Student Number: 164789

1. Approach

The first thing I did when approaching this zero-shot recognition system was to look at the attributes I would need to predict in order for the final classification to be as successful as possible, so I chose the alexnet cnn since its higher dimensionality would mean that the linear SVCs I employed to predict attributes in the next stage would have more data to train on. After this I followed the methodology mentioned in the brief to produce the probabilities of each image existing in each class. The entirety of this recognition system was coded in a Jupyter notebook for easier development.

2. Methodology

2.1 Feature Extraction

I began by loading in the alexnet cnn model using the very handy tools available from opencv. To do this I used the “bvlc_alexnet.prototxt” for the model architecture and the “bvlc_alexnet.caffemodel” for the model’s weights. I went on to create a separate file in each image directory, containing the 4096-dimension features extracted by feeding the images through the net after resizing them to appropriate dimensions. The extracted feature matrices were kept in their corresponding folders instead of a central file to make the creation of the x, y data for the SVCs given the way the labels are produced (based on the class/and folder they belong to).

2.2 Data Preparation

In this step I assembled the input and output data needed to train the classifiers for the attributes. Firstly, I loaded in the files needed to split the data into training and testing sets and began with the x data variable generation. In the function used for this I also saved the number of extracted feature vectors, i.e. input data rows from each class to help populate the label (y) matrix. I then proceeded to fill the y matrix with the class attributes using the list of entry numbers to determine the amount to insert for each class.

2.3 Training of the Classifier

I began training the 85 attribute classifiers using sklearn’s Linear Support Vector Machine[1] classifier model, training each individual one with the appropriate attribute labels, a process made very easy by numpy’s easy-to-use slicing methods. I also tried using more complex models to solve this classification problem but sadly, all the approaches including SVCs with non-linear kernels and Random Forest Classifiers[2] where too computationally demanding and, in most my tests, where actually outperformed by the LinearSVC model with a penalty parameter C of 1e-5.

2.4 Getting Attribute & Class Predictions

Finally, I used the classifiers to get the probability of each attribute existing (i.e. attribute=1) or not existing (attribute=0) in the vectors (and by extension in the images) so I could then use the product of those probabilities to see which class appeared more probable. The brief did request an output of 85 X Ntest so I left the necessary lines of code in a comment in the relevant cell of the jupyter notebook. Finally, I created a rudimentary accuracy metric to judge the predictive model I had implemented.

3. Results and Discussion

The results I got from the zero-shot recognition model with the linear SVC attribute classifier reached their highest accuracy of 49.243% when I implemented a penalty parameter of 1e-5. I believe that this miniscule penalty parameter helped allow outliers to play an important role in the individual classification of characteristics. In contrast, the Random Forest Classifier which, being more complex I thought might perform better, had an overall similar performance in individual attribute prediction and a final accuracy of 39.5% with 10 estimators to average from, so the idea was discarded. Any other more complex approach was rejected as well since training times rose dramatically. The results of all the main classifiers employed in the attribute prediction step can be seen at the end of this section (Table 1).

The results of this implementation were, to me, a pleasant surprise. I considered this task to be difficult to approach, based on both the dataset and the stacking of two separately trained models in order to arrive at the final result. The fact that the error function that trained this system wasn’t based on the classes we were predicting but instead on the attribute prediction accuracy also aided in this pessimistic approach.

All things considered I believe that the project was a success, but I believe there is still room for improvement. My first suggestion is relevant to the dataset. The dataset images were of the animals in question in very noisy environments and this must have interfered with some attribute classifiers. Some cleaner images would probably improve the attribute prediction accuracy. The animal images per class also were slightly imbalanced but were in the same order of magnitude so this might only slightly affect the results.

My second suggestion pertains to some of the attributes themselves since when I inspected the images with the attribute list in hand, I quickly saw that some attributes weren't discernible in many images and their misclassification would sabotage the final class prediction. A dataset with fewer but more easily detectable attributes might outperform this 85x50 problem using the exact same model.

Through this project I got hands on experience on how a very popular tool like the cnn can function as an intelligent encoding tool. On a more practical note, I also acquired valuable experience in creating and transforming a dataset in order to tackle a demanding problem as well as some basic guidelines on how to approach new data centered problems from now on.

All in all, I consider this attempt to implement the Direct Attribute Prediction concept as a mild success, considering that 49.14% accuracy on a 10 class classification is 4.5 times better than random. The reasoning behind the word mild stems from my strong belief that this problem can still be tackled in a more successful way. For example, the alexnet model used in the feature extraction of the images was an ImageNet challenge winner in 2012, meaning that it would perform tremendously when classifying subjects of 1000 different categories, but maybe for this animal detection problem a model trained solely on the various animal types would be more accurate in extracting relevant features.

Model	SVC(Linear)	SVC(rbf)	Random Forest
Test Acc	49.143%	Aborted	39%
Training time	Base Training time	>20xTraining time	6xTraining time

Table 1. System accuracy and training time for the attribute classifiers

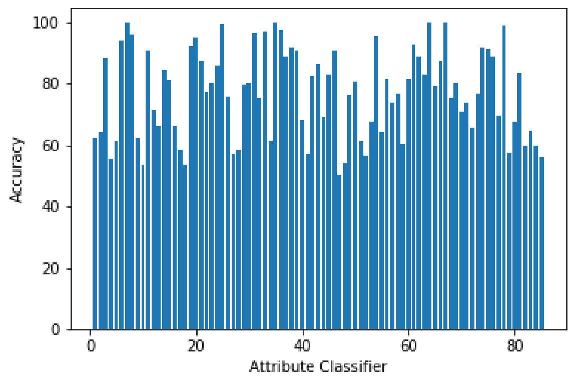

Figure 1. Individual attribute classifiers accuracy for the LinearSVC model

References

[1] Vladimir N. Vapnik, Bernhard E Boser, Isabelle M. Guyon, A training algorithm for optimal margin classifier.

[2] Leo Breiman. Statistics Department, University of California Berkeley, CA94720. Random Forests

[3] LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. Nature, 521(7553)