DL Project Report

The First trial was a Simple CNN: Multi Image Classifier

Set path of train and test

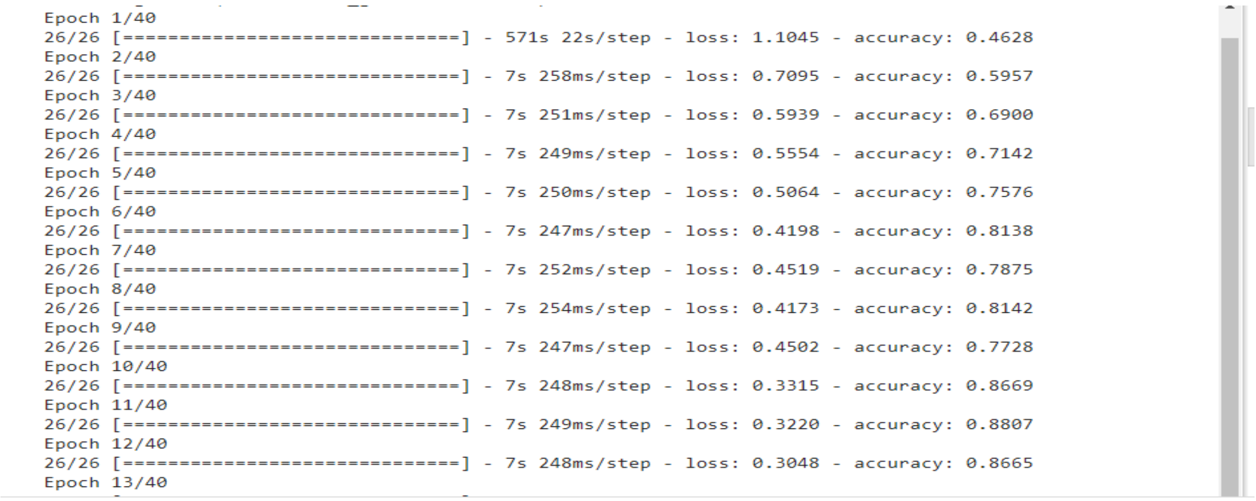
```
import keras
from keras.models import sequential
from keras.layers import Dense,Activation,Dropout,Flatten,Conv2D,MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
import numpy as np

image_shape =(224,224,3)

train_path ='/content/drive/MyDrive/Colab Notebooks/train/train'
test_path = '/content/drive/MyDrive/Colab Notebooks/test/test'
```

- Generate data:

```
from keras.preprocessing.image import ImageDataGenerator

# create a new generator
imagegen = ImageDataGenerator()
# load train data
train = imagegen.flow_from_directory(train_path, class_mode="categorical",
shuffle=False, batch_size=100, target_size=(224, 224))
# load val data
val = imagegen.flow_from_directory(test_path, class_mode="categorical",
shuffle=False, batch_size=100, target_size=(224, 224))
# load val data

from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Flatten, Dense, InputLayer,
BatchNormalization, Dropout
```

- Creating our Convolutional Neural Network code:

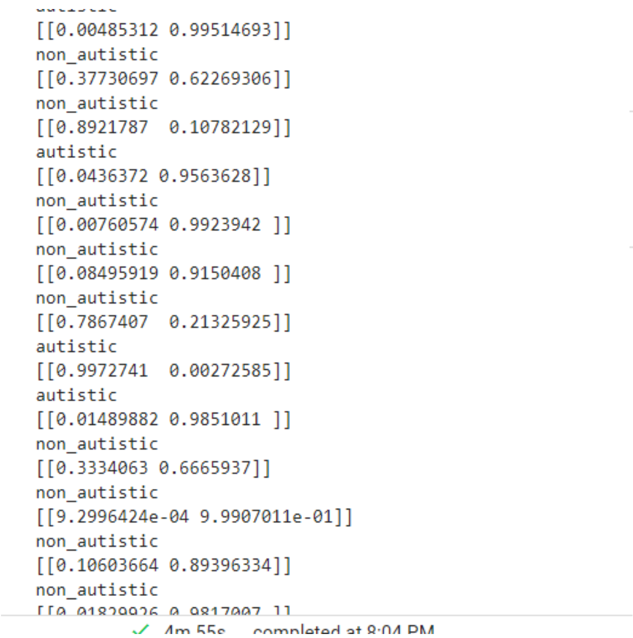
```
• # build a sequential model
model = Sequential()
model.add(InputLayer(input_shape=(224, 224, 3)))

# 1st conv block
model.add(Conv2D(25, (5, 5), activation='relu', strides=(1, 1),
padding='same'))
model.add(MaxPool2D(pool_size=(2, 2), padding='same'))
# 2nd conv block
model.add(Conv2D(50, (5, 5), activation='relu', strides=(2, 2),
padding='same'))
model.add(MaxPool2D(pool_size=(2, 2), padding='same'))
model.add(BatchNormalization())
# 3rd conv block
model.add(Conv2D(70, (3, 3), activation='relu', strides=(2, 2),
padding='same'))
model.add(MaxPool2D(pool_size=(2, 2), padding='valid'))
model.add(BatchNormalization())
# ANN block
model.add(Flatten())
model.add(Dense(units=100, activation='relu'))
model.add(Dense(units=100, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(units=2, activation='softmax'))

# compile model
opt =keras.optimizers.Adam(learning_rate=0.0001)
model.compile(loss='categorical_crossentropy', optimizer=opt,
metrics=['accuracy'])
# fit on data for 30 epochs
model.fit_generator(train, epochs=40, validation_data=val)
model.save("alexnet2.h5")
```



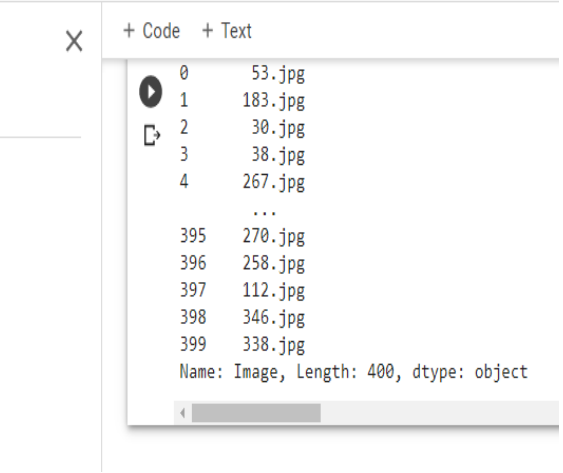
```
Epoch 1/40
26/26 [==============================] - 571s 22s/step - loss: 1.1045 - accuracy: 0.4628
Epoch 2/40
26/26 [==============================] - 7s 258ms/step - loss: 0.7095 - accuracy: 0.5957
Epoch 3/40
26/26 [==============================] - 7s 251ms/step - loss: 0.5939 - accuracy: 0.6900
Epoch 4/40
26/26 [==============================] - 7s 249ms/step - loss: 0.5554 - accuracy: 0.7142
Epoch 5/40
26/26 [==============================] - 7s 250ms/step - loss: 0.5064 - accuracy: 0.7576
Epoch 6/40
26/26 [==============================] - 7s 247ms/step - loss: 0.4198 - accuracy: 0.8138
Epoch 7/40
26/26 [==============================] - 7s 252ms/step - loss: 0.4519 - accuracy: 0.7875
Epoch 8/40
26/26 [==============================] - 7s 254ms/step - loss: 0.4173 - accuracy: 0.8142
Epoch 9/40
26/26 [==============================] - 7s 247ms/step - loss: 0.4502 - accuracy: 0.7728
Epoch 10/40
26/26 [==============================] - 7s 248ms/step - loss: 0.3315 - accuracy: 0.8669
Epoch 11/40
26/26 [==============================] - 7s 249ms/step - loss: 0.3220 - accuracy: 0.8807
Epoch 12/40
26/26 [==============================] - 7s 248ms/step - loss: 0.3048 - accuracy: 0.8665
Epoch 13/40
```

- make prediction:

```
import os
from keras.preprocessing import image
import matplotlib.pyplot as plt
from keras.models import load_model
import pandas as pd

my_img =[]
labels =[]
path =test_path
for i in os.listdir(path):
  my_img.append(i)
  img=image.load_img(path +'//'+i)
  x =image.array_to_img(img)
  x =np.expand_dims(img,axis=0)
  sav= load_model("alexnet2.h5")
  out =sav.predict(x)
  print(out)
  if out[0][1]> out[0][0]:
    print("non_autistic")
    label = 0
    labels.append(label)
  else:
    print("autistic")
    label = 1
    labels.append(label)
submit =pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Submit.csv')
submit['Image'] = my_img
print(submit['Image'])
submit['Label'] = labels
submit.to_csv("submit1.csv",index=False)
#print("Done!")
```

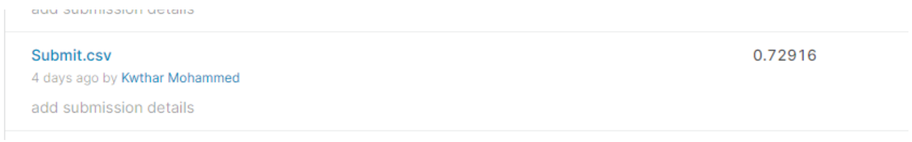


```
autistic
[[0.00485312 0.99514693]]
non_autistic
[[0.37730697 0.62269306]]
non_autistic
[[0.8921787  0.10782129]]
autistic
[[0.0436372 0.9563628]]
non_autistic
[[0.00760574 0.9923942 ]]
non_autistic
[[0.08495919 0.9150408 ]]
non_autistic
[[0.7867407  0.21325925]]
autistic
[[0.9972741  0.00272585]]
autistic
[[0.01489882 0.9851011 ]]
non_autistic
[[0.3334063 0.6665937]]
non_autistic
[[9.2996424e-04 9.9907011e-01]]
non_autistic
[[0.10603664 0.89396334]]
non_autistic
[[0.01829926 0.9817007 ]]
✓ 4m 55s    completed at 8:04 PM
```



```
0       53.jpg
1      183.jpg
2       30.jpg
3       38.jpg
4      267.jpg
         ...
395    270.jpg
396    258.jpg
397    112.jpg
398    346.jpg
399    338.jpg
Name: Image, Length: 400, dtype: object
```

The first accuracy on Kaggle:



add submission details

Submit.csv

4 days ago by Kwthar Mohammed

add submission details

0.72916

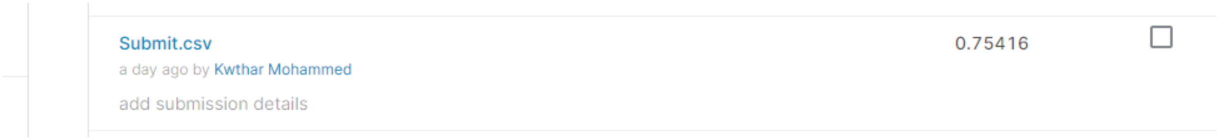
it is the first accuracy as I make the batch size =128

and don’t determine the learning rate of optimizer.

but when I set the batch size with =100

and set learning rate =0.0001 the accuracy increasing

The second accuracy on Kaggle:



Submit.csv

a day ago by Kwthar Mohammed

add submission details

0.75416

□

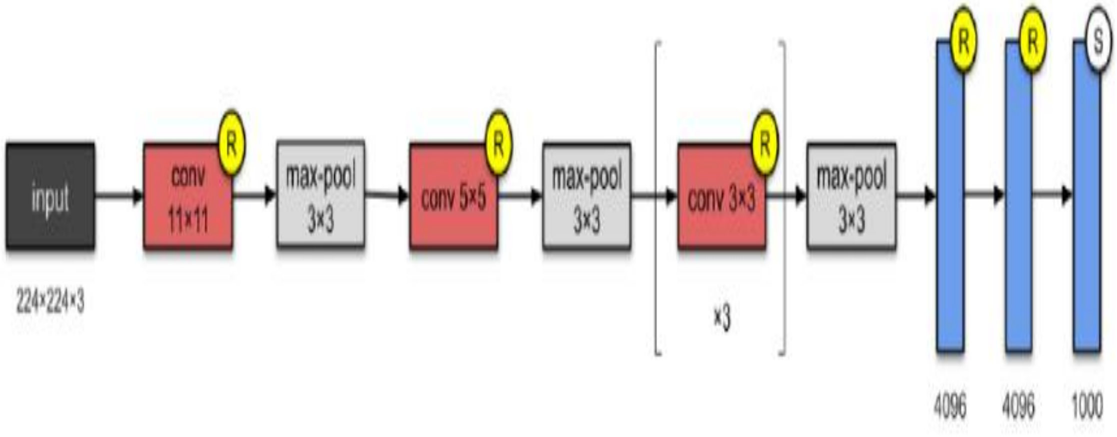
The Second trial was a Simple Alex net: Multi Image Classifier Using Keras

```
import keras
from keras.models import sequential,Model
from keras.layers import
Dense,Activation,Dropout,Flatten,Conv2D,Input,MaxPool2D
from keras.layers.normalization import BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
import numpy as np

image_shape =(224,224,3)

train_path ='/content/drive/MyDrive/Colab Notebooks/train/train'
test_path = '/content/drive/MyDrive/Colab Notebooks/test/test'
```

- Generate data:

```
• from keras.preprocessing.image import ImageDataGenerator

# create a new generator
imagegen = ImageDataGenerator()
# load train data
train = imagegen.flow_from_directory(train_path,
class_mode="categorical", shuffle=False, batch_size=100,
target_size=(224, 224))
# load val data
val = imagegen.flow_from_directory(test_path, class_mode="categorical",
shuffle=False, batch_size=100, target_size=(224, 224))
# load val data
```

- Create Vgg16 Model:
  Architecture :

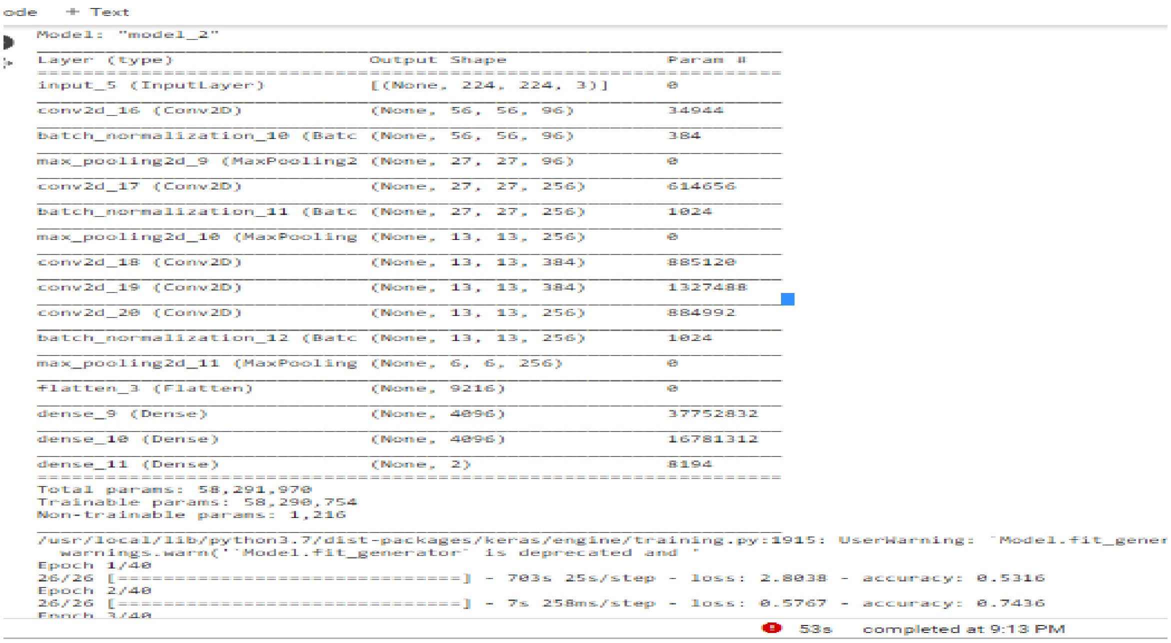


- Function of model:

```
def alexnet(input_shape,n_classes):
  input = Input(input_shape)

  # actually batch normalization didn't exist back then
  # they used LRN (Local Response Normalization) for regularization
  x = Conv2D(96, 11, strides=4, padding='same', activation='relu')(input)
  x = BatchNormalization()(x)
  x = MaxPool2D(3, strides=2)(x)

  x = Conv2D(256, 5, padding='same', activation='relu')(x)
  x = BatchNormalization()(x)
  x = MaxPool2D(3, strides=2)(x)

  x = Conv2D(384, 3, strides=1, padding='same', activation='relu')(x)

  x = Conv2D(384, 3, strides=1, padding='same', activation='relu')(x)

  x = Conv2D(256, 3, strides=1, padding='same', activation='relu')(x)
  x = BatchNormalization()(x)
  x = MaxPool2D(3, strides=2)(x)

  x = Flatten()(x)
  x = Dense(4096, activation='relu')(x)
  x = Dense(4096, activation='relu')(x)

  output = Dense(n_classes, activation='softmax')(x)

  model = Model(input, output)
  return model
```

- calling of model function:

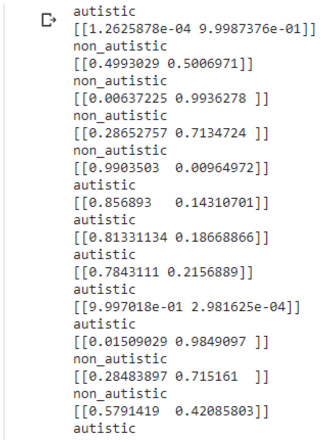
```
num =2
model =alexnet(image_shape,num)
model.summary()
opt =keras.optimizers.Adam(learning_rate=0.00001)
model.compile(loss='categorical_crossentropy', optimizer=opt,
metrics=['accuracy'])
# fit on data for 30 epochs
model.fit_generator(train, epochs=40, validation_data=val)
model.save("alexnet.h5")
```



```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 56, 56, 96)        34944
_________________________________________________________________
batch_normalization_10 (Batc (None, 56, 56, 96)        384
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 27, 27, 96)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 27, 27, 256)       614656
_________________________________________________________________
batch_normalization_11 (Batc (None, 27, 27, 256)       1024
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 13, 13, 256)       0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 13, 13, 384)       885120
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 13, 13, 384)       1327488
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 13, 13, 256)       884992
_________________________________________________________________
batch_normalization_12 (Batc (None, 13, 13, 256)       1024
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 6, 6, 256)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_9 (Dense)              (None, 4096)              37752832
_________________________________________________________________
dense_10 (Dense)             (None, 4096)              16781312
_________________________________________________________________
dense_11 (Dense)             (None, 2)                 8194
=================================================================
Total params: 58,291,970
Trainable params: 58,290,754
Non-trainable params: 1,216
_________________________________________________________________
/usr/local/lib/python3.7/dist-packages/keras/engine/training.py:1915: UserWarning: `Model.fit_gener
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/40
26/26 [==============================] - 703s 25s/step - loss: 2.8038 - accuracy: 0.5316
Epoch 2/40
26/26 [==============================] - 7s 258ms/step - loss: 0.5767 - accuracy: 0.7436
Epoch 3/40
```

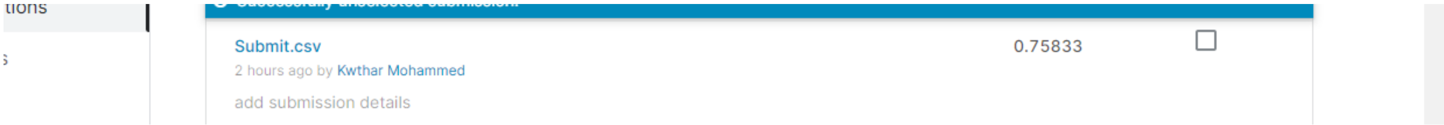
- make prediction:

```
import os
from keras.preprocessing import image
import matplotlib.pyplot as plt
from keras.models import load_model
import pandas as pd

my_img =[]
labels =[]
path =test_path
for i in os.listdir(path):
  my_img.append(i)
  img=image.load_img(path +'//'+i)
  x =image.array_to_img(img)
  x =np.expand_dims(img,axis=0)
  sav= load_model("alexnet.h5")
  out =sav.predict(x)
  print(out)
  if out[0][1]> out[0][0]:
    print("non_autistic")
    label = 0
    labels.append(label)
  else:
    print("autistic")
    label = 1
    labels.append(label)
submit =pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Submit.csv')
submit['Image'] = my_img
print(submit['Image'])
submit['Label'] = labels
submit.to_csv("submit1.csv",index=False)
#print("Done!")
```



```
autistic
[[1.2625878e-04 9.9987376e-01]]
non_autistic
[[0.4993029 0.5006971]]
non_autistic
[[0.00637225 0.9936278 ]]
non_autistic
[[0.28652757 0.7134724 ]]
non_autistic
[[0.9903503  0.00964972]]
autistic
[[0.856893   0.14310701]]
autistic
[[0.81331134 0.18668866]]
autistic
[[0.7843111 0.2156889]]
autistic
[[9.997018e-01 2.981625e-04]]
autistic
[[0.01509029 0.9849097 ]]
non_autistic
[[0.28483897 0.715161  ]]
non_autistic
[[0.5791419  0.42085803]]
autistic
```

Accuracy on Kaggle:



Submit.csv 0.75833 □

2 hours ago by Kwthar Mohammed

add submission details