OUT-OF-DISTRIBUTION CLASSIFICATION AND CLUSTERING

Anonymous authors
Paper under double-blind review

ABSTRACT

One of the long-term goals of machine learning is to develop models which will generalize well enough that they can handle a broader range of circumstances than what they were trained on. In the context of vision, such a model would not get confounded by elements outside of its training distribution. To this end, we propose a new task for training neural networks, in which the goal is to determine if all images in a given set share the same class. We demonstrate that a model trained with this task can classify and cluster samples from out-of-distribution classes. This includes left out classes from the same dataset, as well as entire datasets never trained on. Our experiments also reveal an unreported phenomenon, which is that neural networks can overfit their training classes, leading to poorer out-of-distribution performance. It is our belief that mitigating this effect and improving on our task will lead to better out-of-distribution generalization as well as behaviours more resembling those of humans.

1 INTRODUCTION

A human seeing an elephant for the first time without knowing what such animals are, would quite likely recognize it as something new (out-of-distribution detection). And if this person ever sees an elephant again, he or she should be able to recognize it as the same thing as the one previously seen (out-of-distribution generalization). Designing systems that can handle such out-of-distribution detection and generalization is an open problem in machine learning. Class incremental learning algorithms (Rebuffi et al., 2016; Wu et al., 2019) aim to achieve OOD generalization by training the classifier with new classes as and when they arrive. This retraining of the classifier results in catastrophic forgetting (McCloskey & Cohen, 1989) which is often reduced by replaying examples from previously seen classes (Rebuffi et al., 2016; Chaudhry et al., 2019).

In this paper, we propose an alternate training approach for OOD generalization which is based on the observation that humans don't need class labels to discern among things they have never seen before. Our task requires the model to predict if all samples in a given set come from the same class or not. Such a model can be used as a classifier without any retraining provided it has access to a set of examples from each class that it needs to compare the test sample with. Furthermore, this classifier is not constrained by just the classes used during training since it is agnostic to class labels.

Using the proposed task, we train neural networks on a subset of ImageNet classes (Russakovsky et al., 2014) and show that the model can then correctly classify and cluster samples from the remaining unseen classes. We call these problem settings out-of-distribution classification and out-of-distribution clustering respectively. We also show that the same model trained on ImageNet can correctly classify samples of datasets such as MNIST (Lecun et al., 1998) and CIFAR (Krizhevsky, 2009). We explain how models trained in this fashion could be used in an online setting, where the model would be able to build new classes from samples not belonging to any known class. Alongside, we cover the known issues preventing us from getting there. One such roadblock is the problem of overfitting. It is well known that neural networks can overfit their training data. We show that there is an even more pernicious effect happening, which is that neural networks can overfit their training classes. This directly affects out-of-distribution performance and seems to happen much sooner than regular overfitting.

2 DO CLASSIFIERS EVEN NEED RETRAINING?

Traditionally, when we want a model to be able to classify a new class, we must extend the size of the classifier's output and perform additional training (Rebuffi et al., 2016; Lopez-Paz & Ranzato, 2017; Kirkpatrick et al., 2016). Given that neural network training is time consuming, many different work aim to make this process more efficient (Santoro et al., 2016; Mishra et al., 2017; Finn et al., 2017). Nevertheless solutions remain far from perfect. But, does a classifier actually need to be retrained or would it be possible to generalize to new classes without retraining? To shed some light on these questions, we trained three ResNets, each one on different subsets of ImageNet. One was trained on 800 random ImageNet classes, another one on all classes except canines and the last one on all classes except birds. We then asked the models to classify randomly picked images from classes they didn't train on and applied guided backpropagation to see what features the ResNets focused on. As a comparison, we applied guided backpropagation to the same images with a ResNet trained on all ImageNet classes. Results are presented in Figure 1. As we can see, the baseline and the ResNet which was trained on 800 random classes seem to be focusing on the same features. As for the ResNets which haven't trained on canines and birds, they seem to be picking slightly more details than the baseline. This suggests that a neural network might not necessarily need to be trained on a class for it to be able to discern its features.

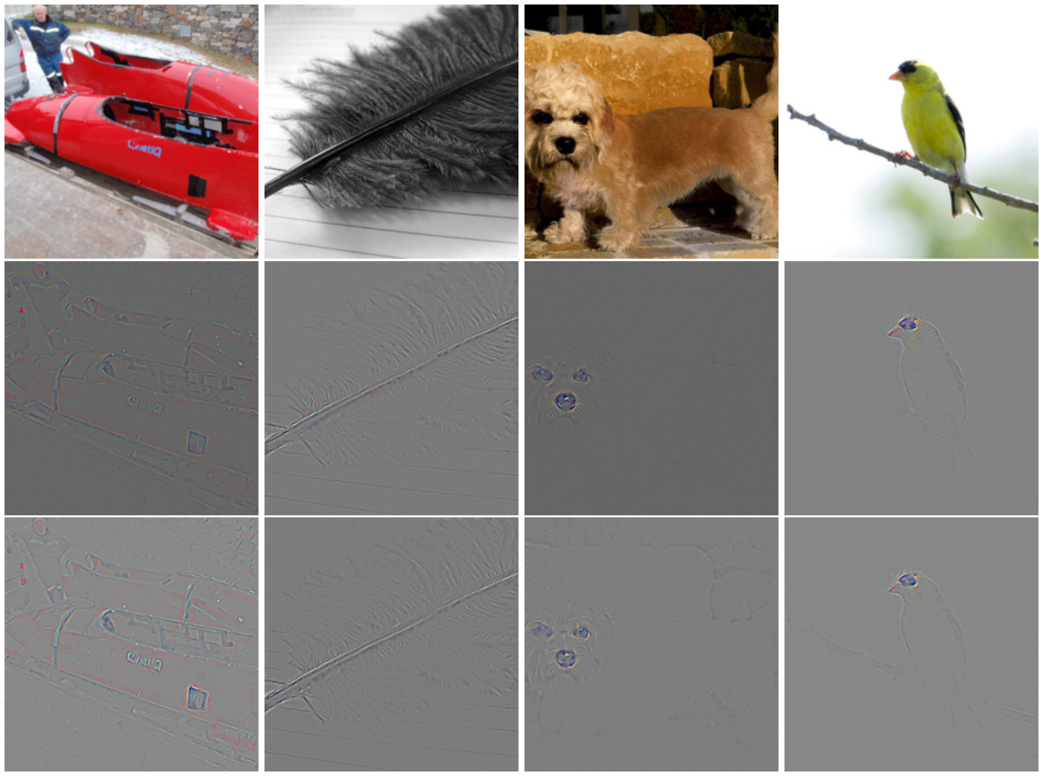

Figure 1: Guided backpropagation of ResNets trained on different subsets of ImageNet. First row is original image. Middle row is the gradient of a ResNet trained on all ImageNet classes. Last row, the first two columns are from a ResNet trained on 800 random ImageNet classes and the last two columns from one ResNet not trained on any canine classes and one not trained on any bird classes.

Our observation that neural networks need not be retrained to recognize features in new classes is not new though. This is precisely the motivation for few-shot meta-learning methods like prototypical networks (Vinyals et al., 2016; Snell et al., 2017). However, prototypical networks exploited this observation only to design classifiers that can perform few-shot classification among a small set of new classes. On the contrary, this paper focuses on classifiers that can classify the given sample among all the seen and unseen classes with only few examples per class.

3 METHODOLOGY

3.1 CLASS HOMOGENEITY

Traditional classification networks output a fixed number of class predictions. This makes it impossible to test their out-of-distribution classification performance. To handle any number of classes at both training and testing time, we propose a new task for neural network training. This task will allow us to evaluate a model for both out-of-distribution classification and out-of-distribution clustering.

The task is defined as follows: given a set of sample-label pairs $\{(x_1, y_1), \cdots, (x_k, y_k)\}$, where x_i is the i-th sample, y_i is its corresponding label and k is a hyperparameter, the task is to determine, using only the samples $\{x_1, \cdots, x_k\}$, if all such samples share the same label ($y_1 = \cdots = y_k$). We call this binary classification task: class homogeneity. The model is not given the explicit class label as supervision. Instead, the class label is only used to construct the binary targets representing whether the samples are from the same distribution/class or not.

3.2 OUT-OF-DISTRIBUTION CLASSIFICATION AND CLUSTERING

We begin by defining out-of-distribution classification and out-of-distribution clustering and then explain how these can be measured using our class homogeneity task. Given a model M trained on a set of sample-label pairs $\{(x_1, y_1), \cdots, (x_n, y_n)\}$, where n is the size of the training set and Y_{train} is the set of labels in the training set. If at test time, M is used to classify $\tilde{x}$ from a sample-label pair $(\tilde{x}, \tilde{y} \mid \tilde{y} \notin Y_{train})$, then we consider this to be out-of-distribution classification. Similarly, if we use M to cluster a set of samples $\{x_1, \cdots, x_n\}$ from $\{(x_1, y_1), \cdots, (x_n, y_n) \mid \exists y_i \notin Y_{train}\}$ where n is the number of samples to cluster, then we consider this to be out-of-distribution clustering.

In other words, when classifying a sample, if its label was never seen by the classifier during training, then we consider this to be out-of-distribution classification. Likewise, when clustering samples, if at least one of the samples' label was never seen by the clusterer during training, then we consider this to be out-of-distribution clustering.

Once a model M has been trained on the class homogeneity task, we can evaluate it for both out-of-distribution classification and out-of-distribution clustering. For the former, in which we are given $\tilde{x}$ from a sample-label pair $(\tilde{x}, \tilde{y} \mid \tilde{y} \notin Y_{train})$, we can classify $\tilde{x}$ by comparing it with samples of each class. That is, we take $k - 1$ samples from every class and predict the label as:

$$\hat{y} = \arg\max_{y \in Y} M(x_1^y, \cdots, x_{k-1}^y, \tilde{x}), \tag{1}$$

where $x_1^y, \cdots, x_{k-1}^y$ are samples with corresponding class y and Y is the set of all classes for which we have $k - 1$ samples to compare with, which also includes classes not trained on. If $\hat{y} = \tilde{y}$, then the model has successfully classified the out-of-distribution sample $\tilde{x}$.

For clustering, there are many ways to predict the labels. We consider the general case here and detail our own setup in the experiment section 5.3. In general, given a set of samples $X = \{x_1, \cdots, x_n\}$ to be clustered, the samples of a subset $S \subseteq X : 2 \leq |S| \leq k$ should be identically labelled if $M(S) > 0.5$. Once all samples are labelled, we can evaluate the clustering accuracy via:

$$accuracy(\{\hat{y}_1, \cdots, \hat{y}_n\}, \{y_1, \cdots, y_n\}) = \max_{p \in P} \sum_{i=1}^{n} \mathbb{1}\left(p(\hat{y}_i) = y_i\right), \tag{2}$$

where $\hat{y}_i$ are the predicted labels, y_i are the targets and P the set of all permutations of the interval $[1, C]$, with C being the number of predicted clusters.

3.3 IMPLEMENTATION FOR IMAGE CLASSIFICATION AND CLUSTERING

In past sections we've described the general definition of our new task and how it can be used to measure out-of-distribution performance. We now detail the specific training setup we used for our image classification and clustering experiments.

We first pretrain to convergence a ResNet-18 (He et al., 2015) to classify the first 800 classes of our modified ImageNet dataset (see section 5.1). Once pretraining is over, the ResNet's final classification layer is discarded. We then train the ResNet and a Transformer encoder (Vaswani et al., 2017) with our class homogeneity task. Specifically, we train two variants, one for out-of-distribution classification and one for out-of-distribution clustering. Note that any of the two could be used for both. Variations are used to improve performance.

For the first one, a training sample is composed of $k = 25$ images, where 24 images are always from the same class and the 25-th image is equally likely to be either from the same or a different class. Both images and classes are sampled uniformly at random. The ResNet first encodes the 25 images, after which the representations are fed to the Transformer and finally passed through a classification layer with a sigmoid activation function. The expected output is the probability that all the images come from the same class. This setup is intended to be used for out-of-distribution classification.

The second variation is exactly the same as the first one, except that we vary the number of images from the same class between 1 and 25. This number is chosen uniformly at random for each training sample. This variation is intended to be used for out-of-distribution clustering.

Additional training details can be found in Appendix section A.1.

4 Related Work

Over the years, many new fields in machine learning have emerged with the shared goal of empowering neural networks with the ability to quickly learn new concepts and adapt to new environments. One such line of work called Class Incremental Learning learns new classes while trying not to forget previous ones. In order to avoid catastrophic forgetting, some methods propose to retrain on a subset of already seen samples while training on the new classes (Rebuffi et al., 2016; Wu et al., 2019; Hou et al., 2019). Another possible approach is to avoid gradient steps which would hurt performance on previous tasks (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018). It has also been proposed that a regularization term be added to the loss in order to penalize changes to parameters important to previous tasks (Kirkpatrick et al., 2016). Our approach is different from these methods since we can classify examples from unseen classes without any additional training.

Few-shot learning focuses on designing models that can learn new classes with just a few samples (Santoro et al., 2016; Mishra et al., 2017; Finn et al., 2017). A more extreme version called zero-shot learning tries to label samples of unseen classes without even training on them. Such methods usually try to encoded images into words (Palatucci et al., 2009; Frome et al., 2013; Norouzi et al., 2013; Socher et al., 2013) or sets of attributes (Lampert et al., 2009; Romera-Paredes & Torr, 2015). More similar to our work, prototypical networks try to classify samples of unseen classes by comparing them with other samples (Koch, 2015; Vinyals et al., 2016; Snell et al., 2017). The main difference between their work and ours is that when classifying a sample, these methods compare it with far fewer classes, 5 to 20, instead of the whole dataset. Also, these approaches still train the model with supervised learning loss while we train our model with the class homogeneity prediction loss.

Recently, Ren et al. (2020) extended prototypical networks (Snell et al., 2017) to online contextualized few-shot learning setting. While Ren et al. (2020) learn to acquire new classes, they are similar to class incremental learning in the sense that they do train the model on new classes. Similarly, Gidaris & Komodakis (2018) also learns to acquire new classes by having an additional training phase to learn the parameters for new classes in a few-shot way. Our approach is much simpler than both these approaches and most importantly, we don't need to retrain on new classes. Also related to our work are the works that focus solely on detecting out of distribution samples (Hendrycks & Gimpel, 2016; Liang et al., 2017; Lee et al., 2017; DeVries & Taylor, 2018) and works that tries to mitigate possible drops in performance caused by potential variance between the test and train distribution (Carlucci et al., 2019; Arjovsky et al., 2019; Gulrajani & Lopez-Paz, 2020).

5 Experiments

5.1 Modified ImageNet

We used ImageNet's validation set as the test set and the first 50 training samples of each class as the validation set. The class order was randomly shuffled. Only the first 800 classes were used for training, validation was done on classes 1-900 and testing on all classes. Everywhere where we mention ImageNet, we mean our modified version.

5.2 Out-of-Distribution Classification

After training our model on ImageNet using the class homogeneity task, we evaluated its out-of-distribution classification performance. To do so, we sampled an image and compared it to 24 images of each class. The highest output probability out of all 1000 ImageNet classes was considered the predicted class. Results were averaged by classifying one thousand images from each class. Images were always drawn uniformly at random per class, from the test set.

For comparison, we report the performance of a ResNet-18 trained to convergence as a classifier on all ImageNet classes as well as one trained on 800 classes. This serves as an upper bound for the performance we can achieve. We also report results for five nearest neighbor variants. In the first nearest neighbor variant, we sampled an image and computed its distance in euclidean space with 24 images of each class. We then averaged the 24 distances and chose the closest class as the prediction. The other four nearest neighbor variations were the same, except that images were first encoded by a ResNet-18 with no classification layer. Respectively, one was a ResNet trained to convergence as a classifier on the first 800 ImageNet classes and the other was the ResNet which we fine-tuned using our class homogeneity task. For both ResNets, we measured nearest neighbors in euclidean space and also with cosine similarity. For all nearest neighbor experiments, performance was averaged by classifying a thousand images for each class, sampling them from the test set uniformly at random.

Table 1: Classification accuracy on our ImageNet test set for various class ranges. Images to be classified are always compared against all thousand classes. We report mean accuracy and standard deviation of five trials. Pretrained embeddings are from the ResNet-18 trained on 800 classes and fine-tuned embeddings from the ResNet-18 trained with our class homogeneity task.

Method	Top-1 Accuracy			
	1-800	801-900	901-1000	1-1000
ResNet-18 (Trained on all classes)	69.65 ± 0.13	70.40 ± 0.25	71.99 ± 0.26	69.96 ± 0.09
ResNet-18 (Trained on 800 classes)	$\mathbf{71.56 \pm 0.19}$	-	-	-
Euclidean Nearest Neighbor (Image Space)	0.55 ± 0.00	0.13 ± 0.00	1.19 ± 0.00	0.58 ± 0.00
Euclidean Nearest Neighbor (Pretrained Embeddings)	51.09 ± 0.24	38.86 ± 0.35	41.85 ± 0.56	48.95 ± 0.17
Euclidean Nearest Neighbor (Fine-tuned Embeddings)	53.15 ± 0.09	41.51 ± 0.23	43.87 ± 0.30	51.06 ± 0.08
Cosine Nearest Neighbor (Pretrained Embeddings)	56.67 ± 0.18	37.28 ± 0.47	38.59 ± 0.21	52.93 ± 0.15
Cosine Nearest Neighbor (Fine-tuned Embeddings)	57.06 ± 0.25	35.70 ± 0.32	37.00 ± 0.34	52.92 ± 0.19
Class Homogeneity	56.78 ± 1.46	$\mathbf{44.13 \pm 0.16}$	$\mathbf{47.47 \pm 0.40}$	$\mathbf{54.59 \pm 1.15}$
Method	Top-5 Accuracy			
	1-800	801-900	901-1000	1-1000
ResNet-18 (Trained on all classes)	89.15 ± 0.09	90.13 ± 0.13	90.34 ± 0.24	89.37 ± 0.07
ResNet-18 (Trained on 800 classes)	$\mathbf{90.04 \pm 0.09}$	-	-	-
Euclidean Nearest Neighbor (Image Space)	2.34 ± 0.00	0.25 ± 0.00	3.76 ± 0.00	2.19 ± 0.00
Euclidean Nearest Neighbor (Pretrained Embeddings)	75.80 ± 0.15	71.35 ± 0.12	74.38 ± 0.22	75.21 ± 0.13
Euclidean Nearest Neighbor (Fine-tuned Embeddings)	77.29 ± 0.23	74.95 ± 0.24	76.36 ± 0.41	76.97 ± 0.20
Cosine Nearest Neighbor (Pretrained Embeddings)	81.28 ± 0.14	72.32 ± 0.24	73.92 ± 0.27	79.65 ± 0.16
Cosine Nearest Neighbor (Fine-tuned Embeddings)	82.24 ± 0.16	72.29 ± 0.15	74.05 ± 0.21	80.43 ± 0.14
Class Homogeneity	82.99 ± 0.72	$\mathbf{76.39 \pm 0.27}$	$\mathbf{77.99 \pm 0.36}$	$\mathbf{81.83 \pm 0.62}$

Reported results are in Table 1. The test accuracy is broken down into various class ranges: classes seen during training (1-800), classes used for validation (801-900) and classes kept for testing (901-1000). We also report the cumulative test accuracy (1-1000). The ResNet trained on all classes is there to show the accuracy of a traditional classifier. For classes seen during training, our model's accuracy is not on par yet with a ResNet classifier trained on the same classes. Maybe there is a trade-off to be made between seen and unseen performance as section 5.4 suggests. We hope though that in future work this performance can be increased. As for classes not seen during training, the model trained using our class homogeneity task performed best.

For further analysis, we also report per class range classification accuracy, where samples to be classified are only compared against classes in each respective range. Results can be found in Table 2. As more classes are considered, the accuracy gradually degrades. Again out of all methods, the model trained with our class homogeneity task performed best.

Table 2: Classification accuracy on our ImageNet test set for various class ranges. This time, images to be classified are only compared against classes in the respective range. We report mean accuracy and standard deviation of five trials. Pretrained embeddings are from the ResNet-18 trained on 800 classes and fine-tuned embeddings from the ResNet-18 trained with our class homogeneity task.

Method	Top-1 Accuracy				
	901-910	901-920	901-940	901-980	901-1000
ResNet-18 (Trained on all classes)	93.80 ± 0.68	93.92 ± 0.27	93.30 ± 0.24	91.67 ± 0.21	91.04 ± 0.096
Euclidean Nearest Neighbor (Image Space)	18.36 ± 0.00	14.07 ± 0.00	6.62 ± 0.00	4.35 ± 0.00	3.65 ± 0.00
Euclidean Nearest Neighbor (Pretrained Embeddings)	85.11 ± 0.73	80.13 ± 0.64	78.67 ± 0.35	71.93 ± 0.35	71.51 ± 0.40
Euclidean Nearest Neighbor (Fine-tuned Embeddings)	83.98 ± 1.02	77.87 ± 0.47	77.44 ± 0.40	72.97 ± 0.41	72.65 ± 0.43
Cosine Nearest Neighbor (Pretrained Embeddings)	88.05 ± 0.45	83.27 ± 0.25	81.25 ± 0.25	76.82 ± 0.26	75.77 ± 0.29
Cosine Nearest Neighbor (Fine-tuned Embeddings)	88.52 ± 0.43	83.78 ± 0.29	81.17 ± 0.14	77.27 ± 0.19	76.22 ± 0.14
Class Homogeneity	**88.83 ± 0.34**	**84.87 ± 0.28**	**83.14 ± 0.24**	**79.18 ± 0.39**	**78.43 ± 0.49**

5.2.1 CLASSIFYING OUT-OF-DISTRIBUTION DATASETS

Using our model trained on ImageNet, we also tried to classify the test sets of MNIST (Lecun et al., 1998), Fashion-MNIST (Xiao et al., 2017), CIFAR10 and CIFAR100 (Krizhevsky, 2009). Performance of ResNet-18 trained on the individual datasets is reported as the upper bound. The nearest neighbor methods used the same ResNets trained on ImageNet as in section 5.2. For each experiment, we classified ten thousand images of each class. Mean accuracy and standard deviation of five trials is reported in Table 3. For FashionMNIST and CIFAR100, computing cosine similarity with ResNet embeddings fine-tuned using our task performed much better than the ResNet and Transformer model. As for CIFAR10, our model performed best, only 7 points behind the ResNet classifier trained on the dataset. We highly suspect that the poorer performance for MNIST and FashionMNIST be due to Transformer overfitting the ImageNet distribution, as using our fine-tuned ResNet embeddings scored much higher. Since CIFAR10 and CIFAR100 are closer distribution wise to ImageNet, this might not affect them.

Table 3: Classification accuracy on the test set of several out-of-distribution datasets. The first reported ResNet-18 was independently trained on each dataset. For all other methods, models were trained on ImageNet. We report mean accuracy and standard deviation of five trials. Pretrained embeddings are from a ResNet-18 trained on 800 ImageNet classes and fine-tuned embeddings from the ResNet-18 trained with our class homogeneity task.

Method	Top-1 Accuracy			
	MNIST	Fashion-MNIST	CIFAR-10	CIFAR100
ResNet-18 (One trained on each dataset)	99.20 ± 0.06	91.51 ± 0.27	77.77 ± 0.89	48.90 ± 0.46
Euclidean Nearest Neighbor (Image Space)	63.68 ± 0.00	60.15 ± 0.00	20.11 ± 0.00	6.05 ± 0.00
Euclidean Nearest Neighbor (Pretrained Embeddings)	75.26 ± 2.49	66.45 ± 1.44	66.21 ± 0.80	37.50 ± 0.62
Euclidean Nearest Neighbor (Fine-tuned Embeddings)	72.87 ± 1.69	67.24 ± 0.69	65.40 ± 0.77	37.51 ± 0.83
Cosine Nearest Neighbor (Pretrained Embeddings)	**76.24 ± 1.38**	67.30 ± 1.06	66.48 ± 0.75	41.86 ± 0.27
Cosine Nearest Neighbor (Fine-tuned Embeddings)	74.14 ± 1.68	**69.01 ± 0.51**	69.88 ± 0.83	**46.10 ± 0.49**
Class Homogeneity	58.99 ± 3.49	64.88 ± 3.37	**70.53 ± 0.96**	42.91 ± 0.68

5.3 OUT-OF-DISTRIBUTION CLUSTERING

Using our models trained with the second variation of our class homogeneity task (see section 3.2), we clustered from scratch images from out-of-distribution classes. Specifically, we set up five experiments in which we gradually increased the number of classes, starting from ten, all the way to one hundred. Images to be clustered were solely from unseen classes. For each class in the experiment, we added its 50 test samples to the pool of images to be clustered.

For clustering, we used the following algorithm. Let $X = \{x_1, \cdots, x_n\}$ be the set of samples to cluster and L the initial empty set of labelled samples. First choose the highest pairwise prediction between all images:

$$x_i, x_j = \arg\max \{M(x_i, x_j) \mid \forall x_i, x_j \in X, x_i, x_j \notin L, i \neq j, [M(x_i, x_j) > 0.5]\}, \qquad (3)$$

and identically label both samples x_i, x_j. Create a new cluster $Q = \{x_i, x_j\}$ and compute for all images not yet labelled the highest probability that an image belongs to the cluster:

$$x_i = \arg\max \{M(Q, x_i) \mid x_i \in X, x_i \notin L, [M(Q, x_i) > 0.5]\}. \qquad (4)$$

Add x_i to Q and label it accordingly. Repeat this cycle until no further images are added to the cluster. Then compute equation 3 again and create a new cluster. Once no more clusters can be built, we are done. Since $k = 25$, if a cluster gets bigger than 24 images, we compute equation 4 by choosing uniformly at random 24 images from that cluster.

We compared our performance against five other methods. For the first one, we clustered image embeddings using k-means. Images were encoded by a ResNet-18 trained to convergence for image classification on all ImageNet classes. For the second method, we used these same embeddings and reduced their dimensionality to 2 via PCA (F.R.S., 1901), then applied t-SNE (van der Maaten & Hinton, 2008) and finally clustered them using DBSCAN (Ester et al., 1996) with $\epsilon = 1.75$ and 5 samples minimum. We also tried two variants to the second method, one where the embeddings were instead encoded by a ResNet-18 trained for classification on the first 800 classes of ImageNet and another by the ResNet which we fine-tuned with our class homogeneity task. For the last method that we tried, we reduced the dimensionality of the images themselves to 2 via PCA, applied t-SNE and clustered them with DBSCAN $\epsilon = 2.3$ and one minimum sample.

Table 4: Clustering accuracy on our ImageNet test set. For each class range, the set of samples to cluster was all 50 test samples of each class in the range. We report mean accuracy and standard deviation of five trials. The first two methods used embeddings from a ResNet-18 trained on all ImageNet classes. Pretrained embeddings are from a ResNet-18 trained on 800 ImageNet classes and fine-tuned embeddings from the ResNet-18 trained with our class homogeneity task. We used k-means with k set to the number of classes in each respective range. For PCA, we reduced the dimensionality to 2.

Method	Clustering Accuracy				
	901-910	901-920	901-940	901-980	901-1000
K-means (Embeddings trained on all classes)	89.04 ± 3.36	84.19 ± 2.50	81.35 ± 2.10	75.59 ± 1.59	75.32 ± 1.12
PCA + t-SNE + DBSCAN (Embeddings trained on all classes)	48.24 ± 2.80	32.74 ± 1.31	22.04 ± 0.75	16.20 ± 0.63	14.12 ± 0.35
PCA + t-SNE + DBSCAN (Images)	16.48 ± 0.80	9.65 ± 1.14	7.53 ± 0.26	5.33 ± 0.36	4.22 ± 0.21
PCA + t-SNE + DBSCAN (Pretrained Embeddings)	47.98 ± 2.95	29.22 ± 3.83	20.05 ± 1.34	14.79 ± 0.27	13.05 ± 0.31
PCA + t-SNE + DBSCAN (Fine-tuned Embeddings)	53.40 ± 4.84	33.47 ± 2.17	21.93 ± 1.32	$\mathbf{15.25 \pm 0.49}$	$\mathbf{13.48 \pm 0.49}$
Class Homogeneity	$\mathbf{57.12 \pm 1.99}$	$\mathbf{41.64 \pm 4.59}$	$\mathbf{34.42 \pm 2.99}$	1.00 ± 0.00	5.59 ± 9.18

We report the mean clustering accuracy and standard deviation of five trials in Table 4. We also showcase a t-SNE plot of our clustering performance for the class range 901-910 in Appendix-A.3. Although k-means scored quite well, the method requires the number of expected clusters as a hyperparameter, unlike DBSCAN and our method. For class ranges up to 901-940, our model performed best, even better than the DBSCAN method using embeddings trained on all classes. For larger numbers of classes, our model's performance drastically fell as it clumped every sample into one big cluster. This might be due to the bottom-up algorithm which we chose to use. There are many different ways that we could of cluster samples using our models. The objective here was simply to demonstrate the ability to cluster out-of-distribution samples. In the classification experiments, we compared against $k - 1$ images of each unseen class. For clustering, we start from scratch. This makes the task much more difficult. The second variant of our class homogeneity task with which we trained our models is also more difficult than the first. Models consistently scored lower training and validation accuracy on that task.

5.4 Overfitting Training Classes

The usual consensus in machine-learning is that a model has overfitted its training data when the validation performance starts to go down. We show that neural networks can overfit the training data much sooner than that. When training our models using our homogeneity class, we noticed that while the validation accuracy of seen classes kept going up, the validation accuracy of classes not trained on went up initially, but then quickly degraded (see Figure 2). This means that neural networks can overfit their training classes. Because of this overfitting, our models could only be fine-tuned on the homogeneity task for less than 3k steps on average. This we think is a major handicap to our model's performance. As for work other than ours, we suspect state-of-the-art image classifiers, despite their superhuman performance, to probably extremely overfit their training classes. For such models to perform well out-of-distribution, we think new methods to avoid overfitting training classes will have to be developed. In our case, we have experimented with weight-decay without any success. It is also possible that areas other than image classification be affected.

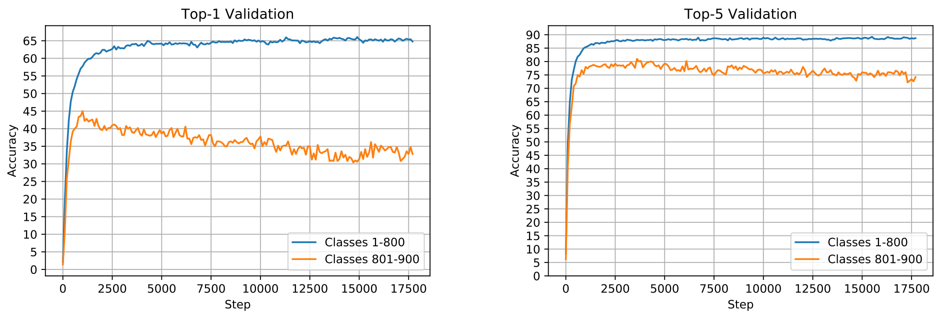

Figure 2: Validation accuracy when training a model using our class homogeneity task. Accuracy is reported for both classes seen during training (1-800) and classes kept for validation (801-900). As we can see, the accuracy drops overtime for unseen classes while the accuracy for seen classes goes up. This indicates that the model is overfitting its training classes.

6 Conclusion & Future Work

In this work, we proposed a new task for training neural networks. Using this task, we showed that we could classify and cluster samples from classes that the network was not trained on. Our work covered the classification and clustering of images, but we see no reasons as to why our approach could not work when applied to other domains. There are many untried variations to our approach and potential research directions. We believe that our class homogeneity task with OOD classification and OOD clustering evaluations could help the community to make measurable progress in OOD generalization of neural networks.

In our classification experiments, images were compared to every class, including the target class. In a more realistic setting, this wouldn't necessarily be possible. A sample might be the first of a new class that we encounter. What we can do though, is if a model is good enough to not classify a sample from a new class with any known class, we can then set that sample aside and attribute it a new label. Once another sample from that same new class shows up, we would compare it with all known classes, including the new class we created. Again, if the model is good enough, it could classify it as belonging to the new class. This example is with one new class, but there could potentially be many new classes which the model would have to build concurrently. In a real-world setting, a model being able to do such a task would be very useful. As of now our models do not perform well enough to be able to do so, but by improving the out-of-distribution performance, we believe that we can get there.

REFERENCES

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant Risk Minimization. *arXiv e-prints*, art. arXiv:1907.02893, July 2019.

Fabio Maria Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain Generalization by Solving Jigsaw Puzzles. *arXiv e-prints*, art. arXiv:1903.06864, March 2019.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient Lifelong Learning with A-GEM. *arXiv e-prints*, art. arXiv:1812.00420, December 2018.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On Tiny Episodic Memories in Continual Learning. *arXiv e-prints*, art. arXiv:1902.10486, February 2019.

Terrance DeVries and Graham W. Taylor. Learning Confidence for Out-of-Distribution Detection in Neural Networks. *arXiv e-prints*, art. arXiv:1802.04865, February 2018.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. KDD'96, pp. 226–231. AAAI Press, 1996.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv e-prints*, art. arXiv:1703.03400, March 2017.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26*, pp. 2121–2129. Curran Associates, Inc., 2013. URL `http://papers.nips.cc/paper/5204-devise-a-deep-visual-semantic-embedding-model.pdf`.

Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11): 559–572, 1901. doi: 10.1080/14786440109462720. URL `https://doi.org/10.1080/14786440109462720`.

Spyros Gidaris and Nikos Komodakis. Dynamic Few-Shot Visual Learning without Forgetting. *arXiv e-prints*, art. arXiv:1804.09458, April 2018.

Ishaan Gulrajani and David Lopez-Paz. In Search of Lost Domain Generalization. *arXiv e-prints*, art. arXiv:2007.01434, July 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv e-prints*, art. arXiv:1512.03385, December 2015.

Dan Hendrycks and Kevin Gimpel. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. *arXiv e-prints*, art. arXiv:1610.02136, October 2016.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, art. arXiv:1412.6980, December 2014.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *arXiv e-prints*, art. arXiv:1612.00796, December 2016.

Gregory R. Koch. Siamese neural networks for one-shot image recognition. 2015.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Christoph H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 951–958, 2009.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples. *arXiv e-prints*, art. arXiv:1711.09325, November 2017.

Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks. *arXiv e-prints*, art. arXiv:1706.02690, June 2017.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. *arXiv e-prints*, art. arXiv:1706.08840, June 2017.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A Simple Neural Attentive Meta-Learner. *arXiv e-prints*, art. arXiv:1707.03141, July 2017.

Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S. Corrado, and Jeffrey Dean. Zero-Shot Learning by Convex Combination of Semantic Embeddings. *arXiv e-prints*, art. arXiv:1312.5650, December 2013.

Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta (eds.), *Advances in Neural Information Processing Systems 22*, pp. 1410–1418. Curran Associates, Inc., 2009. URL `http://papers.nips.cc/paper/3650-zero-shot-learning-with-semantic-output-codes.pdf`.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental Classifier and Representation Learning. *arXiv e-prints*, art. arXiv:1611.07725, November 2016.

Mengye Ren, Michael L. Iuzzolino, Michael C. Mozer, and Richard S. Zemel. Wandering Within a World: Online Contextualized Few-Shot Learning. *arXiv e-prints*, art. arXiv:2007.04546, July 2020.

Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. volume 37 of *Proceedings of Machine Learning Research*, pp. 2152–2161, Lille, France, 07–09 Jul 2015. PMLR. URL `http://proceedings.mlr.press/v37/romera-paredes15.html`.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *arXiv e-prints*, art. arXiv:1409.0575, September 2014.

Adam Santoro, Sergey Bartunov, M. Botvinick, Daan Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical Networks for Few-shot Learning. *arXiv e-prints*, art. arXiv:1703.05175, March 2017.

Richard Socher, Milind Ganjoo, Hamsa Sridhar, Osbert Bastani, Christopher D. Manning, and Andrew Y. Ng. Zero-Shot Learning Through Cross-Modal Transfer. *arXiv e-prints*, art. arXiv:1301.3666, January 2013.

Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv e-prints*, art. arXiv:1706.03762, June 2017.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. *arXiv e-prints*, art. arXiv:1606.04080, June 2016.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large Scale Incremental Learning. *arXiv e-prints*, art. arXiv:1905.13260, May 2019.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv e-prints*, art. arXiv:1708.07747, August 2017.

A Appendix

A.1 Additional Training Details

We found the following hyperparameters to work best. Our Transformer has 6 layers and 16 heads of size 64 each. We concatenate its input with a ”positional” embedding to mark which of the 25 images is the one being compared with the rest. This ”positional” embedding is not necessary, but slightly helps performance. Adam (Kingma & Ba, 2014) is used as the optimizer, with learning rate 3e-5 and betas (0.9, 0.999). The batch size is 2048, but smaller batch sizes also work. This only affects convergence speed.

In practice, any of the outputs can be used as the predicted probability. We chose to apply loss on the first $k-1$ outputs:

$$l(\hat{y}, y) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{k_n-1}\sum_{i=1}^{k_n-1} -(y_n \log(\hat{y}_{n,i}) + (1-y_n)\log(1-\hat{y}_{n,i})) \tag{5}$$

where $\hat{y}$ is our model’s output, y the targets and N the batch size. In the first variant with which we trained our models, k_n is always 25. For the second variation though, each element of the batch has its own k_n value chosen uniformly at random between 2 and 25.

A.2 ImageNet Randomized Class Order

We shuffled the ImageNet classes by first setting `numpy.random.seed(0)` and then used `numpy.random.rand(1000).argsort()` as the randomized order.

A.3 Clustering Visualization

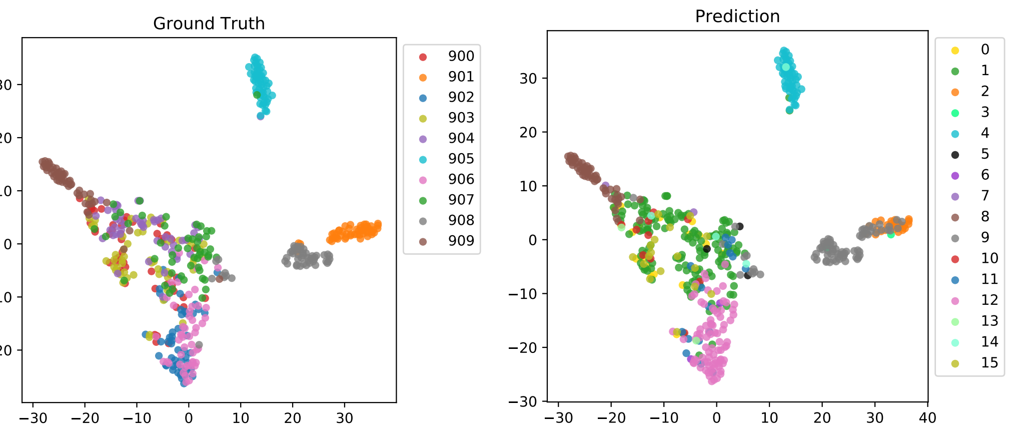

Figure 3: t-SNE plots all test samples in each ImageNet class 901-910. Images were encoded by a ResNet-18 trained with our class homogeneity task before being plotted by t-SNE. On the left, samples are colored using the target labels and on the right using our model's clustering prediction.