Improving Robustness of Prototypical Network in Noisy Few-Shot Settings

Bao Pham

Rensselaer Polytechnic Institute

April 22, 2022

Few Shot Learning (FSL)

Consider a learning task T with a training data set D_{train}, **FSL** is a type of ML setting that deals with:

$$D_{train} = \{(x_i, y_i)\}_{i=1}^{I} \text{ where } I \text{ is small.}$$

Few Shot Learning (FSL)

The empirical risk of the training set D_{train} of I samples:

$$R_I(h) = \frac{1}{I}\sum_{i=1}^{I}\mathcal{L}(h(x_i), y_i)$$

With I being small, R_I is **no longer reliable** —also what if D_{train} is **corrupted**?

The goal of this work is to improve **Prototypical Network** for such an issue.

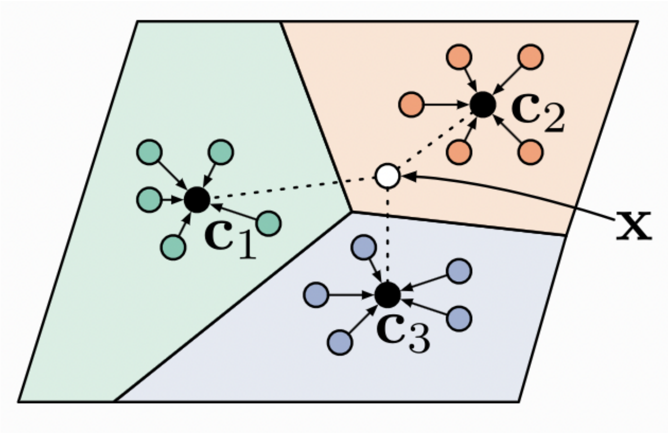
Prototypical Network

Introduced by Snell, J. et al. (2017)

Idea —

> There exists an **embedding** in which points cluster around a single prototype representation for each class.

The model finds the **nearest class prototype** for an embedded query point.



ProtoNet 3-way 5-shot

Prototypical Network

Given a support set S_k and a query set Q_k, determine a prototype c_k that Q_k belongs to.

A prototype c_k is computed as:

$$c_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\phi(x_i)$$

$f_\phi(\cdot)$ is an embedding function.

Prototypical Network

A prototype c_k is then used to classify new examples x_q

$$p_\phi(y = k \mid x_q) = \frac{\exp[-d(f_\phi(x_q), c_k)]}{\sum_{k'} \exp[-d(f_\phi(x_q), c_{k'})]}$$

where d is the Euclidean distance function.

Learning proceeds by minimizing $J(\phi) = -\log p_\phi(y = k \mid x_q)$ via SGD.

Training Prototypical Network

Algorithm 1 Training episode loss computation for prototypical networks. N is the number of examples in the training set, K is the number of classes in the training set, $N_C \leq K$ is the number of classes per episode, N_S is the number of support examples per class, N_Q is the number of query examples per class. $\text{RANDOMSAMPLE}(S, N)$ denotes a set of N elements chosen uniformly at random from set S, without replacement.

Input: Training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where each $y_i \in \{1, \ldots, K\}$. $\mathcal{D}_k$ denotes the subset of $\mathcal{D}$ containing all elements $(\mathbf{x}_i, y_i)$ such that $y_i = k$.

Output: The loss J for a randomly generated training episode.

$V \leftarrow \text{RANDOMSAMPLE}(\{1, \ldots, K\}, N_C)$ ▷ Select class indices for episode

for k in $\{1, \ldots, N_C\}$ **do**

$S_k \leftarrow \text{RANDOMSAMPLE}(\mathcal{D}_{V_k}, N_S)$ ▷ Select support examples

$Q_k \leftarrow \text{RANDOMSAMPLE}(\mathcal{D}_{V_k} \setminus S_k, N_Q)$ ▷ Select query examples

$$\mathbf{c}_k \leftarrow \frac{1}{N_C} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i)$$ ▷ Compute prototype from support examples

end for

$J \leftarrow 0$ ▷ Initialize loss

for k in $\{1, \ldots, N_C\}$ **do**

for $(\mathbf{x}, y)$ in Q_k **do**

$$J \leftarrow J + \frac{1}{N_C N_Q} \left[d(f_\phi(\mathbf{x}), \mathbf{c}_k)) + \log \sum_{k'} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k)) \right]$$ ▷ Update loss

end for

end for

Problem

Recall that the prototype of a class c_k is computed by an arithmetic mean, where **all samples are treated equally**.

> If some samples of S_k are contaminated by **noise**, the performance of c_k may suffer severely, especially under the condition of **data scarcity**.

Goal —Improve Prototypical Network via computing a **more robust prototype** to better its training process.

Weighted Prototypes

Using the work of **Zhu et al.** (2020), an adaptive re-weighting schema is used to counteract the effects of noise on c_k.

Each embedded support sample $f_\phi(x_i)$ is assigned a weight α_i to measure the effect of corresponding sample x_i on the prototype and focus more on samples close to the correct prototype.

$$\alpha_i = \frac{1}{d\big(f_\phi(x_i),\ \frac{1}{|S_k|-1}\sum_{j=1,j\neq i}^{|S_k|} f_\phi(x_j)\big)}$$

$d(\cdot)$ is the squared Euclidean distance metric.

Weighted Prototypes

Using α_i, a more robust prototype μ_k is computed.

$$\mu_k = \frac{\sum_{i=1}^{|S_k|} \alpha_i \, f_\phi(x_i)}{\sum_{i=1}^{|S_k|} \alpha_i}$$

We can also modify the loss function to take into account the distance between **two kinds of prototypes:** μ_k, generated from S_k, and $\tilde{\mu}_k$, generated from Q_k.

Joint Loss Function

Compute $\tilde{\mu}_k$ in the same manner as μ_k but uses Q_k instead.

Then calculate $p_\mu(\tilde{\mu}_k)$

$$p_\mu(\tilde{\mu}_k) = \frac{\exp[-d(\tilde{\mu}_k, \mu_k)]}{\sum_{k'} \exp[-d(\tilde{\mu}_k, \mu_{k'})]}$$

We then minimize over

$$J(\phi, \mu) = -\big(\log p_\phi(y = k \mid x_q) + \lambda \log p_\mu(\tilde{\mu}_k)\big)$$

$\lambda = 0.01$

Experiment

MiniImageNet:

- 64 training classes
- 12 validation classes
- 24 test classes

Network Architecture:

- Same architecture as ProtoNet. (Conv4-backbone)
- Each block has 3x3 Conv2D, BatchNorm2D, ReLU, 2x2 Max Pooling
- The resultant $f_\phi(x_i)$ is flatten into a vector $(C \times H' \times W')$ where $C = 64$

Experiment

Training:

- Adam optimizer with learning rate of 0.001 is used.
- Model is trained accordingly to **Zhu et al.** —40 epochs with 20-way and 5-shot samples that are perturbed by Gaussian noise with $\sigma = 0.7$. with a noise rate of 50%
- Learning rate is halved at 20 epochs.

Results are compared to those of **Zhu et al.** (2020; RRPNet), the focus is on 5-way 5-shot with noise rate of 50%.

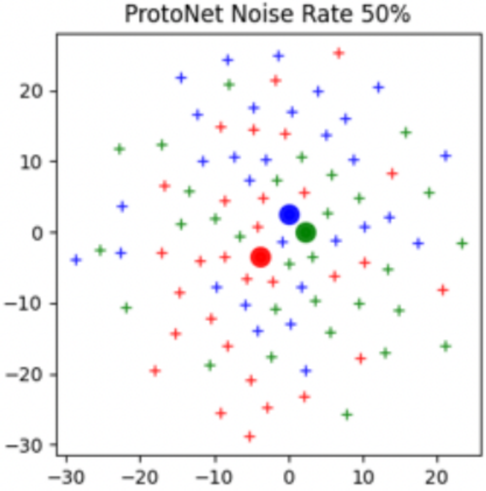
- Test is conducted with 2000 episodes and done 5 times. Accuracy of each test is gathered and averaged altogether.

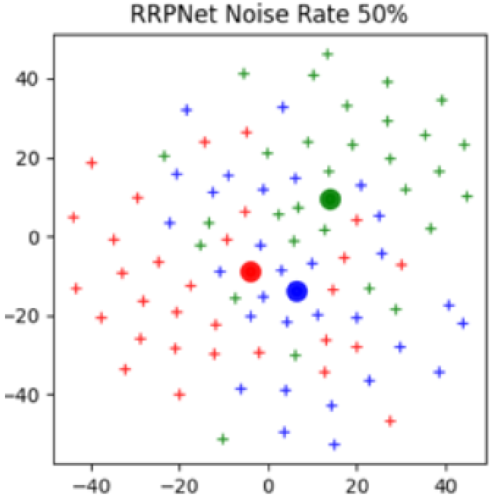
Results

Noise Rate	Model	5 Shot 5 Way Accuracy
10%	ProtoNet	64.15 ±0.66
	RRPNet	**66.30**±0.64
	JRPNet	63.77 ±0.29
30%	ProtoNet	57.87 ±0.58
	RRPNet	61.21 ±0.62
	JRPNet	**63.53** ±0.20
50%	ProtoNet	53.86 ±0.42
	RRPNet	56.11 ±0.46
	JRPNet	**63.59** ±0.19

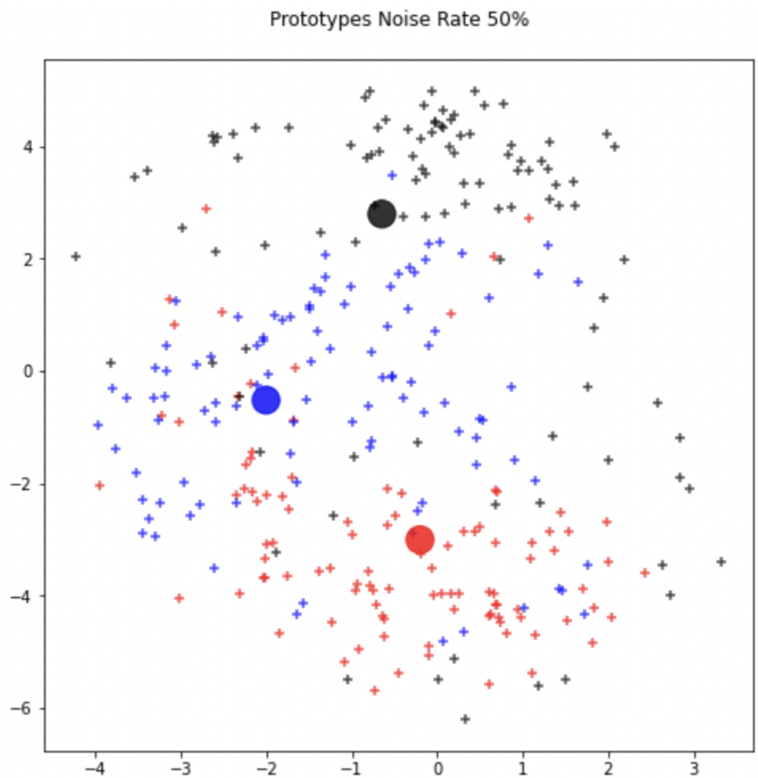
Results

The t-SNE visualization results of prototype representation from **Zhu et al.** (2020).



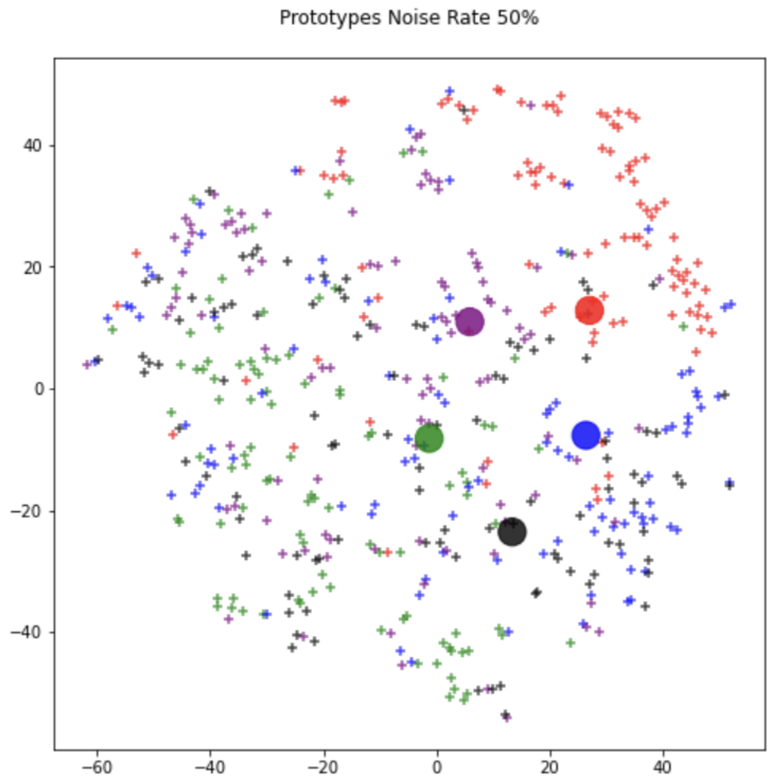


Results



Prototypes View with t-SNE 3-Way 100-Shot Scenario

Results



Prototypes View with t-SNE 5-Way 100-Shot Scenario

Conclusion

The added modifications to ProtoNet illustrate some improvements. However, they need to be further...

- Test with mislabeled S_k
- Test with noisier images (e.g., from adversarial attacks)

References I

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov.
Siamese neural networks for one-shot image recognition.
2015.

Jake Snell, Kevin Swersky, and Richard Zemel.
Prototypical networks for few-shot learning.
In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4080–4090, Red Hook, NY, USA, 2017. Curran Associates Inc.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra.
Matching networks for one shot learning.
In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3637–3645, Red Hook, NY, USA, 2016. Curran Associates Inc.

References II

Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni.
Generalizing from a few examples: A survey on few-shot learning.
ACM Comput. Surv., 53(3), jun 2020.

Junjie Zhu, Xiaodong Yi, Naiyang Guan, and Hang Cheng.
Robust re-weighting prototypical networks for few-shot classification.
In *2020 6th International Conference on Robotics and Artificial Intelligence*, ICRAI 2020, page 140–146, New York, NY, USA, 2020. Association for Computing Machinery.