CSCU9M6 Aerial Car Recognition

Cameron Morrison
Stirling University
Student number 2831609

I. Introduction

The proposed task is a binary classification problem solved with a PyTorch implementation that classifies satellite images containing cars. The data set was comprised based on the suggested M2 implementation, meaning the created data set from multiple large-area images of two cities, Edinburgh and Stirling captured using Google earth pro [1]. The model will identify which images contain cars and which images don't contain cars for traffic monitoring.

II. Proposed Solution with Justifications

Using aerial imaging, the privacy risks associated with data collection such as faces and car registration numbers are mitigated as these features are not distinguishable from an orbital view. Large area images of Edinburgh and Stirling were captured in high resolution and split into small area tiles using IrfanView [2], a graphic viewer software. The selected images were then manually selected and placed within one of two folders to represent positive and negative samples of images containing cars. The Edinburgh data set contains 520 tile images, and Stirling contains 490. The two cities have drastically different appearances to the images, Edinburgh appears high quality, clear and vibrant whilst Stirling's images are dull and noisy. The proposed data loader implementation uses PyTorch to create a train and test loader for each city, using data augmentation the size of the data set was artificially increased. The training and testing data loaders, load in all the images from the same directory, however, a transform performs data augmentation on the train loader so no image is the same as the original images in the testing set. This results in a 50:50 split, where each model has double the images due to the training loader being comprised of rotated, saturated and noisy transformed copies. Neural networks can benefit in classification accuracy strength when trained on more diverse data, typically the addition of noise reduces over-fitting [3]. Both data loaders are additionally normalized using the calculated standard deviation and mean gained from a custom method implemented to loop through all the images within each directory and find the optimal values per city as there is a drastic change in the two sets of images. The proposed solution is a ResNet-18 convolutional neural network. A ResNet-18 model is an 18-layer deep neural network and is known to be one of the most efficient neural networks due to their low error rate and ability to classify images in up to one thousand categories. ResNet-18 networks can also be trained quickly due to skipping connections which allows information to pass through the network without going through every layer. This can prevent the occurrence of the vanishing gradient problem. This is an issue where weights are updated too slowly, causing stagnation within the learning function and ultimately resulting in poor learning performance. The model is trained using NVIDIA CUDA for faster computation on devices with a graphics processing unit. The batch size of the model input is set to the default 32 images and the initial learning rate is set to 0.01, momentum is set to 0.9 and weight decay is 0.3. These values were chosen as PyTorch's tutorials [4] highlighted that they are good starting points. For each epoch, the parameters are updated based on the optimizer and loss function, becoming smarter when the optimizer updates. The benefit of data augmentation is that the optimizer will be able to explore a broader search space, which helps the model to be more reliable when seeing new images. Finally, when the testing and training accuracy reaches 100% or the predetermined amount of epoch iterations runs out, the algorithm is stopped and the model is saved for later use.

III. Results

The models are trained and evaluated with visualisations providing output statements, confusion matrices, scoring metrics and graphs for accuracy and loss to help evaluate the effectiveness of each model. Both cities, Edinburgh [Fig. 2] and Stirling [Fig. 4] scored 100% accuracy, and 1.0 for recall, precision and F1 scores. It took Edinburgh 148 epochs [Fig. 1], and Stirling 97 [Fig. 3]. The epoch size varies per run, however, whilst the epoch numbers sound high it is important to remember that having a smaller batch size means the number of epochs has to be greater (currently the batch size is 32). The two trained models are then evaluated using the other city's testing loader. The Edinburgh model classified 423/490 of Stirling's images correctly (86.327%) with an epoch loss of 3.8156 [Fig. 5], whereas the Stirling model classified 437/520 of Edinburgh's images correctly (84.038%) with an epoch loss of 4.4117 [Fig. 6].

IV. Discussion

The results of the Edinburgh [Fig. 2] and the Stirling [Fig. 4] models were impressive, however, when switching the testing data sets, Edinburgh testing data on the Stirling model [Fig. 6] performs 2.6872% worse. This could in part be due to the Stirling model being trained on a 5.94% smaller image data set or the dissimilarities in the landscape.

V. Conclusion

Both Stirling and Edinburgh ResNet-18 models scored highly even when tested using switched testing data sets due to ResNet-18's impressive classification capabilities and the diverse benefits gained from artificially improving diversity by utilizing data augmentation within the data loader implementation.

References

[1] Google Earth Pro satellite imagery [Online]. Available: https://www.google.com/intl/en_uk/earth/versions/earth-pro

[2] IrfanView image viewer [Online]. Available: https://www.irfanview.com/

[3] Train Neural Networks With Noise to Reduce Overfitting By Jason Brownlee [Online]. Available: https://machinelearningmastery.com/train-neural-networks-with-noise-to-reduce-overfitting/

[4] PyTorch tutorial on how to adjust learning rate [Online]. Available: https://pytorch.org/docs/stable/optim.htmlhow-to-adjust-learning-rate

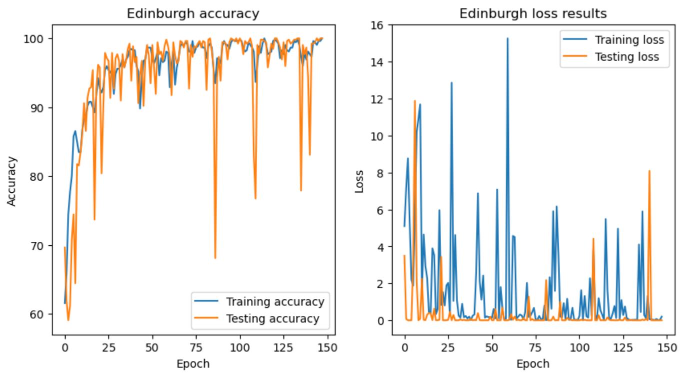

Fig. 1. Edinburgh accuracy (left) and loss (right) visualised. Blue represents training and orange for testing.

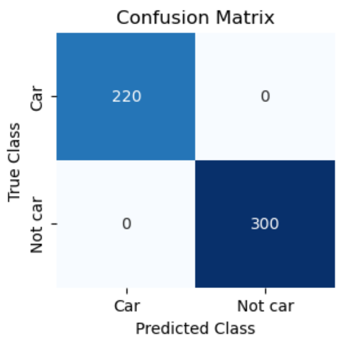

Fig. 2. Edinburgh confusion matrix showing 100% accuracy for correct classifications.

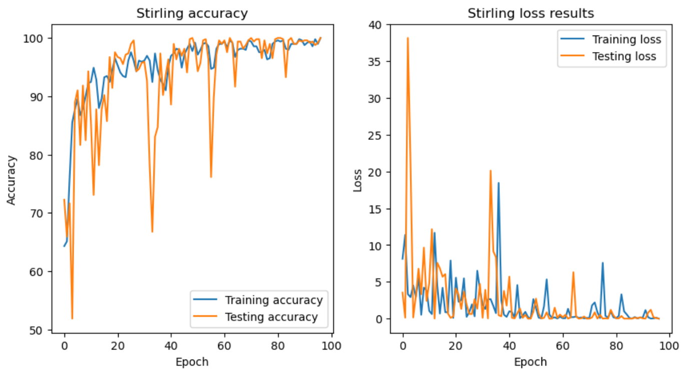

Fig. 3. Stirling accuracy (left) and loss (right) visualised. Blue represents training and orange for testing.

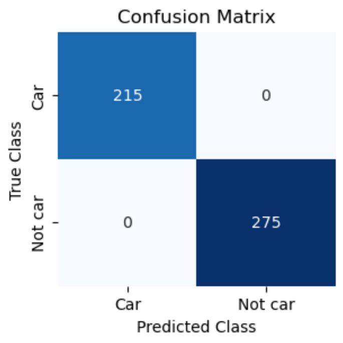

Fig. 4. Stirling confusion matrix showing 100% accuracy for correct classifications.

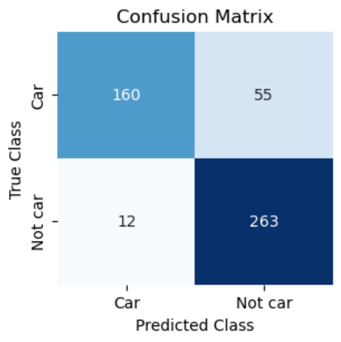

Fig. 5. Stirling test data using the Edinburgh model scoring 86.327% accuracy.

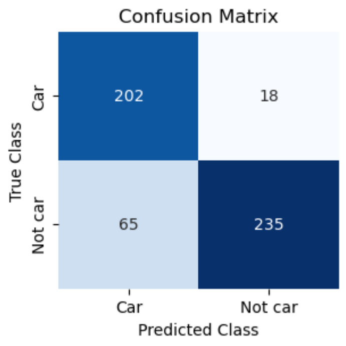

Fig. 6. Edinburgh test data using the Stirling model scoring 84.038% accuracy.