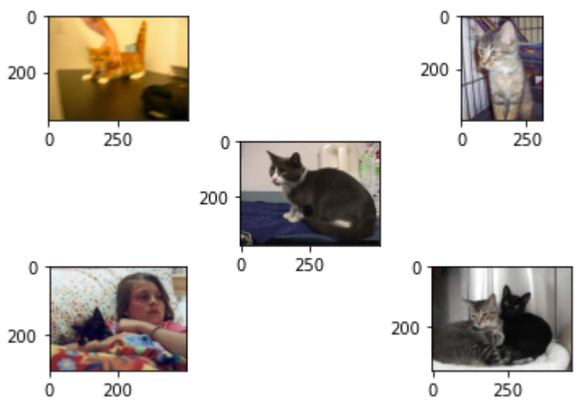
```
!mkdir dataset
!mkdir ./dataset/cat
!mkdir ./dataset/dog
!mkdir validation
!mkdir ./validation/cat
!mkdir ./validation/dog

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

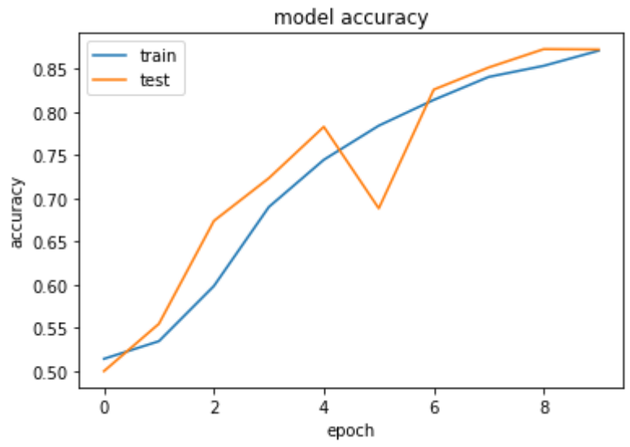
```
model = Sequential()
model.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(48, 48, 3)))
model.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2), strides=(2,2)))

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_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
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: 0.5000
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: 0.5548
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: 0.6738
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: 0.7232
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: 0.7826
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: 0.6882
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: 0.8256
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: 0.8512
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: 0.8724
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: 0.8720
```

```
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

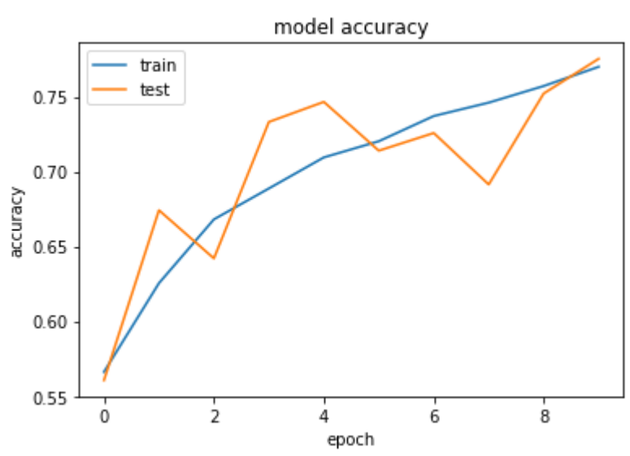
```
model_sgd = Sequential()
model_sgd.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(48, 48, 3)))
model_sgd.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_sgd.add(MaxPooling2D((2, 2), strides=(2,2)))

model_sgd.add(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
```

```
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: 0.5606
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: 0.6742
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: 0.6420
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: 0.7332
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: 0.7466
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: 0.7140
Epoch 7/10
313/313 [==============================] - ETA: 0s - loss: 0.5281 - accuracy: 0.7372
Epoch 00007: val_loss did not improve from 0.51241
313/313 [==============================] - 78s 249ms/step - loss: 0.5281 - accuracy: 0.7372 - val_loss: 0.5375 - val_accuracy: 0.7258
Epoch 8/10
313/313 [==============================] - ETA: 0s - loss: 0.5158 - accuracy: 0.7461
Epoch 00008: val_loss did not improve from 0.51241
313/313 [==============================] - 78s 250ms/step - loss: 0.5158 - accuracy: 0.7461 - val_loss: 0.6087 - val_accuracy: 0.6914
Epoch 9/10
313/313 [==============================] - ETA: 0s - loss: 0.4977 - accuracy: 0.7572
Epoch 00009: val_loss improved from 0.51241 to 0.50194, saving model to best_model_SGD.h5
313/313 [==============================] - 78s 250ms/step - loss: 0.4977 - accuracy: 0.7572 - val_loss: 0.5019 - val_accuracy: 0.7522
Epoch 10/10
313/313 [==============================] - ETA: 0s - loss: 0.4817 - accuracy: 0.7700
Epoch 00010: val_loss improved from 0.50194 to 0.47040, saving model to best_model_SGD.h5
313/313 [==============================] - 78s 249ms/step - loss: 0.4817 - accuracy: 0.7700 - val_loss: 0.4704 - val_accuracy: 0.7754
```

```
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())
```

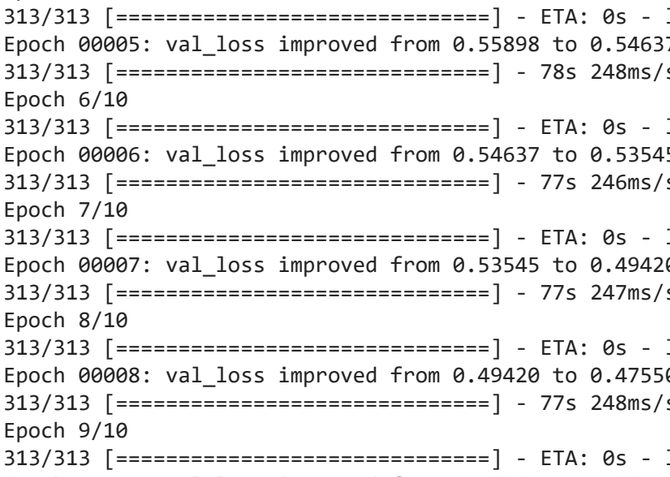
```
model_sgdn.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))

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: 0.6062
Epoch 2/10
313/313 [==============================] - ETA: 0s - loss: 0.6369 - accuracy: 0.6341
Epoch 00002: val_loss improved from 0.65042 to 0.61374, saving model to best_model_SGDN.h5
313/313 [==============================] - 78s 250ms/step - loss: 0.6369 - accuracy: 0.6341 - val_loss: 0.6137 - val_accuracy: 0.6496
Epoch 3/10
313/313 [==============================] - ETA: 0s - loss: 0.6067 - accuracy: 0.6701
Epoch 00003: val_loss did not improve from 0.61374
313/313 [==============================] - 78s 250ms/step - loss: 0.6067 - accuracy: 0.6701 - val_loss: 0.7182 - val_accuracy: 0.6076
Epoch 4/10
313/313 [==============================] - ETA: 0s - loss: 0.5873 - accuracy: 0.6909
Epoch 00004: val_loss improved from 0.61374 to 0.55898, saving model to best_model_SGDN.h5
313/313 [==============================] - 78s 250ms/step - loss: 0.5873 - accuracy: 0.6909 - val_loss: 0.5590 - val_accuracy: 0.7130
Epoch 5/10
313/313 [==============================] - ETA: 0s - loss: 0.5642 - accuracy: 0.7089
Epoch 00005: val_loss improved from 0.55898 to 0.54637, saving model to best_model_SGDN.h5
313/313 [==============================] - 78s 248ms/step - loss: 0.5642 - accuracy: 0.7089 - val_loss: 0.5464 - val_accuracy: 0.7266
Epoch 6/10
313/313 [==============================] - ETA: 0s - loss: 0.5468 - accuracy: 0.7196
Epoch 00006: val_loss improved from 0.54637 to 0.53545, saving model to best_model_SGDN.h5
313/313 [==============================] - 77s 246ms/step - loss: 0.5468 - accuracy: 0.7196 - val_loss: 0.5355 - val_accuracy: 0.7362
Epoch 7/10
313/313 [==============================] - ETA: 0s - loss: 0.5309 - accuracy: 0.7347
Epoch 00007: val_loss improved from 0.53545 to 0.49420, saving model to best_model_SGDN.h5
313/313 [==============================] - 77s 247ms/step - loss: 0.5309 - accuracy: 0.7347 - val_loss: 0.4942 - val_accuracy: 0.7568
Epoch 8/10
313/313 [==============================] - ETA: 0s - loss: 0.5162 - accuracy: 0.7445
Epoch 00008: val_loss improved from 0.49420 to 0.47550, saving model to best_model_SGDN.h5
313/313 [==============================] - 77s 248ms/step - loss: 0.5162 - accuracy: 0.7445 - val_loss: 0.4755 - val_accuracy: 0.7768
Epoch 9/10
313/313 [==============================] - ETA: 0s - loss: 0.4961 - accuracy: 0.7582
Epoch 00009: val_loss improved from 0.47550 to 0.47301, saving model to best_model_SGDN.h5
313/313 [==============================] - 77s 247ms/step - loss: 0.4961 - accuracy: 0.7582 - val_loss: 0.4730 - val_accuracy: 0.7686
Epoch 10/10
313/313 [==============================] - ETA: 0s - loss: 0.4852 - accuracy: 0.7679
Epoch 00010: val_loss did not improve from 0.47301
313/313 [==============================] - 77s 246ms/step - loss: 0.4852 - accuracy: 0.7679 - val_loss: 0.5043 - val_accuracy: 0.7514
```

```
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()
```




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

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()
```

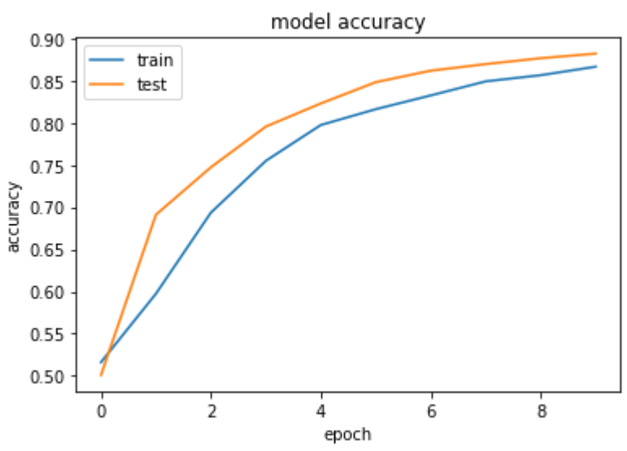
```
Model: "sequential_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_120 (Conv2D)          (None, 48, 48, 256)       7168
_________________________________________________________________
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
_________________________________________________________________
flatten_15 (Flatten)         (None, 2304)              0
_________________________________________________________________
dense_30 (Dense)             (None, 128)               295040
_________________________________________________________________
dense_31 (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_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: 0.5010
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: 0.6910
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: 0.7472
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: 0.7954
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: 0.8228
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: 0.8480
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: 0.8616
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: 0.8694
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: 0.8764
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: 0.8818
```

```
plt.plot(history_rms.history['accuracy'])
plt.plot(history_rms.history['val_accuracy'])
plt.title('model 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'])
```

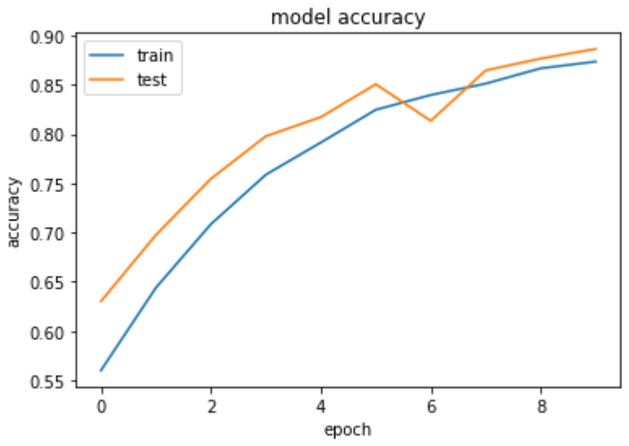
```
model_adam.summary()
```

```
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
_________________________________________________________________
conv2d_130 (Conv2D)          (None, 24, 24, 128)       295040
_________________________________________________________________
conv2d_131 (Conv2D)          (None, 24, 24, 128)       147584
_________________________________________________________________
max_pooling2d_49 (MaxPooling (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_132 (Conv2D)          (None, 12, 12, 64)        73792
_________________________________________________________________
conv2d_133 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_134 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_135 (Conv2D)          (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_50 (MaxPooling (None, 6, 6, 64)          0
_________________________________________________________________
flatten_16 (Flatten)         (None, 2304)              0
_________________________________________________________________
dense_32 (Dense)             (None, 128)               295040
_________________________________________________________________
dense_33 (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_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: 0.6302
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: 0.6976
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: 0.7546
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: 0.7978
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: 0.8172
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: 0.8506
```

```
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(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()
```

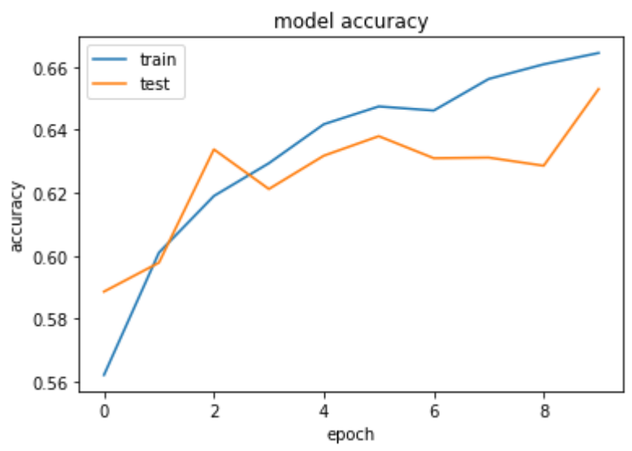
```
Model: "sequential_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_136 (Conv2D)          (None, 48, 48, 256)       7168
_________________________________________________________________
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: 0.5886
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: 0.5978
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: 0.6338
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: 0.6212
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: 0.6318
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: 0.6380
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: 0.6310
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: 0.6312
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: 0.6286
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: 0.6530
```

```
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

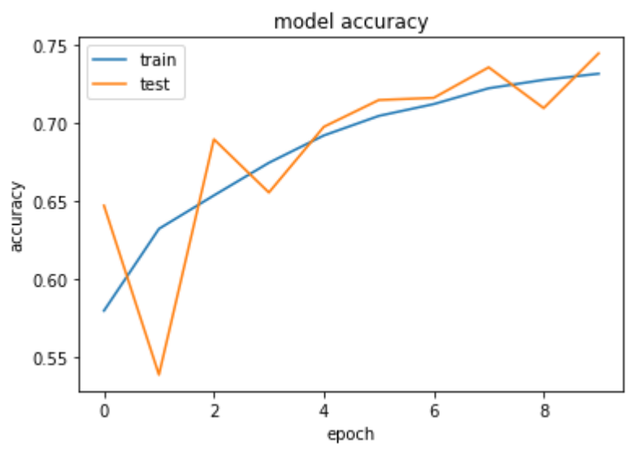
```
model_adag = Sequential()
model_adag.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(48, 48, 3)))
model_adag.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model_adag.add(MaxPooling2D((2, 2), strides=(2,2)))

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
```

```
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: 0.6470
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: 0.5384
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: 0.6896
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: 0.6554
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: 0.6976
Epoch 6/10
313/313 [==============================] - ETA: 0s - loss: 0.5678 - accuracy: 0.7046
Epoch 00006: val_loss improved from 0.57214 to 0.55453, saving model to best_model_ADAGRAD.h5
313/313 [==============================] - 79s 252ms/step - loss: 0.5678 - accuracy: 0.7046 - val_loss: 0.5545 - val_accuracy: 0.7148
Epoch 7/10
313/313 [==============================] - ETA: 0s - loss: 0.5585 - accuracy: 0.7122
Epoch 00007: val_loss improved from 0.55453 to 0.55071, saving model to best_model_ADAGRAD.h5
313/313 [==============================] - 79s 254ms/step - loss: 0.5585 - accuracy: 0.7122 - val_loss: 0.5507 - val_accuracy: 0.7162
Epoch 8/10
313/313 [==============================] - ETA: 0s - loss: 0.5480 - accuracy: 0.7224
Epoch 00008: val_loss improved from 0.55071 to 0.52879, saving model to best_model_ADAGRAD.h5
313/313 [==============================] - 80s 254ms/step - loss: 0.5480 - accuracy: 0.7224 - val_loss: 0.5288 - val_accuracy: 0.7358
Epoch 9/10
313/313 [==============================] - ETA: 0s - loss: 0.5385 - accuracy: 0.7278
Epoch 00009: val_loss did not improve from 0.52879
313/313 [==============================] - 79s 251ms/step - loss: 0.5385 - accuracy: 0.7278 - val_loss: 0.5564 - val_accuracy: 0.7096
Epoch 10/10
313/313 [==============================] - ETA: 0s - loss: 0.5313 - accuracy: 0.7317
Epoch 00010: val_loss improved from 0.52879 to 0.51597, saving model to best_model_ADAGRAD.h5
313/313 [==============================] - 79s 251ms/step - loss: 0.5313 - accuracy: 0.7317 - val_loss: 0.5160 - val_accuracy: 0.7448
```

```
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'))
```

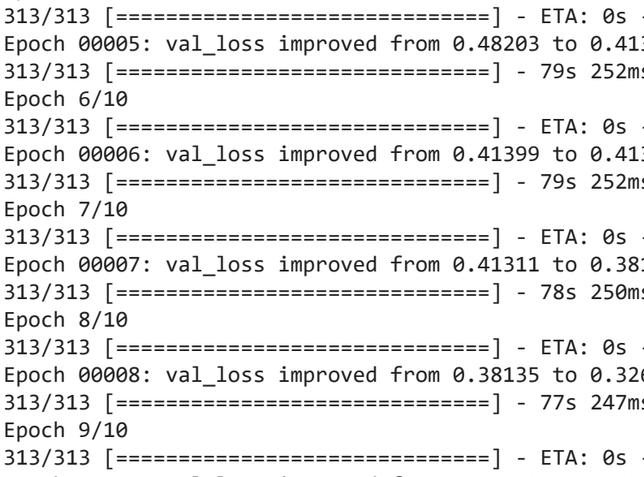
```
opt = tf.keras.optimizers.Adamax(
    learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07,)
model_adamax.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])

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
313/313 [==============================] - ETA: 0s - loss: 0.6777 - accuracy: 0.5832
Epoch 00001: val_loss improved from inf to 0.60260, saving model to best_model_ADAMAX.h5
313/313 [==============================] - 79s 251ms/step - loss: 0.6777 - accuracy: 0.5832 - val_loss: 0.6026 - val_accuracy: 0.6804
Epoch 2/10
313/313 [==============================] - ETA: 0s - loss: 0.5951 - accuracy: 0.6816
Epoch 00002: val_loss improved from 0.60260 to 0.56081, saving model to best_model_ADAMAX.h5
313/313 [==============================] - 79s 252ms/step - loss: 0.5951 - accuracy: 0.6816 - val_loss: 0.5608 - val_accuracy: 0.7026
Epoch 3/10
313/313 [==============================] - ETA: 0s - loss: 0.5446 - accuracy: 0.7276
Epoch 00003: val_loss improved from 0.56081 to 0.50568, saving model to best_model_ADAMAX.h5
313/313 [==============================] - 80s 254ms/step - loss: 0.5446 - accuracy: 0.7276 - val_loss: 0.5057 - val_accuracy: 0.7524
Epoch 4/10
313/313 [==============================] - ETA: 0s - loss: 0.5038 - accuracy: 0.7513
Epoch 00004: val_loss improved from 0.50568 to 0.48203, saving model to best_model_ADAMAX.h5
313/313 [==============================] - 80s 254ms/step - loss: 0.5038 - accuracy: 0.7513 - val_loss: 0.4820 - val_accuracy: 0.7670
Epoch 5/10
313/313 [==============================] - ETA: 0s - loss: 0.4622 - accuracy: 0.7778
Epoch 00005: val_loss improved from 0.48203 to 0.41399, saving model to best_model_ADAMAX.h5
313/313 [==============================] - 79s 252ms/step - loss: 0.4622 - accuracy: 0.7778 - val_loss: 0.4140 - val_accuracy: 0.8086
Epoch 6/10
313/313 [==============================] - ETA: 0s - loss: 0.4156 - accuracy: 0.8101
Epoch 00006: val_loss improved from 0.41399 to 0.41311, saving model to best_model_ADAMAX.h5
313/313 [==============================] - 79s 252ms/step - loss: 0.4156 - accuracy: 0.8101 - val_loss: 0.4131 - val_accuracy: 0.8182
Epoch 7/10
313/313 [==============================] - ETA: 0s - loss: 0.3897 - accuracy: 0.8232
Epoch 00007: val_loss improved from 0.41311 to 0.38135, saving model to best_model_ADAMAX.h5
313/313 [==============================] - 78s 250ms/step - loss: 0.3897 - accuracy: 0.8232 - val_loss: 0.3813 - val_accuracy: 0.8284
Epoch 8/10
313/313 [==============================] - ETA: 0s - loss: 0.3548 - accuracy: 0.8444
Epoch 00008: val_loss improved from 0.38135 to 0.32681, saving model to best_model_ADAMAX.h5
313/313 [==============================] - 77s 247ms/step - loss: 0.3548 - accuracy: 0.8444 - val_loss: 0.3268 - val_accuracy: 0.8584
Epoch 9/10
313/313 [==============================] - ETA: 0s - loss: 0.3409 - accuracy: 0.8500
Epoch 00009: val_loss improved from 0.32681 to 0.31083, saving model to best_model_ADAMAX.h5
313/313 [==============================] - 77s 246ms/step - loss: 0.3409 - accuracy: 0.8500 - val_loss: 0.3108 - val_accuracy: 0.8640
Epoch 10/10
313/313 [==============================] - ETA: 0s - loss: 0.3217 - accuracy: 0.8589
Epoch 00010: val_loss improved from 0.31083 to 0.28348, saving model to best_model_ADAMAX.h5
313/313 [==============================] - 77s 246ms/step - loss: 0.3217 - accuracy: 0.8589 - val_loss: 0.2835 - val_accuracy: 0.8754
```

```
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()
```




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

Testing the Model against custom test data

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

```
import cv2
import glob
predict_files = glob.glob("/content/test/*.jpg")
predictor, image_id = [], []
for file in predict_files:
    img = cv2.imread(file)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img,(48,48))
    img = np.expand_dims(img, 0)
    outcome = [np.argmax(model_rms.predict(img))]
    predictor.extend(list(outcome))
    image_id.extend([i.rsplit("\\")[-1]])
```

```
predict_files = glob.glob("/content/validation/test/*.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_rms.predict(img)

  predict = str(model_rms.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])
```

```
data = []
for file in predict_files:

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

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

df['Predictions'] = predictions

df.head(15)
```

	FileName	Predictions
0	cat	cat
1	cat	cat
2	cat	cat
3	cat	cat
4	cat	cat
5	cat	dog
6	cat	cat
7	cat	cat
8	cat	cat
9	cat	cat
10	cat	cat
11	cat	cat
12	cat	cat
13	cat	cat
14	cat	cat