COMS W4995 004 NNDL Group Project - Kaggle Competition

Jason Bram
Columbia University
New York, NY 10027
jb4613@columbia.edu

Nicko Corriveau
Columbia University
New York, NY 10027
nc2942@columbia.edu

Gokul Natesan
Columbia University
New York, NY 10027
ggn2104@columbia.edu

Abstract

Until recently, ResNets made up of convolutional layers have reigned supreme in the image classification realm, but now there is a new contender for first place; Transformers. Of course, Transformers have typically been used in natural language processing tasks, but now it has been made clear that pure transformers applied directly to sequences of image patches can perform very well on image classification tasks. This finding has been attributed to the self-attention mechanism, which is the most prominent feature that distinguishes Transformers from ResNets. As a result, our team has taken an interest in utilizing the capabilities of these two models in the group Kaggle competition, in an attempt to maximize accuracy. In particular, we are keen on observing an ensemble model of a ResNet and Vision Transformer when there is a distribution shift between training and test data. Through our efforts, we attempt to adapt pre-trained models to the task at hand through a variety of data-processing, fine-tuning procedures, and ensembling, thereby putting our own 'improvements' on the models we find, with the intention of winning the competition.

1 Introduction

We participated in the class competition, which is a multi-label image classification problem where each image will belong to one of three super-classes (bird, dog, reptile). Additionally, each image will also include a sub-class, for example hawks within the bird super-class. The goal is to correctly predict the super-class and sub-class of an image given the constraint that the test set is not necessarily the same distribution as the training set. Distribution shifts are evident in numerous real world applications, and can decrease the accuracy of a machine learning system. In this project, we implement data augmentation methods, large feature extraction models ensembled for prediction, and OOD models using a K-Nearest Neighbors approach to handle two common forms of distribution shifts: domain generalization and subpopulation shifts.

2 Related Works

Looking at past image classification Kaggle competitions, there is a common theme; The vast majority are Deep ResNets. Of course, the first thought then points directly to using a large pre-trained ResNet, and adapting it to our own problem in hopes to reproduce a similar accuracy. However, the majority of these competitions are at least 3 years old, and since then there have been some significant advancements in image classification with the emergence of Vision Transformers. As noted by Dosovitskiy, et al. (2021), it is clear that investigating this prospect is also worthwhile, as the accuracy of the proposed Vision Transformers can outperform that of Deep ResNets. Additionally, the emergence of EfficientNets thanks to Tan, et al. (2020) gave rise to a new neural network model that could scale easily and achieve better accuracy and efficiency than that of ConvNets. As a result,

we decided to work primarily with these three types of models as feature extractors, and experiment with an ensemble model that would incorporate predictions from all three of these models.

3 Method/Algorithm

3.1 Data Preprocessing

The first step of our algorithm was to preprocess the data. In order to handle subpopulation shifts in test data, we decided to augment our training data by implementing various transformations. The goal of our data augmentation was to improve generalization of our model by generating new data points to amplify the data set.

3.2 Super-class Model Training

Our focus was to efficiently recognize what super-class each image belonged to in order to cut the search space for the OOD detection and sub-class classification tasks. To solve this image classification problem for the super-class, we decided to implement an ensemble of the following model architectures:

1. Vision Transformers
2. ResNet50
3. EfficientNet

Once each model was trained with specific hyperparameters we then sent the models to an ensemble network that would properly weight each individual model and combine the predictions into a single output. However, as further explained later in the experiments section, we determined that the EfficientNet model actually hurt the predictions of our ensemble model. Therefore, our ensemble simply consisted of the ResNet50 and Vision Transformer models.

3.3 Out-of-Distribution Detection

One of the unique challenges in this problem was that certain parts of the test data did not reflect what was actually seen during training time. Therefore, we needed a mechanism that could deal with this distribution shift by assigning a sub-class of 'novel' to an unseen example within a certain super-class. To deal with this problem, we created feature embeddings for all training images and grouped them according to the super-class. Then, for each test example, we would check for its embedding similarity to the feature embeddings of the coinciding super-class. This would consequently assign a feature score to each test example, where a higher score indicated the increased likelihood that the given example was in-distribution. Finally, we compared a test example's feature score to a predetermined cutoff threshold and assigned a sub-class of 'novel' if the feature score was less than the threshold value.

3.4 Sub-Class Model Training

An important item of note is that separate models needed to be implemented for both super-class and sub-class classification. While the super-class model was an ensemble, given the Vision Transformer models took almost 5 hours to train on a GPU and each one consumed 1GB of RAM from just loading the weights (and 9GB when training), we decided to use pre-trained ResNet50 networks for our sub-class predictions.

4 Experiments/Results

4.1 Data Preprocessing

All of the models used for super-class and sub-class classification were first pre-trained on an ImageNet-1K dataset and imported from PyTorch. These models have all achieved state-of-the-art performance on image classification benchmarks. To increase model generalization, we incorporated the following data augmentations:

- Random color inversions
- Random horizontal flips
- Resizing
- Trivial augmentations

Figure 1 below displays images before and after transformations, including a label of their subclass.

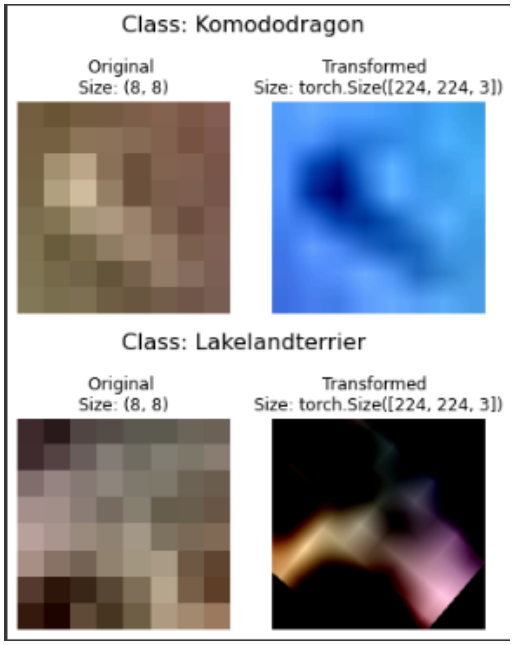

Figure 1: Example Data Augmentations on Training Data.

As displayed in Figure 1, these augmentations increased variance in our training data in hopes of increasing generalization performance. To note, the original data is quite pixelated, so the transformed images follow suit. It is difficult to distinguish classes just from observing the images with the naked eye.

4.2 Super-class Model Training

Training our models for super-class classification involved a two step approach. We first trained classifiers on top of our three large feature extraction networks. After initial training, we performed hyper-parameter optimization on the type of optimizer we use and the learning rate, as displayed in Table 1. We retrained the models using the optimal hyper-parameters with early stopping to achieve best results. The best results for each of three models on training and validation accuracies are displayed in Table 2 below.

Table 1: Model Hyperparameters

Model	Optimizer	Learning Rate
ResNet50	ADAM	0.0005
Vision Transformer	SGD	0.000038
EfficientNet	SGD	0.004016

The EfficientNet model performed significantly worse than the ResNet50 and Vision Transformer models based on the validation accuracy. To improve overall results, we combined our trained models outputs to make an ensemble prediction. The ensemble was created by combining model outputs

Table 2: Training and Validation Accuracies

Model	Training Accuracy (%)	Validation Accuracy (%)
ResNet50	52	61
Vision Transformer	48	49
EfficientNet	41	36

via a shallow neural network, which was tasked with learning the optimal weights of combination to improve validation accuracy. We experimented with using all three models (Resnet50, Vision Transformer, and EfficientNet) and just the top two performers (Resnet50, Vision Transformer). In both cases we were able to improve on the overall accuracy vs the best performing individual model, but found that using just the top two provided slightly better test accuracy (62% vs 61%) and thus used just the Resnet50 and Vision Transformer in the final ensemble.

4.3 Out-of-Distribution Detection

To generate a feature score for each test example, we first used a pre-trained ResNet50 model from the Pytorch Image Models library (timm) to create feature embeddings for the training examples. Once each super-class had all of its training embeddings grouped together, we proceeded to score each test example. Given that we already classified a given test image into a super-class, we used our pre-trained ResNet50 model to create an embedding for the particular test example. Then, with the CleanLab package's OutOfDistribution class, we created distributions for each super-class utilizing the training embeddings. After producing the embedding for a test example, we proceeded to use this vector with our super-class distribution object to generate a feature score for the particular test image. The OutOfDistribution object internally computes the average distance of a test image embedding to its k-nearest neighbors from the super-class training embeddings. With this distance, it would then create a feature score. The scores were on a scale of 0 to 1, where a higher score meant the more likely the image was in-distribution. Therefore, an extremely large average k-nearest neighbors distance would result in a score closer to 0. To classify an image sub-class as novel, we checked to see if the feature score was less than a threshold value of 0.5. As shown below, we included the 15 images from the dog super-class that had the lowest feature scores.

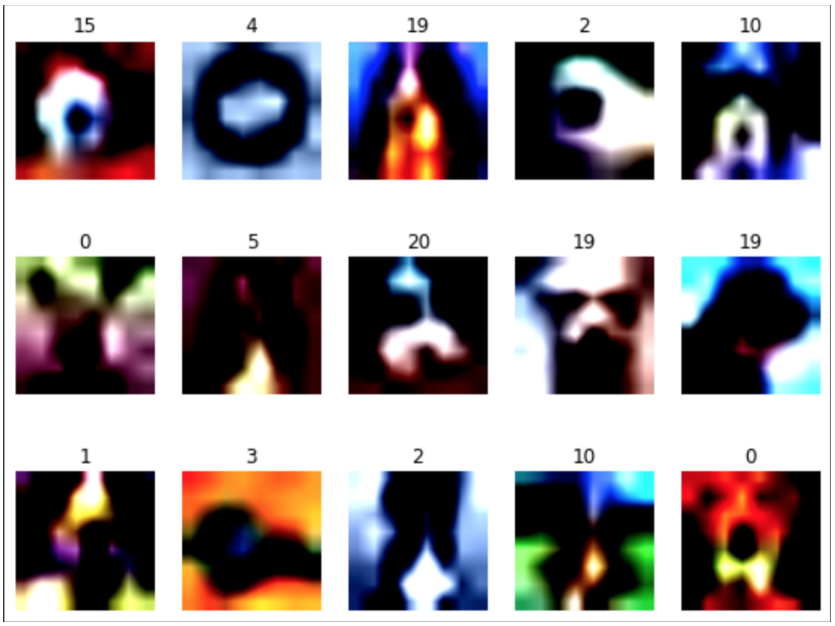

Figure 2: Most Likely OOD Test Examples from Dog Super-class.

4.4 Sub-class Model Training

As part of model selection criteria to handle domain generalization, we explored the validation set strategies described by Gulrajani, et al. (2020) such as having a training-domain validation set, implementing leave-one-domain-out-cross-validation, and using some of our attempts on a test-domain-validation set. Our initial approach considered artificially skewing our validation sets to represent potential domains of the test set, and using the associated model on the test data to guide our model validation. With regards to the overall training and development process for constructing sub-class models, it was very similar to what one would expect. Since we noticed the large discrepancy in accuracy with the provided training and validation data, we decided to only train additional ResNet models. The overall logic was relatively simple; train 3 individual ResNet models on the provided 3 super classes. In doing so, all 3 pre-trained ResNet models were able to adapt well to the training data provided, and achieve incredible accuracy of 95%+ on the provided validation set, as it visible in Table 3.

Table 3: Training and Validation Accuracies for Sub-classes

Model	Training Accuracy (%)	Validation Accuracy (%)
Dog	88	97
Bird	88	97
Reptile	86	97

5 Conclusion

After the countless hours put into training, optimizing, strategizing, and constructing our final model fit for submission, when we finally submitted, there were a few interesting points that we noticed. First and foremost, our methodology for out of sample detection performed incredibly well. We scored 72% for our accuracy in sub-class identification and remain tied for first in terms of performance, and that was after only a single submission. On the other hand, our performance on identifying super-class was very poor. After seeing our ensemble model classify super-class at 62%, it was very strange to see our submission only identify 22% of test sample correctly. We submitted two test predictions overall, but time permitting, further work to increase our super-class classification score could've included 1.) bringing in more training examples from outside sources of dogs, birds, and reptiles 2.) experimenting with three binary classifiers that just predict whether an image is in the category they are trained on or not, and 3.) increasing test submissions to get a better idea of the super class distributions, varying our training distributions and retraining our models accordingly.

Appendix

6 Super-class Ensemble Loss Plots

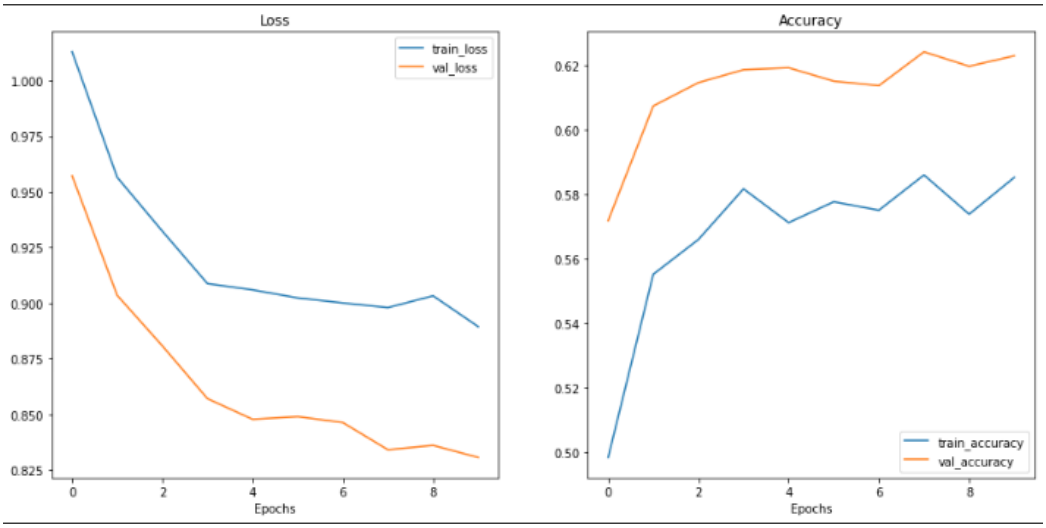

Figure 3: Loss Plot for Super-class Ensemble Model

7 Sub-class Loss Plots

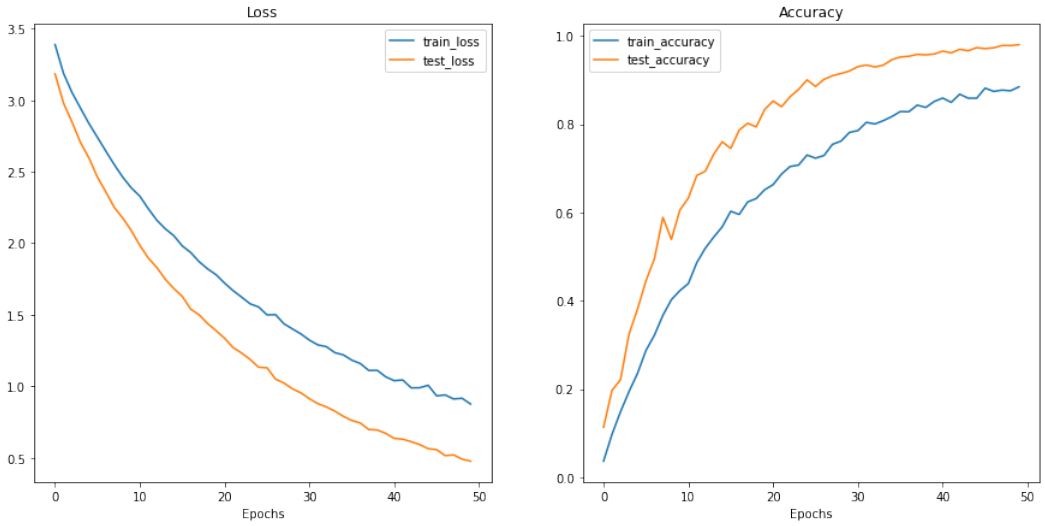

Figure 4: Loss Plot for Dog Sub-class Model

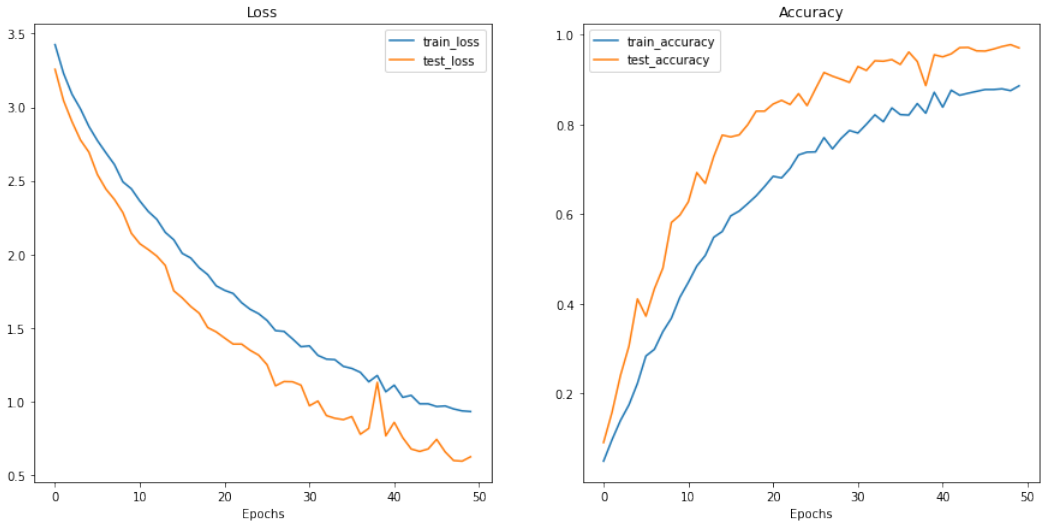

Figure 5: Loss Plot for Bird Sub-class Model

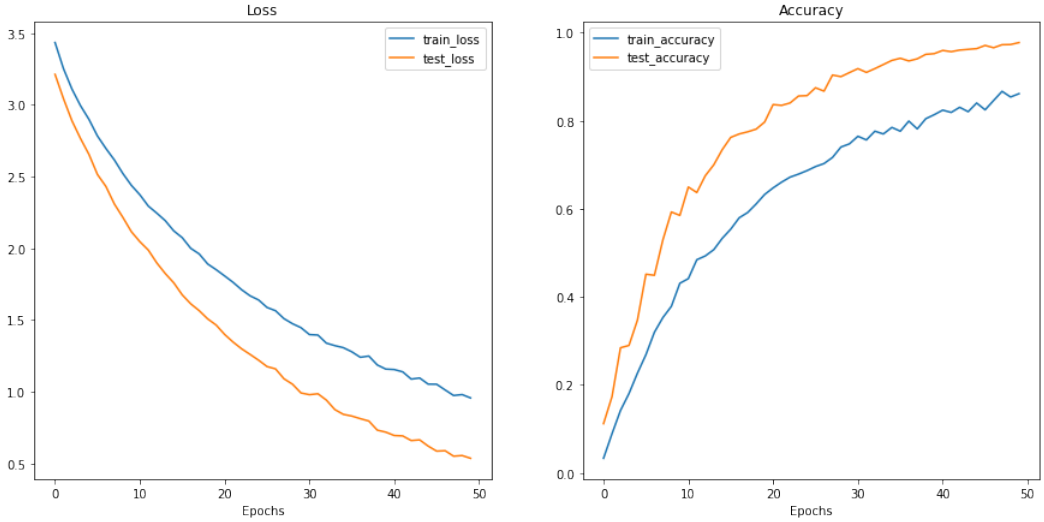

Figure 6: Loss Plot for Reptile Sub-class Model

References

[1] Chen, Chun-Fu, et al. "Crossvit: Cross-Attention Multi-Scale Vision Transformer for Image Classification." ArXiv.org, 22 Aug. 2021, https://arxiv.org/abs/2103.14899v2.

[2] Dosovitskiy, Alexey, et al. "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale." ArXiv.org, 3 June 2021, https://arxiv.org/abs/2010.11929v2.

[3] Fort, Stanislav, et al. "Exploring the Limits of out-of-Distribution Detection." ArXiv.org, 29 July 2021, https://arxiv.org/abs/2106.03004v3.

[4] Gulrajani, Ishaan, and David Lopez-Paz. "In Search of Lost Domain Generalization." ArXiv.org, 2 July 2020, https://arxiv.org/abs/2007.01434.

[5] He, Kaiming, et al. "Deep Residual Learning for Image Recognition." ArXiv.org, 10 Dec. 2015, https://arxiv.org/abs/1512.03385.

[6] Pinto, Francesco, et al. "An Impartial Take to the CNN vs Transformer Robustness Contest." ArXiv.org, 22 July 2022, https://arxiv.org/abs/2207.11347.

[7] Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." ArXiv.org, 11 Sept. 2020, https://arxiv.org/abs/1905.11946.