Student Number:198688

1.1 Outline of methods employed

My methods employed follow a simple progression.

The image data is sent to the feature extraction, 'extract_features()'. The extracted features for the training data are then sent to the method 'train_attribute_models()', which builds a classifier for each attribute. Those classifiers and the test data features are then sent to 'compute_attribute_probs ()' where for each image, its features run through each attribute classifier to determine the probability of that test image containing those attributes. The attribute probabilities for the test images are then sent to 'compute_class_probs()' for the attribute probabilities to be turned into the probability of being one of the unseen classes.

1.2 Feature Extraction

The parameters for this method are; dir, classes, clusters, hessian_threshold and upright_orientation. Dir simply refers to the directory where the image folders are saved. Classes is a list of the animal classes whose features are being extracted. Clusters is the number of clusters used in the k means algorithm as well as the number of features output per image. Hessian threshold literally refers to the hessian threshold used in the SURF algorithm, to determine how many descriptors are found. Upright orientation is used to choose if orientation is calculated when the features are extracted, this option is here for the possibility of speeding up.

As well as the above described standard zero shot approach, I employed a slightly altered second approach with the hope of reducing memory requirements. I will refer to this second approach as the 'double clustering' method. The double clustering method is the same as the standard approach except it clusters the descriptors for each image and saves the clusters. The number of clusters used per image is 500, this does affect the accuracy/size trade-off. I expect this to reduce overall accuracy for comparable picture set sizes but to vastly reduce the memory cost. This second approach could potentially have its accuracy penalty reduced by saving a proportional representative of each image cluster rather than the centres themselves as descriptors.

The reason I tested the double clustering approach as well as the standard approach is because a standard number of descriptors extracted from an image is close to 5000 whereas with the double clustering method I can reduce this to a specified 'inner cluster' amount. Through preliminary test I optimised this inner cluster value to be 500.

The feature extractor extracts SURF descriptors for each image, before clustering them between every image using K-means. Each image then has a histogram applied to its feature distribution, which is normalised to 1.

I have chosen to use a smaller number of clusters/features for my training. I am using 120 features, preliminary tests on a reduced image set showed that this was a performance peak. I performed preliminary tests to optimise these variable, however, my tests were not extensive leaving room for potential further optimisation..

Preliminary tests helped me decide on the final hessian value of 500, once again these tests were not extensive and room for optimisation is still available. Because the features are an average over the training data, this means that attribute classes can share feature meaning, this should mean I have an ample amount of features extracted to test my algorithms. [1]. Features that discriminate for one class can potentially discriminate for another [2].

1.3 Classifiers

The classifier building method has 4 parameters. Classes, features, predicate_binary and attributes. The 'classes' parameter is simply the classes being used to build the classifiers. I pass in training classes for this. Features is the equalised histograms created in the previous section. Predicate binary is used to train the classifier correctly. Attributes is simply the list of attributes being used in the classifier.

To build the classifiers, this method iterates through each attribute, splitting the training features into positive data and negative data for that attribute before feeding them through a linearSVC classifier and calibrating the classifier.

This method simply returns a list of classifiers for each attribute, there is no need to tag the classifiers because they are still in the same order as the attributes.

1.4 Attribute Probabilities

This method takes the classifier and the test data as parameters and returns a matrix of 85 by Ntest, images. Where for each image it has the probability of it containing each attribute.

1.5 Class Probabilities

When computing the class probabilities, this method takes 2 parameters, the predicate matrix for the testing classes and the matrix containing the probabilities each attribute is present in each class. For each image the probability of it being each of the 10 testing classes is then saved and output. This is done by multiplying the probabilities of the test image sharing the same attribute as that class.

2 Results achieved and analysis

Because of the cost requirements of using a sift feature selector, I did not use the full image set when performing my tests. Instead I plotted the accuracy scores against the number of images used per class for both standard zero shot approach and double clustering, this is seen in figure 1.

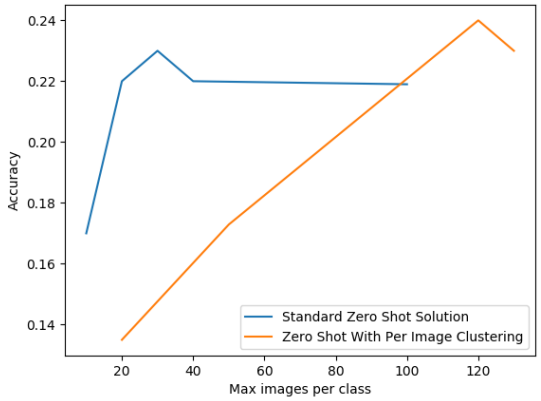

Figure 1: accuracy of both zero shot approaches plotted against number of images used per class.

As you can see from figure 1, both approaches reached a similar maximum accuracy of between 22% and 24%. The methods appear to plateau in accuracy with increase in images unlikely to increase accuracy.

While the double clustering method required much more pictures in order to achieve a similar accuracy, the memory cost was greatly reduced. Figure 2 shows the memory cost required to achieve a specific accuracy rating for each method. Each unit of memory cost represents a 126 sized vector of floating point numbers.

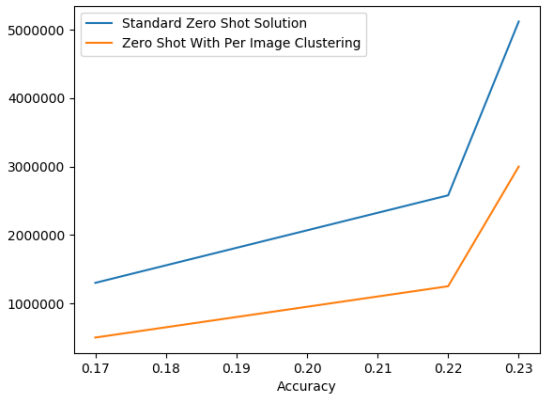

Figure 2: Accuracy score plotted against minimum memory requirements needed.

While the double clustering method does not greatly affect accuracy when a higher image count is used, it does offer a low memory alternative if large amounts of training images are available.

3 Discussion

Exploring different methods of feature extraction and their corresponding accuracy scores would potentially be a useful investigation. It has been shown that feature selection method can influence the overall accuracy while using the same approach to zero shot learning [3]. I would like to explore a colour histogram feature extraction as well as a texture extraction method used in previous studies [4]. These methods are viable for feature extraction, my interest in improving accuracy lies in combining features from multiple extraction methods.

A useful addition to my implementations is a feature selection algorithm used in order to extract the more useful features, this could have a positive effect on the accuracy of the algorithm [4]. Especially if using features from multiple feature selection methods.

The class probabilities could be contributing to a potential loss in accuracy due to the loss in accuracy of the probability itself. This arises from multiplying 85 attribute probabilities which are all in the range $0 \leq x \leq 1$ leading to a small result with a high exponent. This problem can be mitigated by performing an appropriate type of smoothing. Brief test on my data showed that even a simple type of smoothing such as plus one reduces the exponent, potentially leading to more numerical accuracy.

An attribute analysis would help contribute to overall accuracy. This can be determined from figure 3 where I plotted the accuracy using only subsets of the attributes given. The attributes were naturally classed into 4 types. As you can see the sets of attributes had difference performance, this alone is evidence enough to pursue an investigation.

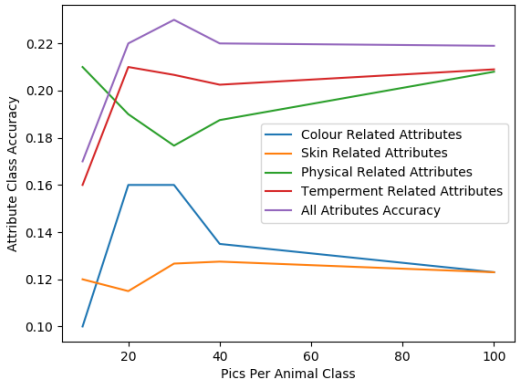

Figure 3: performance of attribute class subsets plotted alongside performance of all attributes.

Attribute selection process could mitigate human error in the generation of the training data e.g. a Persian cat is shown to have the attribute ‘blue’. Attribute selection could potentially remove underperforming attributes with high levels of human error.

Bibliography

[1] A. Torralba, K. P. Murphy and W. T. Freeman, “Sharing visual features for mutliclass and multiview object detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI),* vol. 29, no. 5, pp. 854-869, 2007.

[2] E. Bart and S. Ullman, “Cross-Generalisation: learning novel classes from a single example by feature replacement,” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* vol. 1, pp. 672-679, 2005.

[3] C. A. Caceres, M. J. Roos and K. M. Rupp, “Feature Selection Methods for Zero-Shot Learning of Neural Activity,” Fron Neuroinform, Online, 2017.

[4] M. Liu, D. Zhang and S. Chen, “Attribute relation learning for zero-shot classification,” *Neurocomputing,* pp. 34-46, 2013.