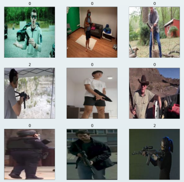
A Multi-Class Classifier for Weapon Threats Detection in Images

Cao Peng (collect dataset, run code, write poster), Cao Qin (collect dataset, run code, write poster),
Fang Junyuan (dataset, write majority of poster, design, code and train machine learning mode), Qin Guorui (collect and review most of dataset, write majority of poster, design, code and train machine learning model)

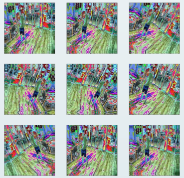
Dataset

Data collection
For each of the 3 categories mentioned above, we collected and labelled **500** images manaully, which are actually screenshots taken from a collection of videos. We also spent a huge amount of efforts to replace duplicate images or obscure images in order to ensure that each image is clear with only a single human and as little background as possible.

Data pre-processing
We created the training and validation batched dataset (batch size 16) using tensorflow inbuilt preprocessing functions, and resized every image to a fixed size of 256 * 256. Then, we applied data augmentation techniques to training dataset, including random flip, rotation, and zoom, to obtain a larger and more diverse data set. Lastly, normallization is applied to both training and validation sets with appropriate mean and variance values.

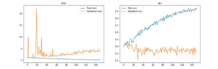


[examples of collected imges]

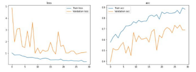


[examples of augmented images]

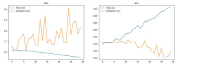
Baseline without batch normalization



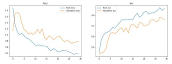
Baseline with batch normalization



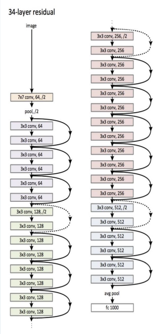
Improved model without batch normalization



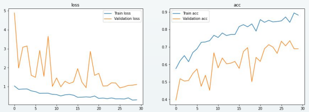
Improved model(drop out 0.5) with batch normalization



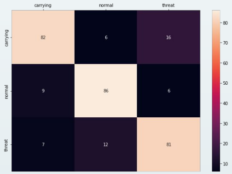
Baseline Model(ResNet with Batch normalization+dropout)



Learning Curve of Baseline Model



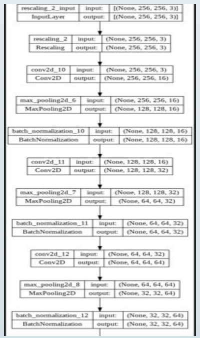
Confusion Matrix of Baseline Model

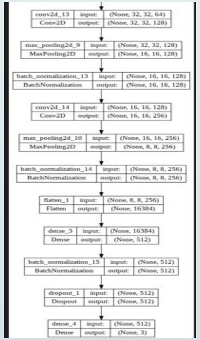


$Accuracy_{overall} = 81.6\%$

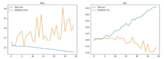
Improved Model

To fit our small dataset, we applied smaller number of convalutional layers.

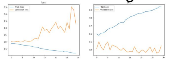




Improved model without batch normalization



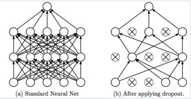
Improved model without batch normalization but augmentation

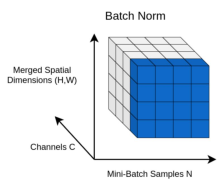


To Prevent Overfitting

Because the dataset is rather small, overfitting is the major challenge for us in the training process.

Dropout
We apply drop outbetween the last 2 dense layers to prevent all neurons synchronously optimize their weights to reduce overfitting.

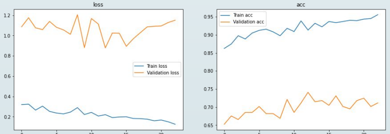




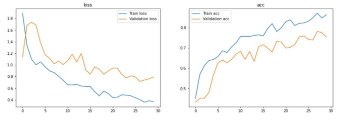
Batch Normalization
We apply batch normalization after each max pooling layer in turn to reduce the dependence on the gradients on the scale of parameters or the initial values.(increase level of generalization, improve the accuracy and reduce overfitting)

Performance

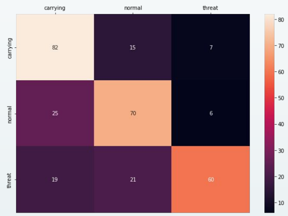
Learning Curve of Improved Model(drop out 0.1)



Learning Curve of Improved Model(drop out 0.5)

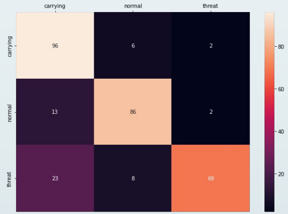


Confusion Matrix of improved model, drop out 0.1



$Accuracy_{normal} = 78.8\%$ $Accuracy_{carrying} = 60.0\ \%$
$Accuracy_{threat} = 69.3\%$ $Accuracy_{overall} = 69.5\%$

Confusion Matrix of improved model, drop out 0.5 (Final model)



$Accuracy_{normal} = 85.1\%$ $Accuracy_{carrying} = 92.3\ \%$
$Accuracy_{threat} = 69.0\%$ $Accuracy_{overall} = 82.3\% > 81.6\%$

Conclusion

The main focus of our model designing is about reducing the chance of overfitting, given the limited datasets (1500 images for 3 classes), compared with 50000 images for cifar-10, which are clearly augmented and grouped.

Our method works well, because the number of convalutional layers is appropriately chosen, especially, it is not over expressive. Also, the use of techniques after each convalutional layer and between layers with many weights to reduce overfitting is effective. Both reasons leads to a more successful model than the baseline model.