Report

Model used --------- CNN

WHY?

A number of different models were experimented with, including decision trees and neural networks before arriving at a final Convolutional Neural Network (CNN) model. CNNs work better for image recognition tasks since they are able to capture spacial features of the inputs due to their large number of filters. The proposed model consists of six convolutional layers, two max pooling layers and two fully connected layers. Upon tuning of the various hyperparameters, this model achieved a final accuracy of 0.96

Data preprocessing

In [1]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import to_categorical
from keras.preprocessing import image
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from keras.preprocessing.image import ImageDataGenerator,array_to_img, img_to_array, load_im
g
from keras.utils import to_categorical
from tqdm import tqdm
```

```
Using TensorFlow backend.
```

In [2]:
```
train = pd.read_csv('../input/mydata/train.csv')
train.head()
```

Out[2]:

	image_file	emotion
0	IMG_10000000fn	1
1	IMG_10000000n	1
2	IMG_10000004f	0
3	IMG_10000004fn	0
4	IMG_10000004n	0

Using image generator to increase dataset

In [3]:
```
train_image = []
y=[]
datagen = ImageDataGenerator(
                    rotation_range = 40,
                    shear_range = 0.2,
                    zoom_range = 0.2,
                    horizontal_flip = True,
                    brightness_range = (0.5, 1.5))

for i in tqdm(range(train.shape[0])):
    img = image.load_img('../input/mydata/train/ntrain/'+str(train['image_file'][i])+'.jpg',
target_size=(38,38,3), grayscale=False)
    img = image.img_to_array(img)
    img = img/255.0
    x=img
    train_image.append(img)
    y.append(train['emotion'][i])
    x = x.reshape((1, ) + x.shape)
    j = 0
    for batch in datagen.flow(x, batch_size = 1):
            j += 1
            bat=batch.reshape(38,38,3)
            bat=bat/255.0
            train_image.append(bat)
            y.append(train['emotion'][i])
            if j > 5:
                break
X = np.array(train_image)
y=np.array(y)
y = to_categorical(y)
```

```
100%|██████████| 1941/1941 [00:28<00:00, 68.03it/s]
```

In [4]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1, test_size=0.2)
```

Using CNN Model

In [5]:
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=(38,38,3)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(5, activation='softmax'))
```

In [6]:
```
model.compile(loss='categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])
```

Accuracy for CNN Model

In [7]:
```
model.fit(X_train, y_train, epochs=30, validation_data=(X_test, y_test))
```

```
Train on 10869 samples, validate on 2718 samples
Epoch 1/30
10869/10869 [==============================] - 38s 4ms/step - loss: 1.3755 - accuracy: 0.4121
- val_loss: 1.1824 - val_accuracy: 0.5401
Epoch 2/30
10869/10869 [==============================] - 37s 3ms/step - loss: 1.1152 - accuracy: 0.5498
- val_loss: 0.9890 - val_accuracy: 0.5997
Epoch 3/30
10869/10869 [==============================] - 44s 4ms/step - loss: 0.9303 - accuracy: 0.6336
- val_loss: 0.7877 - val_accuracy: 0.7167
Epoch 4/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.7879 - accuracy: 0.6932
- val_loss: 0.7000 - val_accuracy: 0.7476
Epoch 5/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.6762 - accuracy: 0.7353
- val_loss: 0.6468 - val_accuracy: 0.7539
Epoch 6/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.5942 - accuracy: 0.7728
- val_loss: 0.5981 - val_accuracy: 0.7877
Epoch 7/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.5159 - accuracy: 0.8009
- val_loss: 0.5643 - val_accuracy: 0.7973
Epoch 8/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.4585 - accuracy: 0.8223
- val_loss: 0.5446 - val_accuracy: 0.8072
Epoch 9/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.4105 - accuracy: 0.8364
- val_loss: 0.5314 - val_accuracy: 0.8197
Epoch 10/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.3823 - accuracy: 0.8517
- val_loss: 0.5685 - val_accuracy: 0.8032
Epoch 11/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.3492 - accuracy: 0.8631
- val_loss: 0.5332 - val_accuracy: 0.8297
Epoch 12/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.3046 - accuracy: 0.8807
- val_loss: 0.5396 - val_accuracy: 0.8249
Epoch 13/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.2941 - accuracy: 0.8858
- val_loss: 0.5589 - val_accuracy: 0.8065
Epoch 14/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.2846 - accuracy: 0.8887
- val_loss: 0.5044 - val_accuracy: 0.8385
Epoch 15/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.2687 - accuracy: 0.8929
- val_loss: 0.5427 - val_accuracy: 0.8308
Epoch 16/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.2440 - accuracy: 0.9041
- val_loss: 0.5320 - val_accuracy: 0.8411
Epoch 17/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.2362 - accuracy: 0.9074
- val_loss: 0.5426 - val_accuracy: 0.8385
Epoch 18/30
10869/10869 [==============================] - 39s 4ms/step - loss: 0.2212 - accuracy: 0.9151
- val_loss: 0.5672 - val_accuracy: 0.8337
Epoch 19/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.2195 - accuracy: 0.9143
- val_loss: 0.5469 - val_accuracy: 0.8407
Epoch 20/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.2052 - accuracy: 0.9228
- val_loss: 0.5434 - val_accuracy: 0.8444
Epoch 21/30
10869/10869 [==============================] - 36s 3ms/step - loss: 0.1999 - accuracy: 0.9212
- val_loss: 0.6584 - val_accuracy: 0.8278
Epoch 22/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.1953 - accuracy: 0.9252
- val_loss: 0.5816 - val_accuracy: 0.8473
Epoch 23/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.1894 - accuracy: 0.9271
- val_loss: 0.6175 - val_accuracy: 0.8458
Epoch 24/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.1814 - accuracy: 0.9275
- val_loss: 0.5880 - val_accuracy: 0.8436
Epoch 25/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.1848 - accuracy: 0.9270
- val_loss: 0.6444 - val_accuracy: 0.8322
Epoch 26/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.1759 - accuracy: 0.9333
- val_loss: 0.5663 - val_accuracy: 0.8503
Epoch 27/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.1807 - accuracy: 0.9296
- val_loss: 0.5726 - val_accuracy: 0.8528
Epoch 28/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.1525 - accuracy: 0.9419
- val_loss: 0.5854 - val_accuracy: 0.8514
Epoch 29/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.1581 - accuracy: 0.9419
- val_loss: 0.5905 - val_accuracy: 0.8547
Epoch 30/30
10869/10869 [==============================] - 37s 3ms/step - loss: 0.1567 - accuracy: 0.9407
- val_loss: 0.6454 - val_accuracy: 0.8514
```

Out[7]:
```
<keras.callbacks.callbacks.History at 0x7ff6d851af98>
```

Testing

In [16]:
```
test = pd.read_csv('../input/finaldata/test.csv')
```

In [17]:
```
test_image = []
for i in tqdm(range(test.shape[0])):
    img = image.load_img('../input/finaldata/test/test/'+str(test['image_file'][i])+'.jpg',
target_size=(38,38,3), grayscale=False)
    img = image.img_to_array(img)
    img = img/255.0
    test_image.append(img)
test = np.array(test_image)
```

```
100%|██████████| 552/552 [00:13<00:00, 41.73it/s]
```

Prediction

In [18]:
```
prediction = model.predict_classes(test)
```

In [19]:
```
sample = pd.read_csv('../input/finaldata/submission.csv')
sample['emotion'] = prediction
sample.to_csv('../working/submission.csv', header=True, index=False)
```