AlphaNet: Improving Long-Tail Classification By Combining Classifiers

Nadine Chang [1,*] nchang1@cs.cmu.edu
Jayanth Koushik [1,*] jkoushik@andrew.cmu.edu
Aarti Singh [1] aartisingh@cmu.edu
Martial Hebert [1] hebert@cs.cmu.edu
Yu-Xiong Wang [2] yxw@illinois.edu
Michael J. Tarr [1] michaeltarr@cmu.edu

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. Analyzing the predictions of long-tail methods for rare classes reveals that a large number of errors are due to misclassification of rare items as visually similar frequent classes. To address this problem, we introduce AlphaNet, a method that can be applied to existing models, performing post hoc correction on classifiers of rare classes. Starting with a pre-trained model, we find frequent classes that are closest to rare classes in the model's representation space and learn weights to update rare classifiers with a linear combination of frequent classifiers. AlphaNet, applied on several different models, greatly improves test accuracy for rare classes in multiple long-tail datasets. We then analyze predictions from AlphaNet and find that remaining errors are to often due to fine-grained differences among semantically similar classes (e.g., dog breeds). Evaluating with semantically similar classes grouped together, AlphaNet also improves overall accuracy, showing that the method is practical for long-tail classification problems.

1. Introduction

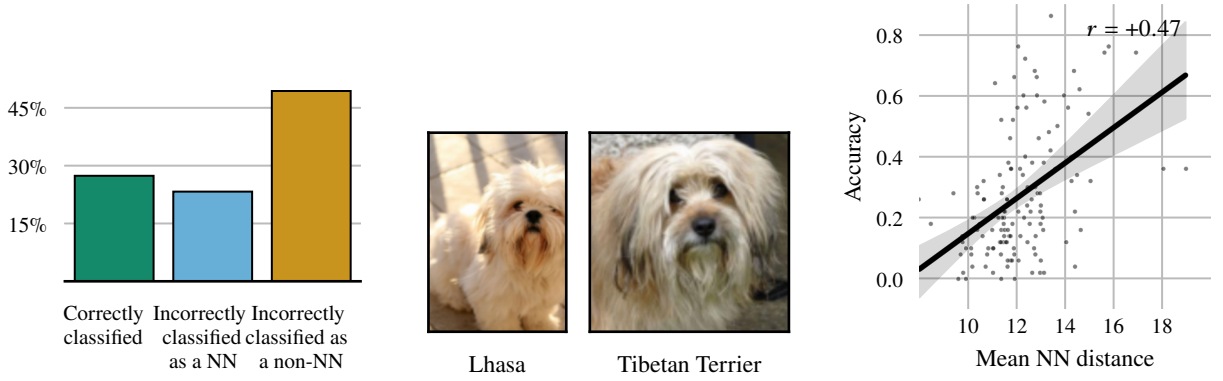
The significance of long-tailed distributions in real-world applications (such as autonomous driving [1] and medical image analysis [2]) has spurred a variety of approaches for long-tail classification [3]. Learning in this setting is challenging because many classes are "rare" – having only a small number of training samples. Some methods re-sample more data for rare classes in an effort to address data imbalances [4, 5], while other methods adjust learned classifiers to re-weight them in favor of rare classes [6]. Both re-sampling and re-weighting methods provide strong baselines for long-tail classification tasks. However, state-of-the-art results are achieved by more complex methods that, for example, learn multiple experts [7, 8], perform multi-stage distillation [9], or use a combination of weight balancing, data re-sampling, and loss decay [10].

Despite these advances, accuracy on rare classes continues to be significantly worse than overall accuracy using these methods. For example, on the ImageNet-LT dataset, the 6-expert ensemble routing diverse experts (RIDE) model [7] has an average accuracy of 68.9% on frequent classes, but an average accuracy of 36.5% on rare classes. In addition to reducing overall accuracy, such performance imbalances raise ethical concerns in contexts where unequal accuracy leads to biased outcomes, for instance in medical imaging [11] or face detection [12]. For example, models trained on chest X-rays consistently under-diagnosed minority groups [13]. Similarly, cardiac image segmentation showed significant differences between racial groups [14].

To understand the poor rare class performance of long-tail models, we analyzed the predictions of the RIDE model [7] on test samples from 'few' split classes (i.e. classes with limited training samples) in ImageNet-LT [15]. Figure 1a shows predictions binned into three groups: 1) samples predicted correctly; 2) samples incorrectly predicted as a visually similar class (e.g., predicting 'husky' instead of 'malamute'); and 3) samples incorrectly predicted as a visually dissimilar class (e.g., predicting 'car' instead of 'malamute'). A significant portion of the misclassifications (about 26%) are to visually similar classes. Figure 1b shows samples from one pair of visually similar classes; the differences are subtle, and can be hard even for humans to identify. We next analyzed the relationship between per-class test accuracy and mean distance of a class to its nearest neighbors (see Section 3 for details). Figure 1c shows a strong positive correlation between accuracy and mean distance – 'few' split classes with close neighbors have lower test accuracy than classes with distant neighbors.

Based on these analyses, we designed a method to directly improve the accuracy of rare classes in long-tail classification. Our method, AlphaNet, uses information from visually similar *frequent* classes to improve classifiers for rare classes. Figure 2 illustrates the pipeline of our method. At a high level, AlphaNet can be seen as moving the classifiers for

This document can be read online at https://jkoushik.me/alphanet.
[*] Equal contribution. [1] Carnegie Mellon University.
[2] University of Illinois Urbana-Champaign.



(a) Predictions from cRT model on test samples from 'few' split of ImageNet-LT. A large percentage of samples are misclassified as a visually similar 'base' split class.

(b) Sample images from two classes in ImageNet-LT. 'Lhasa' is a 'few' split class, and 'Tibetan terrier' is a 'base' split class. The classes are visually similar, leading to misclassifications.

(c) Per-class test accuracy of cRT model on 'few' split of ImageNet-LT versus the mean distance to 10 nearest neighbors from the 'base' split. The line is a bootstrapped linear regression fit, and r (top right) is Pearson correlation. There is a high correlation, i.e., 'few' split classes with close nearest neighbors were more likely to be misclassified.

Figure 1: Analysis of test accuracy for 'few' split of ImageNet-LT.

rare classes based on their position relative to visually similar classes. Importantly, AlphaNet updates classifiers without making any changes to the representation space, or to other classifiers in the model. It performs a *post-hoc* correction, and as such, is applicable to use cases where existing base classifiers are either unavailable or fixed (e.g., due to commercial interests or data privacy protections). The simplicity of our method lends to computational advantages – AlphaNet can be trained rapidly, and on top of any classification model. We will demonstrate that AlphaNet significantly improves the accuracy of rare classes for a variety of long-tail classification models across multiple datasets.

2. Related work

Combining, creating, modifying, and learning model weights are concepts that have been 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-tail 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 [16, 17, 18, 19, 20]. 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 [21, 22, 23, 24, 25, 26, 27, 28]. Other studies go even further and instead of modifying networks, they create entirely new networks exclusively from training samples [29, 30, 31]. In contrast, AlphaNet only combines existing classifiers, without having to create new classifiers or train networks from scratch.

2.2. Classifier or feature composition

There have been works that learn better embedding spaces for image annotation [32], or use classification scores as useful features [33]. However, these approaches do not attempt to compose classifiers nor do they address the long-tail problem. For transfer learning with non-deep methods, there have been attempts to use and combine support vector machines (SVMs). In one method [34], SVMs are trained per object instance, and a hierarchical structure is required for combination in the datasets of interest. However, such a structure is typically not guaranteed nor provided in long-tailed datasets. Another SVM method uses regularized minimization to learn the coefficients necessary to combine patches from other classifiers [35].

While these approaches are conceptually similar to our method, AlphaNet has the additional advantage of *learning* the compositional coefficients. Specifically, different novel classes will have their own set of coefficients, and similar novel classes will naturally have similar coefficients. Learning such varying sets of coefficients is difficult in previous classical approaches, which either learn a fixed set of coefficients for all 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 [30, 36, 37, 38]. However, such composition is often guided by supervision from additional attributes or textual descriptions, which are not needed by AlphaNet.

2.3. Learning transformations between models and classes

Some studies have attempted to learn transformations of model weights with stochastic gradient descent (SGD) optimization [39, 40]. Additionally, there is empirical evidence [41] showing the existence of a generic nonlinear transformation from small-sample to large-sample models for different types of feature spaces and classifier models. Finally, in the case where one learns the transformation from the source function to a related target function, there are theoretical guarantees on performance [42]. AlphaNet is similar in that 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 [20, 41, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]. A wide variety of prior studies have attempted to transfer knowledge from tasks with abundant data to completely novel tasks [23, 40, 53]. However, due to the nature of low-shot learning, these approaches are limited to a small number of tasks, which is problematic since the visual world involves a large number of tasks with varying amounts of information.

2.5. Long-tail learning

The restrictions of low-shot learning have been addressed by the paradigm referred to as long-tail learning, where the distribution of class sizes (number of training samples) closely models that of the visual world; many classes have only a few samples, while a few classes have many samples. Recent work achieves state-of-the-art performance on long-tailed recognition by learning multiple experts [7, 8]. Both of these complex ensemble methods require a two-stage training method. Other approaches re-balance the class sizes at different stages of model training [5], transfer features from common classes to rare classes [15], or transfer intra-class variance [54]. However, approaches for knowledge transfer require complex architectures, such as a specialized attention mechanism with memory [15]. While recent studies have largely focused on representation space transferability or complex ensembles, strong baselines have been established by exploring the potential of operating in classifier space [6]. 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 using only re-sampling strategies achieve good performance [? ?]. These strong baselines support our approach of operating in classifier space – AlphaNet combines strong classifiers to improve weak classifiers.

3. Method

In this work, we define the distance between two classes as the distance between their average training set representation. Given a classification model, let f be the function mapping images to vectors in a d-dimensional space (typically, this is the output of the penultimate layer in convolutional networks). For a class c with n^c training samples $I_1^c, \ldots, I_{n^c}^c$, let $z^c \equiv (1/n^c) \sum_i f\left(I_i^c\right)$ be the average training set representation. Given a distance function $\mu : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$, for classes c_1 and c_2, we define the distance between two classes as $m_\mu(c_1, c_2) \equiv \mu(z^{c_1}, z^{c_2})$.

Given a long-tailed dataset, the 'few' split, C^F, is defined as the set of classes with fewer than T training samples, for some constant T (for the datasets used in our experiments, $T = 20$). The set of remaining classes forms the 'base' split, C^B. AlphaNet is applied to update the 'few' split classifiers using nearest neighbors from the 'base' split. We will use the term 'classifier' to denote the linear mapping from feature vectors to class scores. In convolutional networks, the last layer is generally a matrix of all individual classifiers. The bias terms are not updated by AlphaNet, and are learned separately (more on this later).

3.1. AlphaNet implementation

Figure 2 shows the pipeline of our method. Given a 'few' split class c, let the k nearest 'base' split neighbors (based on m_μ) be $q_1^c, \ldots, q_k^c$, with corresponding classifiers $v_1^c, \ldots, v_k^c$. AlphaNet maps the nearest neighbor classifiers (concatenated together into a vector $\overline{v}^c$) to a set of coefficients $\alpha_1^c, \ldots, \alpha_k^c$. The α coefficients (denoted together as a vector α^c), are then scaled to unit 1-norm (the reasoning behind this will be explained later), to obtain $\tilde{\alpha}^c$:

$$\tilde{\alpha}^c = \alpha^c / \|\alpha^c\|_1. \tag{1}$$

The scaled coefficients are used to update the 'few' split classifier ($w^c \rightarrow \hat{w}^c$) through a linear combination:

$$\hat{w}^c \equiv w^c + \sum_{i=1}^{k} \tilde{\alpha}_i^c v_i^c \tag{2}$$

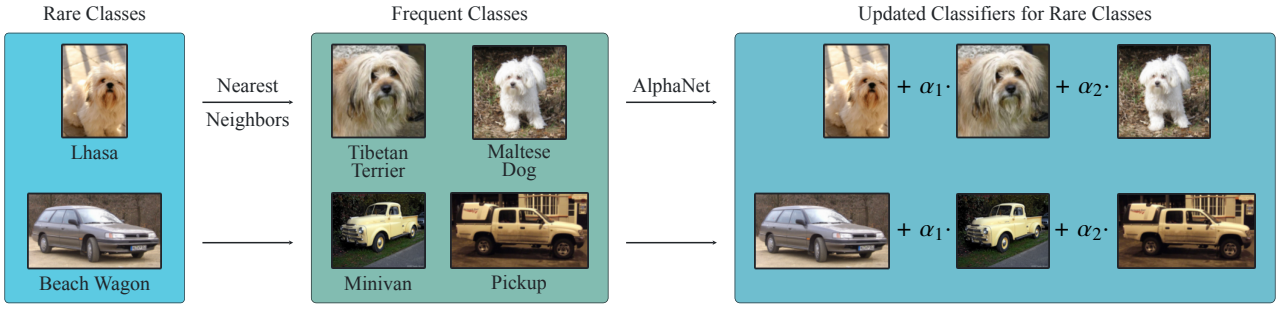


Figure 2: Pipeline for AlphaNet. Given a rare class, we identify the nearest neighbor frequent classes based on visual similarity, and then update the rare class' classifier using learned coefficients. One coefficient, α, is learned for each nearest neighbor. The result is an improved classifier for the rare class.

Due to the 1-norm scaling, we have

$$\begin{aligned}\|\hat{w}^c - w^c\|_2 &\leq \sum_{i=1}^{k} |\tilde{\alpha}_i^c| \left\|v_i^c\right\|_2 \quad \text{(Cauchy-Schwarz inequality)} \\ &\leq \max_{i=1,\ldots,k} \left\|v_i^c\right\|_2 \sum_{i=1}^{k} |\tilde{\alpha}_i^c| \\ &= \max_{i=1,\ldots,k} \left\|v_i^c\right\|_2 \|\tilde{\alpha}^c\|_1 \\ &= \max_{i=1,\ldots,k} \left\|v_i^c\right\|_2, \end{aligned} \tag{3}$$

that is, a classifier's change is bound by the norm of its class' nearest neighbors. Thanks to this, we do not need to update or rescale 'base' split classifiers, which may not be possible in certain domains.

A single network is used to generate coefficients for every 'few' split class. So, once trained, AlphaNet can be applied even to classes not seen during training. This will be further explored in future work.

3.2. Training

The main trainable component of AlphaNet is a network (with parameters θ) which maps $\overline{v}^c$ to α^c. We also learn a new set of bias values for the 'few' split classes, $\tilde{b}_1, \ldots, \tilde{b}_{|C^F|}$ [1]. Given a training image I, the per-class prediction scores are given by

$$s(c; I) = \begin{cases} f(I)^T \hat{w}^c + \hat{b}_c & c \in C^F. \\ f(I)^T w^c + b_c & c \in C^B. \end{cases} \tag{4}$$

These scores are used to compute the softmax cross-entropy loss [2], which is minimized with respect to θ and $\tilde{b}$ using a SGD optimizer.

4. Experiments

4.1. Experimental setup

Datasets. We evaluated AlphaNet using three long-tailed datasets: ImageNet-LT, Places-LT [15], and CIFAR-100-LT [10]. These datasets are sampled from their respective original datasets, ImageNet [55], Places365 [56], and CIFAR-100 [57] such that the new distributions follow a standard long-tailed distribution. For CIFAR-100-LT, we used an imbalance factor of 100 [10].

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 models is iNaturalist [58]. Results for this dataset, however, are much more balanced across splits. For example, the classifier re-training (cRT) model achieves the following per-split accuracies: 75.9% ('many'), 71.4% ('medium'), and 70.4% ('few'). The 'few' split accuracy is only 0.8 points lower than

[1] $|C^F|$ is the cardinality of C^F, i.e., the number of 'few' split classes.

[2] We use softmax cross-entropy loss in our experiments, but any loss function can be used.

the overall accuracy (71.2%); so the dataset does not represent a valid use case for our proposed method, and we omitted the dataset from our main experiments. Results for this dataset are included in the appendix (Section **??**).

Training data sampling. In order to prevent over-fitting 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 size of their class [3]. 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 the following section, and the full set of results is in Section **??**.

Training. [4] All experiments used an AlphaNet module with three 32 unit layers, and Leaky-ReLU activation [59]. 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 [60] 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. Long-tail classification results

Baseline models. First, we applied AlphaNet on the cRT and learnable weight scaling (LWS) models [6]. These methods have good overall accuracy, but accuracy for 'few' split classes is much lower. On the ImageNet-LT dataset, average 'few' split? accuracy using a ResNeXt-50 backbone is nearly 20 points below the overall accuracy, as seen in Table 1, which also shows other baseline models [6] – nearest class mean (NCM), which predicts the nearest neighbor using average class representation, and τ-normalized, which re-balances classifiers by adjusting their classifier weights. Using features extracted from cRT and LWS 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. For $\rho = 1$, average 'few' split accuracy was boosted by 7 points for the cRT model, and about 11 points for the LWS model.

Method	Few	Med.	Many	Overall
ImageNet-LT				
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
α cRT ($\rho = 0.5$)	39.7 $^{1.42}$	42.0 $^{0.66}$	58.3 $^{0.52}$	48.0 $^{0.37}$
α cRT ($\rho = 1$)	34.6 $^{1.88}$	43.7 $^{0.51}$	59.7 $^{0.43}$	48.6 $^{0.24}$
α 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
α LWS ($\rho = 0.5$)	46.9 $^{0.98}$	38.6 $^{0.87}$	52.9 $^{0.86}$	45.3 $^{0.69}$
α LWS ($\rho = 1$)	41.6 $^{1.61}$	42.2 $^{0.53}$	56.0 $^{0.32}$	47.4 $^{0.30}$
α LWS ($\rho = 1.5$)	40.1 $^{1.99}$	43.2 $^{0.98}$	56.9 $^{0.76}$	48.0 $^{0.53}$
Places-LT				
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
α cRT ($\rho = 0.5$)	31.0 $^{0.88}$	34.5 $^{0.17}$	40.4 $^{0.29}$	35.9 $^{0.09}$
α cRT ($\rho = 1$)	27.0 $^{1.02}$	36.1 $^{0.31}$	41.3 $^{0.13}$	36.2 $^{0.10}$
α 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
α LWS ($\rho = 0.5$)	37.1 $^{1.39}$	34.4 $^{0.80}$	37.7 $^{0.52}$	36.1 $^{0.31}$
α LWS ($\rho = 1$)	34.6 $^{0.97}$	35.8 $^{0.54}$	38.6 $^{0.39}$	36.6 $^{0.22}$
α LWS ($\rho = 1.5$)	32.2 $^{1.17}$	37.2 $^{0.36}$	39.5 $^{0.39}$	37.0 $^{0.11}$

Table 1: Mean split accuracy in percents (standard deviation in super-script) of AlphaNet and various baseline methods on ImageNet-LT and Places-LT. cRT and 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 1. 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.

Expert model. Next, we applied AlphaNet on the 6-expert ensemble RIDE model [7]. 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 2 shows the base results for the expert models, along with AlphaNet results for $\rho = 0.5, 1, 2$. On ImageNet-LT,

[3] For example, suppose there are 2 'base' classes; class 1 has 10 samples and class 2 has 100 samples. Then, each class 1 sample is assigned a weight of 0.1, and each class 2 sample is assigned a weight of 0.01. Sampling with this weight distribution, both classes have a 50% chance of being sampled.

[4] Code used for the experiments is available at github.com/jayanthkoushik/alphanet.

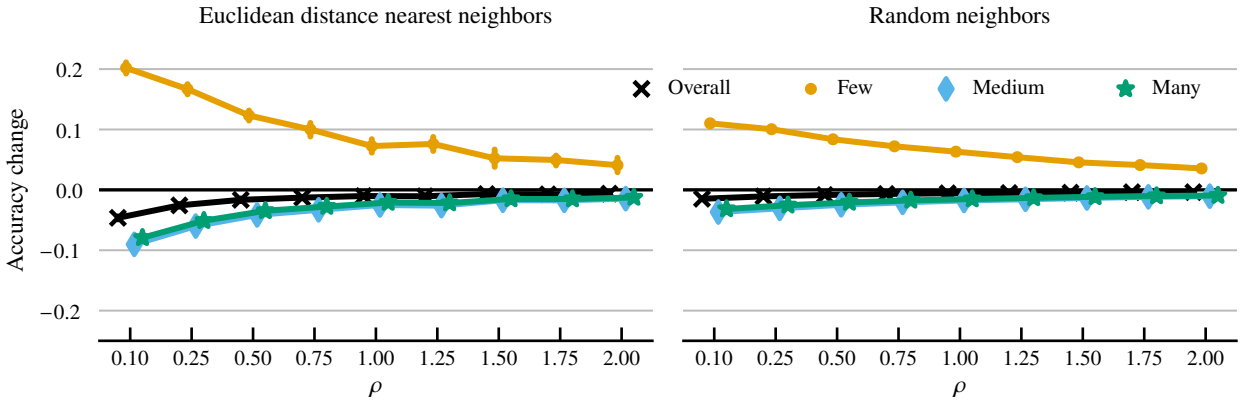


Figure 3: Change in split accuracy for AlphaNet training of cRT model on ImageNet-LT. For each value of ρ, the two plots show the raw change in split accuracy for AlphaNet compared to the baseline cRT model, as well as the change in overall accuracy. Left shows the results for normal training with 5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

Method	Few	Med.	Many	Overall
ImageNet-LT				
RIDE	36.5	54.4	68.9	57.5
α RIDE ($\rho = 0.5$)	$43.5^{0.75}$	$52.3^{0.26}$	$67.3^{0.17}$	$56.9^{0.11}$
α RIDE ($\rho = 1$)	$40.8^{1.00}$	$53.1^{0.21}$	$67.9^{0.18}$	$57.1^{0.11}$
α 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
α RIDE ($\rho = 0.5$)	$30.9^{1.82}$	$47.0^{1.25}$	$65.7^{0.94}$	$48.7^{0.41}$
α RIDE ($\rho = 1$)	$27.2^{1.69}$	$49.1^{0.85}$	$67.4^{0.62}$	$48.9^{0.30}$
α RIDE ($\rho = 1.5$)	$25.0^{1.15}$	$50.1^{1.12}$	$67.9^{1.01}$	$48.8^{0.63}$

Table 2: Mean split accuracy in percents (standard deviation in super-script) on ImageNet-LT and CIFAR-100-LT using the ensemble RIDE model [7]. RIDE applies AlphaNet on average features from the ensemble.

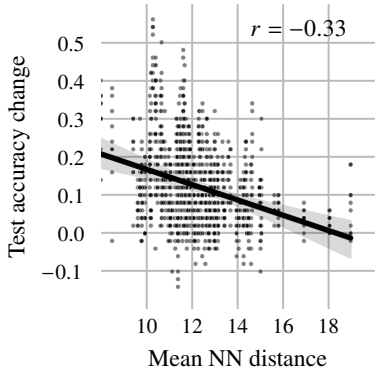


Figure 4: Change in per-class accuracy for AlphaNet applied over cRT features, versus mean Euclidean distance to five nearest neighbors. Comparing with Figure 1c, we can see that AlphaNet provides the largest boost to classes with poor baseline performance, which have close nearest neighbors.

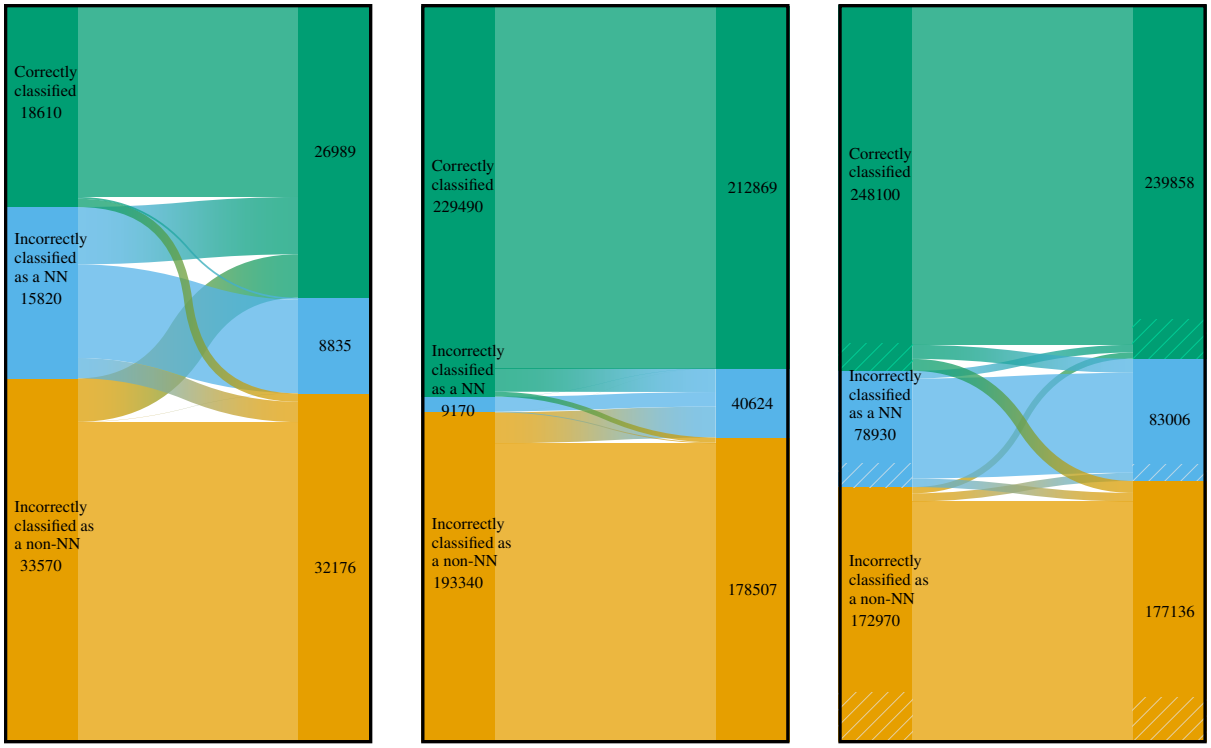
'few' split accuracy was increased by up to 7 points, and on CIFAR-100-LT, by 5 points.

4.3. Comparison with control

Our method is based on the core hypothesis that classifiers can be improved using nearest neighbors. In this section, we directly evaluate 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 the 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 were solely due to extra fine-tuning, we should see similar results. However, as seen in Figure 3, training with 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 information from neighbors to improve classification performance.

4.4. Prediction changes

As shown in Section 1, the cRT model frequently misclassifies 'few' split classes as visually similar 'base' split classes. Using the AlphaNet model with $\rho = 0.5$, we performed the same



(a) Predictions on 'few' split classes, with nearest neighbors selected from 'base' split classes.

(b) Predictions on 'base' split classes, with nearest neighbors selected from 'few' split classes.

(c) All predictions, with nearest neighbors selected from all classes.

Figure 5: Change in sample predictions for AlphaNet applied to cRT features, using $k = 10$ nearest neighbors by Euclidean distance. The bars on the left show the distribution of predictions by the baseline model; and the bars on the right show the distribution for AlphaNet. The counts are aggregated from 10 independent runs of AlphaNet. The "flow" bands from left to right show the changes in individual sample predictions.

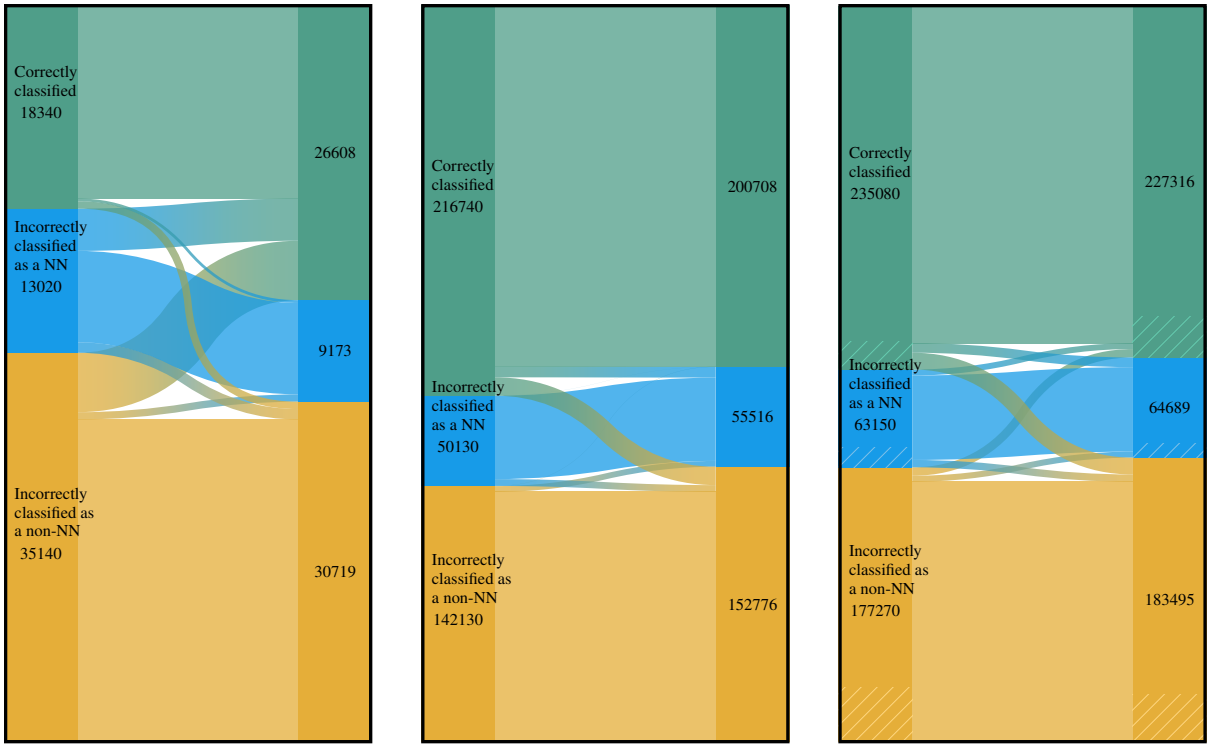
analyses as before. Figure 5a shows the change in sample predictions, A large portion of samples previously misclassified as a nearest neighbor are correctly classified after their classes are updated with AlphaNet. Furthermore, as seen in Figure 4, AlphaNet improvements are strongly correlated to mean nearest neighbor distance. Classes with close neighbors, which had a high likelihood of being misclassified by the baseline model, see the biggest improvement in test accuracy.

4.5. Analysis of AlphaNet predictions

AlphaNet significantly boosts the accuracy of 'few' split classes. However, looking at Table 1 and Table 2, we see that the overall accuracy decreases compared to baseline models, particularly for small values of ρ. It is important to note that the increase in 'few' split accuracy is much larger than the decrease in overall accuracy. As discussed earlier, in many applications it is important to have balanced performance across classes, and AlphaNet succeeds in making accuracies more balanced across splits.

However, we further analyzed the prediction changes for 'base' split samples. Specifically, Figure 5b shows change in predictions for 'base' split samples, with nearest neighbors selected from the 'few' split. We see a small increase in misclassifications as 'few' split classes. This leads to the slight decrease in overall accuracy, which is also evident in Figure 5c where all predictions are shown, and with nearest neighbors from all classes.

The previous analysis was conducted using nearest neighbors identified based on visual similarity. Since this is dependent on the particular model, we conducted an additional analysis to see the behavior of predictions with respect to *semantically similar* categories. For classes in ImageNet-LT, we defined nearest neighbors using distance in the WordNet hierarchy. Specifically, if two classes (e.g., 'Lhasa' and 'Tibetan terrier') share a parent at most four levels higher in WordNet (in this example, 'dog'), we consider them to be nearest neigh-



(a) Predictions on 'few' split classes, with nearest neighbors selected from 'base' split classes.

(b) Predictions on 'base' split classes, with nearest neighbors selected from 'few' split classes.

(c) All predictions, with nearest neighbors selected from all classes.

Figure 6: Change in sample predictions for AlphaNet applied for cRT features, with nearest neighbors identified using WordNet categories. This figure represents the same predictions as Figure 5, but grouped differently.

bors. Figure 6 shows the predictions for AlphaNet with cRT grouped based on these nearest neighbors. As we can see, a large number of predictions which are considered incorrect are among semantically similar categories which can be hard for even humans to distinguish. This suggests that metrics for long-tail classification might need to be re-evaluated for large datasets with many similar classes.

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 [11]. 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 low accuracy for data-poor classes, and 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 greatly improves the accuracy for data-poor classes, and re-balances classification accuracy in a way that 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. We analyzed the predictions of our model, and showed that when considering semantically similar classes together, AlphaNet achieves accuracy on par with baselines, while also improving accuracy for data-poor classes significantly. 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] Athma Narayanan, Yi-Ting Chen, and Srikanth Malla. Semi-supervised learning: Fusion of self-supervised, supervised learning, and multimodal cues for tactical driver behavior detection. *arXiv preprint arXiv:1807.00864*, 2018.

[2] Zhixiong Yang, Junwen Pan, Yanzhan Yang, Xiaozhou Shi, Hong-Yu Zhou, Zhicheng Zhang, and Cheng Bian. Proco: Prototype-aware contrastive learning for long-tailed medical image classification. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2022: 25th International Conference, Singapore, September 18–22, 2022, Proceedings, Part VIII*, pages 173–182. Springer, 2022.

[3] Lu Yang, He Jiang, Qing Song, and Jun Guo. A survey on long-tailed visual recognition. *International Journal of Computer Vision*, 130(7): 1837–1872, 2022.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Tianhao Li, Limin Wang, and Gangshan Wu. Self supervision to distillation for long-tailed visual recognition. *arXiv*, 2021. doi: 10.48550/arxiv.2109.04075.

[10] 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.

[11] 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.

[12] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018.

[13] Laleh Seyyed-Kalantari, Haoran Zhang, Matthew BA McDermott, Irene Y Chen, and Marzyeh Ghassemi. Underdiagnosis bias of artificial intelligence algorithms applied to chest radiographs in under-served patient populations. *Nature medicine*, 27(12):2176–2182, 2021.

[14] Esther Puyol-Antón, Bram Ruijsink, Stefan K Piechnik, Stefan Neubauer, Steffen E Petersen, Reza Razavi, and Andrew P King. Fairness in cardiac mr image analysis: an investigation of bias due to data imbalance in deep learning based segmentation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part III 24*, pages 413–423. Springer, 2021.

[15] 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.

[16] Sebastian Thrun. *Lifelong Learning Algorithms*, chapter 8, pages 181–209. Springer, 1998. doi: 10.1007/978-1-4615-5529-2_8.

[17] 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.

[18] Sinno 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.

[19] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. ISSN 0885-6125. doi: 10.1023/a:1007379606734.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv*, 2016. doi: 10.48550/arxiv.1609.09106.

[25] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *arXiv*, 2018. doi: 10.48550/arxiv.1806.02817.

[26] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *arXiv*, 2017. doi: 10.48550/arxiv.1705.08045.

[27] 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.

[28] Tsendsuren Munkhdalai and Hong Yu. Meta networks. *Proceedings of machine learning research*, 70:2554–2563, 2017.

[29] 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.

[30] 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.

[31] Hyeonwoo 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.

[32] 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.

[33] 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.

[34] 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.

[35] Yusuf Aytar and Andrew Zisserman. Enhancing exemplar svms using part level transfer regularization. *Procedings of the British Machine Vision Conference 2012*, pages 79.1–79.11, 2012. doi: 10.5244/c.26.79.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2017.

[41] 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.

[42] Simon S Du, Jayanth Koushik, Aarti Singh, and Barnabás Póczos. Hypothesis transfer learning via transformation functions. *Advances in neural information processing systems*, 30, 2017.

[43] 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.

[44] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[45] 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.

[46] Zhizhong Li and Derek Hoiem. Learning without forgetting. *arXiv*, 2016. doi: 10.48550/arxiv.1606.09282.

[47] 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.

[48] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 6, 1993.

[49] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv*, 2017. doi: 10.48550/arxiv.1703.05175.

[50] 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.

[51] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. *Advances in neural information processing systems*, 30, 2017.

[52] 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.

[53] 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.

[54] 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.2019.00585.

[55] 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.

[56] 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.

[57] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[58] 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.

[59] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[60] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv*, 2017. doi: 10.48550/arxiv.1711.05101.

[61] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[62] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(03):90–95, 2007.

[63] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. doi: 10.21105/joss.03021. URL `https://doi.org/10.21105/joss.03021`.

Parameter	Value
Optimizer	AdamW [60] with default parameters: $\beta_1 = 0.9$ $\beta_2 = 0.999$ $\epsilon = 0.01$ $\lambda = 0.01$
Initial learning rate	0.001
Learning rate decay	0.1 every 10 epochs
Training epochs	25
Minimum epochs	5
Batch size	64
AlphaNet architecture	3 fully connected layers each with 32 units
Hidden layer activation	Leaky ReLU with negative slope 0.01

Table A1: Hyper-parameters.

Appendix A Implementation details

Experiments were run using the PyTorch [61] library. We used the container implementation provided by NVIDIA GPU cloud (NGC) [A1]. Code to reproduce experimental results is available on GitHub at `github.com/jayanthkoushik/alphanet`, along with instructions to run the code.

Hyper-parameter settings used during training are shown in Table A1. The selection of the best model was controlled by the 'minimum epochs' parameter. After training for at least this many epochs, model weights were saved at the end of each epoch. Finally, the best model was selected based on overall validation accuracy, and used for testing.

All experiments were repeated 10 times from different random initializations, and unless specified otherwise, results (e.g., accuracy) are average values. In tables, the standard deviation is shown in superscript.

Plots were generated with Matplotlib [62] using the Seaborn library [63]. Wherever applicable, error bars show 95% confidence intervals, estimated using 10,000 bootstrap resamples.

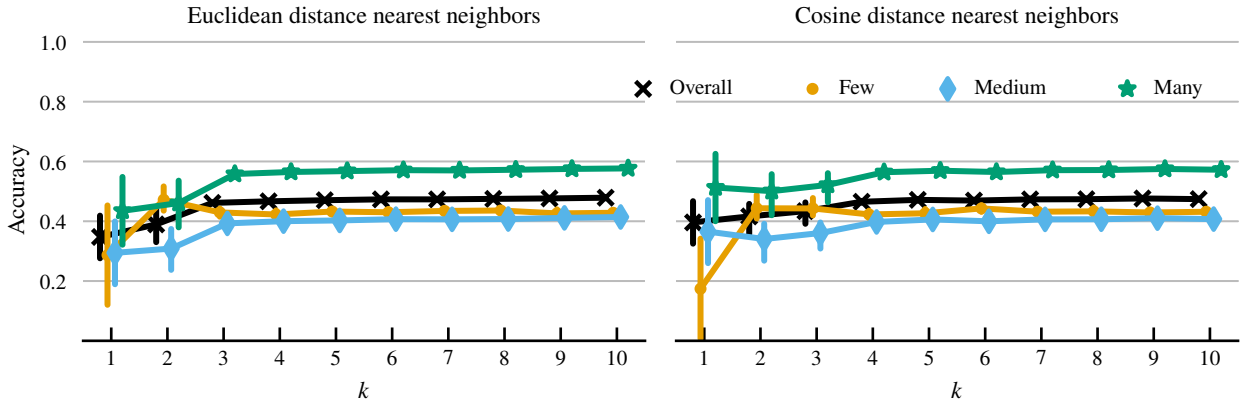
[A1] Version 22.06 from catalog.ngc.nvidia.com/orgs/nvidia/containers/pytorch.

Appendix B Analysis of nearest neighbor selection

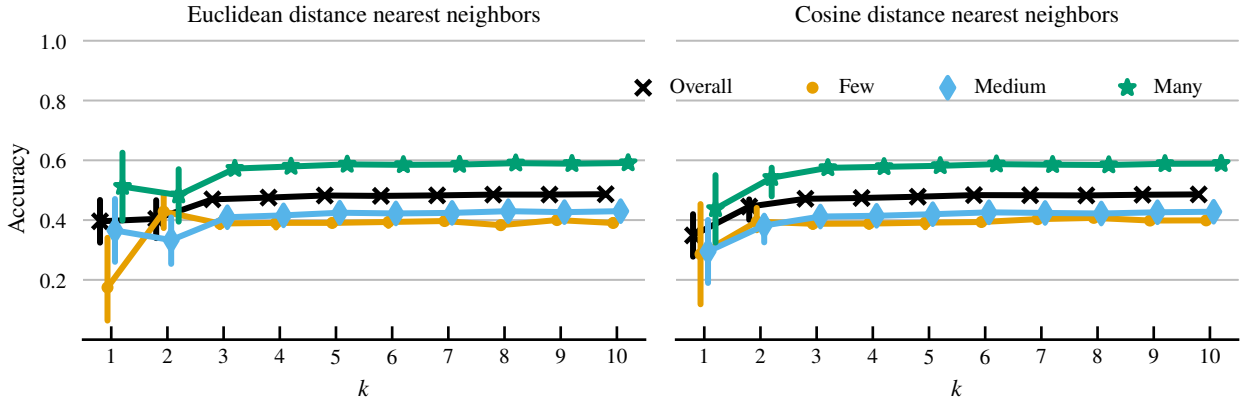
We analyzed the effect of the number of nearest neighbors k, and the distance metric (μ), on the performance of AlphaNet on the ImageNet-LT dataset with the cRT model. We compared two distance metrics: cosine distance ($\mu(z_1, z_2) = 1 - z_1^T z_2 / \|z_1\|_2 \|z_2\|_2$), and Euclidean distance ($\mu(z_1, z_2) = \|z_1 - z_2\|_2$). For each distance metric, we performed 4 sets of experiments, with ρ in $\{0.25, 0.5, 1, 2\}$. For each ρ, we varied k from 2 to 10; all other hyper-parameters were kept the same as described in Section A.

The results are summarized in Figure B1, which shows the per-split accuracies against k for different values of ρ ($\rho = 2$ is omitted from this figure for space – no special behavior was observed for this case). We observe little change in performance beyond $k = 5$, and also observe similar performance for both distance metrics.

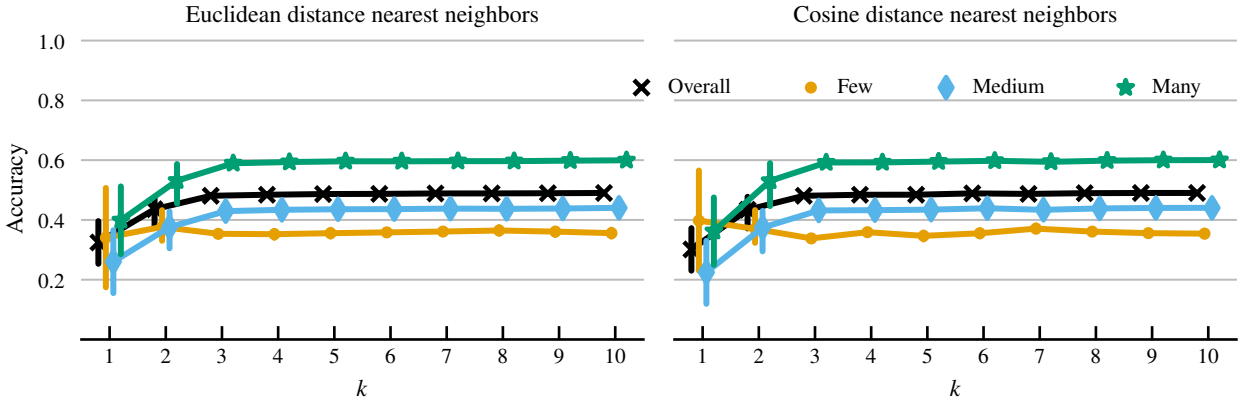
The full set of top-1 and top-5 accuracies is shown in Table B1, B2, B3, B4.



(a) $\rho = 0.25$



(b) $\rho = 0.5$



(c) $\rho = 1$

Figure B1: Per-split accuracies on ImageNet-LT with varying number of nearest neighbors, for AlphaNet with cRT.

Experiment	Few	Med.	Many	Overall
cRT	27.4	46.2	61.8	49.6
α cRT				
ρ = 0.25				
$k = 1$	28.7 $^{29.06}$	29.4 $^{18.36}$	43.5 $^{19.74}$	34.7 $^{12.44}$
$k = 2$	47.2 $^{07.24}$	30.9 $^{11.45}$	46.1 $^{13.08}$	38.9 $^{09.54}$
$k = 3$	43.0 $^{02.40}$	39.3 $^{00.63}$	55.8 $^{00.48}$	46.1 $^{00.37}$
$k = 4$	42.3 $^{02.36}$	40.0 $^{01.00}$	56.5 $^{00.79}$	46.7 $^{00.51}$
$k = 5$	43.2 $^{01.58}$	40.3 $^{00.57}$	56.8 $^{00.48}$	47.0 $^{00.38}$
$k = 6$	43.0 $^{01.81}$	40.7 $^{01.01}$	57.1 $^{00.84}$	47.3 $^{00.56}$
$k = 7$	43.5 $^{01.95}$	40.6 $^{00.81}$	57.0 $^{00.67}$	47.3 $^{00.43}$
$k = 8$	43.6 $^{01.47}$	40.7 $^{00.58}$	57.2 $^{00.50}$	47.5 $^{00.34}$
$k = 9$	42.6 $^{02.26}$	41.1 $^{00.88}$	57.5 $^{00.76}$	47.6 $^{00.43}$
$k = 10$	42.8 $^{01.01}$	41.4 $^{00.58}$	57.7 $^{00.41}$	47.9 $^{00.32}$
ρ = 0.5				
$k = 1$	17.5 $^{26.76}$	36.5 $^{16.92}$	51.2 $^{18.25}$	39.6 $^{11.49}$
$k = 2$	42.8 $^{09.53}$	33.3 $^{12.77}$	48.4 $^{14.22}$	40.4 $^{10.30}$
$k = 3$	38.9 $^{02.06}$	41.0 $^{00.45}$	57.2 $^{00.60}$	46.9 $^{00.40}$
$k = 4$	39.1 $^{02.53}$	41.6 $^{00.90}$	57.9 $^{00.64}$	47.5 $^{00.56}$
$k = 5$	39.1 $^{01.64}$	42.5 $^{00.52}$	58.6 $^{00.39}$	48.2 $^{00.19}$
$k = 6$	39.4 $^{02.05}$	42.2 $^{00.51}$	58.5 $^{00.44}$	48.1 $^{00.21}$
$k = 7$	39.7 $^{01.21}$	42.4 $^{00.51}$	58.6 $^{00.44}$	48.3 $^{00.29}$
$k = 8$	38.3 $^{01.54}$	43.0 $^{00.55}$	59.0 $^{00.39}$	48.5 $^{00.26}$
$k = 9$	40.0 $^{00.99}$	42.7 $^{00.39}$	58.9 $^{00.33}$	48.5 $^{00.21}$
$k = 10$	39.0 $^{01.64}$	43.0 $^{00.51}$	59.1 $^{00.41}$	48.7 $^{00.21}$
ρ = 1				
$k = 1$	34.1 $^{28.61}$	26.0 $^{18.09}$	39.9 $^{19.51}$	32.5 $^{12.28}$
$k = 2$	37.7 $^{08.93}$	37.6 $^{10.62}$	53.1 $^{11.56}$	43.6 $^{08.36}$
$k = 3$	35.4 $^{01.48}$	42.9 $^{00.50}$	59.0 $^{00.46}$	48.1 $^{00.25}$
$k = 4$	35.2 $^{02.02}$	43.4 $^{00.53}$	59.4 $^{00.46}$	48.4 $^{00.30}$
$k = 5$	35.6 $^{01.93}$	43.6 $^{00.59}$	59.6 $^{00.41}$	48.7 $^{00.23}$
$k = 6$	35.8 $^{01.23}$	43.6 $^{00.39}$	59.6 $^{00.32}$	48.7 $^{00.24}$
$k = 7$	36.1 $^{01.30}$	43.8 $^{00.45}$	59.7 $^{00.30}$	48.9 $^{00.17}$
$k = 8$	36.5 $^{01.90}$	43.7 $^{00.43}$	59.7 $^{00.36}$	48.9 $^{00.16}$
$k = 9$	36.1 $^{01.80}$	43.8 $^{00.45}$	59.8 $^{00.40}$	48.9 $^{00.15}$
$k = 10$	35.6 $^{02.01}$	44.0 $^{00.56}$	60.0 $^{00.48}$	49.0 $^{00.22}$
ρ = 2				
$k = 1$	23.0 $^{28.61}$	33.0 $^{18.09}$	47.4 $^{19.51}$	37.2 $^{12.28}$
$k = 2$	29.9 $^{01.82}$	43.8 $^{00.45}$	59.6 $^{00.46}$	48.0 $^{00.16}$
$k = 3$	30.9 $^{01.86}$	44.2 $^{00.35}$	60.1 $^{00.29}$	48.5 $^{00.16}$
$k = 4$	31.2 $^{02.05}$	44.5 $^{00.40}$	60.3 $^{00.39}$	48.8 $^{00.21}$
$k = 5$	33.1 $^{01.45}$	44.3 $^{00.31}$	60.2 $^{00.32}$	48.9 $^{00.15}$
$k = 6$	32.0 $^{01.68}$	44.7 $^{00.34}$	60.5 $^{00.21}$	49.1 $^{00.07}$
$k = 7$	32.3 $^{01.38}$	44.8 $^{00.31}$	60.6 $^{00.25}$	49.2 $^{00.14}$
$k = 8$	32.2 $^{01.70}$	44.8 $^{00.38}$	60.6 $^{00.28}$	49.2 $^{00.11}$
$k = 9$	32.4 $^{01.22}$	44.8 $^{00.29}$	60.6 $^{00.18}$	49.2 $^{00.07}$
$k = 10$	32.4 $^{01.85}$	44.9 $^{00.47}$	60.7 $^{00.37}$	49.3 $^{00.14}$

Table B1: Top-1 accuracy for AlphaNet using varying number of nearest neighbors (k) based on Euclidean distance, with cRT baseline on ImageNet-LT.

Experiment	Few	Med.	Many	Overall
cRT	57.3	73.4	81.8	74.4
α cRT				
ρ = 0.25				
$k = 1$	42.5 $^{41.12}$	63.7 $^{11.28}$	74.8 $^{08.11}$	65.1 $^{02.93}$
$k = 2$	71.4 $^{07.44}$	65.5 $^{07.09}$	76.1 $^{05.04}$	70.4 $^{04.33}$
$k = 3$	67.3 $^{01.26}$	70.4 $^{00.32}$	79.6 $^{00.21}$	73.5 $^{00.22}$
$k = 4$	67.8 $^{01.39}$	70.5 $^{00.35}$	79.7 $^{00.26}$	73.7 $^{00.14}$
$k = 5$	68.2 $^{00.89}$	70.7 $^{00.39}$	79.8 $^{00.24}$	73.9 $^{00.23}$
$k = 6$	68.1 $^{01.28}$	70.8 $^{00.46}$	79.9 $^{00.32}$	73.9 $^{00.17}$
$k = 7$	68.9 $^{00.86}$	70.7 $^{00.37}$	79.8 $^{00.26}$	73.9 $^{00.19}$
$k = 8$	68.7 $^{00.71}$	70.8 $^{00.25}$	79.8 $^{00.18}$	74.0 $^{00.13}$
$k = 9$	68.4 $^{01.27}$	70.9 $^{00.34}$	79.9 $^{00.24}$	74.0 $^{00.13}$
$k = 10$	68.5 $^{00.69}$	71.0 $^{00.33}$	79.9 $^{00.21}$	74.1 $^{00.16}$
ρ = 0.5				
$k = 1$	26.6 $^{37.90}$	68.0 $^{10.37}$	77.9 $^{07.45}$	66.2 $^{02.68}$
$k = 2$	68.1 $^{09.42}$	66.6 $^{07.36}$	76.9 $^{05.22}$	70.8 $^{04.26}$
$k = 3$	64.7 $^{01.62}$	70.9 $^{00.31}$	80.0 $^{00.26}$	73.6 $^{00.18}$
$k = 4$	65.0 $^{02.06}$	71.2 $^{00.51}$	80.1 $^{00.30}$	73.8 $^{00.32}$
$k = 5$	65.3 $^{00.78}$	71.6 $^{00.17}$	80.4 $^{00.14}$	74.1 $^{00.08}$
$k = 6$	65.9 $^{01.05}$	71.3 $^{00.21}$	80.3 $^{00.18}$	74.0 $^{00.12}$
$k = 7$	66.2 $^{00.93}$	71.5 $^{00.34}$	80.3 $^{00.24}$	74.1 $^{00.16}$
$k = 8$	65.4 $^{00.94}$	71.6 $^{00.30}$	80.4 $^{00.20}$	74.1 $^{00.13}$
$k = 9$	66.3 $^{00.71}$	71.6 $^{00.17}$	80.4 $^{00.13}$	74.2 $^{00.09}$
$k = 10$	66.0 $^{00.90}$	71.6 $^{00.25}$	80.5 $^{00.16}$	74.3 $^{00.10}$
ρ = 1				
$k = 1$	50.1 $^{40.52}$	61.6 $^{11.08}$	73.3 $^{07.97}$	64.5 $^{02.87}$
$k = 2$	63.4 $^{08.24}$	69.0 $^{05.95}$	78.6 $^{04.20}$	71.9 $^{03.37}$
$k = 3$	61.9 $^{01.32}$	71.8 $^{00.32}$	80.6 $^{00.26}$	73.8 $^{00.15}$
$k = 4$	62.0 $^{01.36}$	72.0 $^{00.23}$	80.7 $^{00.17}$	74.0 $^{00.15}$
$k = 5$	62.7 $^{01.25}$	72.0 $^{00.27}$	80.8 $^{00.20}$	74.1 $^{00.11}$
$k = 6$	63.2 $^{00.83}$	72.0 $^{00.22}$	80.8 $^{00.18}$	74.2 $^{00.13}$
$k = 7$	63.5 $^{01.07}$	72.0 $^{00.25}$	80.7 $^{00.15}$	74.2 $^{00.09}$
$k = 8$	64.2 $^{01.06}$	72.0 $^{00.23}$	80.7 $^{00.15}$	74.3 $^{00.14}$
$k = 9$	63.7 $^{01.26}$	72.1 $^{00.27}$	80.8 $^{00.17}$	74.3 $^{00.07}$
$k = 10$	63.6 $^{01.51}$	72.1 $^{00.34}$	80.8 $^{00.21}$	74.3 $^{00.08}$
ρ = 2				
$k = 1$	34.4 $^{40.52}$	65.9 $^{11.08}$	76.4 $^{07.97}$	65.6 $^{02.87}$
$k = 2$	56.8 $^{01.38}$	72.2 $^{00.30}$	80.9 $^{00.23}$	73.5 $^{00.16}$
$k = 3$	57.8 $^{01.48}$	72.4 $^{00.19}$	81.1 $^{00.20}$	73.8 $^{00.17}$
$k = 4$	58.8 $^{01.32}$	72.4 $^{00.25}$	81.1 $^{00.19}$	73.9 $^{00.18}$
$k = 5$	60.9 $^{01.15}$	72.3 $^{00.27}$	80.9 $^{00.18}$	74.1 $^{00.12}$
$k = 6$	60.0 $^{01.31}$	72.6 $^{00.17}$	81.2 $^{00.13}$	74.2 $^{00.09}$
$k = 7$	60.5 $^{00.97}$	72.6 $^{00.19}$	81.1 $^{00.15}$	74.2 $^{00.09}$
$k = 8$	60.6 $^{01.15}$	72.6 $^{00.21}$	81.2 $^{00.20}$	74.3 $^{00.06}$
$k = 9$	60.9 $^{00.99}$	72.6 $^{00.19}$	81.2 $^{00.13}$	74.3 $^{00.04}$
$k = 10$	60.9 $^{01.34}$	72.6 $^{00.23}$	81.2 $^{00.16}$	74.3 $^{00.08}$

Table B2: Top-5 accuracy for AlphaNet using varying number of nearest neighbors (k) based on Euclidean distance, with cRT baseline on ImageNet-LT.

Experiment	Few	Med.	Many	Overall
cRT	27.4	46.2	61.8	49.6
α cRT				
ρ = 0.25				
k = 1	17.4 $^{26.99}$	36.5 $^{16.97}$	51.3 $^{18.15}$	39.6 $^{11.45}$
k = 2	44.3 $^{06.71}$	34.1 $^{10.34}$	50.1 $^{11.57}$	41.6 $^{08.50}$
k = 3	44.3 $^{05.19}$	36.1 $^{08.08}$	52.2 $^{09.06}$	43.4 $^{06.69}$
k = 4	42.3 $^{02.12}$	39.8 $^{00.74}$	56.4 $^{00.70}$	46.5 $^{00.52}$
k = 5	42.7 $^{02.81}$	40.6 $^{01.04}$	56.9 $^{00.93}$	47.2 $^{00.52}$
k = 6	44.3 $^{01.89}$	40.0 $^{00.98}$	56.5 $^{00.81}$	46.9 $^{00.54}$
k = 7	43.2 $^{01.60}$	40.6 $^{00.97}$	57.1 $^{00.77}$	47.3 $^{00.55}$
k = 8	43.3 $^{02.31}$	40.7 $^{01.01}$	57.1 $^{00.86}$	47.4 $^{00.52}$
k = 9	42.9 $^{01.23}$	41.0 $^{00.72}$	57.5 $^{00.62}$	47.6 $^{00.45}$
k = 10	43.2 $^{01.54}$	40.7 $^{00.59}$	57.2 $^{00.44}$	47.4 $^{00.32}$
ρ = 0.5				
k = 1	28.6 $^{29.44}$	29.5 $^{18.51}$	43.8 $^{19.80}$	34.9 $^{12.49}$
k = 2	39.5 $^{06.79}$	38.2 $^{08.65}$	54.2 $^{09.53}$	44.5 $^{06.91}$
k = 3	38.8 $^{01.78}$	41.2 $^{00.82}$	57.5 $^{00.78}$	47.1 $^{00.49}$
k = 4	38.8 $^{01.88}$	41.4 $^{00.38}$	57.8 $^{00.44}$	47.4 $^{00.30}$
k = 5	39.1 $^{02.47}$	42.0 $^{00.72}$	58.2 $^{00.62}$	47.8 $^{00.33}$
k = 6	39.4 $^{01.45}$	42.6 $^{00.67}$	58.7 $^{00.44}$	48.4 $^{00.31}$
k = 7	40.3 $^{01.19}$	42.4 $^{00.47}$	58.5 $^{00.43}$	48.3 $^{00.28}$
k = 8	40.7 $^{01.35}$	42.2 $^{00.60}$	58.4 $^{00.42}$	48.3 $^{00.34}$
k = 9	39.8 $^{01.08}$	42.6 $^{00.40}$	58.8 $^{00.31}$	48.5 $^{00.22}$
k = 10	39.9 $^{01.17}$	42.8 $^{00.49}$	58.9 $^{00.31}$	48.6 $^{00.24}$
ρ = 1				
k = 1	39.8 $^{26.99}$	22.5 $^{16.97}$	36.2 $^{18.15}$	30.1 $^{11.45}$
k = 2	37.0 $^{09.44}$	37.4 $^{11.58}$	53.1 $^{12.34}$	43.4 $^{09.03}$
k = 3	33.8 $^{01.49}$	43.2 $^{00.52}$	59.2 $^{00.48}$	48.1 $^{00.25}$
k = 4	35.9 $^{01.10}$	43.3 $^{00.38}$	59.2 $^{00.24}$	48.4 $^{00.27}$
k = 5	34.7 $^{01.99}$	43.4 $^{00.52}$	59.5 $^{00.39}$	48.4 $^{00.22}$
k = 6	35.5 $^{01.79}$	43.9 $^{00.41}$	59.8 $^{00.42}$	48.9 $^{00.14}$
k = 7	37.1 $^{01.72}$	43.4 $^{00.48}$	59.4 $^{00.43}$	48.7 $^{00.22}$
k = 8	36.1 $^{01.47}$	43.8 $^{00.40}$	59.8 $^{00.31}$	48.9 $^{00.15}$
k = 9	35.6 $^{01.37}$	44.0 $^{00.32}$	60.0 $^{00.30}$	49.0 $^{00.14}$
k = 10	35.4 $^{01.68}$	44.0 $^{00.51}$	60.0 $^{00.33}$	49.0 $^{00.15}$
ρ = 2				
k = 1	23.0 $^{28.85}$	33.0 $^{18.14}$	47.5 $^{19.40}$	37.2 $^{12.24}$
k = 2	29.1 $^{01.64}$	44.0 $^{00.36}$	59.9 $^{00.25}$	48.1 $^{00.15}$
k = 3	30.9 $^{01.89}$	44.1 $^{00.59}$	60.0 $^{00.45}$	48.4 $^{00.23}$
k = 4	31.6 $^{02.13}$	44.3 $^{00.49}$	60.3 $^{00.38}$	48.7 $^{00.20}$
k = 5	32.5 $^{02.40}$	44.5 $^{00.57}$	60.3 $^{00.41}$	48.9 $^{00.17}$
k = 6	30.8 $^{01.76}$	44.9 $^{00.35}$	60.7 $^{00.31}$	49.0 $^{00.17}$
k = 7	32.4 $^{01.85}$	44.8 $^{00.34}$	60.6 $^{00.30}$	49.2 $^{00.12}$
k = 8	31.5 $^{01.52}$	45.0 $^{00.27}$	60.7 $^{00.21}$	49.2 $^{00.13}$
k = 9	32.9 $^{01.41}$	44.8 $^{00.25}$	60.6 $^{00.22}$	49.3 $^{00.10}$
k = 10	31.9 $^{02.16}$	45.0 $^{00.42}$	60.7 $^{00.33}$	49.3 $^{00.08}$

Table B3: Top-1 accuracy for AlphaNet using varying number of nearest neighbors (k) based on cosine distance, with cRT baseline on ImageNet-LT.

Experiment	Few	Med.	Many	Overall
cRT	57.3	73.4	81.8	74.4
α cRT				
ρ = 0.25				
k = 1	26.3 $^{38.14}$	68.0 $^{10.37}$	78.0 $^{07.46}$	66.2 $^{02.65}$
k = 2	68.8 $^{07.12}$	67.5 $^{06.28}$	77.6 $^{04.46}$	71.6 $^{03.76}$
k = 3	69.0 $^{05.03}$	68.7 $^{04.72}$	78.4 $^{03.21}$	72.5 $^{02.84}$
k = 4	67.4 $^{01.50}$	70.5 $^{00.29}$	79.8 $^{00.20}$	73.7 $^{00.20}$
k = 5	68.0 $^{01.83}$	70.8 $^{00.39}$	79.9 $^{00.26}$	73.9 $^{00.14}$
k = 6	69.1 $^{01.09}$	70.6 $^{00.37}$	79.7 $^{00.24}$	73.9 $^{00.15}$
k = 7	68.7 $^{01.30}$	70.7 $^{00.43}$	79.8 $^{00.31}$	74.0 $^{00.18}$
k = 8	68.9 $^{01.45}$	70.7 $^{00.32}$	79.8 $^{00.25}$	73.9 $^{00.09}$
k = 9	68.7 $^{00.82}$	70.8 $^{00.35}$	79.9 $^{00.27}$	74.0 $^{00.18}$
k = 10	69.0 $^{00.69}$	70.6 $^{00.30}$	79.7 $^{00.19}$	73.9 $^{00.21}$
ρ = 0.5				
k = 1	42.1 $^{41.61}$	63.7 $^{11.32}$	74.9 $^{08.13}$	65.1 $^{02.89}$
k = 2	64.5 $^{06.59}$	69.5 $^{05.12}$	79.0 $^{03.60}$	72.5 $^{02.96}$
k = 3	64.2 $^{01.95}$	71.2 $^{00.48}$	80.1 $^{00.33}$	73.7 $^{00.17}$
k = 4	65.5 $^{01.10}$	71.1 $^{00.24}$	80.1 $^{00.15}$	73.8 $^{00.18}$
k = 5	65.5 $^{01.55}$	71.3 $^{00.31}$	80.2 $^{00.21}$	73.9 $^{00.13}$
k = 6	65.4 $^{00.95}$	71.7 $^{00.26}$	80.5 $^{00.17}$	74.2 $^{00.11}$
k = 7	66.4 $^{00.82}$	71.5 $^{00.25}$	80.3 $^{00.16}$	74.2 $^{00.13}$
k = 8	66.8 $^{00.88}$	71.4 $^{00.28}$	80.2 $^{00.20}$	74.2 $^{00.17}$
k = 9	66.3 $^{00.93}$	71.6 $^{00.28}$	80.4 $^{00.19}$	74.3 $^{00.13}$
k = 10	66.4 $^{00.68}$	71.6 $^{00.19}$	80.4 $^{00.14}$	74.3 $^{00.12}$
ρ = 1				
k = 1	57.9 $^{38.14}$	59.4 $^{10.37}$	71.8 $^{07.46}$	64.0 $^{02.65}$
k = 2	63.2 $^{09.28}$	68.9 $^{06.35}$	78.6 $^{04.43}$	71.9 $^{03.50}$
k = 3	60.6 $^{01.44}$	71.9 $^{00.34}$	80.6 $^{00.26}$	73.7 $^{00.12}$
k = 4	62.2 $^{01.02}$	72.0 $^{00.32}$	80.7 $^{00.15}$	74.0 $^{00.17}$
k = 5	62.4 $^{01.48}$	71.8 $^{00.25}$	80.7 $^{00.16}$	74.0 $^{00.12}$
k = 6	62.8 $^{01.10}$	72.2 $^{00.20}$	80.8 $^{00.14}$	74.2 $^{00.09}$
k = 7	64.1 $^{01.35}$	71.9 $^{00.22}$	80.6 $^{00.15}$	74.2 $^{00.11}$
k = 8	63.5 $^{01.03}$	72.1 $^{00.20}$	80.8 $^{00.14}$	74.3 $^{00.09}$
k = 9	63.4 $^{01.06}$	72.1 $^{00.23}$	80.8 $^{00.12}$	74.3 $^{00.12}$
k = 10	63.3 $^{01.22}$	72.2 $^{00.26}$	80.9 $^{00.16}$	74.3 $^{00.07}$
ρ = 2				
k = 1	34.2 $^{40.77}$	65.9 $^{11.09}$	76.4 $^{07.97}$	65.6 $^{02.83}$
k = 2	55.7 $^{01.40}$	72.4 $^{00.13}$	81.0 $^{00.09}$	73.4 $^{00.18}$
k = 3	58.1 $^{01.83}$	72.3 $^{00.36}$	81.0 $^{00.23}$	73.7 $^{00.12}$
k = 4	59.4 $^{01.52}$	72.4 $^{00.26}$	81.0 $^{00.19}$	73.9 $^{00.17}$
k = 5	60.2 $^{01.70}$	72.5 $^{00.25}$	81.1 $^{00.19}$	74.2 $^{00.12}$
k = 6	59.3 $^{01.52}$	72.6 $^{00.31}$	81.2 $^{00.16}$	74.1 $^{00.15}$
k = 7	60.5 $^{01.50}$	72.6 $^{00.18}$	81.2 $^{00.15}$	74.3 $^{00.10}$
k = 8	60.0 $^{01.24}$	72.6 $^{00.19}$	81.2 $^{00.11}$	74.2 $^{00.16}$
k = 9	61.0 $^{00.90}$	72.6 $^{00.14}$	81.1 $^{00.10}$	74.3 $^{00.09}$
k = 10	60.5 $^{01.52}$	72.7 $^{00.28}$	81.2 $^{00.16}$	74.3 $^{00.04}$

Table B4: Top-5 accuracy for AlphaNet using varying number of nearest neighbors (k) based on cosine distance, with cRT baseline on ImageNet-LT.