Improvement of Few-shot Learning with random forest

Hyoungsung Kim
School of Electrical Engineering and Computer Science
hyoungsung@gist.ac.kr

1. Introduction

1.1. Topic

Training a neural network with a few data which is known as few-shot learning is one of challenges in Machine learning(ML). To solve this problem, Vinyals et al. [14] proposed matching network which are based on transfer learning. Exactly, matching network is based on metric based method and it is a branch of transfer learning. In metric based method, there are 3 datasets: support set, testing set and training set. More detain about datasets are covered in section 2.

Classification is not a problem that only belonging to neural network. For decades, there are many approaches to improve classification accuracy. Among them, ensemble learning like boosting or bootstrap aggregating(bagging) which are based on decision tree, are shown considerable improvement. For this reason, many improved methods of boosting and bagging have been proposed like random forest[1] and XGBoost [3].

In this report, we propose a method to combine metric based method and ensemble learning method in training phase without changing model. We use CNN as metric based on [13] and random forest[1] as ensemble learning method. We hypothesize 2 things. (1) what a metric based method can learn easily and what a random forest can learn easily are different. Therefore, if we combine them, then we can get a faster convergence. (2) Only using 1 metric method cannot fully use extracted features. Therefore, if metric make a decision by using not only extracted features but also result of random forest, then we can get a higher accuracy. In section 4, we show a result of our hypotheses.

1.2. Importance in visual recognition

Humans can learn with few data. For example, humans can recognize a kind of chairs even though they have seen a single picture of chair. On the contrary, machines need enough data to recognize things. It motivates few-shot learning which uses fewer data for classification.

There has been many researches for few-shot learning [7, 8, 11, 13], and common goal of these is finding a correct cluster of unlabeled input. However, we cannot expect perfect classification in few-shot learning because distance from point of unlabeled images to cluster of labeled images is not clear. For example, when there are 2 clusters which are close from unlabeled image, it is very hard to recognize which cluster is correct one.

Goals of our methods are training faster and using extracted feature as much as we can. These can be used usefully in visual recognition.

2. Related Work

Metric based few-shot learning is a branch of transfer learning based methods which are one of relevant approaches in few-shot learing research. In this paper, we propose a way to combine metric based few-shot learning and random forest in training phase. Therefore, in this section, we briefly review metric based few-shot learning and random forest.

2.1. Metric Based Few-shot learning

2.1.1 Training strategy

Goal of training strategy in metric based learning is finding the nearest cluster of unlabeled data. Exactly, labelled dataset make clusters as a result of trained module and unlabeled data makes a point as a result of modules too. The closest cluster from point is decided by metric method becomes a label of unlabeled data.

However, in few-shot learning, data is not enough therefore it cannot be expected that a label of the closest cluster from point is correct label. It means we cannot obtain remarkable classification accuracy with few data. To solve this problem, our method follows an episode based strategy which are used in [12, 14] that commonly used in metric based few-shot learning.

In episode based strategy, there are 3 datasets: training set, support set and testing set. Training set has its own label spaces which disjoint with support set and testing set. Support set and testing set share same label spaces. When there exists C classes that each class has K labelled data, it is called as C-way K-shot problem. Furthermore, when C-

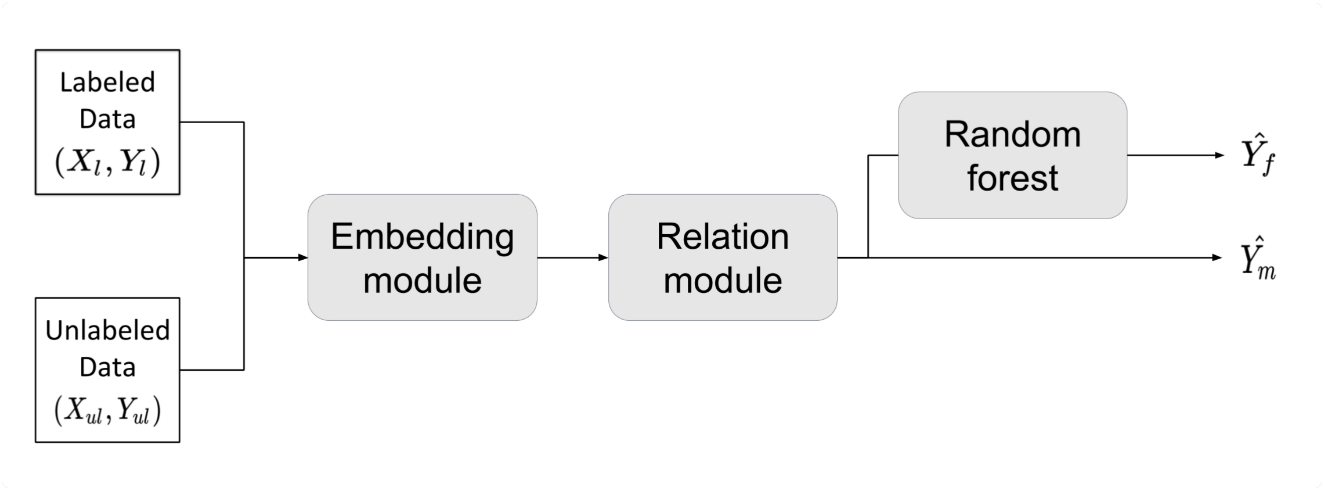

Figure 1. Model architecture of project. Random forest is trained by a result of relation module and predict a class of unlabeled data.

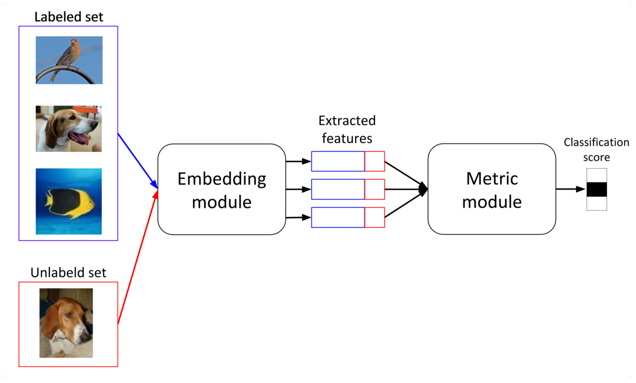

Figure 2. Basic structure of few-shot learning.

way K-shot problem is given with unlabeled data as input of network, it is called as episode.

Unfortunately, lack of data in support set, we cannot make a good classifier. So a feature extractor is trained by training set and this trained knowledge is transferred to classify unlabeled data of episode.

2.1.2 Metric learning approaches

Figure 2 shows basic structure of metric few-shot learning model. There are many ways to improve this structure. [5, 8, 12, 13, 14] changes metric module to improve classification accuracy. Specifically, [5] uses graph neural network, [8, 13] use convolutional network, [12] uses Euclidean distance and [14] uses cosine similarity as metric method. All of them commonly focus on how metric methods work. However, we propose new approach to improve metric based few-shot learning. We use 2 metric methods. One is convolutional network of relation network [13], and the other is random forest[1]. Each method has its own benefit. So to combine their benefits, we propose new objective function. Figure 3, shows a goal of metric learning. Metric is used to find the closest cluster of unlabeled images.

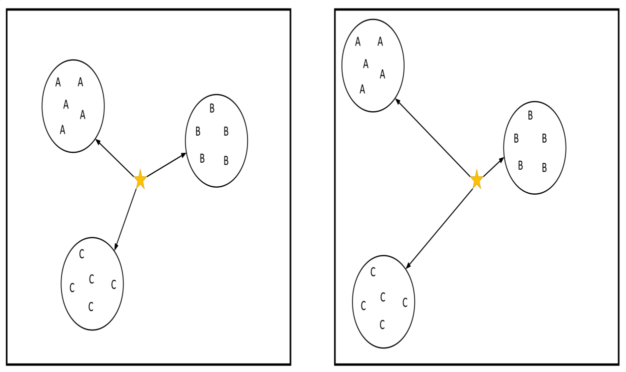

Figure 3. Goal of metric learning

2.2. Random forest

Ensemble learning is a method that uses multiple prediction algorithm to get a better result [10]. Random forest[1] is a group of decision trees which are used to predict result therefore, random forest is a one of ensemble learning methods. Random forest has two phases. One is training and the other is testing. In training phase, data with ground truth labels are given. Decision trees of random forest are trained to decrease a difference between predicted labels of given data and ground truth labels. In test phase, when unknown data is given, each decision tree predict label. After finishing decision of trees, results are combined to predict random forest result.

Random forest has stability and robustness with a small training set [6]. In this project, we train random forest as metric methods and use result to improve classification accuracy.

3. Implementing the state of the art

There are a various way to improve metric based few-shot learning like we have mentioned in section 2.1.2. In this project, we focus on a way that can be incorporated above approaches. As a baseline, we use relation network [13]. It uses convolutional network as metric module in figure 2. [5] use graph convolution neural network as metric module instead of CNN. It shows bare improvement than relation network, but we did not consider this paper because there is an issue in author's github about reproducing result of paper.

3.1. How to reproduce the results in the paper

In relation network, there are two modules. One is embedding module and the other is relation module At first time I was curious that to improve classification accuracy, which module is optimized?

We carried out 2 experiments to know the answer of question. One is not backpropagate a result embedding module like original paper and the other is backpropagate the result of embedding module. We calculated loss of embedding module from each experiment. Difference of loss between first experiment and second experiment was less than 0.01. Furthermore, second one's loss is saturated but first one is not.

The difference of loss was not significant regardless of backpropagation. However, relation modules' loss are different. First experiment's loss decreased faster than second's one. As a result, first experiment shows better classification performance. By these experiments, we confirmed that to get a better result, we have to consider relation module which is used as a metric module of few-shot learning.

3.2. Improvement

Our source code of this project is based on public code. But we fixed depreciated functions and changed to cross entropy instead of mean square error for faster convergence. We use scikit-learn library for random forest. Result of random forest highly depend on hyperparameters(e.g., the number of decision tress, max depth). So currently we are progressing experiments to find optimal hyperparameters of random forest.

3.3. Experimental Results

In figure 4, it shows results of omniglot dataset with various hyperparameters. Result of original paper is the red line that converges faster than any other line. (Technically, Tensorboard uses same red color for 2 results, we will fix it in final report) What we want to check from omniglot experiments are that proposed methods can reach to the same result of original paper and after specific accuracy, which algorithm increase faster to maximum accuracy. It is natural that original paper converges faster than our method because it optimized only one term of loss function. However, we are expecting that after decision trees of random forest are trained well, our methods may increase better than original paper method.

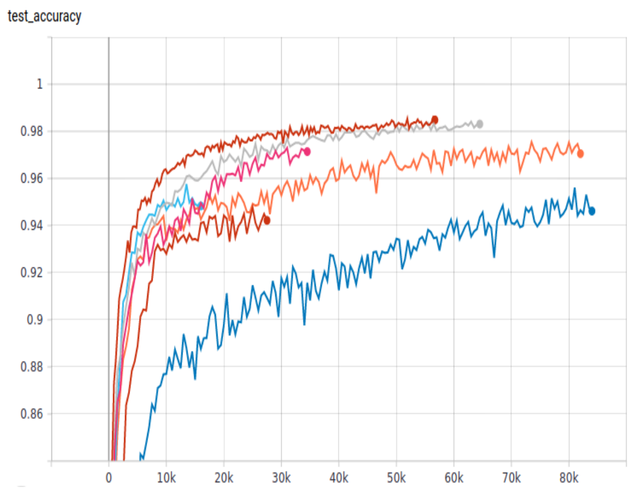

Figure 4. Currently, we are testing to find optimal λ and hyperparameters.

A purpose of our method is getting better result with complex data(e.g., miniimagenet). After finishing hyperparameters tuning, we will carry out experiments with miniimagenet.

Up to here is for the SOTA report.

4. Your New Approach

To improve result of [13], we add random forest module in the end of relation module. In figure 1, we show model architecture. We define new objective function with results of random forest($\hat{Y}_f$) and relation module($\hat{Y}_m$).

$$\mathcal{L} = \lambda \mathcal{L}_f(\hat{Y}_m, \hat{Y}_f) + \mathcal{L}_m(\hat{Y}_m, Y_{ul}) \tag{1}$$

$\mathcal{L}_f$ calculates a difference between results of relation module and results of random forest. A purpose of this term is let neural network know that random forest knows but neural network does not know. We know that results of random forest are not ground truth, so we regulate this term with λ. And $\mathcal{L}_m$ calculate a difference between result of relation module and ground truth.

4.1. Implementation detail

We use 10 queries for training, and tested 1, 5, 10, 15 queries. It means, in 5-way 5-shot with 15 queries, 5 $\times$ 5 + 15 $\times$ 5 = 100 images are used for training. And we used 1, 5, 10, 15 queries for test. Also, we use embedding that used in [2, 12, 13, 14] for pair comparison with references.

And we use convolutional network as metric which is used in [13].

For random forest, we use scikit-learn library. To train random forest with mini-batch, the number of decision tree have to be increased. Therefore, we start with 100 decision trees and increase 1 decision tree with every 100 episode. But scikit-learn library only works with CPU. So we could not use GPU for training. Because of this reason, we restricted increase of decision tree by changing phase. After 30,000 episodes, decision tree increase every 1,000 episodes. This hyperparameter is decided by heuristically. In figure 5, it shows a result of miniimagenet test with different hyperparameters. When we test in omniglot, it converges too fast. Therefore, we test with miniimagenet. Blue means change phase after 20,000, pink means change phase after 40,000 and red means change phase after 30,000. Exponential moving average is used for smoothing in figure 5.

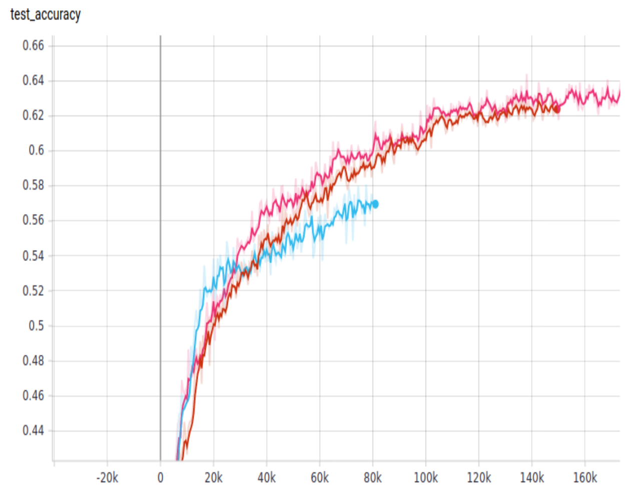

Figure 5. Blue shows when we change phase after 20,000, pink shows when we change phase 40,000 and red shows when we change phase after 30,000

In training phase, training a bunch of decision tree need a lot of computing power. Exactly, training time of decision tree increase exponentially. First 100K training takes about 6 hours. But training from 700K to 800K takes more than 12 hours. So we stopped to train in near 820K.

For test, we set λ of (1) as 0.7. This is also decided by heuristically. In figure 6, we show a result of omniglot with different λ. When λ is 0.7 it shows the best result. This is shown as light blue color, when λ is 0, it shows the worst result as orange color. And green shows result of λ is 0.3, gray shows result λ is 0.8

We use Adam with initial learning rate 1×10^{-3} and reduces it every 100,000 episdes.

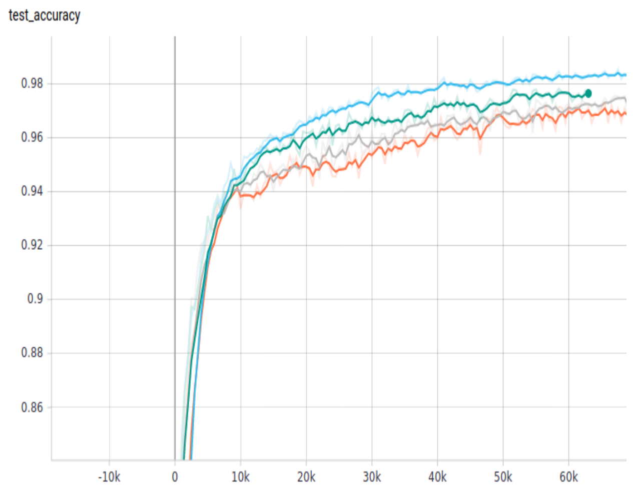

Figure 6. Omniglot result with difference λ

Model	5-way 5-shot Acc
Matching Nets [14]	$55.31 \pm 0.73\%$
Meta-Learn LSTM [9]	$60.61 \pm 0.71\%$
MAML [4]	$63.11 \pm 0.92\%$
Prototypical Nets [12]	$68.20 \pm 0.66\%$
Relation Net [13]	$65.32 \pm 0.70\%$
Ours	65.25 ± 0.64 %

Table 1. 5-way 5-shot result of miniimagenet. Prototypical Nets uses 20-way 15-queries data in training phase. But we use 5-way 10-queries data in training phase.

Number of queries	Acc
1 queries	$68.47 \pm 1.73\%$
5 queries	$65.39 \pm 0.93\%$
10 queries	$65.26 \pm 0.75\%$
15 queries	$65.25 \pm 0.64\%$

Table 2. Result by queires.

4.2. Experimental Results

Our result shows in table 1. We followed the split introduced by [9]. There exists 64 classes for training, 16 classes for validation and 20 classes for testing. In this result, we test 600 episodes with 15 queries. In the table 1, Prototypical network [12] uses 20 way 15 queries in training phase. So it shows the highest result in table 1. Query means the number of unlabeled images per class. For example, if 5 queries are used, it means 25 unlabeled images are given in 1 episode.

In table 2, it shows result by different queries in test phase. In papers which are used for comparison, authors mentions how they used training set. But, in test, few paper did not mention detail how many queries they used for train. So we tested with 4 different queries. When query is 1, it shows the best result. However, it has the highest

95 % confidence interval. When queries is 5, 10 and 15, it shows similar results. Testing with 10, 15 queries show slight less result with original paper [13]. But test with 15 queries show better confidence interval than origin paper. It means variance of our method is 10 % less than original paper.

5. Conclusion

In this project, we propose a way to combine random forest and convolutional network. In [13], they mention that they use 15 queries for test. With same condition, our method shows similar result even though we could not train enoug. Also, we show result with 95% confidential interval. By this result, we can know that our method get 10% less variance. It means random forest helps to decrease variance.

5.1. Discussion

In this project, we took a lot of time to tune hyperparameters. Especially, tuning increase phase of decision tree take almost time. Because, currently, there is no commonly used random forest library except scikit-learn. But like we've mentioned, this library only works with CPU. So to combine random forest with neural network, implementation of random forest which can work with GPU is necessary.

In section 4.1, we shows experiment to decide hyperparameters of random forest. As shown in figure 4 5 6, accuracy of random forest is changed significantly by hyperparameters. In document of scikit-learn libraty, there are more than 10 hyperparameters to tune random forest. Therefore, using random forest need a lot of test to get a best hyper parameters. Also, these hyperparameters have to be changed depend on computing power because of trade-off. We can get a good result when we increase the number of decision tree a lot but it needs a lot of computing power and takes a lot of time for training trees. Therefore, user has to decide how much computing power can use for random forest. For example, in result of 15 queires in table 1, we show better variance than original paper. But because of training trees, it takes much more time to train than original paper.

Up to here is for the final report.

References

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[3] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 785–794, 2016.

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

[5] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*, 2018.

[6] Te Han, Dongxiang Jiang, Qi Zhao, Lei Wang, and Kai Yin. Comparison of random forest, artificial neural networks and support vector machine for intelligent diagnosis of rotating machinery. *Transactions of the Institute of Measurement and Control*, 40(8):2681–2693, 2018.

[7] Lukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. 2017.

[8] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *In ICML workshop*, 2015.

[9] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[10] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8, 2018.

[11] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1842–1850, 2016.

[12] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30*, pages 4077–4087. Curran Associates, Inc., 2017.

[13] 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*, 2018.

[14] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29*, pages 3630–3638. 2016.