EEEM066 Fundamentals of Machine Learning
Coursework Report (Autumn 2023)
Knife Classification in real-world images

Rohit Krishnan, URN: 6839323, r.00088@surrey.ac.uk

Abstract

Increasing crime rate in the UK has made it a necessity to develop a robust AI weapon analysis system to help identify knives. This report looks into techniques used to design, train and evaluate various Deep Neural Networks for a fine-grained real-world classification problem.

Experiments are performed on Nvidia GeForce GTX1650 Ti GPU to achieve faster and exhaustive training on a image dataset of 10279 knife images spanning over 192 classes. The highest accuracy level achieved for knife classification is 84.41%.

1. Introduction

Nowadays, most public places have CCTV cameras for monitoring and preventing crime. However, security staff cannot pay attention to multiple video feeds at the same time and provide reliable monitoring. A performant and robust knife classification model can be used on multiple video feeds to reliably detect knifes and alert the respective authorities.

This report explores the use of CNNs (Convolutional Neural Networks) [1] for classification. Convolutional Neural Networks may comprise of several convolutional, pooling and fully connected layers. Convolutional layers are extremely effective at extracting features from multi-dimensional data.

2. Exploration of Hyper-parameters

Hyper-parameters are parameters that are set before a model is trained. They are not learned from the data and have great influence over the behaviour of the network. This report experiments with various hyper-parameters using a custom Convolutional Neural Network.

2.1. Model Architecture

The custom CNN built for this report is an example of a very minimal convolutional neural network. Figure 1 visualizes the various layers of the model.

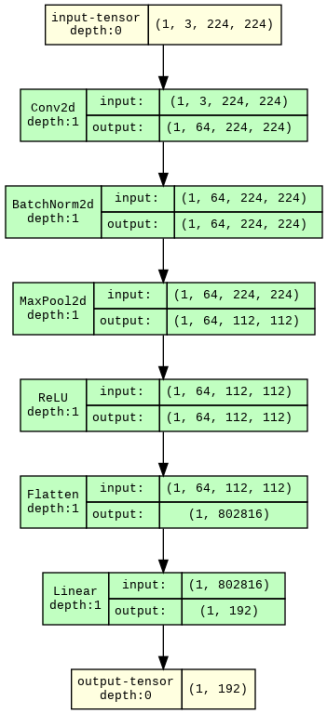

Figure 1. Custom CNN network used to experiment with hyper-parameters

It has one Conv2d layer followed by a BatchNorm2d Layer that is responsible for normalizing the inputs for the next layer. This is a very important step as it prevents the values from scaling to infinity/zero. After the BatchNorm2d layer, the MaxPool2d layer performs a downsampling on the input tensor. This is done to reduce the spatial dimensions while still retaining important features of the image. An activation function, ReLU is used on the output of the MaxPool2d layer. This is done to introduce non-linearity to the model. The outputs of the ReLU activation function are then flattened to be used as the inputs to the fully-connected layer. The fully-connected layer accumulates the features that were gathered in the previous layers and reduces the

dimensions to a desirable size. In this case, it reduces the features to 192 outputs, each of which represents a class of knife.

2.2. Learning Rate and Schedulers

The learning rate is a hyper-parameter that controls the amount of optimization done during the training phase. It is a scalar value that controls how much the optimizers can change the model's weights. Figure 2 shows the affect of varying the learning rate on the mAP of the custom CNN.

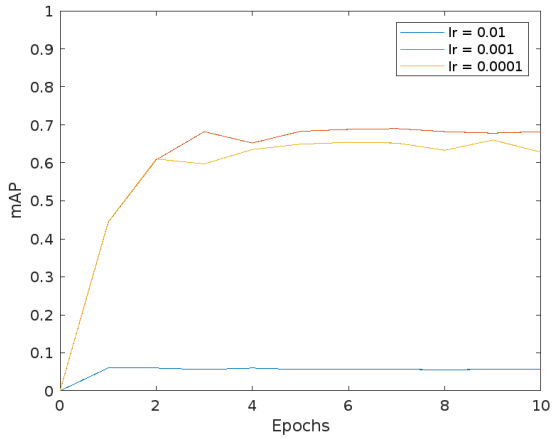

Figure 2. Varying learning rate without a scheduler

TODO: conclusions on lr w/o scheduler

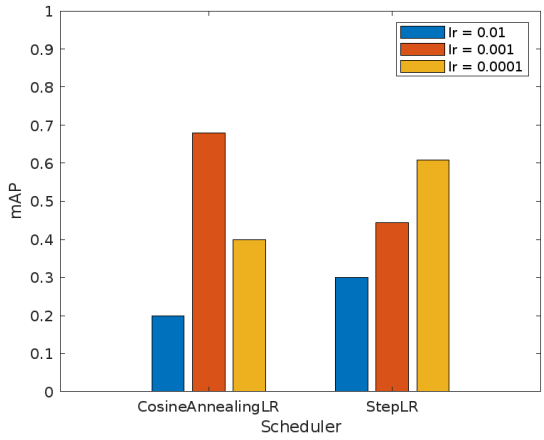

Figure 3. Best mAP values for different learning rates using CosineAnnealingLR and StepLR

2.3. Optimizer

Optimizers are algorithms that are used to modify the model parameters and minimize training loss. It helps in finding the set of parameters that yields the best results on the training data. SGD, Adam and RMSprop are few of the popular algorithms. Try different optimizers (e.g., SGD, Adam, RMSprop) and observe their effects on convergence.

2.4. Learning Rate Schedulers

Implement schedulers like ReduceLROnPlateau or Cyclical Learning Rates to dynamically adjust the learning rate based on model performance.

2.5. Number of Epochs

Experiment with the number of training epochs to prevent underfitting or overfitting.

2.6. Data Augmentation

Explore various data augmentation techniques (e.g., rotation, flipping, scaling) to improve model robustness.

3. Deep model architectures

4. Conclusion

References

[1] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015. 1