Student Number: 169414

1. Introduction

The purpose of this report is to outline and explore an implementation of a zero-shot recognition system employing concepts of Direct Attribute Prediction (DAP).

An attempt will be made to optimize the results of the implementation, as well as opening a discussion on how the implementation might be improved to produce a greater accuracy.

A zero-shot recognition system is one that attempts to label image classes unseen by the system in training. To do so, the system requires a labelled training set of classes and the knowledge of how these seen classes are semantically related to the unseen classes, in the form of a predicate matrix [1].

2. Method

2.1 Feature Extraction

Feature extraction was carried out via the use of a pre-trained convolutional neural network (CNN), ResNet-18 at pool5, the end of the network. The global pooling layers throughout the network pool the input features over all spatial locations, generating a total of 512 features.

Also, a further attempt at feature extraction was carried out, again using a pre-trained CNN, this time AlexNet at layer fc7. As explored in section 4, there was a discreet difference in results between the two.

2.2 Train Attribute Models

The next operation is the training of 85 attribute classifier model's necessary for the 85 different image classes. This was achieved through the use of MatLabs's fitcsvm function.

Fitcsvm returns a support vector machine trained using the extracted image features and trained class labels. In an attempt to improve the performance of the fitcsvm function, a series of hyperparameter optimisation tasks were run. The results of these can be found within Section 3 and discussed in Section 4. The models produced by the fitcsvm function are then passed through MatLab's fitSVMPosterior function, producing a trained support vector machine containing the optimal score-to-posterior probability transformation function for two-class learning.

2.3 Compute Attribute Probabilities

The compute attribute probabilities function outputs a matrix of probabilities representing the probability of each image containing a certain attribute.

2.4 Compute Class Probabilities

The compute class probabilities function equates the probabilities of each class being in each image and takes the attribute probabilities as an input.

2.5 Compute Accuracy

The compute accuracy function equates the final accuracy of the system and received the class probabilities as input.

3. Results

The initial result of the system was a result of roughly 4.7% accuracy, a clearly concerning result, with random selection of the image classes having a greater probability of selecting the correct class at 10%. A series of changes were then made in an attempt to improve the accuracy, as will be explored in sections 3.1 to 3.3.

3.1 Hyperparameter Optimization

As aforementioned in Section 2.2, a series of hyperparameter optimization tasks were run on the fitcsvm function. The function itself takes a series of arguments, ones of interest and used within this experiment are BoxConstraint, KernelFunction and Standardization.

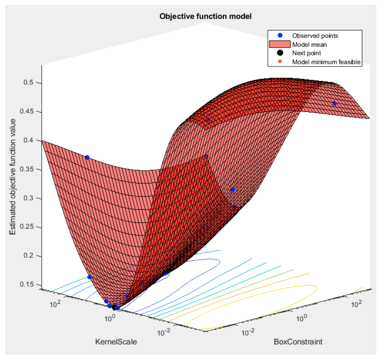

Figure 1 Hyperparameter Optimisation

Figure 1 expresses the results of the optimisation run on the fitcsvm function. Optimisation revealed an optimal BoxConstraint value of 0.0039877 and a Kernel Scale of 4.162233 run on the Radial Basis Function (RBF) kernel, a kernel capable of capturing relationships of greater complexity in the data. Expressed in formulaic terms of

$$G(x^j, x^k) = \exp\left(-\left\|x_j - x_k\right\|\right)$$

The overall accuracy of the system immediately increased to approximately 10.4%%, a significant improvement to the initial result.

3.2 Changing from AlexNet to ResNet-18

A further improvement discovered was changing the neural network used from AlexNet to ResNet. Resnet caused an increase in the accuracy by 1.5%, from 10.4% to 11.9%. Moreover, the greatest benefit of switching the network was the decrease in run time. Prior to changing the network, the system would take 9 hours to produce an accuracy. Changing to ResNet decreased the run time to just over 4 and a half hours. Although the decrease in runtime is to be expected, it could be hypothesised that a network with a dimensionality of 4096 should produce classifiers with a greater accuracy than one of 512 [1].

3.3 Overfitting

A common cause of a poor accuracy result is overfitting [2]. Overfitting is the process of providing the classifiers with too much training data, in this case, too many images of animals. When the classifier is fed too much training data, there is a chance that regularly occurring noise within the pictures can be mistaken for features of interest. This results in the classifier not registering features of an image, despite their presence.

In an attempt to overcome this, the training data was restricted to 100 items per class. This resulted in an increase of the accuracy to 13.4%. Further attempts involving adjusting the box constraint of the fitcsvm function were employed but were unable to increase the accuracy. The results of this experiment can be found in Table 1 below.

Table 1 Box Constraint Optimisation

Box Constraint	Accuracy
0.0039877	10.4%
1	9.6%
10	10.2%

4. Discussion

Further improvements could lie within the introduction of a transductive setting. Transductive Zero Shot Learning implies that unlabeled images from unseen classes are available during training. Theory states that using unlabeled images should improve performance as they contain useful latent information of unseen classes [3].

Furthermore, tuning the parameters for fitcsvm based on the testing set of images could also see an improvement to the accuracy. By performing hyperparameter optimization on the test data, the parameters used to train the classifiers will be the optimal solution for our specific testing data.

Another possible improvement would be to fine tune the images used in testing and training. Removing as much noise as possible from the images, such as the sky and environment would result in a higher accuracy as more features would be extracted from the animal itself and less from the environment.

Continuing on with improvements, long run times of up to 14 hours meant that changes were difficult to implement and test, discouraging tweaking and fine tuning for fear of breaking the implementation. Inefficiencies within code are at fault but could be alleviated by introducing more computing power. The University of Sussex provides specialist computing services in the form of High Performance Computing. Running the system on the high performing computers would allow for a greater level of optimization and tweaking in much smaller time frame.

To conclude, it was interesting to see the development of a zero-shot learning system. Gaining an understanding of how these 'magical' systems work is vital to understanding theories behind Artificial Intelligence. I initially began implementation of the system by using SURF as a means of feature extraction. SURF (Speeded-Up Robust Features) uses a blob detector based on the Hessian matrix to find points of interest, the determinant of the matrix is then used as a measure of local change.

However, disappointingly I was unable to implement this, puzzled by differences in dimensionality fed into the fitcsvm function thus changing to CNN's.

A final change that would be made in future implementations would be switching to Python as the programming language. Although MatLab features useful tools such as the workspace, it is theorized that the computational complexity of the task could be optimized in python, as well as the ability to use toolkits such as OpenCV.

5. References

[1] Fu et al (2017). *Recent Advances in Zero-shot recognition.*

[2]Brownlee, J (2016). *Overfitting and Underfitting with Machine Learning*

[3] V. K. Verm and P. Rai (2017), *A simple exponential family framework for zero-shot learning, pp. 792–808.*

[4] Zhu et al (2019). *Zero Shot Detection.*