COMP6248 REPRODUCIBILITY CHALLENGE FREE LUNCH FOR FEW-SHOT LEARNING

Orry Swift **Neil Patil** **On Law**

Team Name: Classifier? I hardly know her!

Chosen Paper: Free Lunch For Few-Shot Learning: Distribution Calibration
Yang, S., Liu, L. and Xu, M., 2021. Free lunch for few-shot learning: Distribution calibration. arXiv preprint arXiv:2101.06395.

Repository:
https://github.com/COMP6248-Reproducability-Challenge/few-shot-learning-reproducibility

1 TARGET QUESTIONS

The paper 'Free Lunch for Few-Shot Learning: Distribution Calibration' aimed to use transferring statistics on a few sampled classes to calibrate a Gaussian distribution to generate appropriate features for training. By separating the initial dataset into novel (classes with few samples) and base (classes with many samples), the feature distributions in the base classes can be transferred to the novel classes to allow feature sampling. Our targets for this reproducibility study are as follows:

- Re-implement the statistic calibration algorithm for the novel classes
- Generate samples and verify that they are in line with the real class distributions
- Verify that the hyperparameters are optimal at similar values to the paper
- Validate that the sampled training set produces a similar improvement to accuracy over the whole validation set
- Validate that the sampled training set produces a similar improvement to accuracy over the novel validation set

2 EXPERIMENTAL METHODOLOGY

The original paper includes a GitHub link with their code. Although the code is commented well enough for someone familiar with the algorithm, it is not particularly clear what is happening and why to someone who does not already know the algorithm well. Because of this, re-implementing the algorithm as it is presented in the paper would more accurately reflect the reproducibility of the paper. It was well described in the paper, with only a few minor unclear areas that are easily resolved through testing. For example, Equation 6 in the paper is unclear as to whether the sample element $\bar{x}_H$ is part of the sum or not.

To re-implement the N-way K-shot generation procedure, we used the algorithm psuedocode and equations provided in the original paper. As described in the paper's abstract, the algorithm should be able to provide similar results on any dataset and classifier. The models we used were a basic CNN with 2 convolutional layers, and Resnet18, which is one of the models utilised in the original paper. For a dataset, we used the boat dataset (https://artist-cloud.ecs.soton.ac.uk/index.php/s/eAhIkhhdxgmhRHj/download) because it was small and we were already familiar with it. Using Resnet18 from the paper allows us to verify whether our N-way K-shot generation procedure functions as described in the paper, while the custom model and dataset are useful for validating the claim that the procedure will improve performance on any dataset and model. In an ideal scenario, we would also use MiniImageNet, the dataset used in the original project to compare our results more objectively. However, the algorithm is not one that can be GPU-accelerated, and as such would take multiple entire days to run just a single generation process on such a large dataset. Therefore we decided to forego using MiniImageNet.

To verify whether the statistics transfer allows us to properly generate samples in line with the real class distributions, we used t-SNE to project our feature vectors on to 2 dimensions so they can be visualised. The paper uses the same technique, so provides a good comparison between the implementations. We can also take the generated feature vectors and plot them as images, to compare individual samples against real data.

For hyperparameter training, we've used a similar approach to the paper, by running the training procedure and classifier with hyperparameters set across a range of discrete values and results collated at each step. This allows us to compare the results against the graphs provided in the paper.

3 Implementation Details

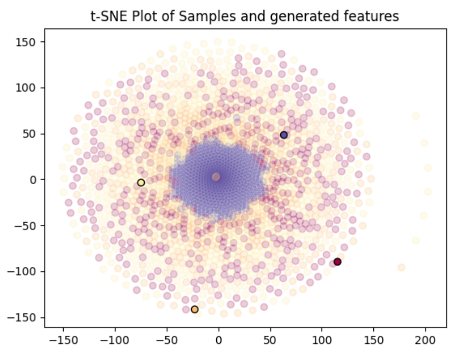
We implemented the algorithm as presented in the original paper such that we could produce the same visualisations and figures as the paper's authors did. The original paper's code utilised the PyTorch library, which we also used instead of alternatives as we were more familiar with it. However, the only PyTorch functions used to implement the algorithm presented by the paper were basic mathematical ones such that the specific implementation of these functions should not influence the reproducibility of our findings.

The original paper claims that the method can be used with generic feature extractors and classification models with minimal modification needed. To attempt to validate this, we used Resnet18 with a simple network and feature extractor. Other than ensuring that the input images are square, no additional changes were needed in connecting the paper's algorithm to the model.

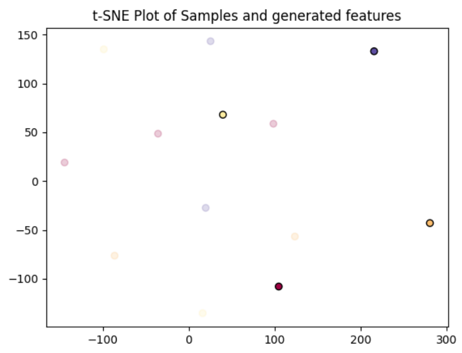
4 Analysis and Evaluations

4.1 Generation

To compare the generated distribution against the values used to make it, we used t-SNE to visualise the original dataset along with the generated set and the validation set for a 5-way 1-shot calibration.



(a) Novel training data with corresponding generated data



(b) Novel training data with validation data

Figure 1: t-SNE plot of the novel training data along with generated data and validation data

Figure 1a shows the novel training set as outlined points and 500 generated features as semi-transparent points. Figure 1b is the same except the semi-transparent points are the validation set. When compared to the graphs shown in the original paper, ours do not display the same tendencies with being centered around the correct points. Rather, the distributions of each generated class seem to be centered around the origin, with some classes largely intersecting.

From this visualisation, it appears that either that our implementation of the algorithm contains errors, or that the base classes used to learn statistics are not applicable to the novel classes for this specific dataset.

4.2 Classification

Tukey Transform	Training with Generated Features	Boat Dataset	
		5 way 1 shot	5 way 5 shot
No	No	44.87 ± 4.46	41.80 ± 3.52
Yes	No	43.89 ± 4.71	43.54 ± 3.83
No	Yes	44.37 ± 4.01	43.55 ± 4.04
Yes	Yes	37.69 ± 4.65	37.36 ± 5.20

Table 1: Data comparing the accuracy when using Tukey transformation and generated features

Having followed their implementation for Tukey transform and generating features as well as 5-way-1-shot and 5-way-5-shot, we generated the table of results as shown in the paper, as well as the accuracy from their hyperparameter training.

Table 1 shows the accuracy our model achieved with combinations of Tukey transform and whether we are generating any features. We see that the accuracy is higher when Tukey transform is not used in preprocessing, and when both Tukey transform and generated features were used, the accuracy was much worse than the other configurations. This is completely opposite from the results shown in the paper which found Tukey transformation with generated features had the best accuracy out of the 4 training procedures. Furthermore, the uncertainty from our training is very high compared to the one in the paper. We believe this is because they ran more rounds in their training, hence collecting more data to generate the accuracy and having a lower variance.

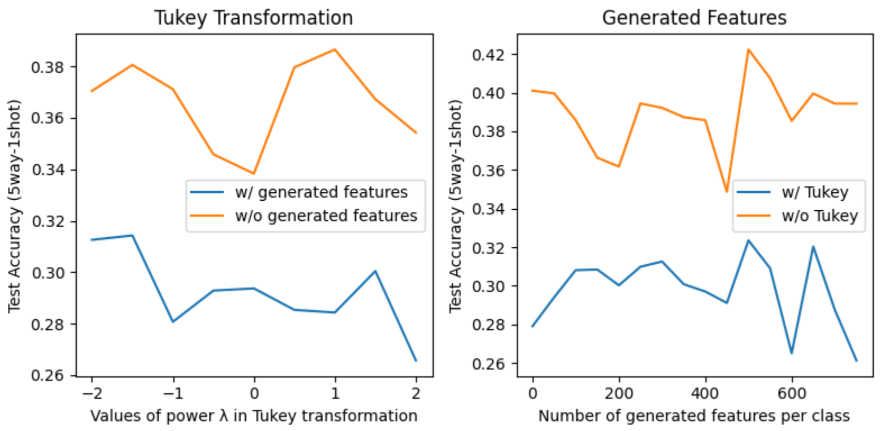


Figure 2: Results from training hyper parameter λ of Tukey transformation and the number of generated features

The results from Figure 2 continues with that trend. We found that over all powers of Tukey, the model which did not have generated features performed better than a model that had generated features. Furthermore, the models without Tukey transform trained better than models with Tukey implemented over all numers of generated features. Compared to the results from the paper, the paper again has generated features and Tukey transform to perform better than not using them. Although the results are vastly different, there are still similarities to be drawn. One of such is from the Tukey results. Our Tukey transform graph shows the models tend to perform the best around λ values of 0.5 and 1. This is consistent with the results from the paper where the accuracy of the models peak when λ is 0.5.

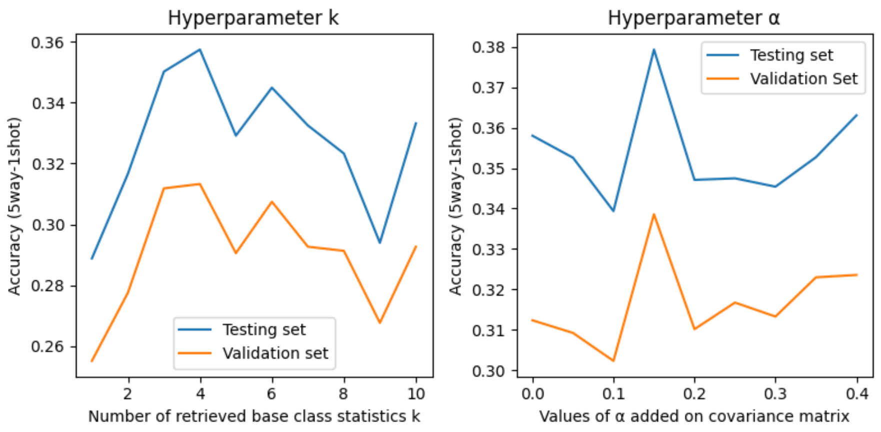


Figure 3: Results from training hyper parameter k and α

The results from training hyperparameters k showed a similar trend to the one in the paper. The models tend to have a lower accuracy when k is at 1, then it rises quickly when k is 2 and 3 and then the accuracy slowly drops back down over time. The graph in the paper shows a similar trend albeit with less fluctuation across the graph. This may be due to the higher number of rounds for collecting data as stated before. Unfortunately, the results from training hyperparameter α did not show the same trend as was in the paper.

Although our results showed that we were not able to reproduce what was shown in the paper, we believe there are multiple factors at hand that may have altered the accuracy of the models. The original paper mentions selecting novel and base classes to generate data from. Different novel and base classes would create wildly different results, therefore without exploring which classes perform better as novel or base, it is hard to reproduce the results from the original paper when not using the same dataset. The same applies to choosing the dataset, as different datasets will see differing performance.

Due to the computation limitations we had, we were not able to train our hyperparameters in a granularity as low as the ones in the paper. The results that were created for this paper took multiple weeks of 24/7 training to produce, while only using the boat dataset which has significantly less data than miniImageNet and the other datasets that were used in the original paper. We believe given more time and computational power, where we can explore finer values for our hyperparameters, as well as testing out the different machine learning models the original paper implemented, we would be able to reproduce similar results as shown in the original paper.

5 Conclusions

Our results indicate that we failed to reproduce the results shown by the original paper. This may indicate a lack of reproducibility. Alternatively, it may just indicate a weakness in the algorithm presented by the original paper - that it does not work well with some datasets, likely those that are fairly small and have difficulty of inter-class transfer. For example, the boat dataset featured very small images of boats on a background of water, so we could not easily find inter-class variance. Although the paper was correct in the assessment that the algorithm can be built on top of off-the-shelf feature extractors, different feature extractors may have improved the beneficial effects of using the algorithm.

As for future improvements, analysing the algorithm using other datasets of differing sizes and varying levels of inter-class variance, such as MNIST and miniImageNet, would more clearly indicate if our implementation and analysis was accurate. A different selection of which classes are chosen as novel classes may improve the performance.