Artificial Intelligence COM2028: Butterfly Classification Coursework

Max Carter • 6567107

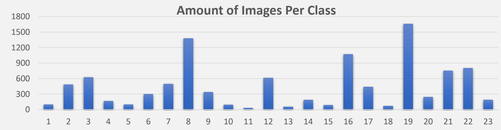
Introduction

Objective
To produce an image classifier for 23 butterfly species through machine learning techniques.

Requirements
The model must have a minimum accuracy of 65%. Optionally, to achieve maximum marks the accuracy must be in the highest tier (tier 10).

Aim
My personal aim for this project was to create an image classifier capable of achieving greater than 90% accuracy which was ambitious and challenging.

Background Research
This type of image classification problem requires a convolutional neural network (CNN) to identify features in the butterflies to identify the species they belong to. A CNN is a specialised neural network model which can be used for 3D images and consists of many different layers and filters. Layers that operate on the raw pixel input data will learn to extract low level features such as lines, whereas deeper layers extract features which are a combination of the lower-level features, such as shapes. Eventually, the deepest layers are able to extract features such as wings or patterns on a butterfly.
Class labels are provided in the training data which means this is a supervised learning problem. Supervised learning uses labelled datasets to train algorithms and accurately predict outcomes.
Since this project has more than two classes, it is known as a multiclass classification, and since there is skewed amount of data for each class it is known as having an imbalanced dataset. Multiclass classification with an imbalanced dataset means the model will be trained more with one class than the rest leading to higher amounts of misclassification.



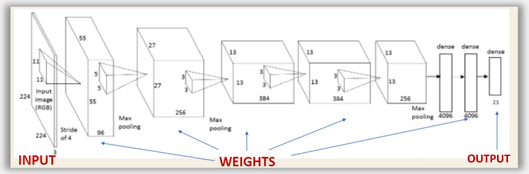
References

CNN, Supervised Learning & Imbalanced Datasets
- https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/
- https://www.ibm.com/cloud/learn/supervised-learning
- https://machinelearningmastery.com/imbalanced-classification-is-hard/

Transfer Learning & VGG19 in Keras
- https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/
- https://keras.io/api/applications/vgg/

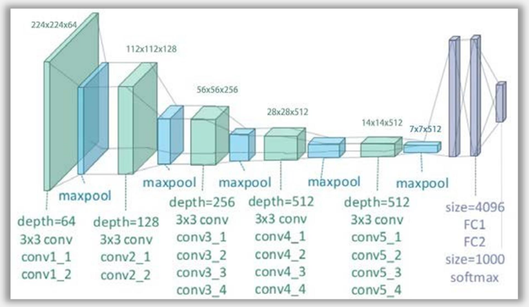
Analysis

Initial Approach
Initially, I created a CNN based on this model:



I saved a few versions of this model with varying hyperparameters that mainly involved changing the resolution, but also involved adding BatchNormalization after each Conv2D layer.

Secondary Approach
I did some research into Transfer Learning (TL) and discovered the VGG19 network. TL allows the use of pre-trained models directly, as feature extraction pre-processing, and integrated into entirely new models. The VGG19 model consists of 19 layers (16 Convolution layers, 3 Fully connected layer, 5 MaxPool layers and 1 SoftMax layer) and can use weights from ImageNet which has millions of pre-trained weights for feature extraction.



However the output layer (image classifier) was replaced with a Flatten layer and a Dense layer of 23 outputs and SoftMax activation so that it could correctly identify the 23 different classes of butterfly. The limitation of this model is that the input is restricted to 224x224x3 since that is what the weights from ImageNet were trained at.

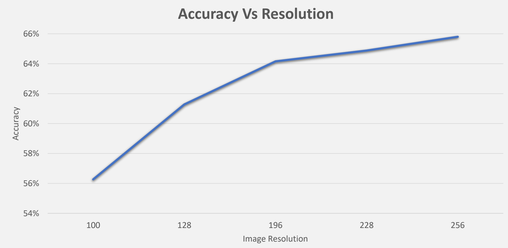
Challenges
- Initially I would load all images into memory before training, however I was limited to 13.9GB of memory, but using Keras' *flow_from_dataframe* I didn't need to load all the images into memory, so I could train at resolutions higher than 200 pixels without crashing.
- Adam optimizer was great for small amounts of parameters in my first approach but underfits with models with large parameters. The Adam optimizer would consistently get stuck at 16.07% when training the VGG19 model. However since the SGD optimizer has less features it yielded significantly greater accuracy results.

Initial Implementation

Initial Approach
Training data was split at 80%, with 8,215 training images and 2054 validation images, although ImageDataGenerator was used which creates about 8000 additional augmented images for training. The initial approach training used consistent batch sizes of 32, 100 epochs and Adam optimizer, and started with 100x100 resolution.

Hyperparameter Change	Accuracy
Default	56.27%
196x196 resolution	64.16%
228x228 resolution	64.88%
256x256 resolution	65.8%
Added BatchNormalization layers	52.28%

Initial Evaluation
During the implementation of the initial approach is was clear that the accuracy could be increased dramatically by increasing the resolution. However as the resolution become much higher, the data started to plateau.



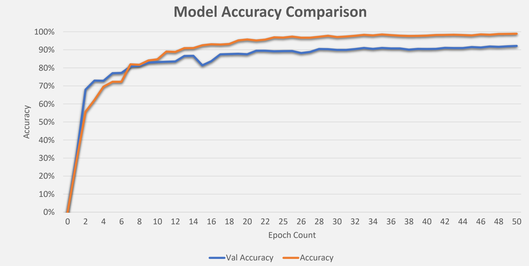
Adding batch normalization to my model culminated in a significantly lower accuracy. In conclusion this model was incapable of achieving an accuracy which could reach my target accuracy. As explained in my analysis section, this was when I started to research into Transfer Learning as a way to use weights from other image classification models into my own.

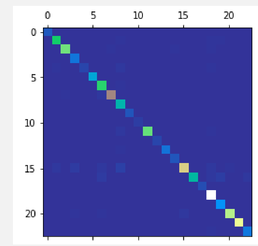
Secondary Implementation

Secondary Approach
I used consistent batch sizes of 32, 50 epochs and SGD optimizer for the VGG19 model at 224x224x3 resolution.

Hyperparameter Change	Accuracy
Default	89.9%
No validation data	93.32%

Secondary Evaluation
The following chart shows the results of the training data which was split at 80%, with 8,215 training images and 2054 validation images. The validation accuracy falls short of the training accuracy suggesting the data was slightly overfit.





Confusion Matrix
A confusion matrix evaluates performance of a classification model which compares the actual target values with those predicted by the model. This matrix represents prediction accuracies of all 23 classes.

The coloured squares diagonally through the middle show high accuracy. The feint blue squares off the middle show false positives (precision), since they are so feint, it proves the model's high accuracy.

Conclusion

As seen in the graph, the initial approach trained the model such that the data was overfitting since the accuracy reached over 90% yet the validation accuracy barely breached 70%.
After my initial approach it was clear that to achieve high accuracy, 10,000 images was simply not enough data on its own to produce a model capable of achieving my target accuracy. The difference in performance in the first and second model proves how valuable transfer learning is to achieving high accuracies in image classification problems.