

A Simple Strategy to Address Imbalanced Long-tail Classification Accuracies

Nadine Chang^{1,*} Jayanth Koushik^{1,*} Michael J. Tarr¹
Martial Hebert¹ Yu-Xiong Wang²

Abstract

Methods in long-tail learning focus on improving performance for data-poor (rare) classes; however, performance for such classes remains much lower than performance for more data-rich (frequent) classes. This is a problem for applications such as autonomous vehicle systems or medical imaging, where balanced performance across all classes is desirable. We propose a simple method that can be applied to existing models, performing a *post hoc* correction, which leads to much higher accuracy on rare classes with negligible change to overall accuracy. Starting with a pre-trained model, we find frequent classes that are closest to rare classes in the model’s representation space and use their classifiers in linear combination, improving rare class accuracy while preserving overall accuracy. This straightforward method is extremely fast to train and can be applied to any model without requiring changes to the base classifiers which may be restricted for a variety of reasons (e.g., ethical considerations, privacy concerns, or intellectual property rights). We observe that our simple correction improves performance for rare classes by up to 10 points on the ImageNet-LT dataset. At the same time, overall accuracy is preserved, leading to more balanced performance across all classes.

1 Introduction

Many AI systems are biased due to underlying biases in training data, in the models themselves, or in the choices of the system designers^[1]. Here we focus on the first of three sources of bias, observing that the prevalence of objects in the real world follows a long-tailed distribution, where some classes occur frequently and others occur rarely^[2,3]. Due to this phenomenon, many applications rely on models that can classify rarer objects alongside common ones. For example, autonomous vehicle systems are expected to classify rare animals, objects, or road configurations in order to avoid potential collisions^[4] and medical image analysis systems are used to spot rare cancers, detect anatomical irregularities, and perform image reconstruction^[1].

The significance of long-tailed distributions in real-world applications has spurred a variety of approaches for long-tailed classification. Some methods re-sample more data for less frequent classes in an effort to address data imbalances^[5,6] re-sampling can be challenging or impossible, e.g., in medical contexts,^[1]. Other methods re-weight the base classifiers so that rarer classifiers are weighted more heavily and, thus, are better at classification^[7]. Both re-sampling and re-weighting

*Equal contribution. ¹Carnegie Mellon University. ²University of Illinois at Urbana-Champaign.
Email: nchang1@cs.cmu.edu, jkoushik@andrew.cmu.edu, michaeltarr@cmu.edu, hebert@cs.cmu.edu,
yxw@illinois.edu.

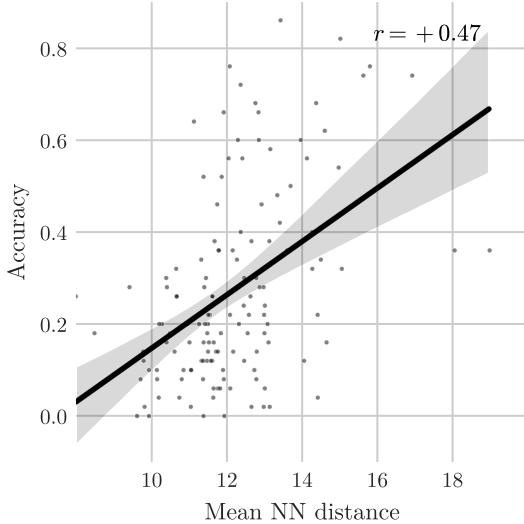


Figure 1: Per-class test accuracy of the cRT model for classes in the ‘few’ split of ImageNet-LT, versus the mean Euclidean distance to five nearest neighbors in the ‘base’ split. Distance between classes was calculated using the mean representation of training data samples. The plot shows a strong correlation (Pearson correlation, r shown in top right) between class test accuracy and the mean nearest neighbor distance, that is, ‘few’ split classes with close neighbors in the ‘base’ split had poor test accuracy.

methods are strong baselines. State-of-the-art results are achieved by more complex and difficult methods that rely on multiple expert learning^[8,9], multi-stage distillation^[10], or a large combination of several balancing, re-sampling, and loss decays^[11].

Although the above methods successfully improve long-tail performance, they present several challenges that we address in our simple model. First, these methods require re-training classifiers and thus are unable to be used in applications where an existing frequent base classifier cannot be altered. For example, some applications deal directly with privacy concerns or intellectual property rights. Second, although these methods improve accuracy for rare classes, they focus on also improving accuracy across *all* classes within a long-tailed distribution. This emphasis on overall accuracy leads the majority of these long-tailed learning methods to suffer from a continued imbalance in accuracy. For example, one highly effective baseline method, cRT^[7] has an average accuracy of 61.8% on frequent classes, but the average accuracy on rare classes is more than 30 points lower, at 27.4%. Such imbalances can be particularly problematic in contexts where ethical considerations come into play, as in medical image computing^[1]. In Figure ??, we illustrate that, much as the classes themselves, *accuracies* across classes follow a long-tailed distribution and that this distribution correlates with class frequency. That is, most long-tail methods focus on data balancing but fail to address accuracy balancing. Here, we argue that accuracy balancing is equally crucial to long-tailed learning.

We address these two challenges by reducing the accuracy differences between common classes and

rarer classes – critically *without* modifying the existing common classifiers that are the majority of the model’s classifiers. Our goal is to balance a set of imbalanced class accuracies in an existing model and *not* to improve upon state-of-the-art. For example, in medical image computing, outcome disparities (imbalanced class accuracies) have been connected to a “lack of diversity and proper representation of the target population in the training databases” i.e., some classes being much rarer than others,^[1]. To reduce accuracy differences, we propose AlphaNet, a simple method that transfers knowledge from strong classifiers (i.e., learned with frequent data) to weak classifiers (i.e., learned with rare data). AlphaNet, achieves more balanced accuracy across classes by strengthening weak classifiers using a shallow network to learn how to linearly combine each weak classifier with relevant strong classifiers. To combine both relevant and non-relevant classifiers, our methods allows for both positive and negative weights respectively. For example, in Figure ??, we illustrate how the weak ‘cat’ classifier can be appropriately adjusted by moving towards relevant strong ‘leopard’ and ‘lion’ classifiers and away from non-relevant strong ‘ostrich’ classifier. Relevant strong classifiers are simply determined by finding the nearest neighbor classes to the target class (e.g., ‘cat’). We emphasize that our method is a *post-hoc* accuracy correction that can be applied on top of any trained model that includes a set of classifiers. Importantly, the adjustments we make to improve classifier balance can be applied without touching the base classifiers. As such, our method is applicable to use cases where existing base classifiers are either unavailable or fixed (e.g., due to commercial interests or data privacy protections). In these instances our method can uniquely provide accuracy balancing updates.

To showcase the efficacy of our method, we applied it on top of strong baselines and approaches for ImageNet-LT, Places-LT, and CIFAR-100-LT. Across these tests, our improvement on rare classes was as high as 10 points. Applying AlphaNet on the strong baselines described by Kang et al. (2019)^[7], rare class accuracy was increased by up to 5 points, while overall accuracy remained within 1 point, showing that our approach provides a simple *post-hoc* adjustment for balancing class accuracies.

Summary of contributions:

- AlphaNet is a frustratingly simple *post-hoc* method that can be applied on top of any classification model with a set of unique classifiers.
- The simplicity of our method lends to two additional advantages: 1) extremely rapid training (e.g. 5 minutes training for ImageNet-LT); 2) light weight computation that can be performed on a single small GPU or even CPU.
- We can strengthen weak classifiers without changing a large set of strong classifiers.
- Our method is robust, as shown by our trivial errors across multiple repetition runs.

2 Related works

Combining, creating, modifying, and learning model weights are concepts that are implemented in many earlier models. As we review below, these concepts appear frequently in transfer learning, meta-learning, zero-shot/low-shot learning, and long-tailed learning.

2.1 Classifier Creation

The process of creating new classifiers is captured within meta-learning concepts such as learning-to-learn, transfer learning, and multi-task learning^[12–16]. These approaches generalize to novel tasks by learning shared information from a set of related tasks. Many studies find that shared information is embedded within model weights, and, thus, aim to learn structure within learned models to directly modify the weights of a different network^[17–24]. Other studies go even further and instead of modifying networks, they create entirely new networks exclusively from training samples^[25–27]. In contrast, AlphaNet only combines existing classifiers, without having to create new classifiers or networks from scratch.

2.2 Classifier or Feature Composition

In various classical approaches, there has been work that learns better embedding spaces for image annotation^[28] or uses classification scores as useful features^[29]. However, these approaches do not attempt to compose classifiers nor do they address the long-tail problem. Within non-deep methods in classic transfer learning, there have been attempts to use and combine SVMs. In Tschantaridis et al. (2005)^[30], SVMs are trained per object instance and a hierarchical structure is required for combination in the datasets of interest. Such a structure is typically not guaranteed nor provided in long-tailed datasets. Additional SVM work uses regularized minimization to learn the coefficients necessary to combine patches from other classifiers^[31]. While these approaches are conceptually similar to AlphaNet, AlphaNet has the additional advantage of *learning* the compositional coefficients without any hyper-parameters and tuning. Specifically, different novel classes will have their own sets of alphas, and similar novel classes will naturally have similar sets of alphas (see ? (?)^[?]). Learning such varying sets of alphas is difficult in previous classical approaches. They either learn a fixed set of alphas for all of the novel classes or are forced to introduce more complex group sparsity-like constraints. Finally, in zero-shot learning there exist methods which compose classifiers of known visual concepts to learn a completely new classifier^[26,32–34]. However, such composition is often guided by additional attribute supervision or textual description – two factors on which AlphaNet does not depend.

2.3 Learning Transformations Between Models and Classes

Other studies have demonstrated different ways of learning transformations to modify model weights in an attempt to learn these transformations during stochastic gradient descent (SGD) optimization^[35,36]. Additionally, Wang and Hebert (2016)^[37] empirically show the existence of a generic nonlinear transformation from small-sample to large-sample models for different types of feature spaces and classifier models. Finally, Du et al. (2016)^[38] provide theoretical guarantees on performance when one learns the transformation from the source function to a related target function. In AlphaNet we likewise infer that our target classifier is a transformation from a set of source classifiers.

2.4 Zero-shot/Low-shot Learning

Meta-learning, transfer learning, and learning-to-learn are frequently applied to the domain of low-shot learning^[16,37,39–48]. A wide variety of prior studies have attempted to transfer knowledge from tasks with abundant data to completely novel tasks^[19,36,49]. However, the explicit nature of

low-shot learning consisting of tasks with small fixed samples means that these approaches do not generalize well beyond the arbitrary few tasks. This is a significant problem as the visual world clearly involves a wide set of tasks with continuously varying amounts of information.

2.5 Long-tailed Learning

The restrictions on low-shot learning have directly led to a new paradigm referred to as “long-tailed” learning, where data samples are continuously decreasing and the data distribution closely models that of the visual world. Recent work achieves state-of-the-art performance on long-tailed recognition by learning multiple experts^[8,9]. Both these complex ensemble methods require a two-stage training method. A somewhat different approach re-balances the samples at different stages of model training^[6], while another study attempts to transfer features from common classes to rare classes^[3] or transfer intra-class variance^[50]. However, both these approaches to knowledge transfer require complex architectures, such as a specialized attention mechanism and memory models as in Liu et al. (2019)^[3]. While most studies have largely focused on representation space transferability or complex ensembles, recent work establishes a strong baseline by exploring the potential of operating in classifier space^[7]. Results suggest that decoupling model representation learning and classifier learning is a more efficient way to approach long-tailed learning. Specifically, methods normalizing classifiers and adjusting classifiers only using re-sampling strategies achieve good performance. Such successes in working only with classifiers support our general concept that combining strong classifiers is a natural and direct way to improve upon weak classifiers.

3 Method

Long-tailed recognition is uniquely characterized by its unbalanced, long-tailed distribution of data. Due to the lack of sufficient data in parts of the distribution, otherwise known as the “tail”, training on the tail classes significantly suffers compared to that of ‘base’ classes that have sufficient data. As a result, tail classes typically see low accuracy performance, and this remains the dominating challenge of long-tailed recognition. AlphaNet directly addresses this issue, and successfully improves the tail performance by combining classifiers in order to adaptively adjust weak classifiers.

3.1 Notation

We first group up our classes into two broad splits: our ‘base’ split with B number of classes and our ‘few’ split with F classes. The ‘base’ classes contain classes with many examples, and conversely the ‘few’ classes contain classes with few examples. We denote by $N = B + F$ the total number of classes.

Additionally, we denote a generic sample by I and its corresponding label by y . We split the training set into two subsets of pairs (I_i, y_i) : the subset X_{base} contains samples from the B classes, and the subset X_{few} contains samples from the F classes.

Finally, we work with a fixed pre-trained model with classifiers W_j and biases b_j , $j \in (1, \dots, N)$; here, classifiers are the last layers of networks, which map input representations to class scores. Given target class j , and its classifier W_j and bias b_j , let class j ’s top k nearest neighbor classifiers be defined as V_i^j , $i \in (1, \dots, K)$. Furthermore, we also define the trivial case where the immediate nearest neighbor is itself, $V_0^j = W_j$.

3.2 AlphaNet

Our combination model, AlphaNet, takes in trained classifiers from the B ‘base’ classes and weak classifiers from the F ‘few’ classes, in order to learn a new classifier composition for each ‘few’ class. The following section will detail the model and our training as depicted in Figure 2.

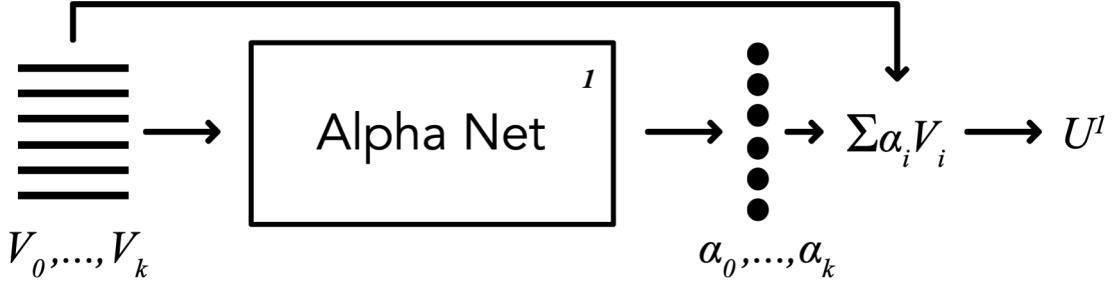


Figure 2: An illustration of AlphaNet. The classifier for a ‘few’ split class, along with the classifiers of k nearest neighbors are passed through a network, which outputs $k + 1$ weights, termed alphas. These alphas are used to linearly combine the input classifiers, producing the updated classifier for the given ‘few’ split class.

3.3 Architecture

AlphaNet is lightweight and uses a small fully connected network. Its input is a flattened concatenated vector of V_k^i , $k \in (0, \dots, K)$, where $V_0^i = W_i$, the original classifier for class i . Its output is an α vector, where α_k , $k \in (0, \dots, K)$ corresponds to input V_k^i . Importantly, AlphaNet does not restrict the range that α coefficients can be. Thus, AlphaNet learns both positive and negative α coefficients. Furthermore, AlphaNet can be used to learn all α coefficients for all classifiers. Once trained, AlphaNet can also be applied on new classes not seen during training.

3.4 Training

Given alphas α_k , and input V_k^i , $k \in (0, \dots, K)$, the updated classifier is given by

$$U_i = V_0 + \sum_{j=1}^K \alpha_j \cdot V_j^i. \quad (1)$$

We use the same set of alphas to learn U_i , $i \in (1, \dots, F)$ for each target class.

$$\begin{aligned} U_1 &= V_0^1 + \sum_{j=1}^K \alpha_j \cdot V_j^1. \\ U_F &= V_0^F \sum_{j=1}^K \alpha_j \cdot V_j^F. \end{aligned} \quad (2)$$

It is important to note that while we are attempting to compose new ‘few’ classifiers, we are still working on a long-tailed recognition problem. Thus, when training we must also consider our ‘base’ classifiers that have been sufficient trained. Now, given a training image sample and label pair $\{I_i, y_i\}$ and its final feature representation x_i , we compute our sample score as

$$\begin{aligned} s_{if} &= x_i \cdot U_f. \quad f \in (1, \dots, F) \\ s_{ib} &= x_i \cdot W_b + b_b. \quad b \in (1, \dots, B) \end{aligned} \tag{3}$$

We combine the two sets of scores and obtain per-class prediction scores. Finally, we compute the softmax cross entropy loss, which is minimized to learn AlphaNet weights.

4 Experiments

4.1 Experiment Setup

4.1.1 Datasets

We evaluated our method using three long-tailed benchmark datasets: ImageNet-LT, Places-LT^[3], and CIFAR-100-LT^[11]. These Datasets are sampled from their respective original datasets, ImageNet^[51], Places365^[52], and CIFAR-100^[53] such that the new distributions follow a standard long-tailed distribution. ImageNet-LT contains 1000 classes with the number of samples per class ranging from 5 to 4980 images. Places-LT contains 365 classes with the number of samples per class ranging from 5 to 1280 images. CIFAR-100-LT contains 100 classes with the number of samples per class ranging from 5 to 500 images. For CIFAR-100-LT, we used the version described by Alshammari et al. (2022)^[11], using an imbalance factor of 100.

The datasets are broken down into three broad splits that indicate the number of training samples per class: 1) ‘many’ contains classes with greater than 100 samples; 2) ‘medium’ contains classes with greater than or equal to 20 samples but less than or equal to 100 samples; 3) ‘few’ contains classes with less than 20 samples. The test set is always balanced, containing an equal number of samples for each class. We use the term ‘base’ split to refer to the combined ‘many’ and ‘medium’ splits.

Note: Another popular dataset used for testing long-tail learning models is iNaturalist^[54]. Results for this dataset, however, are much more balanced across classes. So it does not represent a valid use case for our proposed method, and we omitted the dataset from our experiments.

4.1.2 Training data sampling

In order to prevent overfitting on the ‘few’ split samples, we used a class balanced sampling approach, using all ‘few’ split samples, and a portion of the ‘base’ split samples. Given F ‘few’ split samples, and a ratio ρ , every epoch, ρF samples were drawn from the ‘base’ split, with sample weights inversely proportional to the class size. This ensured that all ‘base’ classes had an equal probability of being sampled. As we show in the following section, ρ allows us to control the balance between ‘few’ and ‘base’ split accuracy. We evaluated AlphaNet with a range of ρ values; results for $\rho = 0.5$, $\rho = 1$, and $\rho = 1.5$ are shown in Section 4.2, and the full set of results is in the appendix.

4.1.3 Training

All experiments used an AlphaNet module with three 32 unit layers, and Leaky-ReLU^[?] activation. Unless stated otherwise, euclidean distance was used to find $k = 5$ nearest neighbors for each ‘few’ split class. Models were trained for 25 epochs to minimize cross-entropy loss computed using mini-batches of 64 samples. Optimization was performed using AdamW^[55] with a learning rate of 0.001, decayed by a factor of 10, every 10 epochs. Model weights were saved after each epoch, and after training, the weights with the best accuracy on validation data were used to report results on the test set. All experiments were repeated 10 times, and we report mean and standard deviation of accuracies across trials.

4.2 Results

4.2.1 Baseline models

First, we applied AlphaNet on several strong baselines proposed by Kang et al. (2019)^[7]. These methods have good overall accuracy, but accuracy for ‘few’ split classes is much lower. On the ImageNet-LT dataset, average accuracy for the cRT and LWS models (using a ResNeXt-50 backbone) is nearly 20 points below the overall accuracy, as seen in Table 1. Using features extracted from these two models, we used AlphaNet to update ‘few’ split classifiers. On both models, we saw a significant increase in the ‘few’ split accuracy for all values of ρ , leading to more balanced accuracies. For $\rho = 1$, average ‘few’ split accuracy was boosted by 7 points for the cRT model, and about 11 points for the LWS model, while overall accuracy was within 2 point of the original. We found this value of ρ to provide a good balance between ‘few’ split and overall accuracy. Lower values of ρ led to much larger ‘few’ split accuracy, and even with $\rho = 1.5$, there was a significant increase, while the accuracy for other splits remained around the same.

Method	Few	Med.	Many	Overall
Cross entropy†	7.7	37.5	65.9	44.4
NCM†	28.1	45.3	56.6	47.3
τ -normalized†	30.7	46.9	59.1	49.4
cRT†	27.4	46.2	61.8	49.6
Alpha-cRT ($\rho = 0.5$)	39.7 ± 1.42	42.0 ± 0.66	58.3 ± 0.52	48.0 ± 0.37
Alpha-cRT ($\rho = 1$)	34.6 ± 1.88	43.7 ± 0.51	59.7 ± 0.43	48.6 ± 0.24
Alpha-cRT ($\rho = 1.5$)	32.6 ± 2.46	44.4 ± 0.49	60.3 ± 0.38	48.9 ± 0.19
LWS†	30.4	47.2	60.2	49.9
Alpha-LWS ($\rho = 0.5$)	46.9 ± 0.98	38.6 ± 0.87	52.9 ± 0.86	45.3 ± 0.69
Alpha-LWS ($\rho = 1$)	41.6 ± 1.61	42.2 ± 0.53	56.0 ± 0.32	47.4 ± 0.30
Alpha-LWS ($\rho = 1.5$)	40.1 ± 1.99	43.2 ± 0.98	56.9 ± 0.76	48.0 ± 0.53

Table 1: Baseline methods on ImageNet-LT. † indicates results that were taken directly from Kang et al. (2019)^[7]. Alpha-cRT, and Alpha-LWS are AlphaNet models applied over cRT and LWS features respectively.

We repeated the above experiment on the Places-LT dataset, where again ‘few’ split accuracy for

the cRT and LWS models is much lower than the overall accuracy (by around 12 and 9 points respectively as seen in Table 2. With $\rho = 1$, AlphaNet improved ‘few’ split accuracy by 2 points on average for the cRT model, and about 6 points on average for the LWS model. In both cases, overall accuracy was within 1 point of the baseline.

Method	Few	Med.	Many	Overall
Cross entropy†	8.2	27.3	45.7	30.2
NCM†	27.3	37.1	40.4	36.4
τ -normalized†	31.8	40.7	37.8	37.9
cRT	24.9	37.6	42.0	36.7
Alpha-cRT ($\rho = 0.5$)	31.0 ± 0.88	34.5 ± 0.17	40.4 ± 0.29	35.9 ± 0.09
Alpha-cRT ($\rho = 1$)	27.0 ± 1.02	36.1 ± 0.31	41.3 ± 0.13	36.2 ± 0.10
Alpha-cRT ($\rho = 1.5$)	25.5 ± 0.89	36.5 ± 0.36	41.6 ± 0.21	36.2 ± 0.11
LWS	28.7	39.1	40.6	37.6
Alpha-LWS ($\rho = 0.5$)	37.1 ± 1.39	34.4 ± 0.80	37.7 ± 0.52	36.1 ± 0.31
Alpha-LWS ($\rho = 1$)	34.6 ± 0.97	35.8 ± 0.54	38.6 ± 0.39	36.6 ± 0.22
Alpha-LWS ($\rho = 1.5$)	32.2 ± 1.17	37.2 ± 0.36	39.5 ± 0.39	37.0 ± 0.11

Table 2: Baseline methods on Places-LT. † indicates results that were taken directly from Kang et al. (2019)^[7]. Alpha-cRT, and Alpha-LWS are AlphaNet models applied over cRT and LWS features respectively.

4.2.2 State-of-the-art

Next, we applied AlphaNet on two state-of-the-art models. First, we used the 6-expert teacher model of Wang et al. (2020)^[8]. We provided the combined feature vectors from all 6 experts as input to AlphaNet. The learned ‘few’ split classifiers were split into 6, and used to update the experts. Prediction scores from the experts were averaged to produce the final predictions, as in the original model. For ImageNet-LT, the experts used a ResNeXt-50 backbone, and for CIFAR-100-LT, a ResNet-32 backbone. Table 3 shows the base results for the expert models, along with AlphaNet results for $\rho = 0.5, 1, 2$. On ImageNet-LT, ‘few’ split accuracy was increased by up to 7 points, and on CIFAR-100-LT, by 5 points. The second state-of-the-art model we used was the weight balancing method proposed by Alshammari et al. (2022)^[11]. The full set of results on the CIFAR-100-LT dataset is shown in the appendix.

4.2.3 Comparison with control

Our method is based on the core hypothesis that classifiers can be improved using nearest neighbors. In this section, we directly test this hypothesis. Based on the results in the previous section, the improvements in ‘few’ split accuracy could be attributed simply to the extra fine-tuning of ‘few’ split classifiers. So, we repeated the experiments of the previous section with randomly chosen neighbors for each ‘few’ split class, rather than nearest neighbors. This differs from our previous experiments only in the nature of neighbors used, so if our method’s improvements are solely due to extra fine-tuning, we should see the same results. However, as seen in Figure 3, training with

Method	Few	Med.	Many	Overall
ImageNet-LT				
RIDE	36.5	54.4	68.9	57.5
Alpha-RIDE ($\rho = 0.5$)	43.5 ± 0.75	52.3 ± 0.26	67.3 ± 0.17	56.9 ± 0.11
Alpha-RIDE ($\rho = 1$)	40.8 ± 1.00	53.1 ± 0.21	67.9 ± 0.18	57.1 ± 0.11
Alpha-RIDE ($\rho = 1.5$)	38.2 ± 1.22	53.6 ± 0.25	68.4 ± 0.17	57.2 ± 0.06
CIFAR-100-LT				
RIDE	25.8	52.1	69.3	50.2
Alpha-RIDE ($\rho = 0.5$)	30.9 ± 1.82	47.0 ± 1.25	65.7 ± 0.94	48.7 ± 0.41
Alpha-RIDE ($\rho = 1$)	27.2 ± 1.69	49.1 ± 0.85	67.4 ± 0.62	48.9 ± 0.30
Alpha-RIDE ($\rho = 1.5$)	25.0 ± 1.15	50.1 ± 1.12	67.9 ± 1.01	48.8 ± 0.63

Table 3: ImageNet-LT and CIFAR-100-LT results using the ensemble model proposed by Wang et al. (2020)^[8]. Alpha-RIDE applies AlphaNet on average features from the ensemble.

nearest neighbors garners much larger improvements in ‘few’ split accuracy, with similar trends in overall accuracy. This supports our hypothesis that data-poor classes can make use of data from neighbors to improve classification performance.

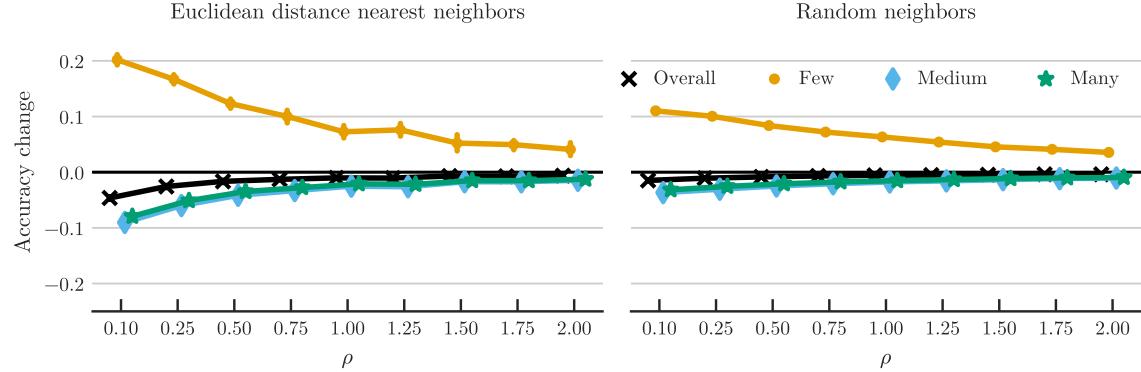


Figure 3: Change in split accuracy for AlphaNet training of cRT model on ImageNet-LT. For each value of ρ , the two plots both show the change in split accuracy for AlphaNet compared to the baseline cRT model, as well as the change in overall accuracy. Plot a) shows the results for normal training with 5 euclidean nearest neighbors, and plot b) shows the results for training with 5 random neighbors for each ‘few’ split class. Training with nearest neighbors leads to much larger increase in ‘few’ split accuracy (especially for small values of ρ), which cannot be accounted for by the additional fine-tuning of classifiers.

...

...

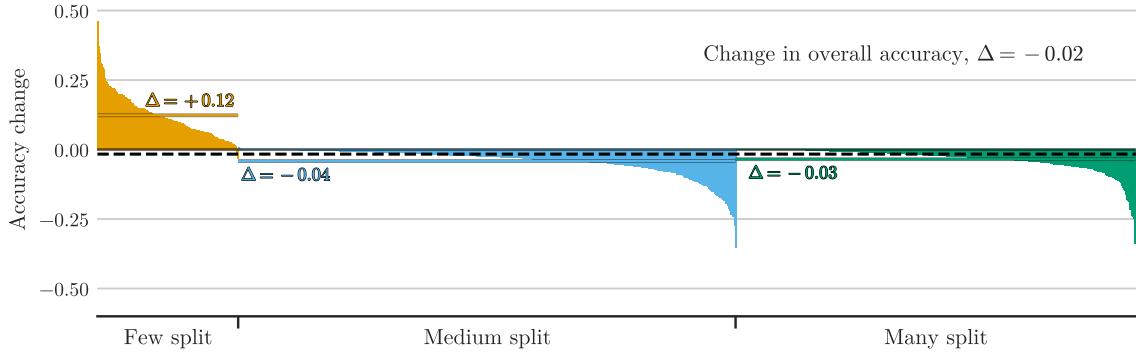


Figure 4: Change in per-class test accuracy for AlphaNet applied over cRT features, with $\rho = 0.5$. Within each split, the classes are sorted by change in accuracy, and the dashed line shows the average per-class change (change value Δ). The spanning dashed line shows the mean overall change in per-class test accuracy. Nearly all ‘few’ split classes have their accuracy improved, and the overall improvement of ‘few’ split accuracy far exceeds the average decrease for ‘many’ and ‘medium’ splits.

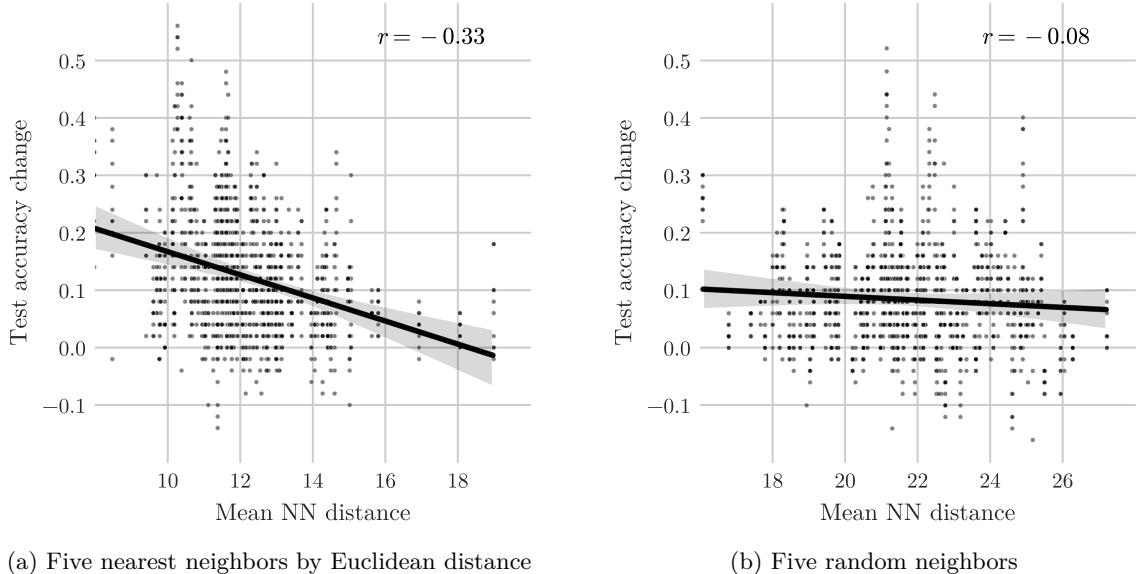


Figure 5: Change in per-class accuracy for AlphaNet applied over cRT features, with $\rho = 0.5$, versus mean Euclidean distance to five neighbors. Figure 5a uses five nearest neighbors by Euclidean neighbors, and Figure 5b uses five random neighbors. Both plots show results from ten runs with random starts. Comparing with Figure 1, we can see that AlphaNet provides the largest boost to classes with poor baseline performance, which have close nearest neighbors.

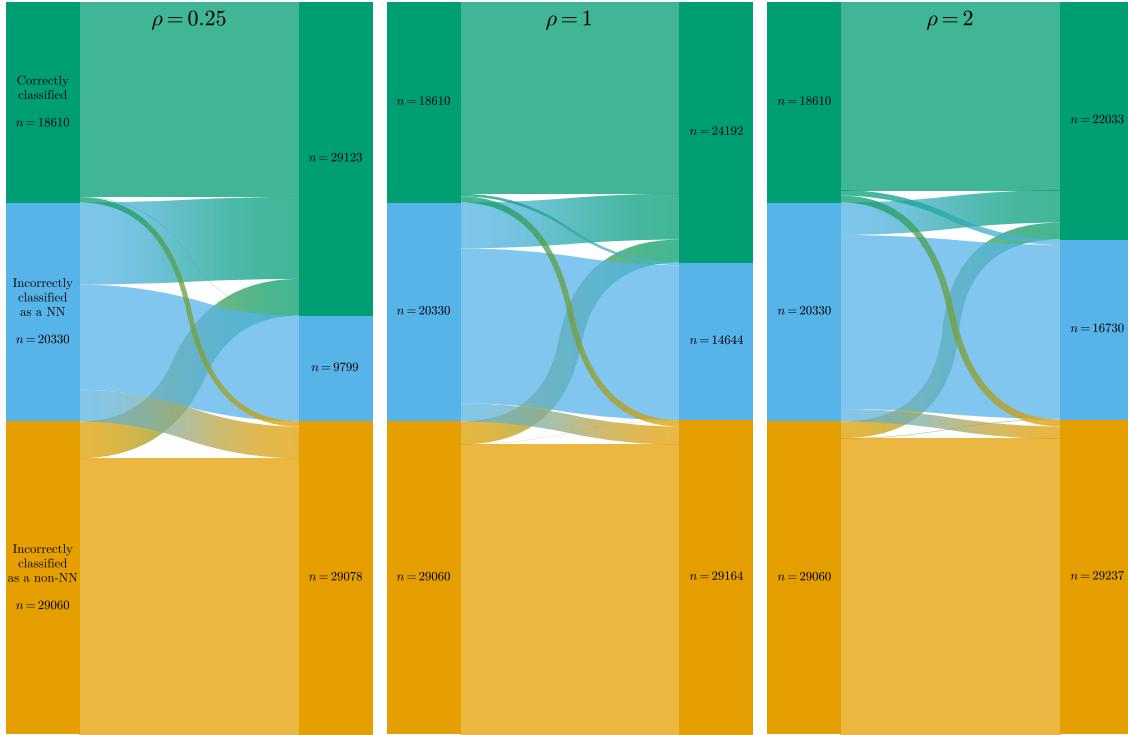


Figure 6: Change in ‘few’ split sample predictions for AlphaNet applied for cRT features, with $k = 10$ nearest neighbors by Euclidean distance. In each chart, the bars on the left show the distribution of predictions for ‘few’ split test samples, by the baseline model; and the bars on the right show the distribution for AlphaNet. The predictions are grouped into three categories: 1) correct predictions, 2) incorrect predictions where the prediction was a nearest neighbor of the original class (e.g., predicting ‘Malamute’ for ‘Husky’), and 3) all other incorrect predictions. The “flow” bands from left to right show the changes in individual sample predictions. There is a large improvement for samples previously misclassified as a nearest neighbor, particularly for small ρ . As ρ is increased, smaller portion of these mistakes are corrected, leading to smaller improvement in ‘few’ split accuracy.

...

5 Conclusion

The long-tailed nature of the world presents a challenge for any model that depends on learning over specific examples. Most long-tailed methods tend to have high overall accuracy, but with unbalanced accuracies where frequent classes are learned well with high accuracies and rare classes are learned poorly with low accuracies. As such, the long-tailed world represents one source of potential bias

[along with the models themselves and system designer decisions; Lara et al. (2022)^[1]]. To address this problem, typical approaches resort to re-sampling or re-weighting of rare classes but still focus on achieving the highest overall accuracy. Consequently, these methods continue to suffer from an accuracy imbalance across data-rich and data-poor classes. In contrast, our method, AlphaNet, provides a rapid 5 minute *post-hoc* correction that can sit on top of any model using classifiers. This simple method re-balances classification accuracy quickly so that rare classes have much higher accuracy, but overall classification accuracy is preserved. In addition to directly addressing training imbalances, AlphaNet is also applicable to re-balancing accuracies across biases arising from model structure or the prioritization of different model parameters. That is, AlphaNet is deployable in any application where the base classifiers cannot be changed, but balanced performance is desirable – thereby making it useful in contexts where ethics, privacy, or intellectual property are concerns.

References

- [1] María Agustina Ricci Lara, Rodrigo Echeveste, and Enzo Ferrante. Addressing fairness in artificial intelligence for medical imaging. *Nature Communications*, 13(1):4581, 2022. doi: 10.1038/s41467-022-32186-3.
- [2] Xiangxin Zhu, Dragomir Anguelov, and Deva Ramanan. Capturing long-tail distributions of object subcategories. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 915–922, 2014. doi: 10.1109/cvpr.2014.122.
- [3] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X. Yu. Large-scale long-tailed recognition in an open world. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 00:2532–2541, 2019. doi: 10.1109/cvpr.2019.00264.
- [4] Ming-Fang Chang, Deva Ramanan, James Hays, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, and Simon Lucey. Argoverse: 3d tracking and forecasting with rich maps. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 00:8740–8749, 2019. doi: 10.1109/cvpr.2019.00895.
- [5] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 00:9260–9269, 2019. doi: 10.1109/cvpr.2019.00949.
- [6] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. *arXiv*, 2019. doi: 10.48550/arxiv.1906.07413.
- [7] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv*, 2019.
- [8] Xudong Wang, Long Lian, Zhongqi Miao, Ziwei Liu, and Stella X Yu. Long-tailed recognition by routing diverse distribution-aware experts. *arXiv*, 2020. doi: 10.48550/arxiv.2010.01809.
- [9] Jiarui Cai, Yizhou Wang, and Jenq-Neng Hwang. Ace: Ally complementary experts for solving long-tailed recognition in one-shot. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 00:112–121, 2021. doi: 10.1109/iccv48922.2021.00018.

- [10] Tianhao Li, Limin Wang, and Gangshan Wu. Self supervision to distillation for long-tailed visual recognition. *arXiv*, 2021. doi: 10.48550/arxiv.2109.04075.
- [11] Shaden Alshammari, Yu-Xiong Wang, Deva Ramanan, and Shu Kong. Long-tailed recognition via weight balancing. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 00:6887–6897, 2022. doi: 10.1109/cvpr52688.2022.00677.
- [12] Sebastian Thrun. Learning to learn. pages 181–209, 1998. doi: 10.1007/978-1-4615-5529-2_8.
- [13] Jürgen Schmidhuber, Jieyu Zhao, and Marco Wiering. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 28(1):105–130, 1997. ISSN 0885-6125. doi: 10.1023/a:1007383707642.
- [14] Simo Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009. ISSN 1041-4347. doi: 10.1109/tkde.2009.191.
- [15] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. ISSN 0885-6125. doi: 10.1023/a:1007379606734.
- [16] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. One-shot learning with memory-augmented neural networks. *arXiv*, 2016. doi: 10.48550/arxiv.1605.06065.
- [17] J. Schmidhuber. A neural network that embeds its own meta-levels. *IEEE International Conference on Neural Networks*, pages 407–412 vol.1, 1993. doi: 10.1109/icnn.1993.298591.
- [18] Jrgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.1.131.
- [19] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip H S Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. *arXiv*, 2016. doi: 10.48550/arxiv.1606.05233.
- [20] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv*, 2016. doi: 10.48550/arxiv.1609.09106.
- [21] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *arXiv*, 2018. doi: 10.48550/arxiv.1806.02817.
- [22] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *arXiv*, 2017. doi: 10.48550/arxiv.1705.08045.
- [23] Abhishek Sinha, Mausoom Sarkar, Aahitagni Mukherjee, and Balaji Krishnamurthy. Introspection: Accelerating neural network training by learning weight evolution. *arXiv*, 2017. doi: 10.48550/arxiv.1704.04959.
- [24] Tsendsuren Munkhdalai and Hong Yu. Meta networks. *Proceedings of machine learning research*, 70:2554–2563, 2017.
- [25] Richard Socher, Milind Ganjoo, Hamsa Sridhar, Osbert Bastani, Christopher D Manning, and Andrew Y Ng. Zero-shot learning through cross-modal transfer. *arXiv*, 2013. doi: 10.48550/arxiv.1301.3666.

- [26] Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, and Ruslan Salakhutdinov. Predicting deep zero-shot convolutional neural networks using textual descriptions. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4247–4255, 2015. doi: 10.1109/iccv.2015.483.
- [27] Hyeyoung Noh, Paul Hongsuck Seo, and Bohyung Han. Image question answering using convolutional neural network with dynamic parameter prediction. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 30–38, 2016. doi: 10.1109/cvpr.2016.11.
- [28] Jason Weston, Samy Bengio, and Nicolas Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine Learning*, 81(1):21–35, 2010. ISSN 0885-6125. doi: 10.1007/s10994-010-5198-3.
- [29] Gang Wang, Derek Hoiem, and David Forsyth. Learning image similarity from flickr groups using fast kernel machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2177–2188, 2012. ISSN 0162-8828. doi: 10.1109/tpami.2012.29.
- [30] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, Yasemin Altun, and Yoram Singer. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(9), 2005.
- [31] Yusuf Aytar and Andrew Zisserman. Enhancing exemplar svms using part level transfer regularization. *Proceedings of the British Machine Vision Conference 2012*, pages 79.1–79.11, 2012. doi: 10.5244/c.26.79.
- [32] Mohamed Elhoseiny, Babak Saleh, and Ahmed Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. *2013 IEEE International Conference on Computer Vision*, pages 2584–2591, 2013. doi: 10.1109/iccv.2013.321.
- [33] Ishan Misra, Abhinav Gupta, and Martial Hebert. From red wine to red tomato: Composition with context. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1160–1169, 2017. doi: 10.1109/cvpr.2017.129.
- [34] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Synthesized classifiers for zero-shot learning. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5327–5336, 2016. doi: 10.1109/cvpr.2016.575.
- [35] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. *arXiv*, 2016. doi: 10.48550/arxiv.1606.04474.
- [36] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.
- [37] Yu-Xiong Wang and Martial Hebert. Computer vision – eccv 2016, 14th european conference, amsterdam, the netherlands, october 11-14, 2016, proceedings, part vi. *Lecture Notes in Computer Science*, pages 616–634, 2016. ISSN 0302-9743. doi: 10.1007/978-3-319-46466-4__37.
- [38] Simon Shaolei Du, Jayanth Koushik, Aarti Singh, and Barnabas Poczos. Hypothesis transfer learning via transformation functions. *arXiv*, 2016. doi: 10.48550/arxiv.1612.01020.
- [39] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006. ISSN 0162-8828. doi: 10.1109/tpami.2006.79.

- [40] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, page 0.
- [41] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. ISSN 0036-8075. doi: 10.1126/science.aab3050.
- [42] Zhizhong Li and Derek Hoiem. Learning without forgetting. *arXiv*, 2016. doi: 10.48550/arxiv.1606.09282.
- [43] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3037–3046, 2017. doi: 10.1109/iccv.2017.328.
- [44] JANE BROMLEY, JAMES W BENTZ, LÉON BOTTOU, ISABELLE GUYON, YANN LE-CUN, CLIFF MOORE, EDUARD SÄCKINGER, and ROOPAK SHAH. Signature verification using a “siamese” time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 07(04):669–688, 1993. ISSN 0218-0014. doi: 10.1142/s0218001493000339.
- [45] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv*, 2017. doi: 10.48550/arxiv.1703.05175.
- [46] Dileep George, Wolfgang Lehrach, Ken Kansky, Miguel Lázaro-Gredilla, Christopher Laan, Bhaskara Marthi, Xinghua Lou, Zhaoshi Meng, Yi Liu, Huayan Wang, Alex Lavin, and D. Scott Phoenix. A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science*, 358(6368), 2017. ISSN 0036-8075. doi: 10.1126/science.aag2612.
- [47] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. *Advances in neural information processing systems*, 30, 2017.
- [48] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1425–1438, 2015. ISSN 0162-8828. doi: 10.1109/tpami.2015.2487986.
- [49] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv*, 2016. doi: 10.48550/arxiv.1606.04080.
- [50] Xi Yin, Xiang Yu, Kihyuk Sohn, Xiaoming Liu, and Manmohan Chandraker. Feature transfer learning for face recognition with under-represented data. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 00:5697–5706, 2019. doi: 10.1109/cvpr.2019.000585.
- [51] 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. *International Journal of Computer Vision*, 115(3):211–252, 2015. ISSN 0920-5691. doi: 10.1007/s11263-015-0816-y.
- [52] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1452–1464, 2017. ISSN 0162-8828. doi: 10.1109/tpami.2017.2723009.
- [53] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

- [54] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8769–8778, 2018. doi: 10.1109/cvpr.2018.00914.
- [55] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv*, 2017. doi: 10.48550/arxiv.1711.05101.

A1 Appendix

A1.1 ImageNet-LT Results

Experiment	Few	Med.	Many	Overall
Baseline	27.4	46.2	61.8	49.6
AlphaNet ($\rho = 0.1$)	47.6 ± 1.60	37.1 ± 0.76	53.8 ± 0.62	45.0 ± 0.41
AlphaNet ($\rho = 0.2$)	45.7 ± 1.56	39.0 ± 0.78	55.7 ± 0.79	46.3 ± 0.52
AlphaNet ($\rho = 0.25$)	44.1 ± 1.44	40.2 ± 0.65	56.7 ± 0.57	47.1 ± 0.36
AlphaNet ($\rho = 0.3$)	42.9 ± 1.29	40.4 ± 0.72	57.0 ± 0.68	47.1 ± 0.47
AlphaNet ($\rho = 0.4$)	40.8 ± 1.85	41.5 ± 0.72	57.8 ± 0.54	47.7 ± 0.36
AlphaNet ($\rho = 0.5$)	39.7 ± 1.42	42.0 ± 0.66	58.3 ± 0.52	48.0 ± 0.37
AlphaNet ($\rho = 0.75$)	37.4 ± 1.93	42.9 ± 0.46	59.0 ± 0.40	48.3 ± 0.16
AlphaNet ($\rho = 1$)	34.6 ± 1.88	43.7 ± 0.51	59.7 ± 0.43	48.6 ± 0.24
AlphaNet ($\rho = 1.25$)	35.0 ± 1.98	43.6 ± 0.70	59.6 ± 0.50	48.6 ± 0.35
AlphaNet ($\rho = 1.5$)	32.6 ± 2.46	44.4 ± 0.49	60.3 ± 0.38	48.9 ± 0.19
AlphaNet ($\rho = 1.75$)	32.3 ± 1.42	44.4 ± 0.32	60.3 ± 0.18	48.9 ± 0.14
AlphaNet ($\rho = 2$)	31.5 ± 1.99	44.7 ± 0.46	60.5 ± 0.30	49.0 ± 0.12
AlphaNet ($\rho = 3$)	29.0 ± 2.05	45.1 ± 0.36	60.9 ± 0.28	49.0 ± 0.08

Table A1: AlphaNet with cRT baseline on ImageNet-LT.

Experiment	Few	Med.	Many	Overall
Baseline	30.4	47.2	60.2	49.9
AlphaNet ($\rho = 0.1$)	53.9 ± 0.77	29.4 ± 1.22	44.2 ± 1.22	38.5 ± 1.03
AlphaNet ($\rho = 0.2$)	52.0 ± 1.21	33.3 ± 1.44	48.0 ± 1.37	41.5 ± 1.08
AlphaNet ($\rho = 0.25$)	51.1 ± 0.63	34.4 ± 1.04	48.9 ± 0.99	42.3 ± 0.82
AlphaNet ($\rho = 0.3$)	49.8 ± 2.20	35.9 ± 1.59	50.4 ± 1.47	43.4 ± 1.10
AlphaNet ($\rho = 0.4$)	48.7 ± 1.17	37.4 ± 1.13	51.6 ± 0.95	44.4 ± 0.80
AlphaNet ($\rho = 0.5$)	46.9 ± 0.98	38.6 ± 0.87	52.9 ± 0.86	45.3 ± 0.69
AlphaNet ($\rho = 0.75$)	45.3 ± 1.89	40.2 ± 1.28	54.4 ± 1.01	46.3 ± 0.76
AlphaNet ($\rho = 1$)	41.6 ± 1.61	42.2 ± 0.53	56.0 ± 0.32	47.4 ± 0.30
AlphaNet ($\rho = 1.25$)	42.5 ± 1.41	42.1 ± 0.74	56.0 ± 0.53	47.5 ± 0.41
AlphaNet ($\rho = 1.5$)	40.1 ± 1.99	43.2 ± 0.98	56.9 ± 0.76	48.0 ± 0.53
AlphaNet ($\rho = 1.75$)	39.4 ± 2.53	43.5 ± 0.88	57.1 ± 0.70	48.2 ± 0.45
AlphaNet ($\rho = 2$)	37.5 ± 2.94	44.3 ± 0.86	57.9 ± 0.72	48.6 ± 0.32
AlphaNet ($\rho = 3$)	34.5 ± 1.91	45.3 ± 0.54	58.7 ± 0.37	49.0 ± 0.21

Table A2: AlphaNet with LWS baseline on ImageNet-LT.

A1.2 Places-LT Results

A1.3 CIFAR100-LT Results

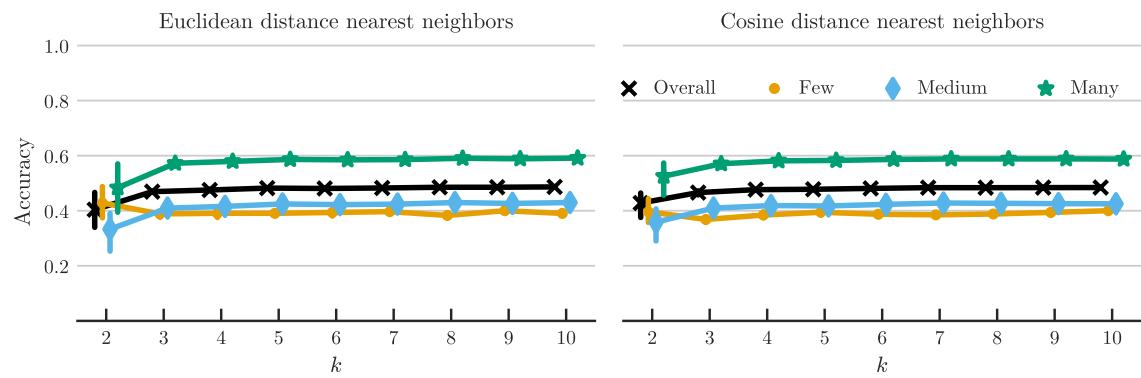


Figure A1: ...

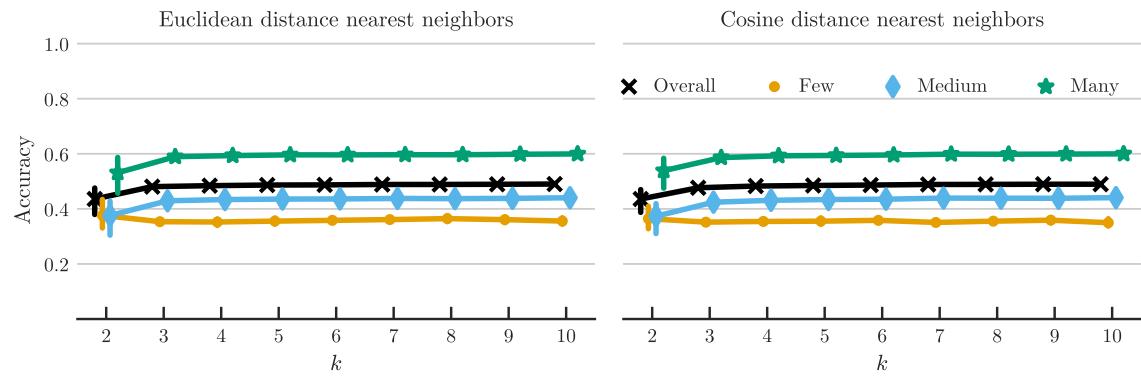


Figure A2: ...

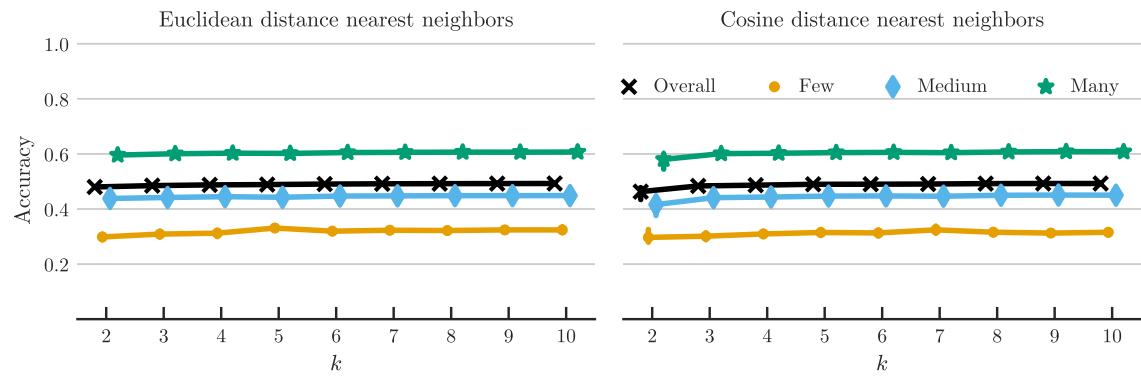


Figure A3: ...

Experiment	Few	Med.	Many	Overall
Baseline	36.5	54.4	68.9	57.5
AlphaNet ($\rho = 0.1$)	49.2 \pm 0.69	49.5 \pm 0.40	65.4 \pm 0.16	55.6 \pm 0.19
AlphaNet ($\rho = 0.2$)	47.1 \pm 0.86	50.5 \pm 0.34	66.0 \pm 0.20	56.0 \pm 0.16
AlphaNet ($\rho = 0.25$)	46.8 \pm 0.59	50.7 \pm 0.29	66.2 \pm 0.25	56.2 \pm 0.23
AlphaNet ($\rho = 0.3$)	46.1 \pm 1.16	51.2 \pm 0.50	66.5 \pm 0.29	56.4 \pm 0.21
AlphaNet ($\rho = 0.4$)	44.7 \pm 1.13	51.7 \pm 0.39	66.9 \pm 0.24	56.6 \pm 0.22
AlphaNet ($\rho = 0.5$)	43.5 \pm 0.75	52.3 \pm 0.26	67.3 \pm 0.17	56.9 \pm 0.11
AlphaNet ($\rho = 0.75$)	41.7 \pm 0.63	52.8 \pm 0.18	67.7 \pm 0.15	57.0 \pm 0.12
AlphaNet ($\rho = 1$)	40.8 \pm 1.00	53.1 \pm 0.21	67.9 \pm 0.18	57.1 \pm 0.11
AlphaNet ($\rho = 1.25$)	39.0 \pm 1.28	53.4 \pm 0.22	68.2 \pm 0.16	57.2 \pm 0.05
AlphaNet ($\rho = 1.5$)	38.2 \pm 1.22	53.6 \pm 0.25	68.4 \pm 0.17	57.2 \pm 0.06
AlphaNet ($\rho = 1.75$)	37.7 \pm 0.89	53.7 \pm 0.14	68.4 \pm 0.10	57.2 \pm 0.05
AlphaNet ($\rho = 2$)	37.1 \pm 0.98	53.8 \pm 0.12	68.5 \pm 0.11	57.2 \pm 0.06
AlphaNet ($\rho = 3$)	34.5 \pm 1.23	54.3 \pm 0.14	68.8 \pm 0.11	57.2 \pm 0.08

Table A3: AlphaNet with RIDE baseline on ImageNet-LT.

Experiment	Few	Med.	Many	Overall
Baseline	24.9	37.6	42.0	36.7
AlphaNet ($\rho = 0.1$)	38.2 \pm 1.30	30.4 \pm 0.84	37.3 \pm 0.72	34.4 \pm 0.41
AlphaNet ($\rho = 0.2$)	34.8 \pm 1.42	32.4 \pm 0.66	39.0 \pm 0.29	35.3 \pm 0.15
AlphaNet ($\rho = 0.25$)	34.4 \pm 1.27	32.8 \pm 0.82	39.3 \pm 0.34	35.5 \pm 0.29
AlphaNet ($\rho = 0.3$)	33.2 \pm 1.57	33.4 \pm 0.75	39.5 \pm 0.60	35.6 \pm 0.25
AlphaNet ($\rho = 0.4$)	32.4 \pm 1.47	33.8 \pm 0.69	39.8 \pm 0.46	35.7 \pm 0.22
AlphaNet ($\rho = 0.5$)	31.0 \pm 0.88	34.5 \pm 0.17	40.4 \pm 0.29	35.9 \pm 0.09
AlphaNet ($\rho = 0.75$)	28.6 \pm 1.27	35.5 \pm 0.40	41.0 \pm 0.13	36.1 \pm 0.11
AlphaNet ($\rho = 1$)	27.0 \pm 1.02	36.1 \pm 0.31	41.3 \pm 0.13	36.2 \pm 0.10
AlphaNet ($\rho = 1.25$)	26.9 \pm 1.23	36.0 \pm 0.47	41.3 \pm 0.14	36.1 \pm 0.16
AlphaNet ($\rho = 1.5$)	25.5 \pm 0.89	36.5 \pm 0.36	41.6 \pm 0.21	36.2 \pm 0.11
AlphaNet ($\rho = 1.75$)	25.4 \pm 1.32	36.5 \pm 0.29	41.5 \pm 0.23	36.2 \pm 0.11
AlphaNet ($\rho = 2$)	24.9 \pm 1.38	36.6 \pm 0.49	41.5 \pm 0.20	36.1 \pm 0.10
AlphaNet ($\rho = 3$)	22.3 \pm 1.56	37.3 \pm 0.28	41.9 \pm 0.20	36.1 \pm 0.14

Table A4: AlphaNet with cRT baseline on Places-LT.

Experiment	Few	Med.	Many	Overall
Baseline	28.7	39.1	40.6	37.6
AlphaNet ($\rho = 0.1$)	42.5 ± 1.03	29.5 ± 1.01	34.4 ± 0.94	33.8 ± 0.58
AlphaNet ($\rho = 0.2$)	41.3 ± 1.30	31.0 ± 0.81	35.5 ± 0.81	34.6 ± 0.44
AlphaNet ($\rho = 0.25$)	40.5 ± 2.25	30.9 ± 2.92	35.6 ± 2.20	34.4 ± 1.67
AlphaNet ($\rho = 0.3$)	38.7 ± 1.13	33.2 ± 1.12	37.0 ± 0.70	35.6 ± 0.51
AlphaNet ($\rho = 0.4$)	38.0 ± 1.45	33.6 ± 1.18	37.3 ± 0.65	35.8 ± 0.50
AlphaNet ($\rho = 0.5$)	37.1 ± 1.39	34.4 ± 0.80	37.7 ± 0.52	36.1 ± 0.31
AlphaNet ($\rho = 0.75$)	35.5 ± 1.25	35.3 ± 0.59	38.5 ± 0.28	36.5 ± 0.16
AlphaNet ($\rho = 1$)	34.6 ± 0.97	35.8 ± 0.54	38.6 ± 0.39	36.6 ± 0.22
AlphaNet ($\rho = 1.25$)	32.6 ± 1.28	36.8 ± 0.60	39.3 ± 0.39	36.9 ± 0.19
AlphaNet ($\rho = 1.5$)	32.2 ± 1.17	37.2 ± 0.36	39.5 ± 0.39	37.0 ± 0.11
AlphaNet ($\rho = 1.75$)	31.7 ± 1.35	37.3 ± 0.43	39.5 ± 0.27	37.0 ± 0.09
AlphaNet ($\rho = 2$)	30.9 ± 1.28	37.6 ± 0.40	39.8 ± 0.18	37.1 ± 0.14
AlphaNet ($\rho = 3$)	27.8 ± 1.94	38.5 ± 0.51	40.2 ± 0.29	37.1 ± 0.10

Table A5: AlphaNet with LWS baseline on Places-LT.

Experiment	Few	Med.	Many	Overall
Baseline	25.8	52.1	69.3	50.2
AlphaNet ($\rho = 0.1$)	38.2 ± 3.22	38.1 ± 5.01	56.3 ± 7.65	44.5 ± 3.50
AlphaNet ($\rho = 0.2$)	36.0 ± 4.34	41.4 ± 5.41	59.1 ± 8.31	45.9 ± 3.58
AlphaNet ($\rho = 0.25$)	34.0 ± 1.70	44.1 ± 1.68	62.6 ± 1.54	47.6 ± 0.76
AlphaNet ($\rho = 0.3$)	32.9 ± 1.20	45.3 ± 1.02	64.0 ± 0.78	48.1 ± 0.51
AlphaNet ($\rho = 0.4$)	31.1 ± 1.27	45.6 ± 1.63	64.8 ± 1.22	48.0 ± 0.84
AlphaNet ($\rho = 0.5$)	30.9 ± 1.82	47.0 ± 1.25	65.7 ± 0.94	48.7 ± 0.41
AlphaNet ($\rho = 0.75$)	28.6 ± 1.63	48.3 ± 0.84	66.6 ± 0.65	48.8 ± 0.26
AlphaNet ($\rho = 1$)	27.2 ± 1.69	49.1 ± 0.85	67.4 ± 0.62	48.9 ± 0.30
AlphaNet ($\rho = 1.25$)	26.1 ± 1.54	49.8 ± 0.59	68.0 ± 0.47	49.1 ± 0.42
AlphaNet ($\rho = 1.5$)	25.0 ± 1.15	50.1 ± 1.12	67.9 ± 1.01	48.8 ± 0.63
AlphaNet ($\rho = 1.75$)	24.8 ± 1.09	50.3 ± 0.61	68.4 ± 0.33	49.0 ± 0.27
AlphaNet ($\rho = 2$)	24.3 ± 1.74	51.0 ± 0.55	68.9 ± 0.47	49.2 ± 0.34
AlphaNet ($\rho = 3$)	22.2 ± 1.71	51.6 ± 0.86	69.3 ± 0.58	49.0 ± 0.31

Table A6: AlphaNet with RIDE baseline on CIFAR-100-LT.

Experiment	Few	Med.	Many	Overall
Baseline	33.6	49.4	69.7	51.8
AlphaNet ($\rho = 0.1$)	40.9 ± 4.12	37.4 ± 7.63	62.1 ± 5.07	47.1 ± 3.27
AlphaNet ($\rho = 0.2$)	42.6 ± 2.47	32.9 ± 4.98	58.7 ± 3.36	44.8 ± 2.43
AlphaNet ($\rho = 0.25$)	42.3 ± 2.45	34.4 ± 6.36	59.5 ± 4.83	45.5 ± 3.31
AlphaNet ($\rho = 0.3$)	40.9 ± 3.77	36.6 ± 6.76	61.3 ± 4.78	46.6 ± 3.14
AlphaNet ($\rho = 0.4$)	42.3 ± 1.73	33.1 ± 6.20	58.9 ± 4.24	44.9 ± 3.25
AlphaNet ($\rho = 0.5$)	43.2 ± 1.17	32.1 ± 4.73	58.7 ± 3.55	44.7 ± 2.64
AlphaNet ($\rho = 0.75$)	43.1 ± 1.45	31.3 ± 5.14	57.7 ± 4.01	44.1 ± 2.95
AlphaNet ($\rho = 1$)	42.5 ± 2.31	31.3 ± 6.45	57.4 ± 4.50	43.8 ± 3.21
AlphaNet ($\rho = 1.25$)	41.7 ± 4.03	33.4 ± 8.11	59.1 ± 5.61	44.9 ± 3.91
AlphaNet ($\rho = 1.5$)	43.6 ± 0.87	31.3 ± 3.49	57.8 ± 2.57	44.3 ± 1.95
AlphaNet ($\rho = 1.75$)	42.7 ± 0.70	30.0 ± 4.75	56.5 ± 3.51	43.1 ± 2.93
AlphaNet ($\rho = 2$)	43.1 ± 0.79	28.4 ± 1.86	55.4 ± 1.59	42.2 ± 1.39
AlphaNet ($\rho = 3$)	43.2 ± 0.93	28.0 ± 2.00	55.3 ± 1.45	42.1 ± 1.43

Table A7: AlphaNet with LTR baseline on CIFAR-100-LT.

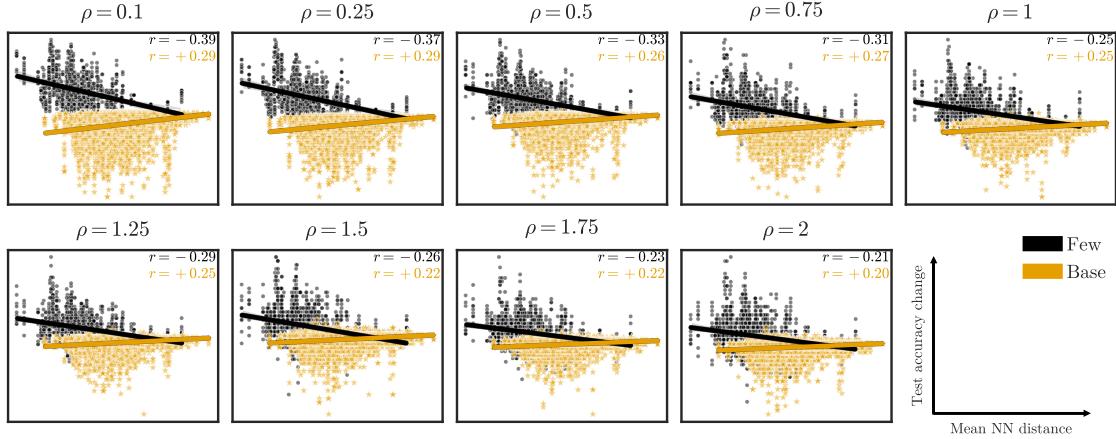


Figure A4: ...

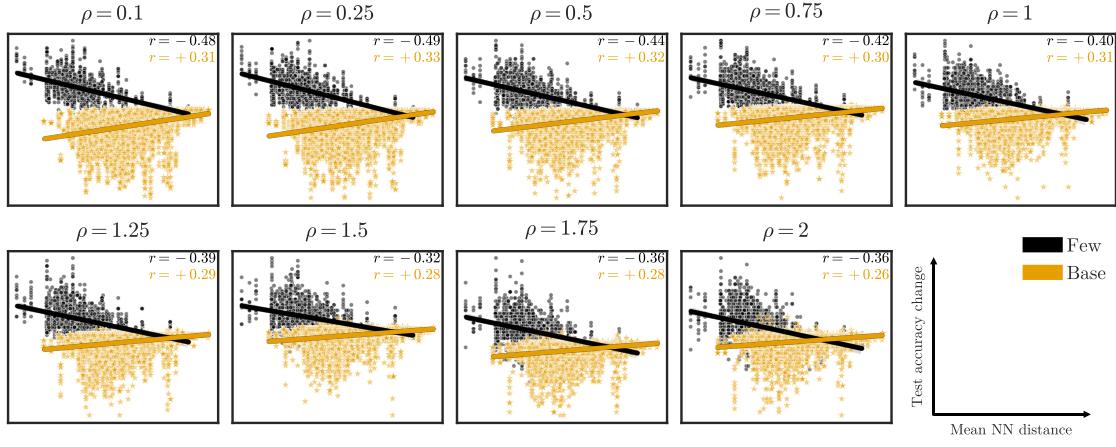


Figure A5: ...

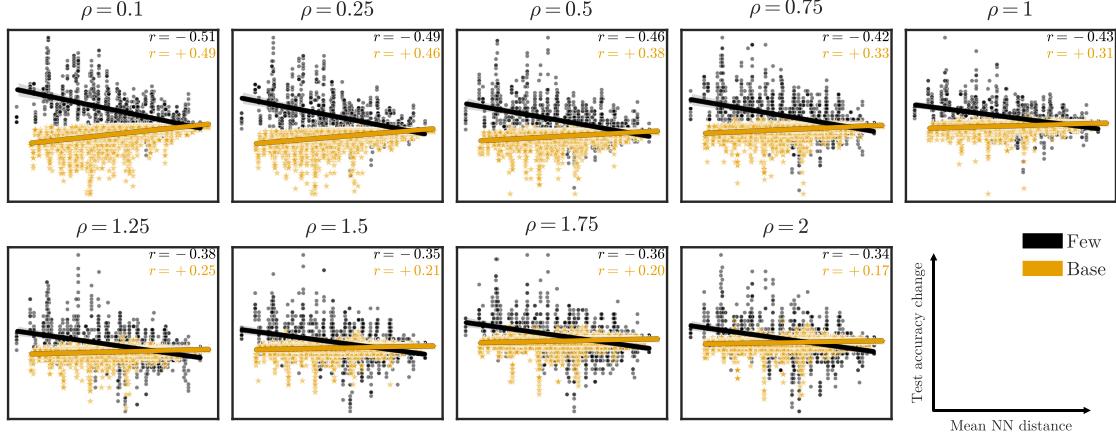


Figure A6: ...

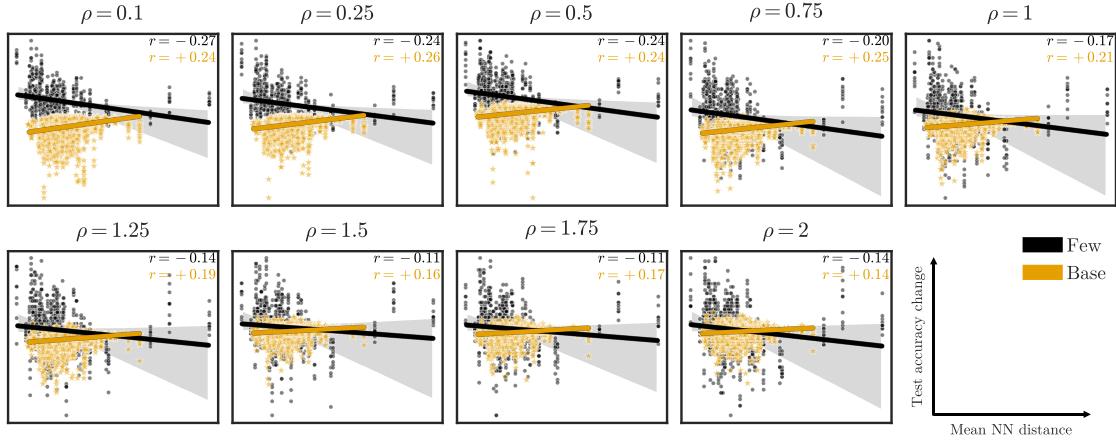


Figure A7: ...

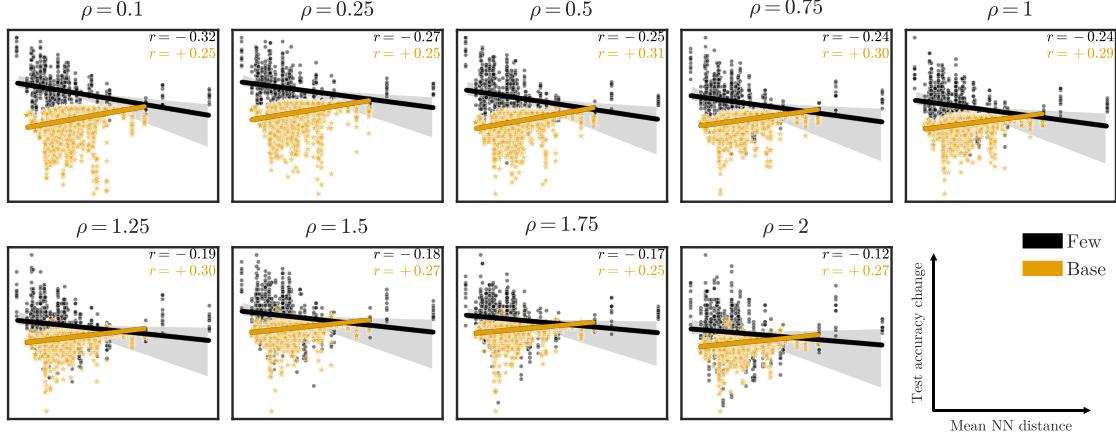


Figure A8: ...

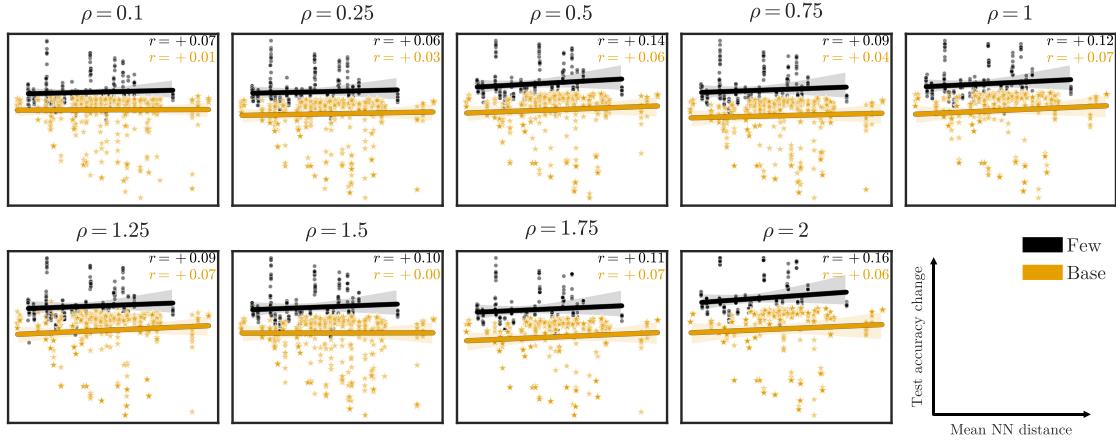


Figure A9: ...

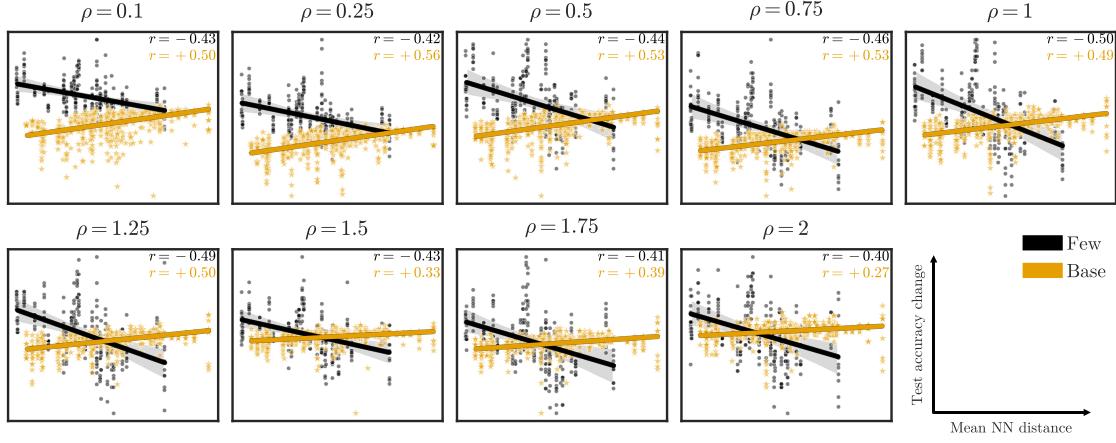


Figure A10: ...

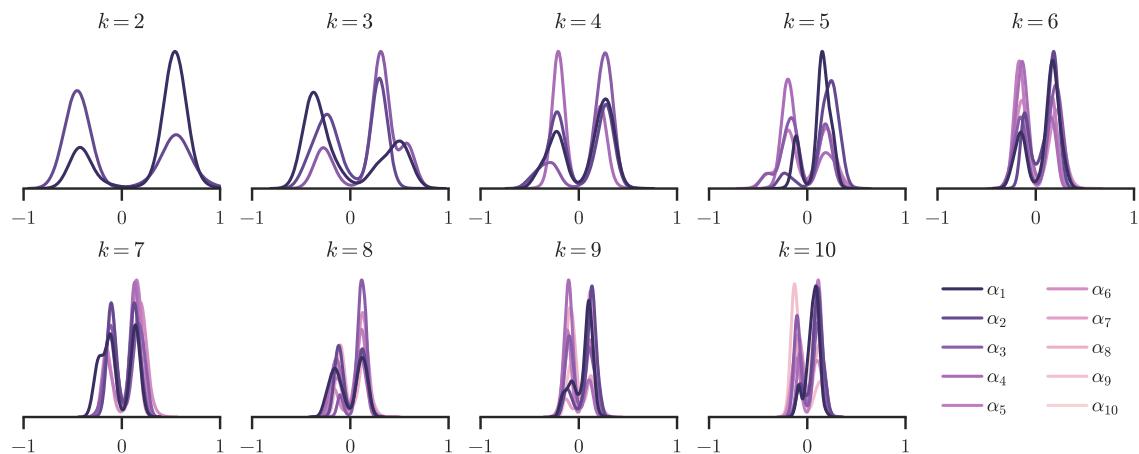


Figure A11: ...

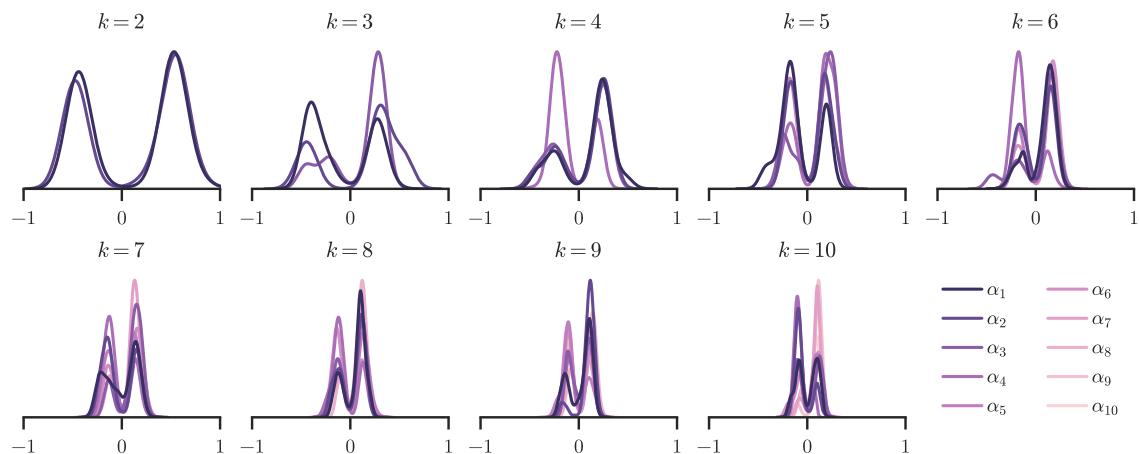


Figure A12: ...

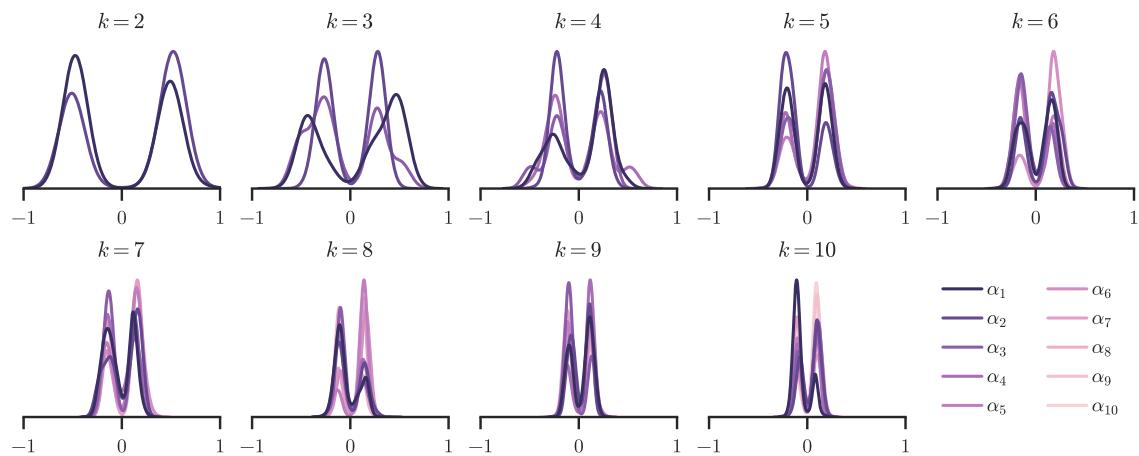


Figure A13: ...

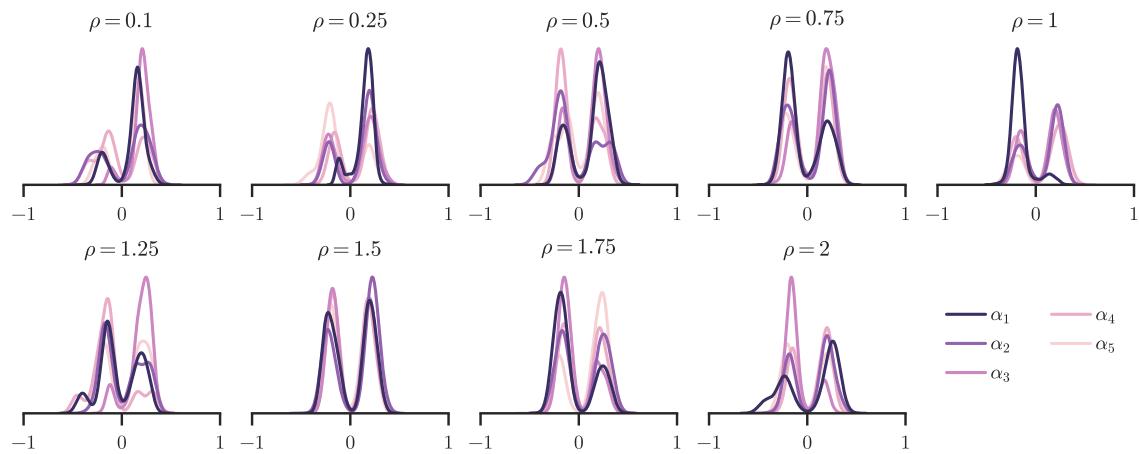


Figure A14: ...

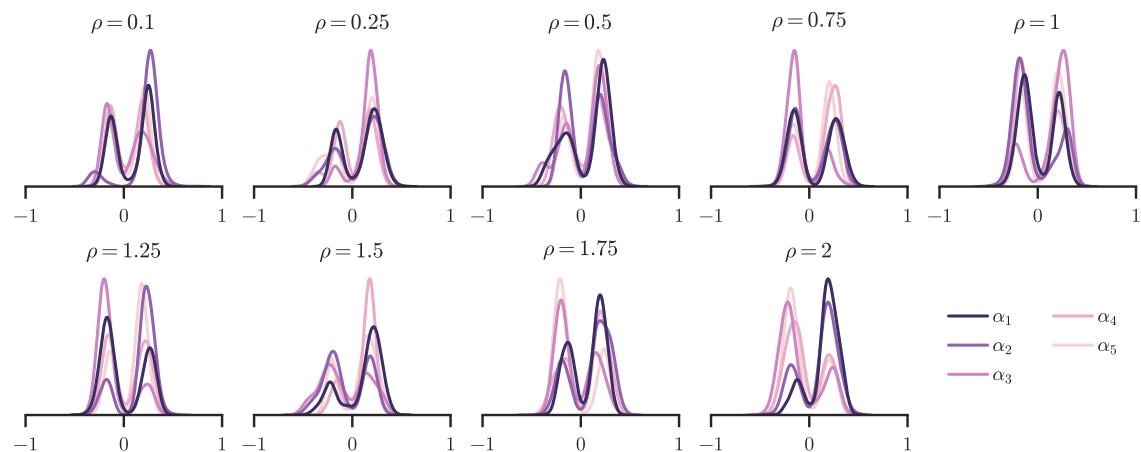


Figure A15: ...

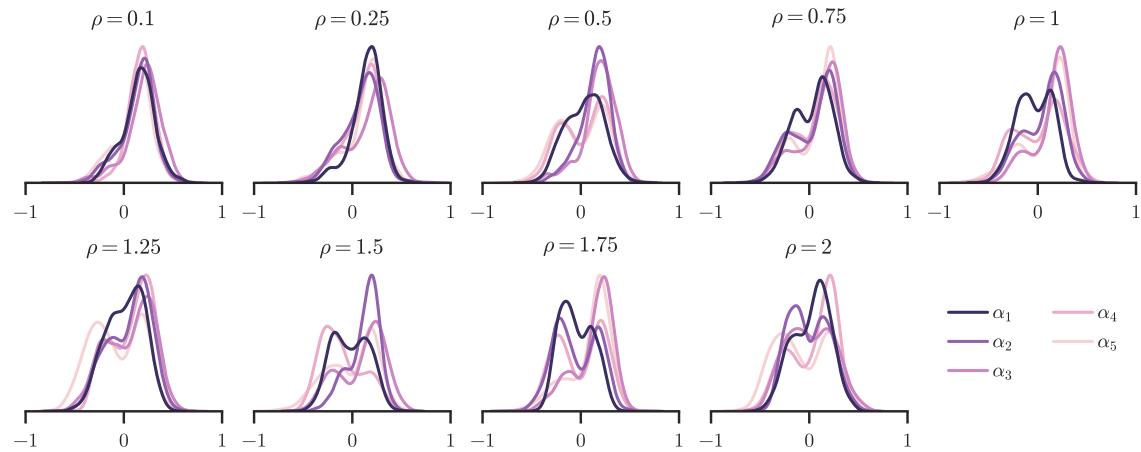


Figure A16: ...

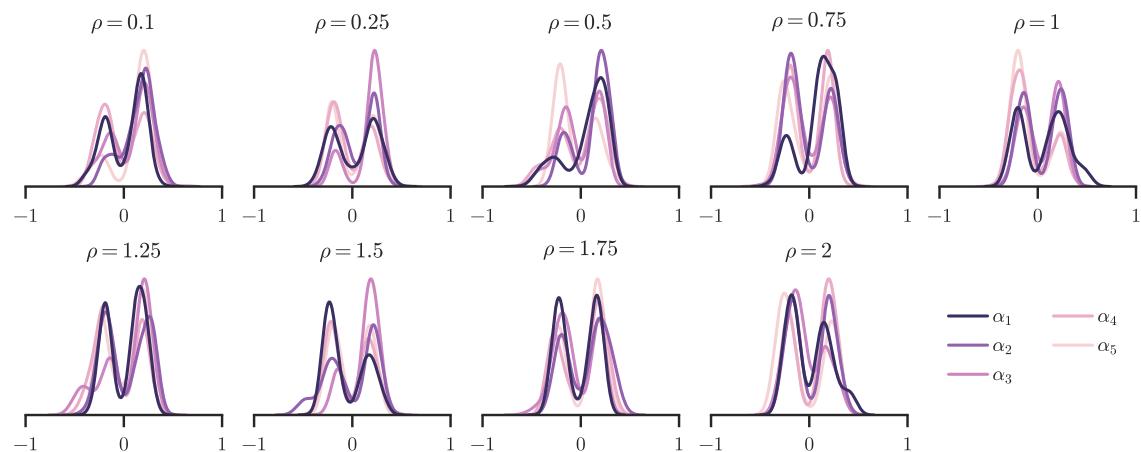


Figure A17: ...

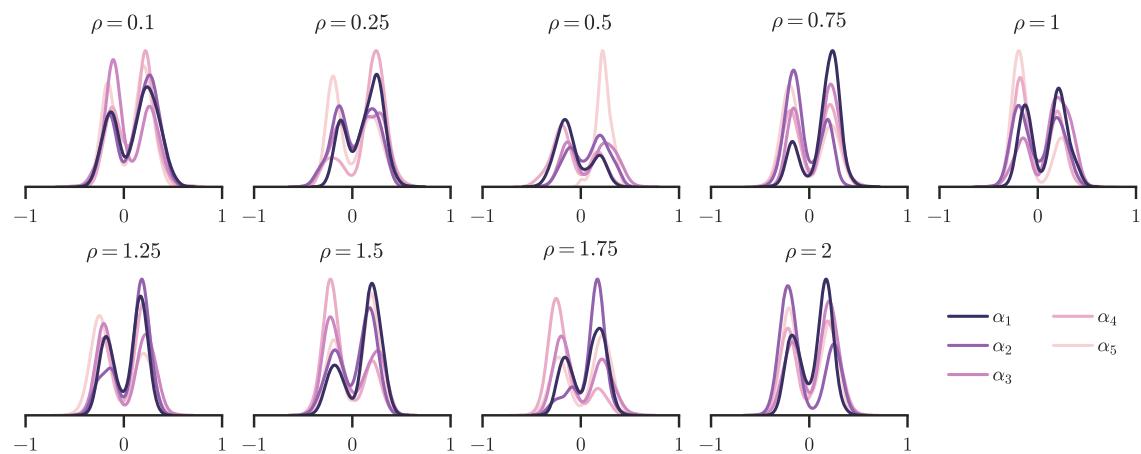


Figure A18: ...

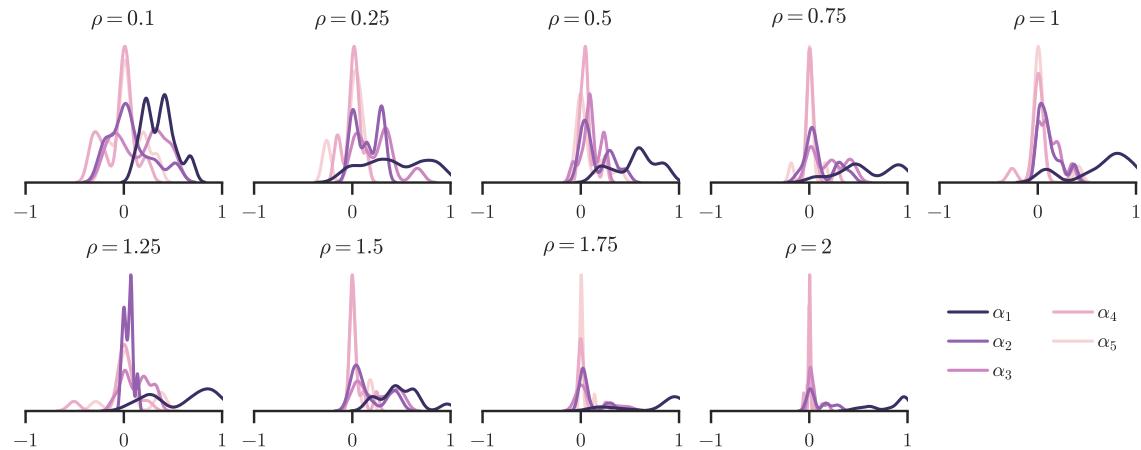


Figure A19: ...

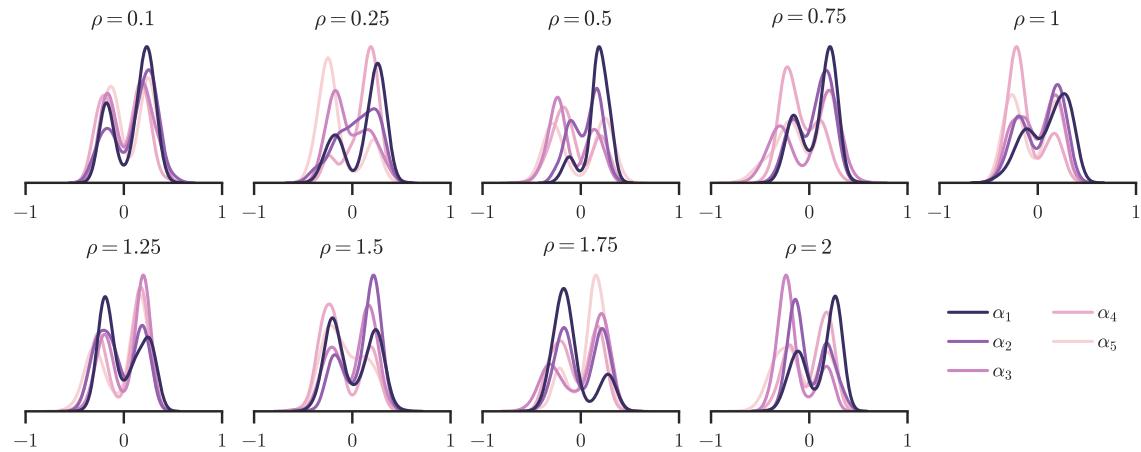


Figure A20: ...