CS4243: Identifying Threatening Images

Preparing the Dataset

Data Cleaning

Skimming through the categorised dataset given, we found many irrelevant photos and removed them. There were also some mislabeled photos and we moved them to the correct categories.

Removed weird/blurry photos

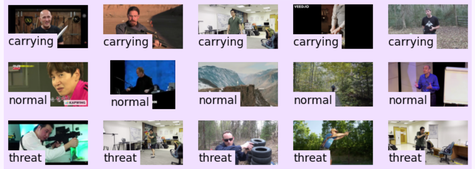
Was catergorised as "carrying". Moved to "threat"

Exploratory Data Analysis / Image Pre-processing

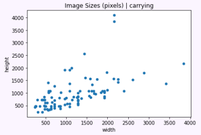
Although we've already looked through every image in the data cleaning process, we randomly sampled a few images from each class to double check

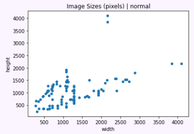
The classes seem to be quite evenly distributed

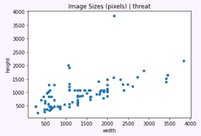
Normal	1672	34.7%
Carrying	1482	30.7%
Threat	1669	34.6%



The images are of many different sizes and resolutions, some with paddings of various colors, most without. In these scatterplots we can see that the majority of images are smaller than 2000x2000. Additionally, we confirmed that all images are **.jpg** files.







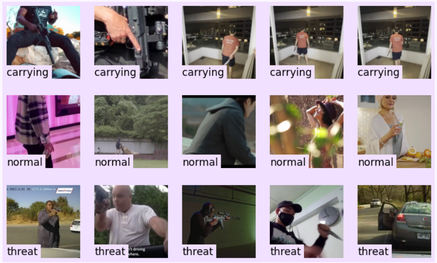
However, for many neural network architectures and machine learning algorithms, the input has to be of the same size. We could do a center crop for all images. However we realised that there could be heavy information loss for some images, with an example on the right.

Thus, we went through the image individually and manually cropped the images ourselves. There were also a few images that couldn't be cropped at all, for these images we added padding instead.

For this image, if we cropped it as shown on the left, we felt that we will be removing crucial pixels conveying information about the person's posture and body language that could be important in differentiating between carrying and threat. For these images, we chose to pad them instead with the images' mean color.

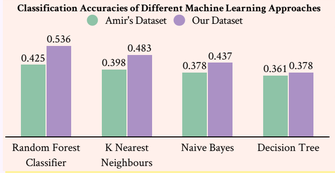
Final Data/Train-Test Split

Lastly, we **normalise**d the images, to speed up the model learning process, and decided on a **85-15** train-test split. Finally, we obtained these normalised images of the same size, ready to be used for machine learning.



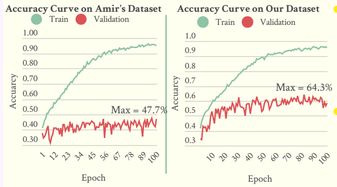
Developing Vanilla Baselines

Baseline Machine Learning Classifiers



- Give us a good idea whether the data is actually learnable
- Serve as **baselines** for neural networks and deep learning

Training a Custom CNN from Scratch

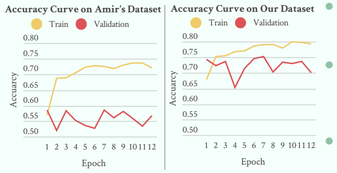


- Architecture:
  A recursion of Conv2D, ReLU, BatchNorm, Max Pooling Layers, followed by a classification layer
- Performance was better than ML baselines but not that great. Time consuming to train from scratch!

Improving the Baselines

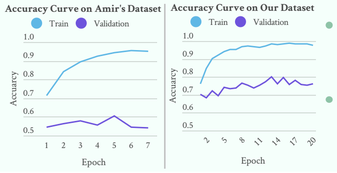
Transfer Learning and Finetuning on InceptionV3 (ImageNet)

Step 1: Transfer Learning



- A classifier with one hidden layer is added atop of InceptionV3
- Added dropout layer (20%), weight regularization, and early stopping to minimize overfitting
- Max validation accuracy achieved after transfer learning: **58.6% (Amir), 75.2% (Ours)**

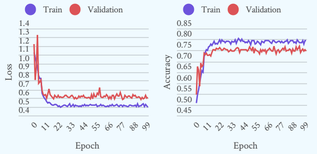
Step 2: Further Finetuning



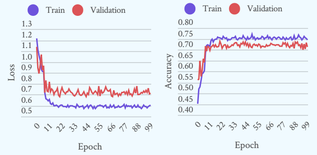
- Unfrozen last few layers in InceptionV3 to further fine-tune them on our dataset
- Max validation accuracy achieved after finetuning: **60.5% (Amir), 80.2% (Ours)**

RESNET18

Training with colored images

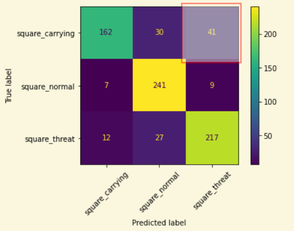


Training with grayscaled images



- Original thought: For the classification task, the pose of the people should matter more, and the color should have less impact
- RESNET 18 is trained by two different datsets, one with colored and on grayscaled, and below are the val accuracy results:
  Colored dataset: 76.4%
  Grayscaled dataset: 70.9%
- Opposite to the original thought, the model trained with the colored images performed better. Colored images still have better features for classification

Other Observations

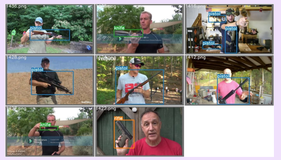


Unsurprisingly, our models have the most difficulty differentiating between non threatening and threatening weapon situations. Our models might not be complex enough to capture the nuances in the posture, expression and finger positions of the human carrying the gun.

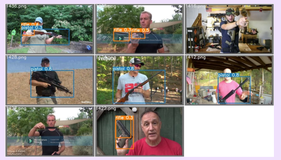
Other Experimentations

Weapon detection with Yolov7

Labeled data



Prediction



(The labels are misarranged, where the rifle in the prediction should be pistol, and the pistol in the prediction should be rifle)

An example for pistol detection

- **Goal**: To focus more on the weapon parts of the image, we also tried to perfrom weopon detections on the dataset.
- **Method**: We labeled a portion of the dataset, and tried to train a YOLOv7 model to detect "knife", "pistol" and "machine gun"
- **Outcome**: The model can detect weopon with a satisfying result, but there are still some problems:
  1. Knifes are less likely to be detected, since there was less data labeled as "knife" while training
  2. "Pistol" maybe predicted in different styles, since in the training data its lower part is sometimes covered by hand
- **Conclusion:** YOLO models may be helpful for the classification, where we can further differentiate an image under "threat" or "carrying" after we have detected some weapons in an image. To achieve better results, we need to label data with more amount and variety for training.