

RNNP: A Robust Few-Shot Learning Approach

Pratik Mazumder[†]

Pravendra Singh[†]

Vinay P. Namboodiri^{†*}

[†]Department of Computer Science and Engineering, IIT Kanpur, India

^{*}University of Bath, United Kingdom

{pratikm, psingh}@iitk.ac.in, vpn22@bath.ac.uk

Abstract

Learning from a few examples is an important practical aspect of training classifiers. Various works have examined this aspect quite well. However, all existing approaches assume that the few examples provided are always correctly labeled. This is a strong assumption, especially if one considers the current techniques for labeling using crowd-based labeling services. We address this issue by proposing a novel robust few-shot learning approach. Our method relies on generating robust prototypes from a set of few examples. Specifically, our method refines the class prototypes by producing hybrid features from the support examples of each class. The refined prototypes help to classify the query images better. Our method can replace the evaluation phase of any few-shot learning method that uses a nearest neighbor prototype-based evaluation procedure to make them robust. We evaluate our method on standard mini-ImageNet and tiered-ImageNet datasets. We perform experiments with various label corruption rates in the support examples of the few-shot classes. We obtain significant improvement over widely used few-shot learning methods that suffer significant performance degeneration in the presence of label noise. We finally provide extensive ablation experiments to validate our method.

1. Introduction

Consider that you need to train a robot on a new concept of spectacles. You ideally would like to do so by providing a few examples. However, you make a mistake and provide an example of sunglasses as spectacles instead of using its original class. What will your robot do now? Will it retrieve your spectacles when you ask it to do so? Most probably not if your robot is trained using existing deep learning techniques. Through this paper, we address this issue and provide a method to perform robust prototype-based classification on a limited data setting.

Deep learning methods need vast amounts of annotated data for training, which can be very difficult and costly to

obtain. Such models are unable to perform well for categories having very few training samples. On the other hand, humans can quickly learn new categories by simply looking at a few samples and can perform very well in identifying them. We need to reduce this gap between deep learning and humans. Few-shot learning algorithms address this shortcoming of deep learning. Few-shot learning involves training networks in such a way that they can achieve good results for classes that have very few training data (few-shot classes). Researchers have proposed several methods to deal with this problem [19, 2, 17, 12, 21, 4, 26]. However, it is possible that these few-shot classes contain some support examples with corrupted labels. This makes the few-shot learning problem even more challenging. In this paper, we deal with the problem of few-shot learning, where the few-shot classes contain support examples with corrupted labels. In the few-shot learning setting, the network receives one episode at a time, and each episode contains a set of support examples and a set of query examples. Each episode contains a few classes and has very few support examples per class. The query examples also belong to one of the classes present in the episode. In this setting, the task of the network is to use the limited support data present in an episode to classify the query data points in the episode.

During the base class (defined in Sec. 3.1) training phase, the model trains on episodes containing classes from the base class set only. During the novel class (defined in Sec. 3.1) evaluation phase (testing), the model receives episodes from novel/few-shot classes (not part of the base class set), and it has to successfully classify the query examples of the novel class episode using the support examples in that episode only. Since the number of support examples per class is very low, standard classification training makes the network overfit to the limited training data in the episode and fail miserably in classifying the query data. A large number of few-shot learning methods perform few-shot testing by applying the nearest neighbor prototype-based testing approach (NNP) [19, 12, 30, 29, 24, 8], i.e., for each class, they learn a prototype, and for each query ex-

ample, they calculate the distance of the query feature from the prototypes. They predict the class prototype closest to the query example in the feature space as the output class.

In the above setting, during the novel class evaluation phase, if some of the support examples of the novel/few-shot classes have corrupted labels, then the few-shot learning algorithms will suffer from significant performance degradation. We propose a novel method to address the loss in performance (due to corrupted support examples) of few-shot learning methods that use the NNP testing approach. This work focuses specifically on the novel class evaluation phase (testing) of few-shot learning algorithms. Several methods deal with corrupted labels in the full-shot settings (with a huge number of samples per class) [18, 5, 10, 3, 31]. However, these methods are training-based and do not perform well when trained on the few support examples in the novel class episode due to overfitting, as shown in Sec. 5.9. Therefore, there is an urge to make few-shot learning robust to label corruption.

The support examples with corrupted labels (in a novel class episode) corrupt the prototypes computed for each class and harm the performance. Our method refines the prototypes in order to reduce the level of label corruption in them. It produces hybrid features from the features of the support examples extracted by the network and performs a soft k-means clustering on the support features and hybrid features in order to refine the prototypes. Since the corrupt support examples are relatively closer to the other support examples of their actual class, the clustering procedure helps correct the labels for many such support examples. This, in turn, helps reduce the influence of the corrupt support examples in the class prototypes of the wrong class. Consequently, the refined class prototypes perform better in classifying the query data points of that episode. We call our method the Robust Nearest Neighbor Prototype-based testing approach (RNNP) (depicted in Fig. 1). Our method can replace the testing process of any few-shot learning algorithm that uses the nearest neighbor prototype-based evaluation. Our method does not make use of any external data or additional labels of any form. In order to demonstrate the efficacy of our method, we replace the testing process of a state-of-the-art few-shot learning method FEAT [30] and a very popular few-shot learning method prototypical network [19]. Our experiments show that our testing strategy works for multiple datasets and few-shot learning methods.

To the best of our knowledge, this is the first work that tries to make the testing strategy of few-shot image classification algorithms robust to any label corruption in the support examples. Our major contributions are as follows:

- We propose a novel approach called Robust Nearest Neighbor Prototype-based testing (RNNP) that makes the few-shot evaluation process robust to noisy/corrupted labels in the novel class episodes.

- We experimentally show that our method significantly improves the performance of few-shot learning methods that use a nearest neighbor prototype-based testing strategy, in the presence of noisy labels.
- We validate the different components of our method using extensive ablation studies.

2. Related Works

2.1. Few-Shot Classification

Researchers have proposed several approaches to deal with the problem of few-shot classification. Siamese Neural Network [6] trained models to learn to discriminate between pairs of images. This training strategy helped the network to even discriminate between images from previously unseen classes at test time. Prototypical Network [19] produces class prototypes by averaging the features of support examples of that class and then uses the nearest neighbor approach to find the nearest prototype to the query feature in order to predict its class. Model Agnostic Meta Learning [2] method trains the network to quickly adapt to new data by taking an average gradient step that benefits multiple non-similar tasks at once. RelationNet [20] trains the network to generate a similarity score between a query image and support examples of a class.

PFA [14] tries to predict the parameters for adapting a network to the new class by using the activations for new images. TADAM [12] uses task-based embedding as attention to the network to produce better features for the images in the episode. MetaOptNet [7] trains the network to learn features that work well with linear classifiers when used for few-shot classification. CTM [8] trains a module to select important dimensions in the features that help improve the classification performance. Simple Shot [23] optimizes the representations to perform better on the nearest neighbor classifier.

FEAT - Few-shot Embedding Adaptation with Transformer [30] uses multi-headed attention to adapt the class prototypes to draw them away from each other. This helps the nearest neighbor classification further. PARN [27] trains a position-aware relation network (PARN) that tries to make the relation module invariant to changes in the spatial position of the object. MetaMG [28] trains a network to perform fast few-shot incremental learning

2.2. Learning with Noisy Labels

Researchers have proposed various methods to help the network learn better in the presence of noisy labels. Some methods focus on estimating the latent noisy transition matrix [9, 11, 18, 13]. Co-teaching [3], involves training two networks simultaneously, and each network provides samples with possibly clean labels from each mini-batch to the other network for training.

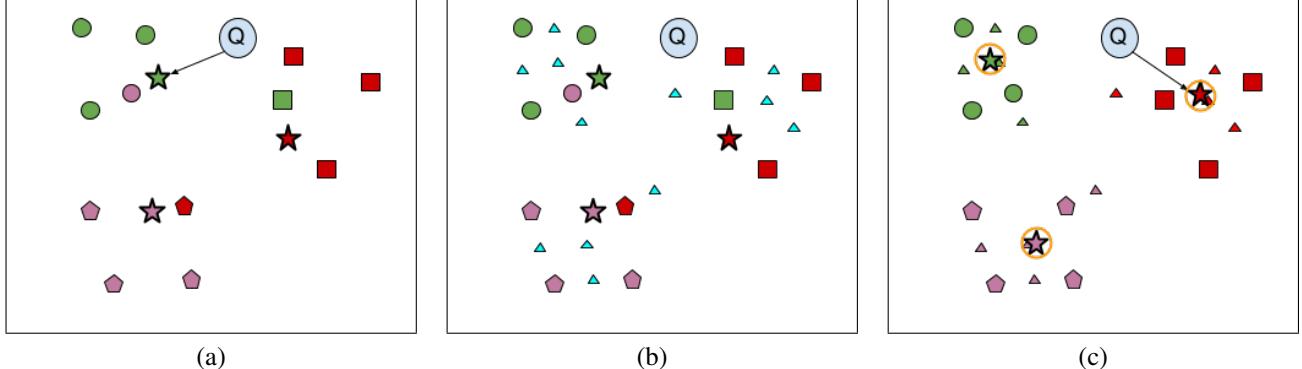


Figure 1: (a) Green, red, and magenta refer to support examples of 3 classes, say A, B, and C, respectively. Let us assume that due to label corruption, 1 support example of each class is corrupted (marked with the color of a different class). Stars represent the prototype for each class (marked with the color of the corresponding class) computed by averaging the support features. Standard nearest neighbor prototype-based classification (NNP) results in wrongly predicting class A for query data point Q. (b, c) Triangles represent the unlabeled hybrid features produced from the support example features. RNNP produces unlabeled hybrid features from the support examples and performs clustering on the hybrid features, support features, and the query feature. It updates the cluster centers and rectifies the labels of the support examples. The newly computed class prototypes (depicted as stars marked with the color of the corresponding class and surrounded by an orange circle) result in correctly predicting class B for the query data point Q. The arrow points to the class prototype closest to Q.

Methods like Decoupling [10] use a “Disagreement” strategy of training, where the network updation is based on the level of disagreement between two different networks on the same data point. Co-teaching+ [31] combines the “Disagreement” strategy with Co-teaching to improve the robustness of the model. However, there is no guarantee that the examples chosen for the “Disagreement” strategy have the correct labels. Another approach of tackling this problem is to utilize regularization from peer networks to improve the generalization ability of networks by encouraging agreement between them. However, these methods still suffer from the memorization of the noisy labels [33]. JoCoR (Joint Training with Co-Regularization) [25] trains two networks with a conventional supervised loss and a co-regularization loss. It also employs an example selection strategy, which chooses examples with a lower total loss for training the networks. This minimizes the flow of erroneous information.

Our few-shot setting has corrupt labels only in the novel class episodes, which have very few samples. These methods require a large number of labeled data for training, and therefore, they overfit to the scant data in the few-shot setting. We experimentally show in Sec. 5.9 that such methods do not perform well and are not suitable for the few-shot setting.

3. Proposed Method

3.1. Problem Setting

In the few-shot learning setting, there are two sets of classes of images, i.e., the base class set and the novel class set. The unique nature of this setting is that these two sets of

classes do not share any common classes. The base classes C_{base} have many labeled data points, but the novel/few-shot classes C_{few} only have a few labeled examples.

An episodic formulation is generally followed in the few-shot learning setting. In this setting, the network receives 1 episode at a time and each episode contains a set of support examples and a set of query examples. In an episode, there are N classes and each class has K support examples per class i.e. $S = \{(x_1, y_1), (x_2, y_2) \dots (x_{K \times N}, y_{K \times N})\}$. The query examples also belong to one of the N classes i.e. $Q = \{(x_1^*, y_1^*), (x_2^*, y_2^*) \dots (x_q^*, y_q^*)\}$. Here x_i are images and $y_i \in \{1..N\}$ are labels. Such an episode is referred to as a K -shot N -way episode. The number of support examples per class is very low (e.g. $K=1,5,10$).

There are two phases in this setting: the base class training phase and the novel class evaluation (testing) phase. During the base class training phase, the network trains on episodes of the base classes. Each such episode contains a few classes from the base class set. The network trains on the support examples per class (S) present in each episode. During this phase, the network can carry over the knowledge gained from one episode to the next.

During the novel class evaluation phase, the network receives episodes of the novel classes. The network has to use the few support examples per class (S) present in an episode to classify the query data points (Q) in the episode. During this phase, the network cannot carry over the knowledge gained from one episode to the next. In the novel class episodes, some of the support examples labels are incorrect. This type of corruption makes the few-shot classification of

the novel classes challenging. The distribution of label corruption is uniform across the classes in each episode. The objective of the network is to deal with this corruption and classify query examples in each novel class episode.

3.2. Method Overview

In the few-shot learning problem setting with label corruption (in novel class episodes), since there are a limited number of support examples per class and some support examples may also be corrupted (belonging to a different class), the computed class prototypes are not correct. We propose a novel method (RNNP) to improve the robustness of the few-shot learning method when used for few-shot testing on episodes that contain support examples with corrupted labels (Fig. 1). Since RNNP is a testing method, it can replace the testing process of any few-shot learning method that utilizes a nearest neighbor prototype-based (NNP) few-shot testing mechanism.

Let us assume that we have a base network B already trained on the episodes from the base classes. During the novel class evaluation phase (testing), given a novel class episode, we first extract features for the support images and query images using the trained base network (B).

$$z_i = B(x_i) \quad (1)$$

where x_i refers to the support or query image, and z_i is the extracted feature of that image.

Next, we calculate the prototypes of each class using the support image features of that class. Many methods use simple averaging to get the mean prototype per class [19]. Some methods adjust this mean prototype to further help in classifying the query examples [30].

$$p_n = \sum_{i=1}^K \frac{z_i^n}{K} \quad (2)$$

where p_n is the prototype for the n^{th} class and $z_1^n, z_2^n, \dots, z_K^n$ are the features of the support examples of the n^{th} class.

The computation process of the prototypes unknowingly utilizes the support examples with corrupted labels too. Therefore, the class prototypes are not fully correct and need refinement. In order to refine them, we first create unlabeled hybrid features by combining the support images features from the same class using proportion hyperparameter α . For each support image feature, we generate β unlabeled hybrid features.

$$z_u = \alpha * z_i^n + (1 - \alpha) * z_j^n \quad (3)$$

where z_u is the generated unlabeled hybrid feature, z_i^n, z_j^n are features of support examples of class $n \in C_{few}$ and $i \neq j$. $\alpha \in (0, 1)$ is the proportion hyper-parameter.

Next, for each query image, we perform soft k-means clustering on the combined set of the support image features, the hybrid features, and the corresponding query image feature. We use the class prototypes computed in Eq. 2 as the initial cluster centers for the clustering algorithm. Using the current cluster centers, we assign soft labels to the support and hybrid features and the single query feature. We update the cluster centers using these soft labels. We repeat this process a few times and use the final cluster centers as the refined class prototypes. Since the corrupt support features are relatively closer to the features of the support examples of their actual classes, the clustering process can correctly assign many of them to their actual classes. This process serves two purposes. Firstly, it reduces the influence of the corrupted labels on the class prototypes. Secondly, this process also reduces the problem of overfitting in the prototypes because we compute the refined prototypes using both hybrid and support features.

After obtaining the refined prototypes, we find the distance from each of refined class prototypes in the feature space to the query image. Next, we apply a softmax function over the negative values of the distances so that refined class prototypes at smaller distances from the query example get higher softmax probabilities.

$$W_q^i = \frac{\exp(-d(z_q, p_i^*))}{\sum_{j=1}^N \exp(-d(z_q, p_j^*))} \quad (4)$$

where W_q^i is the class probability for q^{th} query example for the i^{th} class, $d(.,.)$ is the distance metric, p_i^* is the refined class prototype for the i^{th} class, z_q is the feature for the q^{th} query example. N is the total number of classes in the episode.

The class for which the above probability is highest, is predicted as the output class for that query image.

$$\hat{y}_q = \text{argmax}_n W_q^n \quad (5)$$

where $n \in \{1, 2, \dots, N\}$ refers to the n^{th} class, \hat{y}_q refers to the predicted class for the q^{th} query example.

3.3. Different from Transductive and Semi-Supervised FSL

Our method should not be confused with transductive few-shot learning. Transductive few-shot learning makes use of all the query points simultaneously to help in classifying themselves by making use of the structure of this bulk data. However, our setting is the standard few-shot setting, and we cannot use the query images to help classify other query images. Instead, we utilize hybrid features to help in classifying the query images better.

Our method also does not fall under the semi-supervised few-shot learning setting. Semi-supervised few-shot learning uses unlabeled real data from the same dataset, either

belonging to the classes in that episode or to other classes in the dataset. These data are used for training and also for testing. However, our method does not use extra real data and only uses an interpolation of support features of the same class. This makes our method very efficient. We consider them as unlabeled data points and use them in our clustering algorithm only during testing.

4. Experiments

4.1. DataSets

We perform few-shot classification experiments on 2 few-shot classification datasets, namely, mini-ImageNet [22] and tiered-ImageNet [15].

mini-ImageNet [22] is the most popular few-shot learning dataset. It is derived from the ImageNet dataset [16]. It has 100 classes and 600 images of size 84×84 pixels for each class. The classes consist of 64 train classes, 16 validation classes, and 20 test classes.

tiered-ImageNet [15] is larger subset of the ImageNet [16] dataset. It has 608 classes divided into 351 train, 97 validation, and 60 test classes. It groups similar classes into higher-level classes. In tiered-ImageNet, the training classes differ significantly from the testing classes as compared to mini-ImageNet.

4.2. Implementation Details

Since the mini-ImageNet and tiered-ImageNet datasets do not contain corrupted labels, we manually corrupt the support image labels for the novel classes during the novel class evaluation phase. We perform experiments with 0%, 20%, and 40% label corruption rates in the support examples of the novel classes. 20% label corruption rate refers to the setting where 20% of the support examples in an episode have incorrect labels. The distribution of label corruption is uniform across all the classes in each episode. For each novel class episode, we assume that the support image with corrupted labels belongs to another class from the same episode. We use this assumption so that the episodic formulation holds, i.e., all the examples in an episode are from the classes in the episode. For fairness, we perform all the experiments in this setting. We perform experiments in this corrupted label setting for 5-shot 5-way and 10-shot 5-way episodes.

We replace the testing process of two few-shot learning methods (ProtoNet and FEAT) with our proposed method, RNNP, and perform few-shot testing experiments on the trained models. We do not retrain the models. The first method is the prototypical network (ProtoNet)[19], which is a very popular few-shot learning method. It trains and tests the network on episodes and predicts the output class using the nearest neighbor prototype-based (NNP) approach to find the closest class prototype to the query feature. We

also perform experiments on the recent state-of-the-art few-shot learning method FEAT - (Few-shot Embedding Adaptation with Transformer) [30]. FEAT adapts the class prototypes using a multi-headed self-attention based transformer to increase the gap between them. It also uses NNP for testing. We use ResNet-12 [30] as the backbone architecture.

We follow the testing protocol that we have defined earlier. Whenever we calculate/re-calculate the class prototypes (cluster centers), we use the mechanism proposed by the respective method (FEAT or ProtoNet) for each case. For the prototypical network (ProtoNet), we compute the class prototypes by averaging the support image features of the corresponding class. For FEAT, we compute the class prototypes by averaging the support image features of the corresponding classes and adapting them using a trained transformer.

We can choose the proportion hyper-parameter α between 0 and 1. We use $\alpha = 0.8$ for our experiments and validate this choice through ablation experiments. In the 5-shot episodes, for each support example, we have 4 other support examples of the same class. Similarly, in the 10-shot episodes, for each support example, we have 9 other support examples of the same class. For each support example in the 5-shot episodes, we generate $\beta = 4$ hybrid features using the 4 other support examples of the same class. Similarly, for each support example in the 10-shot episodes, we generate $\beta = 9$ hybrid features. For our experiments, we repeat the clustering and label assignment process for three times. We validate these choices using ablation experiments. We evaluate both the models with and without RNNP. We evaluate the models for 1000 episodes and report the best out of 5 runs for each method. The above-mentioned methods (FEAT, ProtoNet) with the traditional testing protocol form our baseline for those methods.

Since there are no previous works that make the few-shot evaluation procedure robust to noisy support examples of novel classes, we compare our method with two popular full-shot noisy label training methods in our setting (Sec. 5.9).

4.3. mini-ImageNet Results

Table 1 reports the results for few-shot classification on 5-shot 5-way episodes of mini-ImageNet with various levels of corruption. The results indicate that for both FEAT and ProtoNet, the performance drops significantly with the increase in the level of corruption of the support image labels when using the standard nearest neighbor prototype-based testing process (NNP). For 20% label corruption, the performance of FEAT drops by absolute margins of 5.38% and 2.8% for 5-shot and 10-shot 5-way episodes, respectively. Similarly, for 20% label corruption, the performance of ProtoNet drops by absolute margins of 4.96% and 2.7% for 5-shot and 10-shot 5-way episodes, respectively. When

Table 1: Performance of FEAT and ProtoNet with NNP/RNNP on the mini-ImageNet dataset over 5/10-shot 5-way episodes with 0%, 20%, 40% label corruption.

Corruption Rate	Testing Method	FEAT (CVPR'20)		ProtoNet (NIPS'17)	
		5-shot	10-shot	5-shot	10-shot
0%	NNP	$82.05 \pm 0.14\%$	$85.03 \pm 0.44\%$	$80.53 \pm 0.14\%$	$83.61 \pm 0.45\%$
	RNNP (Ours)	$82.76 \pm 0.42\%$	$85.52 \pm 0.46\%$	$81.62 \pm 0.45\%$	$83.63 \pm 0.47\%$
20%	NNP	$76.67 \pm 0.44\%$	$82.23 \pm 0.47\%$	$75.57 \pm 0.45\%$	$80.91 \pm 0.43\%$
	RNNP (Ours)	$78.98 \pm 0.42\%$	$83.68 \pm 0.43\%$	$76.72 \pm 0.42\%$	$81.69 \pm 0.46\%$
40%	NNP	$65.07 \pm 0.58\%$	$74.37 \pm 0.60\%$	$64.63 \pm 0.55\%$	$73.27 \pm 0.57\%$
	RNNP (Ours)	$70.97 \pm 0.57\%$	$79.72 \pm 0.54\%$	$69.02 \pm 0.60\%$	$77.52 \pm 0.58\%$

Table 2: Performance of FEAT and ProtoNet with NNP/RNNP on the tiered-ImageNet dataset over 5/10-shot 5-way episodes with 0%, 20%, 40% label corruption.

Corruption Rate	Testing Method	FEAT (CVPR'20)		ProtoNet (NIPS'17)	
		5-shot	10-shot	5-shot	10-shot
0%	NNP	$84.79 \pm 0.16\%$	$87.72 \pm 0.46\%$	$84.03 \pm 0.16\%$	$87.25 \pm 0.45\%$
	RNNP (Ours)	$85.57 \pm 0.45\%$	$88.35 \pm 0.43\%$	$84.77 \pm 0.45\%$	$87.25 \pm 0.47\%$
20%	NNP	$80.07 \pm 0.43\%$	$85.35 \pm 0.45\%$	$80.23 \pm 0.44\%$	$84.95 \pm 0.42\%$
	RNNP (Ours)	$81.69 \pm 0.44\%$	$86.52 \pm 0.43\%$	$81.25 \pm 0.46\%$	$85.79 \pm 0.45\%$
40%	NNP	$69.57 \pm 0.58\%$	$79.17 \pm 0.55\%$	$69.08 \pm 0.54\%$	$78.61 \pm 0.59\%$
	RNNP (Ours)	$73.04 \pm 0.59\%$	$84.55 \pm 0.54\%$	$72.35 \pm 0.58\%$	$82.54 \pm 0.55\%$

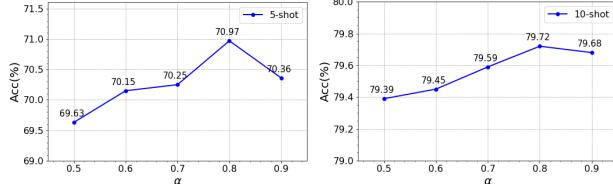


Figure 2: Performance of FEAT with RNNP on the mini-ImageNet dataset over 5/10-shot 5-way episodes with 40% label corruption for different values of α .

our method RNNP replaces the testing process of FEAT and ProtoNet, their performances improve significantly. Specifically, for FEAT, using RNNP improves the performance in the 20% label corruption setting by absolute margins of 2.31% and 1.45% for 5-shot and 10-shot 5-way episodes, respectively. The performance in the 40% label corruption setting improves by absolute margins of 5.9% and 5.35% for 5-shot and 10-shot 5-way episodes, respectively. As the level of corruption increases, the performance improvement due to our method also increases. Even when the label corruption rate is 0%, our method does not have any side effect on the performance. In fact, it increases the performance in some cases.

4.4. tiered-ImageNet Results

Table 2, reports the results for few-shot classification on 5-shot 5-way episodes of tiered-ImageNet with various levels of corruption. For 20% label corruption, the performance of FEAT drops by absolute margins of 4.72% and 2.37% for 5-shot and 10-shot 5-way episodes, respectively. Similarly, for 20% label corruption, the performance of ProtoNet drops by absolute margins of 3.8% and 2.3% for 5-shot and 10-shot 5-way episodes, respectively. When our

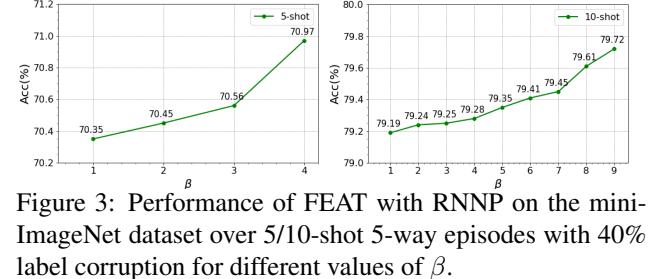


Figure 3: Performance of FEAT with RNNP on the mini-ImageNet dataset over 5/10-shot 5-way episodes with 40% label corruption for different values of β .

method RNNP replaces the testing process of FEAT and ProtoNet, their performances improve significantly. Specifically, for FEAT, the performance in the 40% label corruption setting improves by absolute margins of 3.47% and 5.38% for 5-shot and 10-shot 5-way episodes, respectively.

5. Ablation

5.1. Value of α

Fig. 2 depicts the performance of FEAT with our proposed RNNP for different values of α . Since we combine the support features using α and $1 - \alpha$, therefore, we perform ablation experiments with α values from 0.5 to 0.9. We observe the best results for $\alpha = 0.8$ for both 5-shot 5-way and 10-shot 5-way episodes. We use this value of α for all our experiments.

5.2. Value of β

Fig. 3 depicts the performance of FEAT with our proposed RNNP for different values of β . We observe the best results for $\beta = 4$ and $\beta = 9$ in the 5-shot and 10-shot 5-way episodes, respectively, which are also the maximum possible value for β in the respective cases.

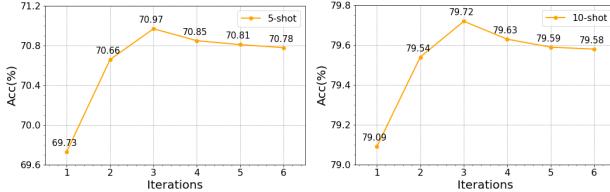


Figure 4: Performance of FEAT with RNNP on the mini-ImageNet dataset over 5/10-shot 5-way episodes with 40% label corruption for different number of iterations of the clustering process.

Table 3: Performance of FEAT with RNNP on the mini-ImageNet dataset over 5/10-shot 5-way episodes with 40% label corruption using hard/soft k-means.

K-means	FEAT	
	5-shot	10-shot
Hard	$70.63 \pm 0.55\%$	$79.23 \pm 0.55\%$
Soft	$70.97 \pm 0.57\%$	$79.72 \pm 0.54\%$

5.3. Number of Iterations

Fig. 4 depicts the performance of FEAT with our proposed RNNP for different numbers of iterations of the clustering process used in our method. We observe the best results for 3 iterations of our process for both 5-shot and 10-shot settings.

5.4. Labeled vs. Unlabeled Hybrid Features

In our method, we keep the hybrid features as unlabeled. The hybrid features may not necessarily belong to the class of the parent features since classification boundaries are usually non-linear, and in such cases, points that lie between 2 features from the same class may not necessarily belong to that class. Further, because of the presence of noisy labels, the hybrid features may have been created by combining features of two different classes. To validate this point, we performed an experiment using the FEAT w/ RNNP model with 40% label corruption, assuming that the hybrid features belonged to the class of the parent features and directly updated the class prototypes without using soft k-means. The resulting model performed similarly to the FEAT model without RNNP. This is why we keep the hybrid features as unlabeled.

5.5. Hard Clustering vs. Soft Clustering

In our experiments, we use a soft k-means algorithm for performing clustering on the features. Hard k-means clustering assigns points to a single cluster, whereas in soft k-means clustering, the points do not fully belong to any one cluster. We perform an ablation to check whether hard k-means clustering performs better than soft k-means clustering when used in our method. From Table 3, we can see that RNNP based on soft k-means clustering performs bet-

Table 4: Performance of FEAT with RNNP on the mini-ImageNet dataset over 5/10-shot 5-way episodes with 40% label corruption by creating hybrid features from the same class or from different classes.

Same/Different class	5-shot	10-shot
Different	$70.63 \pm 0.60\%$	$79.21 \pm 0.56\%$
Same	$70.97 \pm 0.57\%$	$79.72 \pm 0.54\%$

Table 5: Performance of FEAT with RNNP on the mini-ImageNet dataset over 5/10-shot 5-way episodes with 40% label corruption by using noise, hybrid features, features of new images generated using Mixup or CutMix data augmentation.

New features	5-shot	10-shot
Noise	$58.63 \pm 0.59\%$	$67.89 \pm 0.60\%$
Mixup [34] (ICLR'17)	$66.34 \pm 0.55\%$	$75.92 \pm 0.58\%$
CutMix [32] (ICCV'19)	$69.23 \pm 0.57\%$	$78.45 \pm 0.53\%$
Hybrid (Ours)	$70.97 \pm 0.57\%$	$79.72 \pm 0.54\%$

ter than RNNP based on hard k-means clustering. Soft k-means clustering allows the features to refine multiple clusters (based on their distance from the cluster centers), and this helps it to perform better than the hard k-means algorithm in this case.

5.6. Hybrid Features from Different Classes

Table 4 depicts the performance of FEAT with our proposed RNNP using hybrid features produced by combining support features from the same or different classes. The results indicate that we achieve better results when we use support features from the same class rather than from different classes for 5-shot and 10-shot episodes.

5.7. Hybrid Features vs. Noise

Table 5 depicts the performance of FEAT with our proposed RNNP using random Gaussian noise in place of hybrid features. This ablation experiment involves randomly sampling noise from a Gaussian distribution as unlabeled features instead of generating hybrid features from the support examples for our method. The results indicate that using noise significantly hurts the performance for both 5-shot and 10-shot episodes.

5.8. Effect of using CutMix and Mixup

In our method, we generate hybrid features from the support image features. Since we have very limited support examples per class, we cannot use the data generation techniques such as generative adversarial networks and variational auto encoders, which require a huge amount of data for training. We can also use data augmentation techniques such as CutMix [32] or MixUp [34] to produce hybrid images for our method. We can use the features of these hybrid images as unlabeled features for our method.

Table 6: Performance of FEAT with RNNP/JoCoR/Co-Teaching+ on the mini-ImageNet dataset over 5/10-shot 5-way episodes with 40% label corruption.

Noise Correction	5-shot	10-shot
Co-Teaching+ [31] (ICML'19)	$59.34 \pm 0.56\%$	$68.23 \pm 0.57\%$
JoCoR [25] (CVPR'20)	$59.89 \pm 0.55\%$	$68.56 \pm 0.59\%$
RNNP (Ours)	$70.97 \pm 0.57\%$	$79.72 \pm 0.54\%$

Table 5 depicts the performance of FEAT with our proposed RNNP using CutMix [32] or MixUp [34] to produce hybrid images from the support images. We use the features extracted from these hybrid images in place of the hybrid features generated by our approach. The results indicate that when we use mixup in our method, the resulting performance is significantly lower than our method’s performance using hybrid features. When we use CutMix in our method, the results are better than that of mixup but still lower than our method’s performance with hybrid features. A possible reason for this is that both these methods are used for increasing the training data (data augmentation), and in this case, we do not train the base network B .

5.9. Comparison with Noisy Label Approaches

As discussed in Sec. 2.2, there are many approaches for learning from data with noisy labels. However, these methods require large amounts of data for training. In our setting, the novel classes, which contain the corrupted labels, have very few support examples. Therefore, these methods suffer from overfitting and do not perform well. To validate this point, we experimented with JoCoR [25] and Co-Teaching+ [31] in this setting. For every novel class episode, we trained the base network B on the support examples using JoCoR and Co-Teaching+. Both methods require two networks for training, and so we make two copies of the base network for each episode. We do not carry forward the knowledge from one episode to the next, as mandated during the novel class evaluation phase. Table 6 reports the results for these experiments. The results indicate that the network cannot perform well using these methods because of the extremely limited training data.

6. Analysis

6.1. Effect of Prototype Refinement

We perform experiments to analyze how our method RNNP rectifies the incorrectly labeled support examples. Fig. 5 depicts the change in the number of support examples that have correct labels after RNNP is applied (on FEAT backbone) for four randomly chosen 5-shot 5-way episodes with 40% label corruption. The figure indicates that our method consistently corrects many incorrectly labeled support examples in each episode. Our method is able to rectify the labels of support examples and thereby refine the class prototypes. The refined prototypes help to achieve better

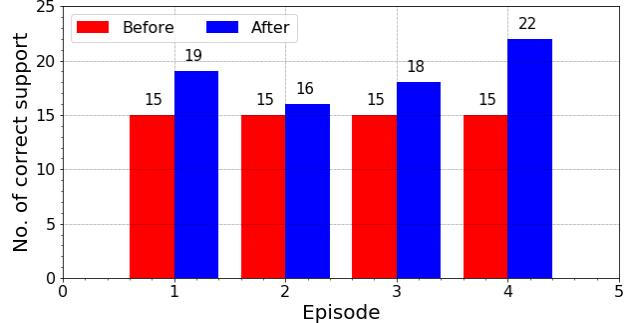


Figure 5: Effect of FEAT with RNNP on the support examples with noisy labels in the novel class 5-shot 5-way episodes of mini-ImageNet with 40% label corruption.

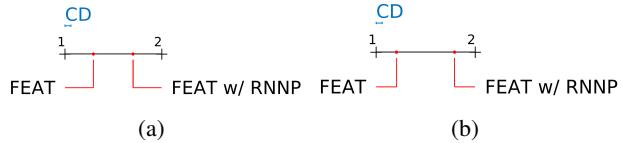


Figure 6: Statistical significance test to show that FEAT w/ RNNP is statistically different from FEAT for 5-shot 5-way novel class episodes of mini-ImageNet dataset with (a) 20% label corruption rate (b) 40% label corruption rate with a significance level of 0.05.

performance in this setting.

6.2. Statistical Significance

We studied the statistical significance [1] for FEAT with our method RNNP against FEAT. If the difference in the rank of the two methods lies within Critical Difference [1], then they are not significantly different. Fig. 6 visualizes the post hoc analysis using the CD diagram for few-shot classification with noisy support labels. We observe in Fig. 6 that the statistical difference between FEAT with our method RNNP (FEAT w/ RNNP) and FEAT is more than the Critical Difference (CD=0.062) for both 20% and 40% label corruption rates. Therefore, FEAT with RNNP is statistically different from FEAT.

7. Conclusion

We propose a novel approach called Robust Nearest Neighbor Prototype-based testing (RNNP) that makes the few-shot testing process robust to noisy labels in the novel class episodes. Our approach does not require any additional training and can replace the testing process of any few-shot learning method that uses a nearest neighbor prototype-based testing strategy. We empirically show that our method improves the performance of multiple few-shot learning methods in the presence of noisy labels. Through extensive validation experiments, we validate the different components of our method.

References

[1] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[3] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*, pages 8527–8537, 2018.

[4] Kyle Hsu, Sergey Levine, and Chelsea Finn. Unsupervised learning via meta-learning. *arXiv preprint arXiv:1810.02334*, 2018.

[5] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*, 2017.

[6] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

[7] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.

[8] Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang. Finding task-relevant features for few-shot learning by category traversal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2019.

[9] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(3):447–461, 2015.

[10] Eran Malach and Shai Shalev-Shwartz. Decoupling “when to update” from “how to update”. In *Advances in Neural Information Processing Systems*, pages 960–970, 2017.

[11] Aditya Menon, Brendan Van Rooyen, Cheng Soon Ong, and Bob Williamson. Learning from corrupted binary labels via class-probability estimation. In *International Conference on Machine Learning*, pages 125–134, 2015.

[12] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, pages 721–731, 2018.

[13] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1944–1952, 2017.

[14] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7229–7238, 2018.

[15] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

[16] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[17] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

[18] Tyler Sanderson and Clayton Scott. Class proportion estimation with application to multiclass anomaly rejection. In *Artificial Intelligence and Statistics*, pages 850–858, 2014.

[19] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[20] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[21] Eleni Triantafillou, Richard Zemel, and Raquel Urtasun. Few-shot learning through an information retrieval lens. In *Advances in Neural Information Processing Systems*, pages 2255–2265, 2017.

[22] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[23] Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019.

[24] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7278–7286, 2018.

[25] Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An. Combatting noisy labels by agreement: A joint training method with co-regularization. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020.

[26] Xiu-Shen Wei, Peng Wang, Lingqiao Liu, Chunhua Shen, and Jianxin Wu. Piecewise classifier mappings: Learning fine-grained learners for novel categories with few examples. *IEEE Transactions on Image Processing*, 28(12):6116–6125, 2019.

[27] Ziyang Wu, Yuwei Li, Lihua Guo, and Kui Jia. Parn: Position-aware relation networks for few-shot learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6659–6667, 2019.

[28] Shudong Xie, Yiqun Li, Dongyun Lin, Tin Lay Nwe, and Sheng Dong. Meta module generation for fast few-shot in-

cremental learning. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

- [29] Chen Xing, Negar Rostamzadeh, Boris Oreshkin, and Pedro O Pinheiro. Adaptive cross-modal few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4847–4857, 2019.
- [30] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
- [31] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? volume 97 of *Proceedings of Machine Learning Research*, pages 7164–7173, Long Beach, California, USA, 09–15 Jun 2019. PMLR.
- [32] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019.
- [33] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.
- [34] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.