Gun Detection

Bahkari Bridgewater and Zachary Villanueva

The Goal

OR

Moving from CCTV

In light of the dataset you have shown us that you will be evaluating our work on, we have decided to move away from CCTV datasets.

Data Collection Methods

Collection Site for images including guns:

https://chrome.google.com/webstore/detail/imageye-image-downloader/agionbommeaifngbhincahgmoflcikhm?hl=en

ImageEye for Chrome

Download all images (Extention)

https://chrome.google.com/webstore/detail/download-all-images/ifipmflagepipjokmbdecpmjbibjnakm?hl=en

Approach

We plan to use a Convolutional Neural Net with training images from kaggle to identify an image as either contains a gun, or not containing a gun. Will then use python to output a probability score from the trained model describing likelihood (based on our model's evaluation) that the image contains a gun. We would like to ensure that we can also identify with only partial view of the weapon, though that is a stretch goal.

UPDATE:

We found FE methods 1 and 2 to be the most effective in our efforts maximize accuracy

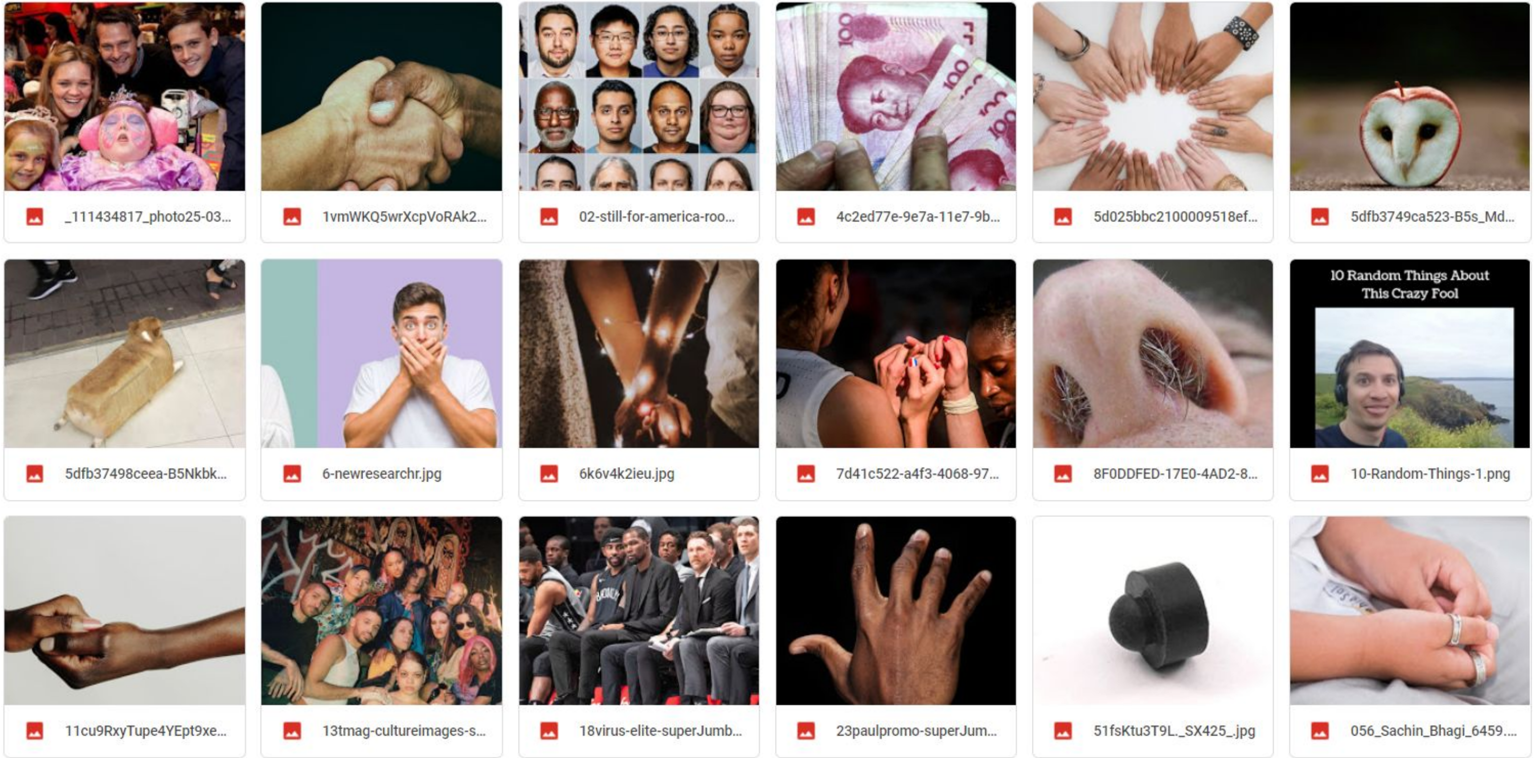
FE 1 and FE 2

FE 1 - First we use the VGG16 pretrained convolutional base to extract features from our unaugmented data set. The weight we used was imagenet. The convolutional base outputs features that can be fed to our classification model.

FE 2 - First the convolutional base is frozen, the classifier is appended to the end, then it is trained. When the classifier has been trained, we use data augmentation when inputting any images to the model.

DATA COLLECTION

Examples (No_Gun)



Examples (Gun)


_87024558_87017927.jpg


_110385184_hi0589149...


5a31ecfd452ef3f26f9ef...


7c7ced609da2364ef501...


7c525ce7c25fbb072c75...


9c807d2fd68fa6fffbde6...


09e92036a7dc88e34301...


41Sy2pT-RDL._AC_.jpg


81L0r445TsL._AC_SX42...

81VtXpu0qQL._SL1500_....


87cf15d33a8bc7a247c7...


143d251670d22fdcd3cb...


500_F_231940812_wOm...


800px_COLOURBOX312...


920x920.jpg


32472ED8-8A08-49F5-9...


101317_ac_soapbox-fire...


170322-london-attack-m...

Roles

Bahkari - Initial dataset collection for training validation and testing sets

Zack - Advanced training set for both training and validation and testing sets

Both- Independent Auditing and pruning of datasets the other member created

Training, validation, diagrams, and commenting

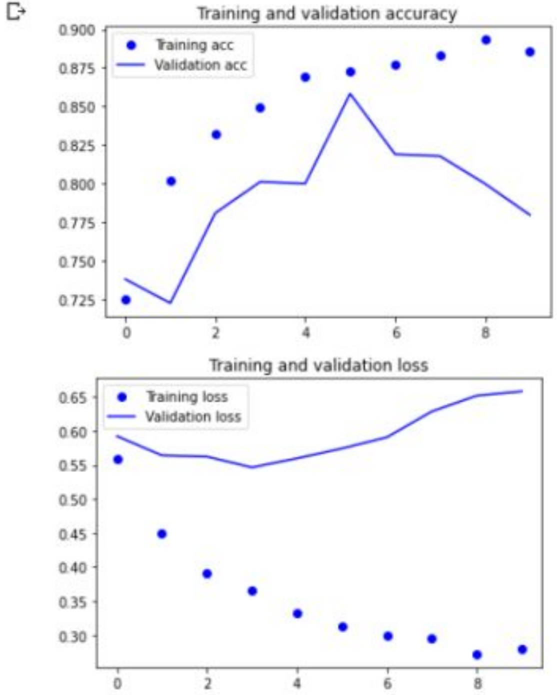
Collaborative Team coding

Challenges

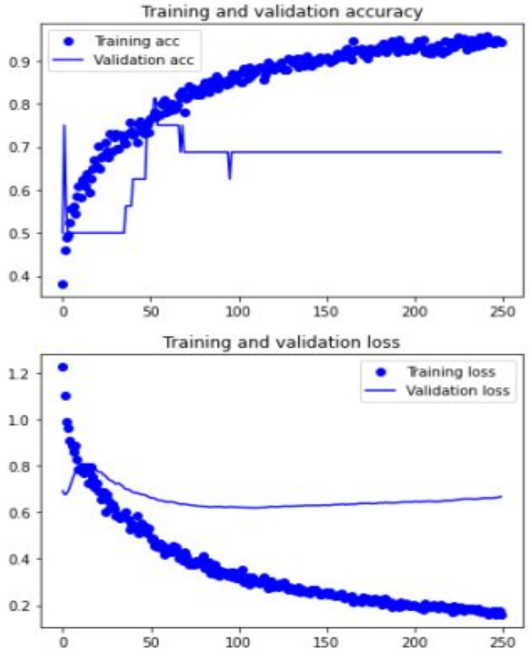
- Overfitting was common in training with more than 100 epochs, independent of training or testing data size (changing the content of the training and testing set had a larger impact)
- Dataset revision was necessary for images containing transparencies or excessive white regions
- Too many hands! Dataset pruning to avoid more overfitting problems

FINE TUNING

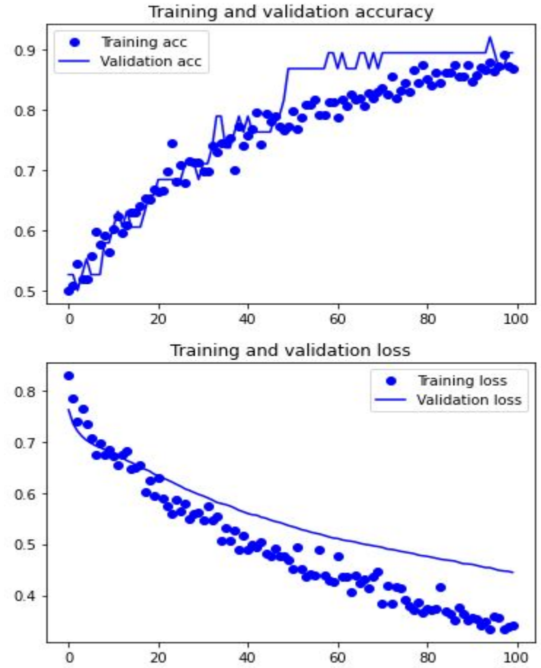
FIRST RUN (FE 2)



ONE DAY FINE TUNING (FE 1)



3 DAYS FINE TUNING (FE 2)



Final Testing Results

```
[ ]  1 #Show final loss and accuracy on unseen dataset of our own
     2 loss, acc = model.evaluate(test_features, test_labels)
     3 print("Using first model, validation accuracy: {:5.2f}%".format(100*acc))
```

```
1/1 [==============================] - 0s 781us/step - loss: 0.4839 - acc: 0.7500
Using first model, validation accuracy: 75.00%
```

```
[ ]  1 #Show final loss and accuracy on unseen dataset of our own
     2 loss, acc = model2.evaluate(t_data_batch,t_labels_batch)
     3 print("Using second model, validation accuracy: {:5.2f}%".format(100*acc))
     4 print("Using second model, validation loss: {:5.2f}".format(loss))
```

```
1/1 [==============================] - 0s 4ms/step - loss: 0.4115 - acc: 0.8500
Using second model, validation accuracy: 85.00%
Using second model, validation loss:  0.41
```

The Following is a comparison between the two FE methods we ecplored above:

FE1: We found this method to train faster and better than the FE2 method. Despite the lack of data augmentation available FE1 aided us in reaching over 75% accuracy after fine tuning and heavy curation of the Testing, training,m and validation datasets.

For the testing on our FE 1 model, we attained **75% accuracy** with .48 loss.

For the testing on our FE 2 model, we attained **85% accuracy** with .41 loss

Questions?