BASELINE

230/229 [==============================] - 20s 85ms/step - loss: 1.8339 - categorical\_accuracy: 0.9913 - val\_loss: 4.3627 - val\_categorical\_accuracy: 0.4646

BASELINE (MASK)

230/229 [==============================] - 31s 137ms/step - loss: 1.6915 - categorical\_accuracy: 0.9632 - val\_loss: 8.7832 - val\_categorical\_accuracy: 0.1475

ALS

230/229 [==============================] - 19s 84ms/step - loss: 2.7355 - categorical\_accuracy: 0.9907 - val\_loss: 3.7228 - val\_categorical\_accuracy: 0.4384

ALS (Beta = 0.25)

230/229 [==============================] - 19s 82ms/step - loss: 2.1241 - categorical\_accuracy: 0.9943 - val\_loss: 3.8658 - val\_categorical\_accuracy: 0.4597

ALS (Beta = 0.75)

230/229 [==============================] - 19s 83ms/step - loss: 2.5535 - categorical\_accuracy: 0.9940 - val\_loss: 3.7104 - val\_categorical\_accuracy: 0.4569

ALS (MASK)

230/229 [==============================] - 19s 82ms/step - loss: 2.6950 - categorical\_accuracy: 0.9686 - val\_loss: 4.5321 - val\_categorical\_accuracy: 0.2271

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Method | Train N | ACC | ECE  100 | ECE  15 | MCE | O.conf | U.conf |
|  |  |  |  |  |  |  |  |
| Hard Label | 3.668k | 0.4646 |  |  |  | 0.98115385 | 0.019000035 |
| A. L. S. | 3.668k | 0.4384 |  |  |  | 0.691008 | 0.3093609 |
| A. L. S. (Beta=0.75) | 3.668k | 0.4569 |  |  |  | 0.72354233 | 0.27702665 |
| A. L. S. (Beta=0.25) | 3.668k | 0.4597 |  |  |  | 0.9060857 | 0.093837395 |
|  |  |  |  |  |  |  |  |
| Hard Label (incorrect mask) | 3.668k | 0.1475 |  |  |  |  |  |
| A. L. S. (incorrect mask) | 3.668k |  |  |  |  |  |  |

Table 1: Classification and calibration results with Oxford IIIT Pets

I used the Oxford IIIT Pets dataset. This dataset consists of around 200 images per class for 37 classes. However, the training and test sets were created with a 50-50 split. I initially wanted to recombine them and create my own 80-20 or 90-10 split, but I found that the images in the test set did not have bounding boxes, so I could not use them for training. This meant that for training, I have around 100 images per class for 37 classes, which is a very small amount. I suspected this would easily overfit, so I used TensorFlow’s ImageDataGenerator to augment the training examples, creating new examples.

I decided to exclude some examples for two main reasons. The first reason is that some of the images were unusable; They were either stored under the incorrect format (.jpg instead of .png or .gif), or they were corrupted. The second reason is that some of the training images did not proper bounding boxes. Some of them did not have an .xml file, meaning there was no associated bounding box for the image. One of the images had two bounding boxes, and the authors mentioned a strategy (masking) for dealing with images with multiple bounding boxes. However, as there was only one image with more than one bounding box, I decided it would not significantly change the results if it was not included. I excluded this image for consistency.

I decided not to use two validation sets like the authors did. This was because the V2 validation set that the authors used was a set of “harder” images compared to the V1 validation set. However, the Oxford IIT Pets dataset does not have a comparable split. Randomly splitting the validation set into two independent sets would have been pointless, as they would both have similar results.

I initially misunderstood what the authors meant by ‘masking’. I initially understood it to mean replacing every pixel outside of the bounding box with a constant value. However, after getting terrible results, I realized that the authors use it to deal with images with multiple bounding boxes. The authors mask every bounding box except for one, which also results in additional training examples. Again, this is not very applicable to the dataset I used, because all of the examples except for one have a single bounding box. Since I did not use masking, I replicated the masking results achieved by the authors without masking.

I resize every image to 100 by 100 pixels. I initially resized to 150 by 150 pixels, but I decided to reduce the size to facilitate faster training. I also add 6 pixels to each image and randomly crop it back to 100 by 100 pixels.

I use ResNet50. I train for 200 epochs using SGD with a learning rate of 0.001 decayed by 0.1 at epochs 50, 100, and 150 and a momentum of 0.9. I use a batch size of 16 and L2 regularization with a penalty term of 0.01.

At the end of training, my model achieves over 99% training accuracy, while my validation accuracy is around 47%. Not only is there significant overfitting, but the ridiculously high training accuracy suggests my model is memorizing the training data.

The results in Table 1 are consistent with the results achieved in the paper. Hard label is equivalent to beta=0. A.L.S. is equivalent to beta=1. With a decreasing beta, the overconfidence increases while the underconfidence decreases. The accuracies achieved are also consistent. The accuracy increases with a decreasing beta.

The overconfidence of the Hard Label approach is 0.98, reflecting the extreme overfitting (>99% training accuracy). A.L.S. helped to regularize the model, so although it similarly achieved >99% training accuracy, it is less overconfident.

Overall, a lot of my problems were caused by the dataset I selected. However, I chose the dataset, so I had to work through the problems. It is very likely that I could have achieved better results with a better dataset. If I had more computing power, I could have resized the images to a large size. Currently, resizing to 100 by 100 pixels discards a lot of information for the larger images.