Reimplementation of SimpleShot

Kevin Chen, Sophia Wang, Katherine Wei, Anya Yerramilli

1. Introduction

Few-shot learning requires models to recognize new object categories using only 1-5 labeled examples per class. Traditionally, it is believed that meta-learning, a method that involves training the model to learn how to learn, is required for these tasks. This can be computationally expensive and very complex training (due to nested loops). *SimpleShot* (Wang et al.) challenges this notion, and explores if a more naive solution can perform comparably. The SimpleShot model focuses on supervised training on base cases and nearest neighbor classification for new classes. It adds two critical feature transformations: L2-normalization and centering. The advantages of this model will be discussed in the latter sections.

2. Chosen Result

We aimed to reproduce the results of using the architecture described in *SimpleShot* (Wang et al.) for different feature transformations (UN, L2N, CL2N) across various CNN backbones (Conv-4, ResNet-10, ResNet-18) on 5-way 1-shot and 5-way 5-shot classification tasks using the miniImageNet dataset. Specifically, we focused on validating whether simple feature transformations could make nearest-neighbor classifiers competitive with meta-learning approaches, as the original paper claimed. We aimed to reproduce the performance improvements from UN (unnormalized), L2N, and CL2N.

3. Methodology

For re-implementing *SimpleShot*, we decided to focus on one of the datasets presented in the original paper, *miniImageNet*. This dataset has 100 classes and 600 examples per class. Following the structure from the paper, we split the dataset into 64 base training classes, 16 validation classes, and 20 test classes. Images were also resized to be 84 x 84 pixels, and we used center cropping.

Feature Extraction and Transformation

We implemented multiple CNN architectures for feature extraction: Conv-4, ResNet-10, and ResNet-18. After feature extraction, we applied three feature transformation methods:

1. **Unnormalized (UN):** Raw features without any normalization
2. **L2-Normalization (L2N):** $x \leftarrow \frac{x}{||x||_2}$ for feature vector x
3. **Centering + L2-Normalization (CL2N):** First centering data with $x \leftarrow x - \bar{x}$, then applying L2-normalization

L2-Normalization scales all of the feature vectors to a unit length. This helps reduce any magnitude bias due to a variety of different factors, such as lighting or sizing in images. For example, if one image has bright lighting and another is dark, the L2-Normalization helps scale them so they are comparable.

Centering helps to remove dataset-wide bias by subtracting the mean feature vector ($\bar{x}$) from each feature vector. This operation removes noisy information or dominant patterns that might be present across the entire dataset but aren't distinctive for classification.

Training and Evaluation

We trained for 90 epochs and used SGD and cross-entropy loss. The model starts with a learning rate of 0.1, then is reduced by a factor of 10 at epochs 45 and 66. We did 1-shot and 5-shot 5-way tasks. *x*-shot is using *x* examples per novel class, and 5-way is using 5 novel classes per task.

For evaluation, we tested all three of these feature transformations, and for validation, we used CL2N. After feature transformation, we used nearest-neighbors classification to label the images and get the predicted classes. The metrics on which we evaluated our model were the percent accuracy for labeling images.

Testing

While the training architecture uses a linear classifier, the testing architecture uses the nearest-neighbor classification for novel classes. The testing pipeline consists of extracting features from both query (test) images and support (example) images using the trained CNN backbone, applying feature transformations (UN, L2N, or CL2N) to both query and support features, and then performing the nearest-neighbor classification using Euclidean distance.

4. Results and Analysis

We created this graph for a run of the model using the Conv-4 CNN for feature extraction and using CL2N for feature transformation. Training accuracy rose to ~80% by epoch 45, and jumped to > 95% after the LR drop. Validation accuracy didn't jump as significantly, hovering at around 40%, and peaking at 43.17% after the LR drop. These results make sense because our model is really good at learning the classes it sees in training, but not as good at learning novel classes.

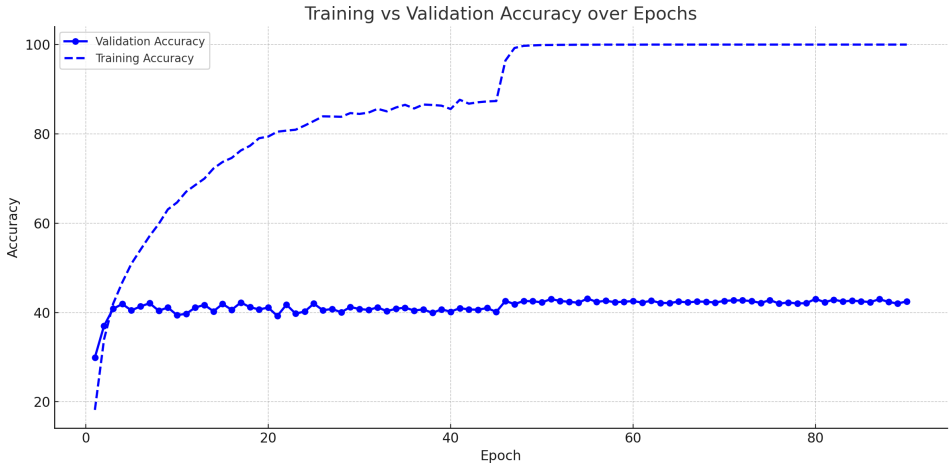

Figure 1: Training and validation accuracy over epochs under Conv-4 backbone

Our testing results for 5-way 1-shot and 5-way 5-shot classification with a Conv-4, ResNet-10, and ResNet-18 are as follows:

Network	**Five Shots**	**Five Shots (original paper)**	**One Shot**	**One shot (original paper)**
Conv-4 (UN)	58.41 ± 0.17%	*63.25 ± 0.17*	30.76% ± 0.15%	*33.17 ± 0.17*
Conv-4 (L2N)	61.09% ± 0.17%	*66.49 ± 0.17*	43.53% ± 0.18%	*48.08 ± 0.18*
Conv-4 (CL2N)	61.19% ± 0.17%	*66.92 ± 0.17*	44.65% ± 0.18%	*49.69 ± 0.19*
ResNet-10 (UN)	70.09% ± 0.16%	*76.98 ± 0.15*	50.70% ± 0.20%	*54.45 ± 0.21*
ResNet-10 (L2N)	70.96% ± 0.16%	*78.73 ± 0.15*	52.41% ± 0.20%	*57.85 ± 0.20*
ResNet-10 (CL2N)	70.92% ± 0.16%	*78.40 ± 0.15*	54.84% ± 0.20%	*60.85 ± 0.20*
ResNet-18 (UN)	69.91% ± 0.16%	*78.63 ± 0.15*	48.58% ± 0.20%	*56.06 ± 0.20*
ResNet-18 (L2N)	71.10% ± 0.16%	*79.94 ± 0.14*	53.08% ± 0.20%	*60.16 ± 0.20*

ResNet-18 (CL2N)	71.46% ± 0.16%	*80.02 ± 0.14*	55.71% ± 0.20%	*62.85 ± 0.20*

Our results reinforce the findings of the original SimpleShot (Wang et al.) paper: feature normalization, particularly CL2N, significantly improves few-shot classification performance. Despite slightly lower testing accuracies for all settings compared to the original paper, our results still confirm the impact of feature transformation techniques and the number of shots on model performance. While ResNet-10 and ResNet-18 achieve higher accuracy overall, Conv-4 remains more competitive relative to the original paper's results, given its simplicity. The impact of normalization is most pronounced in 1-shot settings, where structured embeddings are critical.

5. Reflections

Our results support the hypothesis that the SimpleShot nearest-neighbor classifier, when paired with proper feature normalization (such as Centering and L2N or CL2N) and model backbone, can achieve comparable performance to meta-learning methods in few-shot learning tasks. The SimpleShot methods provide significant advantages in training efficiency and simplicity compared to meta-learning. Future work in few-shot classification can explore integrating more advanced feature embeddings, investigating the impact of different backbone architectures, and combining SimpleShot with lightweight fine-tuning or adaptation techniques to enhance performance beyond meta-learning techniques.

GitHub: https://github.com/anya-yerramilli/cs4782_final_project

References:

[1] Wang, Y., Chao, W.-L., Weinberger, K. Q., & van der Maaten, L. (2019). SimpleShot: Revisiting nearest-neighbor classification for few-shot learning. arXiv. https://arxiv.org/abs/1911.04623
[2] Zagoruyko, S., & Komodakis, N. (2016). Wide Residual Networks. arXiv. https://arxiv.org/abs/1605.07146

Include other networks (will format properly later):

- Conv-4: https://arxiv.org/pdf/1703.05175
- ResNet: https://arxiv.org/pdf/1512.03385
- MobileNet: https://arxiv.org/pdf/1704.04861
- DenseNet: https://arxiv.org/pdf/1608.06993