

# SimpleShot: Revisiting Nearest-Neighbor Classification for Few-Shot Learning

Yan Wang  
Cornell University  
yw763@cornell.edu

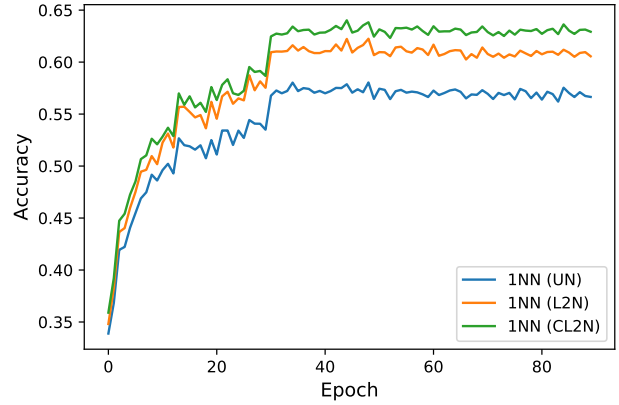
Wei-Lun Chao  
Ohio State University  
chao.209@osu.edu

Kilian Q. Weinberger  
Cornell University  
kgw4@cornell.edu

Laurens van der Maaten  
Facebook AI Research  
lvdmaaten@fb.com

## Abstract

*Few-shot learners aim to recognize new object classes based on a small number of labeled training examples. To prevent overfitting, state-of-the-art few-shot learners use meta-learning on convolutional-network features and perform classification using a nearest-neighbor classifier. This paper studies the accuracy of nearest-neighbor baselines without meta-learning. Surprisingly, we find simple feature transformations suffice to obtain competitive few-shot learning accuracies. For example, we find that a nearest-neighbor classifier used in combination with mean-subtraction and L2-normalization outperforms prior results in three out of five settings on the miniImageNet dataset.*



**Figure 1: Feature transformations matter in few-shot learning using nearest neighbors.** We train a DenseNet on *miniImageNet* and use the learned features to perform few-shot learning using a nearest-neighbor classifier with Euclidean distance. We measure the one-shot five-way accuracy on 10,000 tasks sampled from the validation classes during training. We compare un-normalized (UN), L2-normalized (L2N), and centered L2-normalized (CL2N) features. CL2N features outperform UN features, highlighting the importance of feature transformations in few-shot learning.

## 1. Introduction

The human visual system has an ability to recognize new visual classes (for instance, greebles [7]) based on a few examples that is, currently, unmatched by computer vision. The development of computer-vision systems that can perform such *few-shot learning* [3, 30, 34] is important, *e.g.*, for developing systems that can recognize the millions of natural or man-made classes that appear in the world [12].

Few-shot learning is generally studied in a learning setting in which the visual-recognition system is first trained to recognize a collection of *base classes* from a large number of training examples. Subsequently, the system receives a small number of training examples (so-called “*shots*”) for a few novel visual classes that it needs to recognize thereafter. In order to be robust to overfitting, a successful few-shot learning model must efficiently re-use what it learned from training on the base classes for the novel classes.

Many current few-shot learners extract image features using a convolutional network, and use a combination of meta-learning and nearest-neighbor classification to perform the recognition [24, 34, 30, 31, 36]. Prior studies suggest that using meta-learning outperforms “vanilla” nearest neighbor classification [26, 30].

This study challenges the status quo by demonstrating that nearest-neighbor classifiers can achieve state-of-

the-art performance on popular few-shot learning benchmarks without meta-learning. Specifically, we find that applying simple feature transformations on the features before nearest-neighbor classification leads to very competitive few-shot learning results. For example, we find that a nearest-neighbor classifier that uses DenseNet features [14] to which mean subtraction and L2-normalization are applied outperforms a long list [1, 4, 5, 6, 8, 9, 10, 15, 17, 21, 22, 23, 24, 26, 29, 25, 30, 31, 32, 34, 37] of recent, arguably more complex few-shot learning approaches on the popular *miniImageNet* [34] and *tieredImageNet* [27] benchmarks (see Table 1 and 2). These observations generalize to other convolutional network architectures [11, 13, 38]. We refer to our few-shot learner as SimpleShot. We hope to re-establish nearest-neighbor classification as an obvious but competitive baseline for few-shot learning.

## 2. Nearest Neighbors for Few-Shot Learning

Denoting an image by  $\mathbf{I}$ , we assume we are given a training set,  $\mathcal{D}_{\text{base}} = \{(\mathbf{I}_1, y_1), \dots, (\mathbf{I}_N, y_N)\}$ , that contains  $N$  labeled images from  $A$  base classes; that is,  $y_n \in \{1, \dots, A\}$ . Furthermore, we assume we are given a support set  $\mathcal{D}_{\text{support}}$  of labeled images from  $C$  novel classes, where each novel class has  $K$  examples. The goal of few-shot learning is to construct a model that accurately recognizes the  $C$  novel classes. This learning setting is referred to as the  $K$ -shot  $C$ -way setting.

We study a **few-shot learner** based on **nearest-neighbor classification**, called **SimpleShot**. The nearest-neighbor classifier operates on **features  $\mathbf{x} \in \mathbb{R}^D$**  that were extracted from **image  $\mathbf{I}$**  using a **convolutional network  $f_\theta(\mathbf{I})$  with parameters  $\theta$** . The feature-producing convolutional network,  $f_\theta(\mathbf{I})$ , is trained to minimize the loss of a linear classifier (with  $\mathbf{W} \in \mathbb{R}^{D \times A}$  in the last network layer) on  $\mathcal{D}_{\text{base}}$ :

$$\arg \min_{\theta, \mathbf{W}} \sum_{(\mathbf{I}, y) \in \mathcal{D}_{\text{base}}} \ell(\mathbf{W}^\top f_\theta(\mathbf{I}), y),$$

where the loss function  $\ell$  is selected to be the cross-entropy loss. The convolutional network and the linear classifier are trained jointly using stochastic gradient descent.

**Nearest Neighbor Rule.** Once the feature extraction network,  $f_\theta$ , is trained on the base classes, we access images exclusively in feature space and consider all subsequent images as readily provided in feature space. For simplicity of notation, we denote  $\mathbf{x} = f_\theta(\mathbf{I})$  as an image in feature space. In this space we perform nearest-neighbor classification using some distance measure,  $d(\mathbf{x}, \mathbf{x}') \in \mathbb{R}_0^+$ . We first consider the **one-shot** setting, that is, the setting in which  $\mathcal{D}_{\text{support}}$  contains only  $K = 1$  labeled example for each of the  $C$  classes:  $\mathcal{D}_{\text{support}} = \{(\hat{\mathbf{x}}_1, 1), \dots, (\hat{\mathbf{x}}_C, C)\}$ , where we use the notation  $\hat{\mathbf{x}}$  to distinguish **images in the novel  $C$  classes** from images  $\mathbf{x}$  in  $\mathcal{D}_{\text{base}}$ . The nearest-neighbor rule assigns the label of the most similar support image (in feature space) to a test image  $\hat{\mathbf{x}}$ :

$$y(\hat{\mathbf{x}}) = \arg \min_{c \in \{1, \dots, C\}} d(\hat{\mathbf{x}}, \hat{\mathbf{x}}_c). \quad (1)$$

In **multi-shot** settings, we use a nearest-centroid approach. Specifically, we compute the averaged feature vector (centroid) for each class in  $\mathcal{D}_{\text{support}}$  and treat each of the centroids as a one-shot example for the corresponding class. We then apply Equation 1 on the centroids.

### 2.1. Feature Transformations

In this study, we use the Euclidean distance,  $d(\hat{\mathbf{x}}, \hat{\mathbf{x}}') = \|\hat{\mathbf{x}} - \hat{\mathbf{x}}'\|_2$ , as the distance measure for nearest-neighbors classification. We only consider two feature transformations that are well-established and may be considered trivial but, empirically, we find that they can have a positive effect on the accuracy of the SimpleShot few-shot learner.

**Centering.** We compute the **mean feature vector on the base classes**,  $\bar{\mathbf{x}} = \frac{1}{|\mathcal{D}_{\text{base}}|} \sum_{\mathbf{x} \in \mathcal{D}_{\text{base}}} \mathbf{x}$ , and subtract it from a feature vector  $\hat{\mathbf{x}}$  to normalize it:  $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} - \bar{\mathbf{x}}$ . Centering (or mean subtraction) in itself does not alter Euclidean distances between feature vectors, but can become effective in combination with L2-normalization.

**L2-normalization (L2N).** Given a feature vector  $\hat{\mathbf{x}}$ , we normalize it to have unit  $\ell_2$  norm:  $\hat{\mathbf{x}} \leftarrow \frac{\hat{\mathbf{x}}}{\|\hat{\mathbf{x}}\|_2}$ .

## 3. Experiments

Following prior work, we measure the efficacy of feature transformations in nearest-neighbor classifiers for few-shot learning in a series of image-recognition experiments.<sup>1</sup>

### 3.1. Experimental Setup

**Datasets.** We experiment on three image datasets.

The **miniImageNet dataset** [34] is a subset of ImageNet [28] that is commonly used to study few-shot learning. The dataset contains 100 classes and has a total of 600 examples per class. Following [26] and subsequent work, we split the dataset to have 64 base classes, 16 validation classes, and 20 novel classes. Following [34] and subsequent studies, we resize the images to  $84 \times 84$  pixels via rescaling and center cropping.

We also perform experiments on the **tieredImageNet dataset** [27], which is also constructed from ImageNet but contains 608 classes. The dataset is split into 351, 97, and 160 classes for base, validation, and novel classes, respectively. The class split is performed using WordNet [20] to ensure that all the base classes are semantically unrelated to the novel classes. Again, we resize images to  $84 \times 84$  pixels.

Following [24], we also perform experiments on the **CIFAR-100** [16] dataset, which contains 100 image classes. Each of the classes in the dataset has 600 images of size  $32 \times 32$  pixels. We follow [24] and split the classes into 60 base, 20 validation, and 20 novel classes.

**Evaluation protocol.** Following [29], we measure the accuracy of SimpleShot and the other few-shot learners by drawing 10,000  $K$ -shot  $C$ -way tasks from the novel classes: each task has  $C$  novel classes and  $K$  labeled (support) images and 15 test (query) images per class. Following prior work, we focus on one-shot and five-shot, five-way tasks.

We average observed accuracies over all test images and over all the tasks, and report the resulting average accuracy and 95% confidence interval.

**Model and implementation details.** We evaluate our methods using five different convolutional-network architectures as the basis for the feature-generating function  $f_\theta(\mathbf{I})$ . We study five different network architectures:

<sup>1</sup>Code at [https://github.com/mileyan/simple\\_shot](https://github.com/mileyan/simple_shot).

- **Four-layer convolutional networks (Conv-4):** We follow [30, 34] to implement this baseline model.
- **Wide residual networks (WRN-28-10) [38]:** We follow [29] and use the architecture with 28 convolutional layers and a widening factor of 10.
- **Dense convolutional networks (DenseNet-121) [14]:** We use the standard 121-layer architecture but remove the first two down-sampling layers (*i.e.*, we set their stride to 1) and change the first convolutional layer to use a kernel of size  $3 \times 3$  (rather than  $7 \times 7$ ) pixels.
- **Residual networks (ResNet-10/18) [11]:** We use the standard 18-layer architecture but we remove the first two down-sampling layers and we change the first convolutional layer to use a kernel of size  $3 \times 3$  (rather than  $7 \times 7$ ) pixels. Our ResNet-10 contains 4 residual blocks; the ResNet-18 contains 8 blocks.
- **MobileNet [13]:** We use the standard architecture for ImageNet [28] but, again, we remove the first two down-sampling layers from the network.

We train all networks for 90 epochs from scratch using stochastic gradient descent to minimize the cross-entropy loss of  $A$ -way classification ( $A$  is the number of base classes). We perform the data augmentation proposed in [11]. We set the initial learning rate to 0.1 and use a batch size of 256 images. On *miniImageNet*, We shrink the learning rate by 10 at 45 and 66 epoch respectively. On *tieredImageNet*, we divide the learning rate by 10 after every 30 epochs. We perform early stopping according to the one-shot five-way accuracy (measured using SimpleShot (L2N)) on the validation classes.

**Feature transformations.** We evaluate the effectiveness of three feature transformations in our experiments:

- **UN:** Unnormalized features.
- **L2N:** L2-normalized features.
- **CL2N:** Centered and then L2-normalized features.

These transforms are followed by nearest-neighbor classification using the Euclidean distance measure.

**Comparison.** We compare our baselines to a range of state-of-the-art few-shot learners [1, 4, 5, 6, 8, 10, 15, 17, 21, 22, 23, 24, 26, 29, 25, 30, 31, 32, 34, 36]. We do not compare to approaches that were developed for semi-supervised and transductive learning settings, as such approaches use the statistics of query examples or statistics across the few-shot tasks. We note that the network architectures used in prior studies may have slight variations; we have tried our best to eliminate the effect of such variations on our observations as much as possible.<sup>2</sup>

<sup>2</sup>For example, we report results for ResNet-10 models because it is the shallowest ResNet architecture used in prior work on few-shot learning.

## 3.2. Results

Table 1, 2, and 3 present our results on *miniImageNet*, *tieredImageNet*, and CIFAR-100, respectively. In line with prior work, we observe that nearest-neighbor classifiers using “vanilla” Euclidean distance (UN) do not perform very well. However, simply applying L2-normalization (L2N) consistently leads to accuracy gains of at least 3% on these datasets. Subtracting the mean before L2-normalization (CL2N) leads to another improvement of 1–3%.

Our SimpleShot nearest-neighbor / nearest-centroid classifiers achieve accuracies that are comparable with or better than the state-of-the-art. For example, on the *miniImageNet* dataset, our simple methods obtain the highest one-shot and five-shot accuracies for three of five network architectures.

We perform a simple experiment measuring the effectiveness of feature transformations at various stages of convolutional-network training. We train a DenseNet on *miniImageNet* for 90 epochs, and measure the one-shot five-way accuracy on 10,000 tasks sampled from the validation classes after each epoch. The results of this experiment are shown in Figure 1: they show that nearest-neighbor classifiers using CL2N feature transformation consistently outperform their UN and L2N counterparts. This suggests that our observations on the role of feature transformations do not depend on how long the network is trained.

We also investigate the effect of feature transformations on more complex few-shot learning algorithms. Specifically, we trained a Conv-4 architecture with the ProtoNet [30] loss, which uses unnormalized Euclidean distances. After training, we apply feature transformations before computing pairwise Euclidean distances between features in a nearest-neighbor approach. Table 4 presents the results of this experiment, which shows that CL2N normalization can also improve the performance of ProtoNet.

## 4. Conclusion

We analyzed the effect of simple feature transformations in nearest-neighbor classifiers for few-shot learning. We observed that such transformations — in particular, a combination of **centering** and **L2-normalization** — can improve the quality of the representation to a degree that the resulting classifiers outperforms several state-of-the-art approaches to few-shot learning. We hope that the SimpleShot classifiers studied in this paper will be used as a competitive baseline in future studies on few-shot learning.

**Acknowledgments** The authors thank Han-Jia Ye for helpful discussions. Y.W. and K.Q.W. are supported by grants from the NSF (III-1618134, III-1526012, IIS-1149882, IIS-1724282, and TRIPODS-1740822), the Bill and Melinda Gates Foundation, and the Cornell Center for Materials Research with funding from the NSF MRSEC program (DMR-1719875); and are also supported by Zillow, SAP America Inc., and Facebook.

Table 1: Average accuracy (in %; measured over 600/10,000 rounds\*) of one-shot and five-shot classifiers for five-way classification on *miniImageNet*; higher is better. The best result of each network architecture of each column is in **bold** font. Results of our approaches are in **blue**. Best viewed in color.

Approach	Network	One shot	Five shots
Meta LSTM [26]	Conv-4	43.44 $\pm$ 0.77	60.60 $\pm$ 0.71
MatchingNet [34]	Conv-4	43.56 $\pm$ 0.84	55.31 $\pm$ 0.73
MAML [4]	Conv-4	48.70 $\pm$ 1.84	63.11 $\pm$ 0.92
LLAMA [10]	Conv-4	49.40 $\pm$ 1.83	–
ProtoNet [30]	Conv-4	49.42 $\pm$ 0.78	68.20 $\pm$ 0.66
Reptile [23]	Conv-4	49.97 $\pm$ 0.32	65.99 $\pm$ 0.58
PLATIPUS [5]	Conv-4	50.13 $\pm$ 1.86	–
mAP-SSVM [32]	Conv-4	50.32 $\pm$ 0.80	63.94 $\pm$ 0.72
GNN [6]	Conv-4	50.33 $\pm$ 0.36	66.41 $\pm$ 0.63
RelationNet [31]	Conv-4	50.44 $\pm$ 0.82	65.32 $\pm$ 0.70
Meta SGD [18]	Conv-4	50.47 $\pm$ 1.87	64.03 $\pm$ 0.94
MTNet [17]	Conv-4	51.70 $\pm$ 1.84	–
Qiao <i>et al.</i> [25]	Conv-4	54.53 $\pm$ 0.40	67.87 $\pm$ 0.20
FEAT [36]	Conv-4	<b>55.15 <math>\pm</math> 0.20</b>	<b>71.61 <math>\pm</math> 0.16</b>
SimpleShot (UN)	Conv-4	33.17 $\pm$ 0.17	63.25 $\pm$ 0.17
SimpleShot (L2N)	Conv-4	48.08 $\pm$ 0.18	66.49 $\pm$ 0.17
SimpleShot (CL2N)	Conv-4	49.69 $\pm$ 0.19	66.92 $\pm$ 0.17
MAML [4] <sup>†</sup>	ResNet-18	49.61 $\pm$ 0.92	65.72 $\pm$ 0.77
Chen <i>et al.</i> [2]	ResNet-18	51.87 $\pm$ 0.77	75.68 $\pm$ 0.63
RelationNet [31] <sup>†</sup>	ResNet-18	52.48 $\pm$ 0.86	69.83 $\pm$ 0.68
MatchingNet [34] <sup>†</sup>	ResNet-18	52.91 $\pm$ 0.88	68.88 $\pm$ 0.69
ProtoNet [30] <sup>†</sup>	ResNet-18	54.16 $\pm$ 0.82	73.68 $\pm$ 0.65
Gidaris <i>et al.</i> [8]	ResNet-15	55.45 $\pm$ 0.89	70.13 $\pm$ 0.68
SNAIL [21]	ResNet-15	55.71 $\pm$ 0.99	68.88 $\pm$ 0.92
Bauer <i>et al.</i> [1]	ResNet-34	56.30 $\pm$ 0.40	73.90 $\pm$ 0.30
adaCNN [22]	ResNet-15	56.88 $\pm$ 0.62	71.94 $\pm$ 0.57
TADAM [24]	ResNet-15	58.50 $\pm$ 0.30	76.70 $\pm$ 0.30
CAML [15]	ResNet-12	59.23 $\pm$ 0.99	72.35 $\pm$ 0.71
SimpleShot (UN)	ResNet-10	54.45 $\pm$ 0.21	76.98 $\pm$ 0.15
SimpleShot (L2N)	ResNet-10	57.85 $\pm$ 0.20	78.73 $\pm$ 0.15
SimpleShot (CL2N)	ResNet-10	60.85 $\pm$ 0.20	78.40 $\pm$ 0.15
SimpleShot (UN)	ResNet-18	56.06 $\pm$ 0.20	78.63 $\pm$ 0.15
SimpleShot (L2N)	ResNet-18	60.16 $\pm$ 0.20	79.94 $\pm$ 0.14
SimpleShot (CL2N)	ResNet-18	<b>62.85 <math>\pm</math> 0.20</b>	<b>80.02 <math>\pm</math> 0.14</b>
Qiao <i>et al.</i> [25]	WRN	59.60 $\pm$ 0.41	73.74 $\pm$ 0.19
MatchingNet [34] <sup>‡</sup>	WRN	64.03 $\pm$ 0.20	76.32 $\pm$ 0.16
ProtoNet [30] <sup>‡</sup>	WRN	62.60 $\pm$ 0.20	79.97 $\pm$ 0.14
LEO [29]	WRN	61.76 $\pm$ 0.08	77.59 $\pm$ 0.12
FEAT [36]	WRN	<b>65.10 <math>\pm</math> 0.20</b>	<b>81.11 <math>\pm</math> 0.14</b>
SimpleShot (UN)	WRN	57.26 $\pm$ 0.21	78.99 $\pm$ 0.14
SimpleShot (L2N)	WRN	61.22 $\pm$ 0.21	81.00 $\pm$ 0.14
SimpleShot (CL2N)	WRN	63.50 $\pm$ 0.20	80.33 $\pm$ 0.14
SimpleShot (UN)	MobileNet	55.70 $\pm$ 0.20	77.46 $\pm$ 0.15
SimpleShot (L2N)	MobileNet	59.43 $\pm$ 0.20	78.00 $\pm$ 0.15
SimpleShot (CL2N)	MobileNet	<b>61.30 <math>\pm</math> 0.20</b>	<b>78.37 <math>\pm</math> 0.15</b>
SimpleShot (UN)	DenseNet	57.81 $\pm$ 0.21	80.43 $\pm$ 0.15
SimpleShot (L2N)	DenseNet	61.49 $\pm$ 0.20	81.48 $\pm$ 0.14
SimpleShot (CL2N)	DenseNet	<b>64.29 <math>\pm</math> 0.20</b>	<b>81.50 <math>\pm</math> 0.14</b>

<sup>†</sup>: Results reported in [2]. <sup>‡</sup>: Results reported in [36].

\*: [29, 36] and our results are averaged over 10,000 rounds.

Table 2: Average accuracy (in %; measured over 600/10,000 rounds\*) of one-shot and five-shot classifiers for five-way classification on *tieredImageNet*; higher is better. The best result of each network architecture of each column is in **bold** font. Results of our approach are in **blue**. Best viewed in color.

Approach	Network	One shot	Five shots
Reptile [23] <sup>‡</sup>	Conv-4	48.97 $\pm$ 0.21	66.47 $\pm$ 0.21
ProtoNet [30] <sup>‡</sup>	Conv-4	<b>53.31 <math>\pm</math> 0.89</b>	<b>72.69 <math>\pm</math> 0.74</b>
SimpleShot (UN)	Conv-4	33.12 $\pm$ 0.18	65.23 $\pm$ 0.18
SimpleShot (L2N)	Conv-4	50.21 $\pm$ 0.20	69.02 $\pm$ 0.18
SimpleShot (CL2N)	Conv-4	51.02 $\pm$ 0.20	68.98 $\pm$ 0.18
SimpleShot (UN)	ResNet-10	58.60 $\pm$ 0.22	79.99 $\pm$ 0.16
SimpleShot (L2N)	ResNet-10	64.58 $\pm$ 0.23	82.31 $\pm$ 0.16
SimpleShot (CL2N)	ResNet-10	65.37 $\pm$ 0.22	81.84 $\pm$ 0.16
SimpleShot (UN)	ResNet-18	62.69 $\pm$ 0.22	83.27 $\pm$ 0.16
SimpleShot (L2N)	ResNet-18	68.64 $\pm$ 0.22	84.47 $\pm$ 0.16
SimpleShot (CL2N)	ResNet-18	<b>69.09 <math>\pm</math> 0.22</b>	<b>84.58 <math>\pm</math> 0.16</b>
Meta SGD [18] <sup>†</sup>	WRN	62.95 $\pm$ 0.03	79.34 $\pm$ 0.06
LEO [29]	WRN	66.33 $\pm$ 0.05	81.44 $\pm$ 0.09
SimpleShot (UN)	WRN	63.85 $\pm$ 0.21	84.17 $\pm$ 0.15
SimpleShot (L2N)	WRN	66.86 $\pm$ 0.21	<b>85.50 <math>\pm</math> 0.14</b>
SimpleShot (CL2N)	WRN	<b>69.75 <math>\pm</math> 0.20</b>	85.31 $\pm$ 0.15
SimpleShot (UN)	MobileNet	63.65 $\pm$ 0.22	84.01 $\pm$ 0.16
SimpleShot (L2N)	MobileNet	68.66 $\pm$ 0.23	<b>85.43 <math>\pm</math> 0.15</b>
SimpleShot (CL2N)	MobileNet	<b>69.47 <math>\pm</math> 0.22</b>	85.17 $\pm$ 0.15
SimpleShot (UN)	DenseNet	64.35 $\pm$ 0.23	85.69 $\pm$ 0.15
SimpleShot (L2N)	DenseNet	69.91 $\pm$ 0.22	86.42 $\pm$ 0.15
SimpleShot (CL2N)	DenseNet	<b>71.32 <math>\pm</math> 0.22</b>	<b>86.66 <math>\pm</math> 0.15</b>

<sup>†</sup>: Results reported in [29]. <sup>‡</sup>: Results reported in [19].

\*: [29] and our results are averaged over 10,000 rounds.

Table 3: Average accuracy (in %; measured over 600/10,000 rounds\*) of one-shot and five-shot classifiers for five-way classification on CIFAR-100; higher is better. The best result is in **bold** font. Results of our approach are in **blue**. Best viewed in color.

Approach	Network	One shot	Five shots
TADAM [24]	ResNet	40.10 $\pm$ 0.40	<b>56.10 <math>\pm</math> 0.40</b>
SimpleShot (UN)	ResNet-10	36.38 $\pm$ 0.17	52.67 $\pm$ 0.18
SimpleShot (L2N)	ResNet-10	38.47 $\pm$ 0.17	53.34 $\pm$ 0.18
SimpleShot (CL2N)	ResNet-10	<b>40.13 <math>\pm</math> 0.18</b>	53.63 $\pm$ 0.18

\*: Our results are averaged over 10,000 rounds.

Table 4: **Feature transformations matter in 1NN classification with ProtoNet [30]**. We report average accuracy (in %; measured over 10,000 rounds) of five-way one-shot / five-shot ProtoNet classifiers on *miniImageNet* with and without feature transformations (applied after training).

1NN (UN) [30]	1NN (UN; ours)	1NN (L2N)	1NN (CL2N)
49.42 / 68.20	49.56 / 67.79	49.55 / 67.84	<b>50.12 / 68.51</b>



## References

- [1] M. Bauer, M. Rojas-Carulla, J. B. Swiatkowski, B. Scholkopf, and R. E. Turner. Discriminative k-shot learning using probabilistic models. *arXiv preprint arXiv:1706.00326*, 2017. 1, 3, 4
- [2] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. In *ICLR*, 2019. 4
- [3] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *PAMI*, 28(4):594–611, 2006. 1
- [4] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 1, 3, 4
- [5] C. Finn, K. Xu, and S. Levine. Probabilistic model-agnostic meta-learning. In *NeurIPS*, 2018. 1, 3, 4
- [6] V. Garcia and J. Bruna. Few-shot learning with graph neural networks. In *ICLR*, 2018. 1, 3, 4
- [7] I. Gauthier. *Dissecting face recognition: The role of expertise and level of categorization in object recognition*. PhD thesis, Yale University, 1998. 1
- [8] S. Gidaris and N. Komodakis. Dynamic few-shot visual learning without forgetting. In *CVPR*, 2018. 1, 3, 4
- [9] J. Gordon, J. Bronskill, M. Bauer, S. Nowozin, and R. Turner. Meta-learning probabilistic inference for prediction. In *ICLR*, 2019. 1
- [10] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. In *ICLR*, 2018. 1, 3, 4
- [11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 3
- [12] G. V. Horn and P. Perona. The devil is in the tails: Fine-grained classification in the wild. In *arXiv 1709.01450*, 2017. 1
- [13] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 3
- [14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 1, 3
- [15] X. Jiang, M. Havaei, F. Varno, G. Chartrand, N. Chapados, and S. Matwin. Learning to learn with conditional class dependencies. In *ICLR*, 2019. 1, 3, 4
- [16] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 2
- [17] Y. Lee and S. Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*, 2018. 1, 3, 4
- [18] Z. Li, F. Zhou, F. Chen, and H. Li. Meta-sgd: Learning to learn quickly for few shot learning. *arXiv preprint arXiv:1707.09835*, 2017. 4
- [19] Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. J. Hwang, and Y. Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, 2019. 4
- [20] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 2
- [21] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. In *ICLR*, 2018. 1, 3, 4
- [22] T. Munkhdalai, X. Yuan, S. Mehri, and A. Trischler. Rapid adaptation with conditionally shifted neurons. In *ICML*, 2018. 1, 3, 4
- [23] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018. 1, 3, 4
- [24] B. N. Oreshkin, A. Lacoste, and P. Rodriguez. Tadam: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, 2018. 1, 2, 3, 4
- [25] S. Qiao, C. Liu, W. Shen, and A. L. Yuille. Few-shot image recognition by predicting parameters from activations. In *CVPR*, 2018. 1, 3, 4
- [26] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017. 1, 2, 3, 4
- [27] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2018. 1, 2
- [28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F.-F. Li. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 2, 3
- [29] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell. Meta-learning with latent embedding optimization. In *ICLR*, 2019. 1, 2, 3, 4
- [30] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017. 1, 3, 4
- [31] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018. 1, 3, 4
- [32] E. Triantafillou, R. Zemel, and R. Urtasun. Few-shot learning through an information retrieval lens. In *CVPR*, 2017. 1, 3, 4
- [33] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018. 6
- [34] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *NIPS*, 2016. 1, 2, 3, 4
- [35] D. Wertheimer and B. Hariharan. Few-shot learning with localization in realistic settings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6558–6567, 2019. 6
- [36] H.-J. Ye, H. Hu, D.-C. Zhan, and F. Sha. Learning embedding adaptation for few-shot learning. *arXiv preprint arXiv:1812.03664*, 2018. 1, 3, 4
- [37] J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn. Bayesian model-agnostic meta-learning. In *NeurIPS*, 2018. 1
- [38] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016. 1, 3

## A. Meta-iNat Results

We also investigate the role of feature transformations in SimpleShot on the long-tailed iNaturalist dataset [33]. Following the meta-iNat benchmark [35], we split the dataset to have 908 base classes and 227 novel classes. We follow the evaluation setup of [35] and perform 227-way multi-shot evaluation. (In the meta-iNat benchmark, the number of shots varies per class.) We train all networks for 90 epochs using stochastic gradient descent. We set the initial learning rate to be 0.1 and batch size to be 256. We scale the learning rate by 0.1 after every 30 epochs.

The results of our meta-iNat experiments with SimpleShot are presented in Table 5. The table reports the averaging the accuracy on each class over all test classes (per class) and the average accuracy over all test images (mean). To the best of our knowledge, our highest accuracy of 62.13% (per class) and 65.09% (mean) is the current state-of-the-art on the meta-iNat benchmark. Figure 2 shows the absolute accuracy improvement (in %) of each of the classifiers compared to the baseline nearest-neighbor classifier without feature normalization (UN). In line with prior experiments, L2-normalization (L2N) leads to accuracy improvements in few-shot learning. Different from the other experiments, centering after L2-normalization (CL2N) does not improve the accuracy of SimpleShot further.

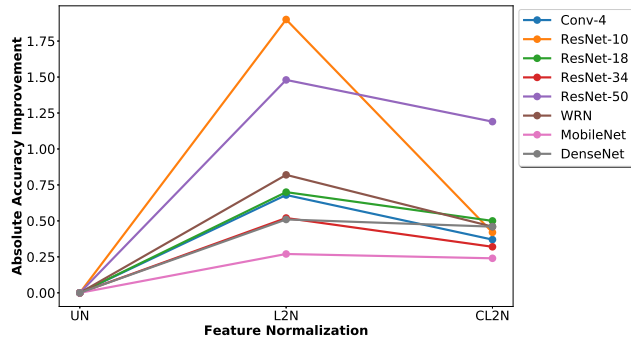


Figure 2: Absolute accuracy improvement (per class; in %) on the meta-iNat dataset of SimpleShot classifiers with L2-normalization (L2N) and centering and L2-normalization (CL2N) compared to a SimpleShot classifier without feature normalization (UN).

Table 5: Accuracy (in %) of SimpleShot classifiers in 227-way multi-shot classification on the meta-iNat benchmark [35]. Accuracy is measured by averaging the accuracy on each class over all test classes (per class) and by averaging accuracy over all test images (mean). Higher is better.

	SimpleShot (UN)		SimpleShot (L2N)		SimpleShot (CL2N)	
	Per class	Mean	Per class	Mean	Per class	Mean
<b>Conv-4</b>	21.32	22.93	22.00	23.73	21.69	23.21
<b>ResNet-10</b>	40.50	42.06	42.40	43.86	40.92	42.19
<b>ResNet-18</b>	55.33	58.06	56.03	58.50	55.83	58.33
<b>ResNet-34</b>	59.98	62.43	60.50	62.65	60.30	62.50
<b>ResNet-50</b>	54.13	56.85	55.61	57.77	55.32	57.47
<b>WRN</b>	60.48	63.22	61.30	63.77	60.94	63.42
<b>MobileNet</b>	52.01	53.92	52.28	54.06	52.25	54.01
<b>DenseNet</b>	61.62	64.77	62.13	65.09	62.08	65.02