HW4

Eyebrow CNN Training

Trenton Ruf

December 11, 2022

1 IMPORTING DATA

The dataset was split into 80% Training and 20% Validation. The only pre-processing done was a normalization of the image pixel values to be between 0 and 1. This pre-processing was done "in model" though. This way when using the model later all I have to do is pass it a raw image and it will normalize it for me.

2 DATA AUGMENTATION

To make up for my small dataset I implemented four common data augmentation layers. The augmentation only applies to the training data, and only while training. These augmentations are a ±2% random rotation, zoom, contrast, and a ±5% random brightness. I did not apply the common image flip augmentation because then it couldn't tell the difference between left or right when only one eyebrow was raised.

3 MODEL

I added three convolutional layers followed by a 2x2 pooling layer. I've read that many computer vision researchers dislike pooling because it decreases the feature space, but implemented it because it should offer a real-time speed advantage for a trade-off in accuracy. I then added a series of dropout layers and dense hidden layers before ending it with a softmax. The output of the convolutions was flattened before the first dense layer to coalesce the features into a 1-D array. I want to experiment more with the number of dense layers and their perceptron count. I will do that during the winter break before next term. As it is the model is far larger than I want it to be, so I will decrease the number of perceptrons and convolutional filters in the future.

4 Training

The model was trained until its validation loss value no longer decreased after 15 epochs in a row. This was done with keras' early stopping feature. Since I don't have a dedicated GPU the training was very slow. It took over 40 hours to train this model. Another reason I want to try decreasing the number and size of the layers.

5 Performance

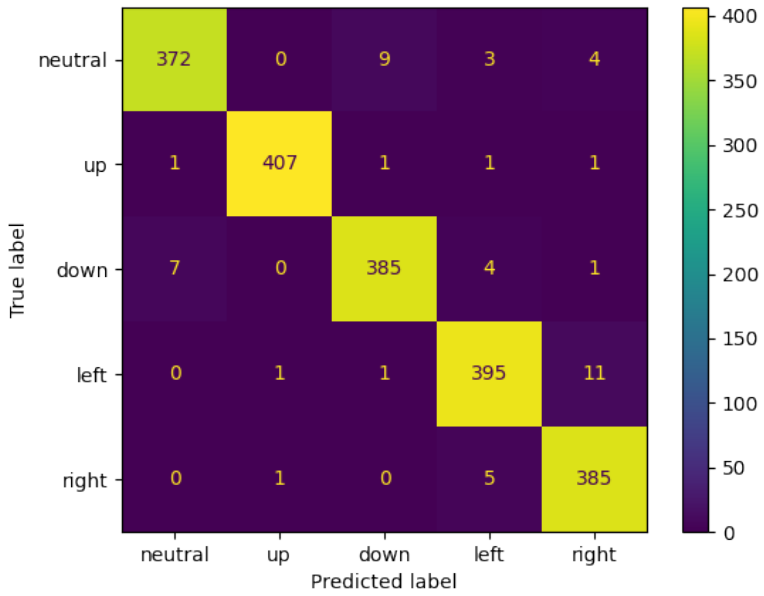

Figure 5.1: Confusion matrix

The model converged around 500 epochs at over 97% accuracy on the validation data. It had the most trouble distinguishing between a couple categories. Neutral and down eyebrows was one. The distance covered by the eyebrows is the least between these two, so maybe that is the reason. The other category pair that had difficulties was left and right. This initially didn't make much sense to me because these should be the two that are the most different from each other since they are opposite. I believe the issue stems from the dataset rather than the model. When looking through the dataset manually I found quite a few pictures of left raised eyebrows that were actually right. I will remake the left and right portion of the dataset and train it again in the future.

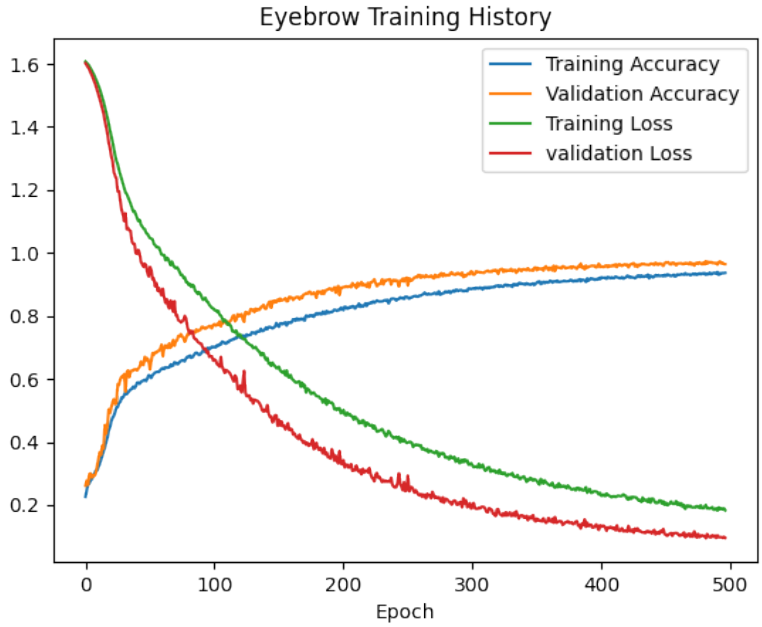

Figure 5.2:

6 FUTURE TESTING

Along with decreasing the size of the model I want to do more accuracy comparisons with varying:

- Data augmentation levels
- Dataset size
- Un-even dataset distributions
- Color - RGB vs Grayscale

7 CODE

All files related to this project can be found at:
`https://github.com/Trenton-Ruf/Intelligent_Robotics`

Listing 1: faceTrain.py

```
#!/usr/bin/env python
import os
# I don't have an NVidia GPU :(
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2' # disable annoying Tensorflow warnings
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Rescaling
from keras.layers import RandomRotation, RandomZoom, RandomBrightness, RandomContrast,
    ↪ RandomCrop

from matplotlib import pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import numpy as np

expressions=['neutral','up','down','left','right']

face_ds = keras.utils.image_dataset_from_directory(
    directory='./dataset',
    labels='inferred',
    class_names=expressions,
    label_mode='categorical', # catagorical for catagorical_crossentropy
    batch_size=32,
    image_size=(50, 100),
    shuffle=True,
    seed=3,
    validation_split=0.2,
    subset="both"
)
train_ds, validation_ds = face_ds

data_augment = Sequential([
    RandomRotation(factor=(0.02), fill_mode="nearest"),
    RandomZoom(height_factor=(0.02),width_factor=(0.02), fill_mode="nearest"),
    RandomContrast(factor=(0.02)),
    RandomBrightness(factor=(0.05))
    #might add more, but not image fliping because it needs to know if right or left
        ↪ eyebrow raised
    ])

data_rescale=Sequential([Rescaling(1./255)])

input_shape=(50, 100, 3) # IMG_SIZE x IMG_SIZE RGB

model = Sequential()
model.add(data_augment) # augmented only during model.fit
model.add(data_rescale) # rescale data in model
model.add(Conv2D(32, kernel_size = (3, 3), input_shape=input_shape,activation='relu'))
model.add(Conv2D(64, (3, 3), activation = 'relu'))
model.add(Conv2D(128, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2,2))) # 2x2 pooling, probably don't need
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024, activation = 'relu'))
```

```
model.add(Dropout(0.15))
model.add(Dense(512, activation = 'relu'))
model.add(Dropout(0.05))
model.add(Dense(5, activation = 'softmax'))

model.compile(
    loss = keras.losses.categorical_crossentropy,
    #optimizer = keras.optimizers.Adam(learning_rate=0.001),
    optimizer = keras.optimizers.Adadelta(learning_rate=0.001),
    metrics = ['accuracy']
)

callback = keras.callbacks.EarlyStopping(
    monitor="val_loss",
    min_delta=0,
    patience=15,
    verbose=0,
    mode="auto",
    baseline=None,
    restore_best_weights=True
)

history = model.fit(
    train_ds,
    validation_data = validation_ds,
    batch_size = 32,
    epochs = 6000,
    verbose = 1,
    shuffle = True,
    callbacks=[callback]
)

#print(history.history.keys())

#  Need to save training graph
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='validation Loss')
plt.title('Eyebrow Training History')
plt.xlabel('Epoch')
#plt.ylabel('Y')
plt.legend(loc='best')  # legend text comes from the plot's label parameter.
plt.savefig("./trainingGraph.png",transparent=True)

#  Might as well put the Confusion Matrix too.
y_test = np.concatenate([y for x, y in validation_ds], axis=0)
pred = model.predict(validation_ds)

plt.clf()
cm = confusion_matrix(np.argmax(y_test, axis=-1), np.argmax(pred, axis=-1) )
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=expressions)
disp.plot()
plt.savefig("./confusionMatrix.png", transparent=True)

model.save('./faceModel')
```