NOVA
IMS
2021

DEEP LEARNING – GROUP PROJECT REPORT

THREAT TYPE DETECTOR - BINARY IMAGE CLASSIFICATION WITH CNNS

GROUP 15

OGUZ KOKES - 20201006

SERDAR CETINER – 20201016

BERFIN SAKALLIOGLU – 20200545

1. Introduction

Crime detection is a popular and controversial subject that has always been a source of discussion. In George Orwell's 1984 "Big Brother is watching you", in 2002 movie "Minority Report" the illegal activities are stopped even before they are committed as technology allows computers to "previsualize" crimes, and the list can go on and on. However, advancements in Machine Learning and Deep Learning in recent years made this topic more relevant and plausible than ever before. As Deep Learning models are now able to analyse and interpret video and image data very rapidly and precisely, and there are various projects and attempts to make this phenomenon a reality.

In this project, the main objective is to create a CNN model using Keras API that can successfully classify Knife vs Pistol images to determine the threat type of a crime taking place.

2. Project Objective & Scope

The project scope was quite dynamic as it was shaped and changed throughout the project through research, trial, and error. The initial scope of the project was pistol "detection", rather than classification. In this more ambitious and broader version of the project, the way of thinking was to treat pistol images as one class and use random images as the "other" class which then was fed to the CNN models. However, analysing model performance proved that this aim was a bit above the groups' current knowledge on CNNs, as the "other" class prediction accuracies were lower than expected, and the overall model performance never increased over 73% validation accuracy. Studies on this subject indicate that using Faster R-CNN models might be a better model type for successful detection.

Another discussion topic was to whether or not use other "weapons" for multiclass classification, however this was also dropped due to time constraints and the models required a very long time over large number of epochs for training.

With these trials, the main objective was changed to binary image classification with high accuracy to correctly identify the weapon (knife vs pistol) of an offender.

3. Project Preparation

Data Source

As the project scope and possible objectives were decided, a large and extensive data source was required. To this end, images of guns, knives and similar hand-held objects were gathered from the database published by the Andalusian Research Institute in Data Science and Computational Intelligence, "**Weapons detection for security and video surveillance"**.

This dataset was used as it consists of handheld and standalone shots of guns, knives, long barrels and similar objects as in cards, keys and even contextually unrelated objects as leopards and aeroplanes. that can be used with all possible project scopes.

7,700 pistol and 5,062 knife images were taken directly from this public dataset and was merged into the “raw” images folder. Images in different directories were named the same, to be complied each image was renamed with random numbers. These raw images were then split into three groups, train, test and validation. As the number of images for each class was not equal, 5062 of the pistol images were used during the project to prevent bias and equalling the amount of data for both classes.

```
|-- data
|   |-- raw
|   |   |-- knives
|   |   |-- others
|   |   |-- pistols
|   |-- cooked
|   |   |-- test
|   |   |   |--knives
|   |   |   |--pistols
|   |   |-- train
|   |   |   |--knives
|   |   |   |--pistols
|   |   |-- validation
|   |   |   |--knives
|   |   |   |--pistols
```

Project Evaluation

The topic at hand is quite delicate and required upmost precision to correctly inform law enforcement about the situation, we decided to use validation accuracy as our metric for model selection & evaluation.

Project Roadmap

As the final aim of the project is to submit a single accurate CNN binary image classification model, it requires different trials to achieve a final best-performing model. To better understand the effect of any potential changes, we decided to create a baseline model with acceptable (70-80%) accuracy, which then can be improved by checking and comparing other models online, changing model architecture, layer sizes & numbers, and parameter tuning.

As the project dataset was quite large in size and models required constant modifications and trials, the project was set up on a GitHub repository. This repository contains all the necessary files, notebooks used during trial, and the “final_notebook” which contains the code & topics discussed in this report.

We decided not to use Keras Tuners to follow a more manual (and admittedly slower) but in-depth approach that would give us a better understanding of how models and parameters operate.

Defined Functions

Several functions were defined and used throughout the project to simplify the process of dividing, loading images, and creating & evaluating models. These functions can be found in the .py files inside the modules folder.

`cleaner():` for cleaning “cooked” folders for re-loading or re-splitting images to the directory

`sorter():` splits the cooked images to train test valid folders based on the ratio specified

`loader():` for simplifying ImageDataGenerator() process, with augmentation argument allowing quick and easy changes if necessary

`sample_print():` for printing & checking augmented images from loader

`create_model():` for simplifying the model creation process, this function takes the Conv2D layer sizes as a list and other parameters & model architecture choices can be specified as arguments.

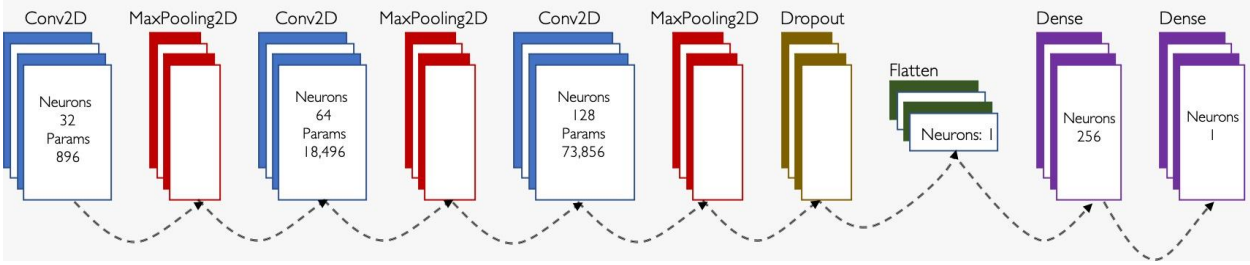
`model_fitter():` for training the model using the specified datasets, number of epochs

`pred_perf():` for evaluating model performance, it prints out the confusion matrix and classification report for the given test_set, as well as returning the prediction results

`hist_acc():` to easily print out loss and accuracy plots for different models

4. Model Trials

Baseline Model



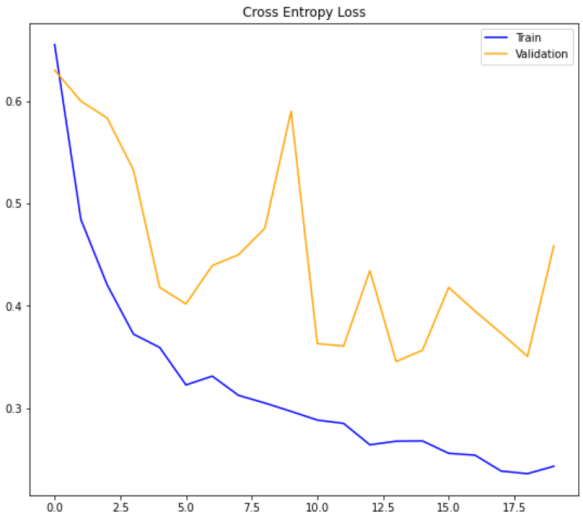
The model summary for the baseline model can be found above. The model is composed of 3 Conv2D layers, each followed by MaxPooling to reduce dimensionality. In various models on the internet, these MaxPooling layers either been used after every Conv layer or after every couple of Conv layers depending on the model. For this model, early testing suggested that using a MaxPooling layer improved model performance and was used in all the following models.

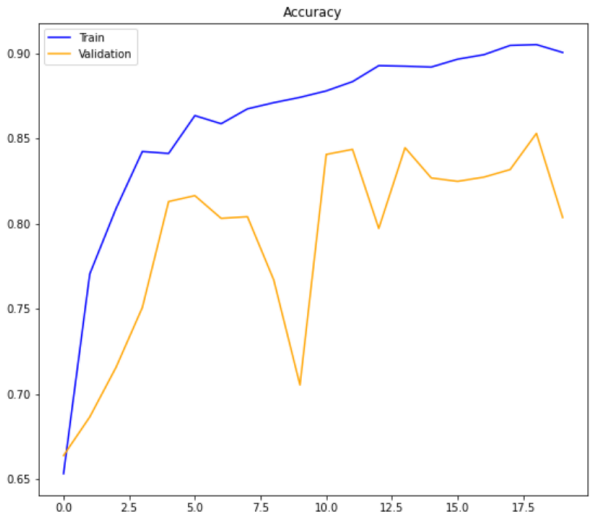
These layers are followed by a Dropout layer with 0.3 rate. This layer was added as Dropout layers improve generalization by randomly setting neuron weights to zero, in a way forcing the model learn from more robust features and not overfit to the training set.

The Dropout layer outputs are then passed through a Flatten layer, which is followed by a Dense layer with 256 neurons and a final Dense layer with a single “sigmoid” activated neuron to output class predictions.

The baseline model was also used for determining image (150x150 or 200x200) and kernel size (2x2 or 3x3). These parameters established earlier in the trial process because testing them out in later stages would need different train/test/valid sets as well as models with different input

shapes. As the later models will be built upon this model, we decided to go with 200x200 image size with 3x3 kernels as this was the bst performing combination.





The baseline model's prediction accuracy over 20 epochs can be found above. The model is performing above initial expectations of 70-80% accuracy as the prediction set predicted with 81%. This was both pleasing and challenging as it meant that we would be able to submit a fairly successful model, but also meant that we had to focus on how to improve the model to actually show our efforts.

Dropout

As explained above, Dropout was the first regularization technique utilized in the project as it is a very common method in Deep Learning models. In the earlier stages Dropout layer did not seem to have a significant impact on the model performance. During later stages of testing, this layer was tried again, and dropped as the model accuracies showed a slight improvement when excluding it. Model performances with and without Dense layer can be seen below.

```
Epoch 00015: val_accuracy did not improve from 0.83753
190/190 [==============================] - 149s 786ms/step - loss: 0.2560 - accuracy: 0.8966 - val_loss: 0.3841 - val_accuracy: 0.8326
Epoch 00015: early stopping
```

```
Epoch 00017: val_accuracy did not improve from 0.88049
190/190 [==============================] - 149s 782ms/step - loss: 0.2286 - accuracy: 0.9096 - val_loss: 0.3419 - val_accuracy: 0.8430
Epoch 00017: early stopping
```

Number of Neurons & Layers

It is expected that the increases in number of neurons and number of layers will increase the models ability to learn. As simply stated by Head of Montreal Institute for Learning Algorithms, Professor Bengio, the number of neurons and layers were added until the validation score stopped improving. 4 Conv2D layers were added until model stopped improving. The model was able to reach 84% validation accuracy, showing slight improvement compared to the base model.

Kernel Initializer

By default Keras Conv2D layers uses “glorot_uniform” to initialize kernel weights. As several forums and discussions suggesting using “he_normal” or “he_uniform”, these initializers were also tested. Although there were no difference between the two, both were actually improvements over the default initializer and thus were added to the model architecture.

Kernel Regularizer

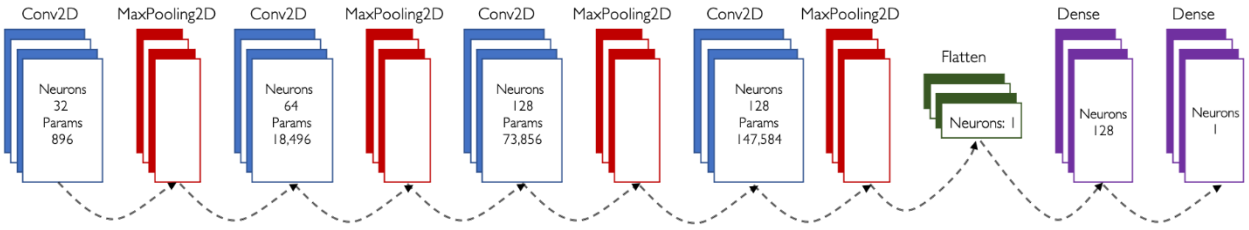
As Dropout layer was removed from the model, other regularization methods were also tested. From previous examples and lessons, L2 regularization seemed like a potential improvement method. However, testing proved that this method penalized the model performance even with 0.01 regularization and model performance never increased over 80%.

```
Epoch 00018: val_accuracy did not improve from 0.80691
190/190 [==============================] - 147s 771ms/step - loss: 0.4513 - accuracy: 0.8559 - val_loss: 0.6445 - val_accuracy: 0.7551
Epoch 00018: early stopping
```

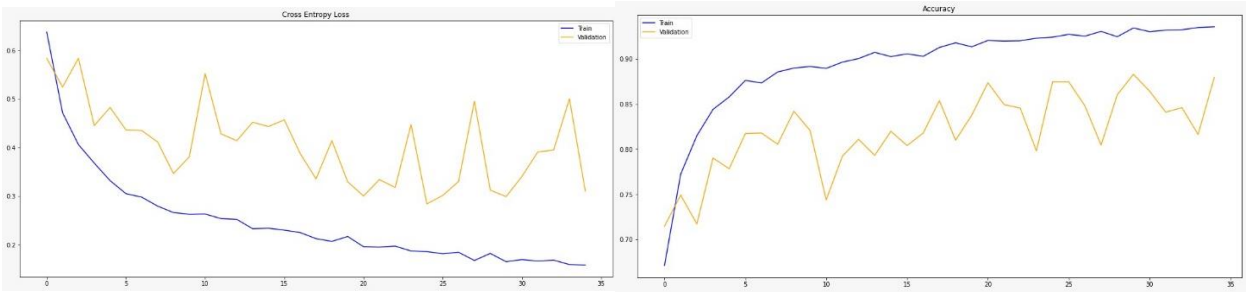
Callbacks

As more model architectures and parameters were tested, we required to automatically stop unfruitful training sessions and save possible best models for later use. To do this two callbacks, EarlyStopping with 10 patience (over 50 epochs) and ModelCheckpoint were used.

Final Model



The final and best performing model (best_model.hdf5) architecture can be seen above. The model utilizes all performance improving tunings mentioned earlier and was trained over 50 epochs. Model performance throughout the training can be seen below.



The model achieves 88% validation accuracy during training and was stopped after 35 epochs.

```
Classification Report
              precision    recall  f1-score   support

      Knives       0.93      0.83      0.88      1013
     Pistols       0.85      0.93      0.89      1012

    accuracy                           0.88      2025
   macro avg       0.89      0.88      0.88      2025
weighted avg       0.89      0.88      0.88      2025
```

The test set prediction also gives out the same score for accuracy. When the report is analysed it is seen that the model is more successful with classifying knife images (with 93% precision).

5. Conclusion & Discussion

In this project the main aim was to train and submit a highly accurate binary image classifier that can be used as a threat type detector (knives vs. pistols) for law enforcement purposes. Various models and methods were used to improve performance and the resulting model has an 88% classification accuracy. This score is actually over minimum threshold of 85% that we set for ourselves when starting the project, but still not as higher as some other models on the internet and not over the 90% threshold we were aiming for.

One of the main constraints faced during this project was the project scope. As we had no prior Keras experience beforehand, setting up a realistic target proved to be harder than expected. In the earlier stages, we had many trials with the aim of pistol detection which was ultimately dropped. However, this meant re-evaluating the training data and the model architecture which gave us less time than before.

Another constraint was the training times and of the models. To get a reliable understanding of each model trial, we had to use longer epoch times (preferably 20 epochs) which meant a training period of 2.5 hours. We tried to use Google Cloud Services to run the notebook on the cloud, but were not able to as we had the images locally and couldn't successfully export them to the Google platform.

The model performance can definitely be improved with more trials and tuning. One of the most anticipated improvements we had in mind was to use batch normalization to observe and understand its affect. This was initially scraped as the model included a Dropout layer and several studies and articles suggest that using the two methods together may cause disharmony between the two and result in a worse performing model. However, as the Dropout layer was dropped later in the model trials, using Batch Normalization on the current model may potentially increase model performance. This was disappointingly not tested due to time constraints.

6. References

1. Soft Computing and Intelligent Systems research group, https://sci2s.ugr.es/weapons-detection
2. DasCI Weapon Detection Open Data https://dasci.es/transferencia/open-data/24705/
3. Weapon Detection Open Repository https://github.com/ari-dasci/OD-WeaponDetection
4. Bengio, Yoshia; answer to “Artificial Neural Networks: How can I estimate the number of neurons and layers?”, Quora.com, https://bit.ly/39GuVwa
5. os Miscellaneous operating system interfaces documentation, https://docs.python.org/3/library/os.html
6. Keras Image Preprocessing documentation https:/keras.io/api/preprocessing/image
7. Brownlee, Jason; How to Configure Image Data Augmentation in Keras https://bit.ly/2PYJdkp
8. Keras Initializers documentation, https://www.tensorflow.org/api_docs/python/tf/keras/initializers
9. Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun; Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks https://arxiv.org/abs/1506.01497
10. Ananth, Shilpa; Faster R-CNN for object detection https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46-