Final Assignment

Inon Katz 208084004
Netanel Sabah 207846643

Pseudo Code:

Baseline

The baseline used is a simple resnet, fitted to a custom number of outputs.

DREML pseudo-code

the model used here is the very same as the one used in the baseline
D - number of meta classes in a partition
L - number of child models in the ensemble

- fit

```
fit(X, y):
        initialize models[L]
        Initialize model_results_matrix[L, length of X]
        ** train the DREML ensemble **
        For model in models:
                y' = meta_class_partition(model, D)
                train model(X, y')
        ** get the embeddings of each model **
        For model in models:
                model_results_matrix[model] = predict model(X)
        ** concatenate the embeddings for each image **
        initialize ensemble_results[length of X, L*D]
        ensemble_results = concatenate results of model_results_matrix
        ** feed the results into the svm model **
        initialize SVM_model
        fit SVM_model(ensemble_results, y)
```

- predict/predict_proba

```
predict/predict_proba(X):
        Initialize model_results_matrix[L, length of X]
        for model in models:
                model_results_matrix[model] = predict model(X)
        initialize ensemble_results[length of X, L*D]
        ensemble_results = concatenate results of model_results_matrix
```

predict/predict_proba SVM_model(ensemble_results)

Improved-DREML

Mostly uses the same code as the base DREML.
The only difference is found in calculating the value of y' during the fitting process, where instead of

y' = meta_class_partition(model, D)

we shall call

y' = meta_class_partition(model, f_D(model)),

where f_D is a function that, when given the current model, calculates the value of D. In our implementation, f_D is an increasing, positive function whose value is bound by the total number of classes in y.

Results:

We ran our three algorithms over 4 different data sets. For each dataset, we randomly sampled a smaller number of classes and images 5 times, for a total of 4 * 5 = 20 such dataset-partitions.
To check the full, 600-entry long result spreadsheet please check out the results file.

Friedman test

We performed the Friedman test over the values of the accuracy metric.
First, for every model/dataset-partition we calculated the mean accuracy for the 10 outer cross validation folds and placed the values in a table:

	DREML	Improved DREML	Baseline
car-0	0.722	0.715	0.147
car-1	0.827	0.807	0.067
car-2	0.821	0.867	0.057
car-3	0.767	0.678	0.123
car-4	0.78	0.775	0.249
cub-0	0.89	0.876	0.095
cub-1	0.905	0.889	0.144
cub-2	0.9	0.906	0.211
cub-3	0.922	0.918	0.106
cub-4	0.926	0.865	0.285
flowers-0	0.989	0.984	0.243
flowers-1	0.994	0.983	0.304
flowers-2	0.982	0.987	0.302
flowers-3	0.965	0.922	0.216
flowers-4	0.976	0.979	0.197
caltech-0	0.92	0.91	0.333
caltech-1	0.959	0.959	0.196
caltech-2	0.926	0.915	0.338
caltech-3	0.906	0.891	0.086
caltech-4	0.928	0.921	0.208

Following that, we ranked the values of each row:

	DREML	Improved DREML	Baseline
car-0	1	2	3
car-1	1	2	3
car-2	2	1	3
car-3	1	2	3
car-4	1	2	3
cub-0	1	2	3
cub-1	1	2	3
cub-2	2	1	3
cub-3	1	2	3
cub-4	1	2	3
flowers-0	1	2	3
flowers-1	1	2	3
flowers-2	2	1	3
flowers-3	1	2	3
flowers-4	2	1	3
caltech-0	1	2	3
caltech-1	1	2	3
caltech-2	1	2	3
caltech-3	1	2	3
caltech-4	1	2	3

Calculating the mean of each column (algorithm)'s rank gives us the following ranking:
DREML: 1.2
Improved DREML: 1.8
Baseline: 3

L, which is the number of algorithms, is equal to 3.
N, which is the number of dataset-partitions we ran our code on, is equal to 20.

Our null hypothesis states that there is no difference between the three algorithms.

After calculating the f value using the Friedman test's formula, we reached the value of 99.75.

Our degrees of freedom are 2 and 38. For every meaningful alpha (not too small), the critical value of the F distribution is smaller than 99.75, meaning that we can safely reject the null hypothesis, and thus, there is a significant difference between all three algorithms.

Post-hoc test

For the post-hoc test, we employed a paired, post-hoc comparison between each pairing of algorithms: between DREML and Improved DREML, between DREML and the Baseline, and between Improved DREML and the baseline.
For that end, we calculated the absolute value of the differences between each pairing's ranks:

- DREML and Improved DREML: 1.8 - 1.2 = 0.6
- DREML and baseline: 3 - 1.2 = 1.8
- Improved DREML and baseline: 3 - 1.8 = 1.2

For alpha = 0.05/0.10, the critical difference is 0.741/0.649 respectively. As we can see, from these two values we can infer that while there is a significant difference between the Baseline and each of the other two algorithms, there is no difference between DREML and Improved DREML.

Conclusions

- Training a single DREML model, on average, takes more time than a simple, basic Resnet model on a factor of ~42. Inference time is worse by a factor of ~8.
- The DREML method of image embedding allows for the creation of a more robust, accurate classification model over a larger number of classes, even when given a small number of samples within each class.
- While our suggested improvement over the DREML algorithm is not statistically different in its performance from that of the normal DREML, the rankings calculated during the Friedman tests concludes that there is indeed an edge in the accuracy performance of the normal DREML. This makes us wonder if a more randomized implementation of our idea would have better sufficed, where for each child model D would be randomly set instead of an incrementing value. Due to a lack of time (since running the DREML/improved DREML algorithm is a long-winding process), we were sadly unable to test this hypothesis out.