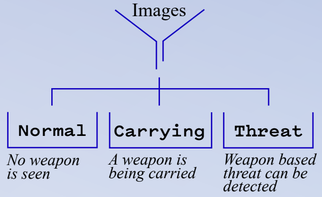
THREAT DETECTION IN IMAGES

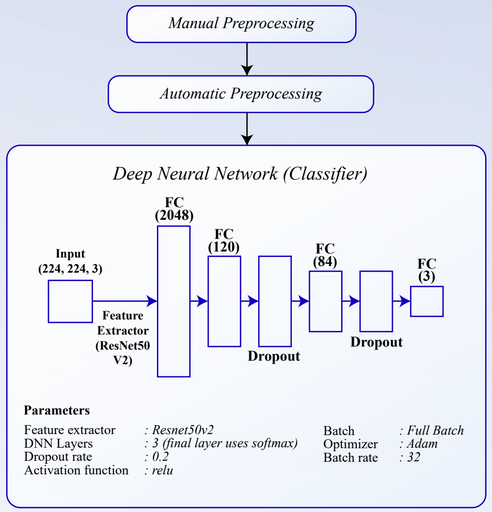
Credits:
SON LINH ▒ Designed and coded image classifier and pipeline/ Trained model/ Compiled results ▌WANYIN ▒ Manual image preprocessing/ Removed poor & misclassified images/ Designed poster ▌YU SHING ▒ Analyzed model results/ Future improvements/ Flowcharts and models

1. Objective

To develop an image classifier capable of detecting possible dangers in images.



2. Flowchart



Parameters

Feature extractor	: *Resnet50v2*	Batch	: *Full Batch*
DNN Layers	: *3 (final layer uses softmax)*	Optimizer	: *Adam*
Dropout rate	: *0.2*	Batch rate	: *32*
Activation function	: *relu*		

3. Preprocessing

`MANUAL` Original dataset contained many falsely labeled images. Hence, manual preprocessing was necessary. We manually removed blurry and misclassified images while ensuring that each class had `1320` images each, and that images fit the class description.

`AUTOMATIC` We created an adaptable, general pipeline that can fit any models consisting of the following steps:

- *Parse images and labels*
- *Create train/test/validation sets with configurable ratio*
- *Resize and rescale to model requirements*
- *Shuffling (Optional)*
- *Perform data augmentation (Optional)*
- *Batching and prefetching (Optional)*

We made use of the pipeline for this project by setting the train/ test/ validation ratio to 80:10:10, image dimensions to 224 x 224 px for ResNet50V2, turning on reproducible shuffling, batching and prefetching. Data augmentation was not employed due to its unpredictable impact on model performance.

4. Model

Our model contains a feature extractor using ResNet50V2 architecture trained on ImageNet dataset and a classification head of interleaved fully connected and dropout layers as depicted in the flowchart to the left. We chose this as our baseline model and fine-tuned it.

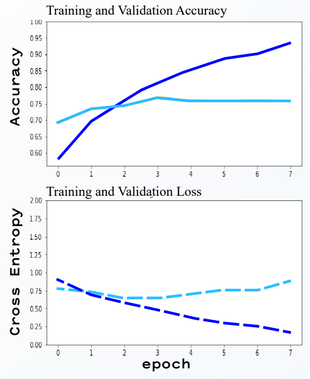
Training Procedure:

- Transfer learning:
  - Train for 10 epochs
  - Optimizer: `Adam`
- Fine tuning:
  - 10 epochs
  - Optimizer: RMSProp with learning rate 1e-5
- Both stages use `SparseCategoricalLoss` and `SparseCategoricalAccuracy` as loss function and metrics
- *Early stopping* is configured for both stages

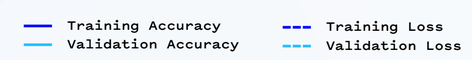
This approach is ideal as our dataset has a total of approximately *4000 images* only, which is insufficient to train a deep model from scratch. Therefore *transfer learning* is a better option, and fine-tuning can improve accuracy by a few percentage points.

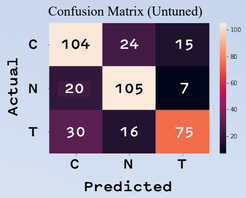
5. Results

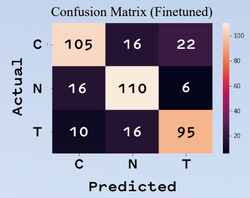
Untuned results



Finetuned results







Final model accuracy

Fine-tuned
`losses` : *0.5828*
`accuracy` : *0.7828*

Untuned
`losses` : *0.6457*
`accuracy` : *0.7172*

	N	C	T
TP	*0.833*	*0.734*	*0.785*
TN	*0.913*	*0.897*	*0.961*
FP	*0.086*	*0.103*	*0.038*
FN	*0.166*	*0.266*	*0.214*

As seen in the loss graphs, both untuned and finetuned models suffer from high degree of overfitting. Finetuning improved accuracy on the test set by nearly 7%.

6. Limitations

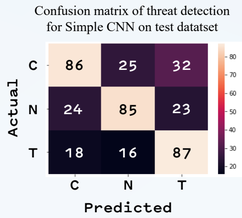
Even our fine-tuned model can only achieve an accuracy of around 78%, which is nowhere near good enough to be deployed.

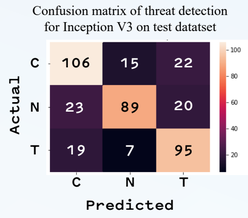
Reasons:

- Data was *insufficient*, *blurry*, and *mislabelled*
- Subtle concept difference between threat/ carrying
- Hard to distinguish between normal/ carrying where weapons are small or concealed carry

To prove this claim, we trained a simple 3-layer CNN and replaced ResNet50V2 with `InceptionV3` feature extractor.

The simple model achieved respectable accuracy, while InceptionV3 got similar results to ResNet50V2 in the untuned version. Therefore, it is likely that low accuracy is due to problems with training data.





Some examples of misclassification by our model:

`Predicted (P) vs Actual (A)`

P: Threat
A: Carrying

P: Normal
A: Carrying

P: Threat
A: Normal

P: Carrying
A: Normal

P: Carrying
A: Threat

P: Normal
A: Threat

While our model classified some images wrong, there are images that can be put in any of the 3 classes, proving the point on subtlety.

7. Future Improvements

I. `Increase training datasets to discover more fringe cases`

The sample size we have may be insufficient to help the model discover the full range of poses a human posing a threat would have in an image.
[Drawback] This might lead to further over-fitting resulting in a model that is unable to discover the patterns that differentiate the three classes.

II. `Video input classifier instead of image input classifier`

A temporal dimension will allow the neural network to detect motion thus help better identify threats. We can use recurrent neural networks such as LSTM and GRU.
[Drawback] Recurrent neural networks are more computationally taxing.

III. `Ensemble approach`

Instead of only training one model, we train multiple models. Then we take the weighted average result of all the models to classify the images. As such, if one model classified an image wrongly, the other models will rectify the mistake.
[Drawback] Training and processing time will be increased due to multiple models.