Final Project Report – RandAugment

INTRO

In this work, I evaluated the algorithm ‘RandAugment’ as proposed in citation .

Data augmentation is a powerfull technique to increase the generalization capability and reduce overfitting of deep learning models. However, it is not obvious which augmentations should be applied for a certain task, nor at which frequncy or strength. In recent years, many different augmentation policies were proposed, such as ‘AutoAugment’ (AA) (citation).

However, thses policies usually require a second optimization procedure to select the optimal augmentation strategy for the task.

In the paper ‘’, the authors propose a simple augmentation strategy which performs about as good as, and sometimes better than, methods such as AA but with a much simpler optimization procedure, with a considerably smaller search space.

ALGORITHM

Key idea:

\* Define a list of K transformations which make sense for the task at hand (*some* domain knowledge is required)

\* Define a linear scale M of strength for each transformation, say 0 to 10.

\* Now you have 2 parameters to optimize, M and N – number of transfromations to choose out of possible K.

\* Finally, at training time, to each sample apply N random transformations with strength M.

The advantage or RA is that its very straightforward to implement and optimize, only roughly 102 search space size, and it allegedly performs quite well.

The downside, is that a more thorough algorithms can likely perform better (as evident even by the RA paper authors).

As a baseline comparison, I used a simple random flip + random resized crop augmentation policy.

DATA

Due to the nature of the algorithm, I had to work with image data (classification).

Finding 20 image datasets which were of reasonably size proved to be quite difficult. I ended up following Prof. Lior Rokach’s suggestion of picking 4-5 datasets and splitting them to artificially create 20 datasets:

[VGG-Flowers](https://www.robots.ox.ac.uk/~vgg/data/flowers/102/index.html) x 1 (102 category, no split)

[The Oxford-IIIT Pet Dataset](https://www.robots.ox.ac.uk/~vgg/data/pets/) x 1 (only the cats, 12 category, no split)

[Stanford Dogs Dataset](http://vision.stanford.edu/aditya86/ImageNetDogs/) x 6 (120 categories, randomly split into 6 datasets of 20 each)

[CINIC-10](https://paperswithcode.com/dataset/cinic-10) x 2 (10 categories, split into 2 datasets of 5 each, randomly selected 10% of the images per class. Left with 2,700 samples per class)

[CIFAR-100](https://www.cs.toronto.edu/~kriz/cifar.html) x 5 (100 categories, randomly split into 5 datasets of 20 each)

[COIL-100](https://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php) x 5 (100 categories, randomly split into 5 datasets of 20 each)

After some cleaning up and organizing, I ended up with 20 datasets as described above with a total of more than 125,000 images(!)

METHOD

Since I was required to compare 2 algorithms, each across 20 datasets, each with a 10 fold cross-validation loop, each with a 3 fold inner cross-validation loop with 50 hyperparameter trials..

Needless to say, this was an absurd amount of models that I had to train, which was only made worse by the fact that I had to do it on images.

Some concessions had to be made:

\* I used a relatively small model (ResNet-18)

\* I used a pretrained model and only replaced the last layer for each dataset.

\* I only trained for 2 epochs per inner CV fold and 10 epochs per outer CV fold.

\* I used [optuna](https://optuna.org/) to search for hyperparametrs and early pruned unpromising trials.

\* Worst of all was realizing I could set up the experiment better, but only half way through training.

Sadly I could not afford to rerun everything.

Still, it took nearly 2 weeks to train everything on 4 machines:

2 x Computers from work with GTX 1080TI (only partially available to me)

1 x Microsoft Azure VM with a Tesla V100 (only partially available to me)

1 x My own machine with RTX 3060

(I added a ‘GPU’ column to the results tables to help make sense of the train\infer times.)