Label Propagation for Few-Shot Learning

Ming Lei
Department of Computer Science
Carleton University
Ottawa, ON K1S 5B6
minglei@cmail.carleton.ca

Yanan Mao
Department of Computer Science
Carleton University
Ottawa, ON K1S 5B6
yananmao@cmail.carleton.ca

Abstract

Few-shot learning is an eye-catching area in images classification. However, correctly classifying the images based on a few examples is not hard for humans but extremely difficult for machines. In this project, based on the matching network, we propose to replace the nearest neighbor classifier with label propagation. More fine-tuning strategies will be shown in the report.

Index Terms —— few-shot learning, label propagation, transfer learning

1 Introduction

It is easily observed that people who learn how to program in the C language will also be faster learners of Java, C++, and python languages. Kids are more likely to recognize a new species of animals based on the features on different animals and combine the features based on the semantic description. When it comes to the machine learning area, transfer learning is the one that aims to build a framework like that.

However, transfer learning required a deep learning size of dataset, which might be thousands of well-labeled data. However, there is an increasing number of categories that we cannot collect enough data for each of them. Whether the data is expensive to get access to, such as diagnosing images, or well-labeled the data we have, which is impossible except everyone is an expert.

A specific case in transfer learning is few-shot learning, which will classify the images based on a few examples of each class in the training stage and adjust the classifier to have the ability to distinguish the differences. Few-shot learning is an exciting topic aiming to close the gap between machines and humans in learning concepts from a few labeled instances.

In general, we have four types of metric-based deep learning networks on few-shot learning:

- Siamese neural network
- Matching network
- Prototypical network
- Relation network

The main idea of the siamese neural network is to output binary conclusions. Koch et al.[4] proposed the base idea of image classification. There are two identical neural networks in the system. Input one image for each network, the corresponding feature vectors could be calculated, and the feature vectors distance will decide whether those two input images are similar or not. However, the network could only do pairs. The training amount would be the square level of our dataset, which is enormous and easily overtraining the network.

The attention mechanism is the crucial component of the matching network. Vinyals et al.[6] introduced the matching network, which uses the attention mechanism to predict the query set based on comparing the feature embedding of the support set. Then, the neural network extracts the feature from images and compares the similarities. For example, we could normalize the distance and get the attention matrix or weight matrix. The weighted nearest-neighbor classifier is used after the feature matrix is collected.

Prototypical networks are based on the idea of points cluster [5]. In each cluster, a single prototype will represent the features of the cluster in embedding space. And they are then finding the nearest class prototype. Furthermore, their paper also points out that, Euclidean distance has better performance than cosine distance.

The relation network [3] is to combines the features. The embedding features of the query set will be attached after each support set image feature vector. During the training, relation scores could be obtained and whether two images are related. Moreover, the class with the maximum value in relation score matric will be the class of query image.

In our project, we try to propose a novel idea to improve the performance of the matching network.

While dealing with an image classification task, however, visual data contains additional information like spatial layout, which is unlabeled by humans but the machine itself. K-NN method only shows the human-labeled relationship in each image. Furthermore, the distance between each test point and the training set must be calculated. When the training set or image features are large, the calculation is overloaded and time-consuming. K-NN is also a sample imbalance method that produces low prediction accuracy for low-resource categories. Besides, K-NN is not based on learning or lower down gradient, and slow prediction speed will be given. Even if we could set up weight parameters to improve the performance of the K-NN network, setting the values of parameters is a tricky question.

In this case, we aim to use label propagation (LP) [1] to pseudo-labels for unlabeled data, which will train the classifier. LP is a graph-based method and works in embedding obtained by classification networks. The network will be trained from labeled and pseudo-labeled data at the beginning of the project. Then, embeddings of the network will combine the previous network with training the new layer of the nearest neighbor graph. Each dataset will be split into labeled training data and unlabelled training data. The labels will be updated by learning from labels and pseudo-labels. Finally, output the accuracy from the label probability distributions, and build predict functions.

2 Related Work

Vinyals et al. [6] mentions that the learning of parametric methods is slow, requiring many weight updates using stochastic gradient descent and based on large datasets while the non-parametric methods such as nearest-neighbor don't require any training but their performance depends on the chosen distance metric that how to compute the difference of two images. Inspired by Vinyals et al.[6], we employ the idea of combining the advantages of both parametric and non-parametric methods to reduce the number of trainable parameters as well as improve the model performance.

The authors start their idea on the non-parametric kernel, the attention kernel that softmax over cosine similarities between the full context embedding of query and support set samples. In the Matching Networks, a fully differentiable nearest neighbors classifier is trained. It first embeds all samples with a simple neural network, takes the original embedding as input to calculate full context embedding (FCE) with a bi-LSTM [8] model on support set and query samples, then calculates the cosine distance between the embeddings of query samples and support sets to make a class prediction by taking the weighted average of the support sets with normalized distance using an attention kernel.

Label propagation algorithm was introduced in reference [9]. As we know, the weight matrix is the essential parameter in the LP algorithm, which decides the surrounding neighbors and values the label propagation. Liu et al.[9] chose to obtain the proper neighbor by generating the weight matrix based on similarities. Both the embedding feature vectors of the support set and the query set are components of the example-wise length-scale matrix. In this case, the authors use Euclidean distance as their distance measurement method. Every two samples will have a responded relationship weight value, and all of the weight values form the weight matrix. Assume there are K classes, then the top K values in each row of the matrix will be reserved in normalized values, while others are overwritten with value 0. The sample with the highest score will be the most reliable reference in label propagating and vice versa.

After the dig deeper of reference [6] [9], we could find out that relation network has better performance than matching network. And the different classifiers might be the reason why the relation network has a 9% improvement in nimi Image dataset. Thus, we aim to change the classifier to figure out the answers in this project.

3 Method

3.1 Few-shot Learning Concept

There is a gap between machine and human in the task of image classification. For example, a child can easily recognizes an apple even if he or she has seen one single apple image. In this way, inspired by the human, a few-shot learning model aims to close this gap between machine and human.

Few-shot learning is a solution of making predictions based on a limited number of samples. The goal of training the few-shot network is to know the similarity and difference between samples. Unlike the traditional supervised learning approaches that using same training and test classes, we employ the few-shot model to recognize sign letter classes that are never seen during training and It is supposed to learn only few samples in training phase. Compare to the supervised learning, the few-shot learning model separates data samples into the support set and query set.

Specifically, the support set includes training data with labeled samples and the query data set includes the unlabeled samples that are be predicted by our model. There are two important terminologies in few-show learning: K-way, N-shot. K-way means the support set has K classes, while N-shot means every class in support set has N samples. In general, with the increasing number of shots, the predict accuracy improves.

3.2 Long Short Term Memory networks (LSTM)

Some neural networks have been used to classify images. The advantage of using neural networks is that they learn the most important classification features. However, they require more time and many weight update using stochastic gradient descent when it comes with a large amount samples.

A special neural network family is based on Recurrent Neural Networks (RNN). As humans who will develop new skills based on the old ones, and never understand brand new information from scratch. This type of network also has a memory system to help react faster and precisely.

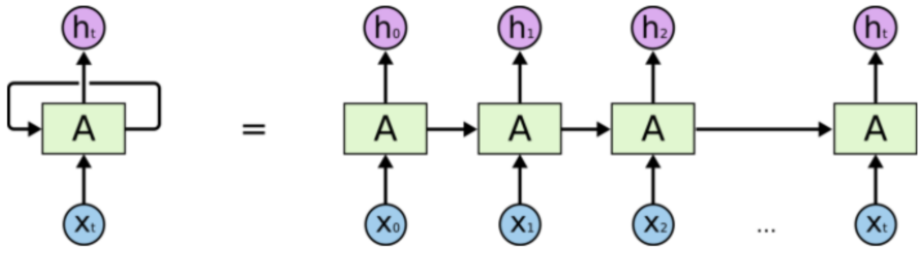


Figure 1: RNN

As shown in Figure 1, the network has a loop, which serves the function as a memory that allows previous information to play apart in the new loop, and parameters are shared. The chain-like network indicts that this type of network has the advantage of dealing with sequences but irrelevant data, such as language or images. However, for long sequences, RNN gets into the vanishing gradient. But, LSTM, as a very special kind of RNN, could learn in a long term.

In this project, we are based on matching network [6], which uses LSTM as their model to embedding images features. As shown in Figure 2, the structure of LSTM are similar to RNN. At the

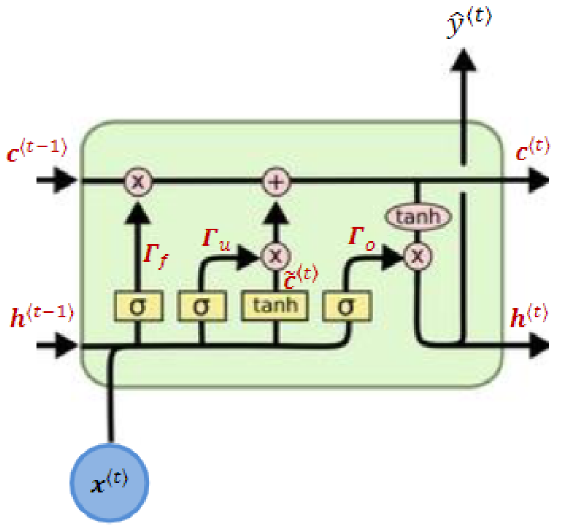

$\hat{y}^{\langle t \rangle}$
$c^{\langle t-1 \rangle}$
$c^{\langle t \rangle}$
Γ_f
Γ_u
Γ_o
tanh
$\tilde{c}^{\langle t \rangle}$
$h^{\langle t-1 \rangle}$
σ
σ
tanh
σ
$h^{\langle t \rangle}$
$x^{\langle t \rangle}$

Figure 2: LSTM

meantime, LSTM [7] is governed by the following update equation:

$$\begin{aligned}
C^{(t)} &= tanh(W_c[h^{(t-1)}, x^{(t))}] + b_c) \\
T_u &= \sigma(W_u[h^{(t-1)}, x^{(t)}] + b_u) \\
T_f &= \sigma(W_f[h^{(t-1)}, x^{(t)}] + b_f) \\
T_o &= \sigma(W_o[h^{(t-1)}, x^{(t)}] + b_o) \\
C^{(t)} &= \Gamma_u C^{(t)} + \Gamma_f C^{(t+1)} \\
h^{(t)} &= \Gamma_o tanh(C^{(t)})
\end{aligned}$$

U stands for update gate, which will pass the information that we want to transfer to cell state. F represents the information we want to keep from the previous cell memory, and o are the information from the cell state that we need to make the next prediction.

3.3 K-Nearest Neighbor

The non-parametric models such as K-Nearest Neighbor (KNN) doesn't require any training. For example, given two image embedding, KNN calculates the approximate distances between them and then assign the unknown image to the class of its K-nearest neighbors. However, this method makes test phase slower since the model structure is determined from the data it self and it's performance depends on the chosen metric, e.g., the similarity function to calculate the distance between data points. We will further explore the chosen metrics for our model.

KNN algorithm, the concept is also shown in Figure 3, assumes that those samples that share high similarity should be in dense clusters, and low similarities samples are far from each other. K is the number of the nearest neighbours we want, and usually, the value of K would be an odd number. In order to find the closest samples, calculating distance is necessary. But, at the same time, KNN is also a lazy learning method, as mentioned before, the training time will be exponentially increased. When we increase the dimension, the overfitting problem appears. Thus LP is what we proposed in this project to replace the KNN method.

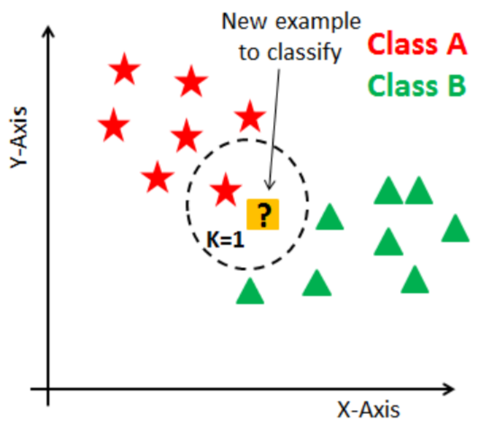


Figure 3: KNN

3.4 Matching Networks

The attention function is given by:

$$a(\hat{x}, x_i) = \frac{e^{c(f(\hat{x}),g(x_i))}}{\sum_{j=1}^{k} e^{c(f(\hat{x}),g(x_i))}}$$

In above equation, $\hat{x}$ represents query images, x_i represents images in support set, c is cosine similarity and f and g are full context embedding functions.This attention kernel is the key part of the non-parametric method. The k-nearest neighbor method uses cosine similarity function to calculate the distance in embedding space of query and support sets samples that given from the embedding function. The embedding function they used is a simple CNN which is differentiable. This means it can be used to fit the whole model end-to-end with typical methods such as stochastic gradient descent.

3.5 Label Propagation

LP is a graph-based algorithm. Instead of calculating the distance between query set and support set, LP aims to predict unlabeled samples based on all the labels. The structure of LP is shown as Figure 4.

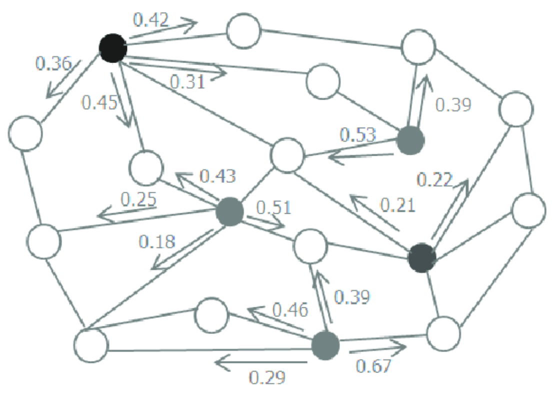


Figure 4: Label Propagation

A vivid example from daily life is that we are most likely surrounded by someone who shares the same characters. However, the wrong predicted labels will affect the accuracy, and sparsely connected regions in the graph may have trouble propagating. Still, LP will reach convergence when the unlabeled sample has the majority label of its surrounded neighbors. One of the best features of LP is that the labeling process could be narrowed down to only the related samples and generate the solution.

The weight matrix for LP is the key point of getting a high predict accuracy, which we borrowed the idea from reference [9], the example-wise length-scale parameter is defined as:

$$W_{i,j} = exp(-\frac{1}{2}d(\frac{f_{\varphi(x_i)}}{\sigma_i}, \frac{f_{\varphi(x_j)}}{\sigma_j}))$$

Where $X_i \epsilon$ S $\cup Q$, φ is generat from embedding features. After we generate the W matric, we will select the top K samples that are related to the query set, and we get Y.

The label propagation formula is thus, shown as below:

$$F_{t+1} = \alpha S F_t + (1 - \alpha) Y$$

4 Experiments

Firstly, We reproduce the Matching Network model as the baseline model to evaluate the few-shot classification task on two data sets: Omniglot and American Sign Language. After that, we fit the label propagation approach to the Matching Network model to evaluate the same data sets. The results from Matching Networks model as the baseline that are used to compare our label propagation approach. Although the miniImageNet is also a popular few-shot learning benchmarks, we skip it due to our limited GPU performance and time.

4.1 Datasets

4.1.1 Omniglot

The Omniglot data set [2] is designed for developing more human-like learning algorithms. It contains 1,623 different handwritten characters collected from 50 different alphabets.Each of the 1623 characters was drawn online via Amazon's Mechanical Turk by 20 different people and each image has 105x105 pixels.

4.1.2 American Sign Language

The ASL Alphabet Test data set is our extend data set that retrieved from Kaggle website [11]. This data set is not large, it contains 870 images, each image contains a had making the shape of an ASL letter. There are 29 classes including letter A-Z delete, space and nothing in total. Each class contains 30 200x200 pixel images.

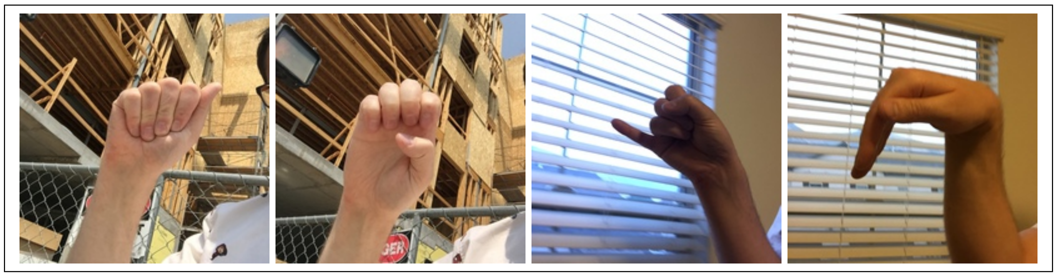


Figure 5: Samples for ASL sign A, E, J and Delete.

4.2 Data Preparation

In order to fairly compare with Matching Network models results, we following the same image processing procedures. Firstly, all images from Omniglot are resized to 28x28 pixels (see Figure 6) and it reduces training time as well. Furthermore, all images are rotated in multiples of 90 degrees randomly since this way can improve the model performance, this is proved method that is introduced by Santoro et al.[10]. Lastly, we use the 1200 characters (total 4800 characters including rotations) for training and remaining characters for test. Similarly, all images from American Sign Language are resized to 80x80 pixels and randomly rotated in multiples of 90 degrees as well in order to increase number of classes. Finally, we have 2400 samples in training set and 1080 samples in test after image rotation.

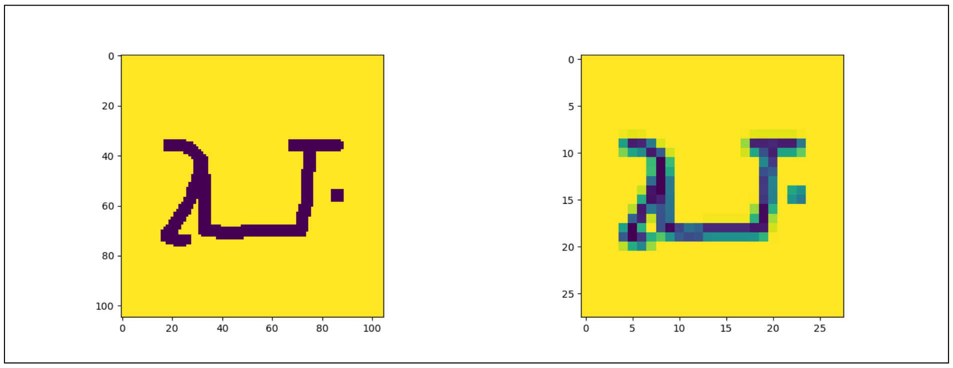


Figure 6: Original Image (Left) vs. Resized Image (Right)

4.3 Images Embedding

We use simple and widely-used Convolutional Neural Network (CNN) as the feature embedding function. It is ued to extract features of an input x_i, where $f(x_i)$ is the embedding images in the support set S and query set Q. We employ the same architecture used in Matching Network so that we can provide more fair comparisons in the experiments. The CNN has 4 blocks and two fully=connected layers, each block contains 3x3 convolution, batch normalization, ReLU activation followed by 2x2 max pooling.The number of filter in each convolution block is 64. The output dimension of embedded images is 1x1x64.

4.4 Distances Metrics Choice

Distance metrics is a key part of KNN algorithm and it aims to optimize the transferable embedding. Matching Networks (Vinyals et al., 2016) produce a weighted nearest neighbor classifier given the support set to adjust feature embedding according to the performance on the query set. It compute the distance between small number of input images and target images.

To have better performance on different datasets, we need to choose the proper distance measurements accordingly. Euclidean distance and Cosine distance are what we prepare to use in the project, and both of the distance output the value bounce between 0 and 1.

Cosine Distance: Output the similarity between two vectors. This method measures the cosine angle between two vectors and determines the directions of two vectors are same or not.

Pairwise Euclidean Distance: The default metric in SKlearn library of K-Nearest Neighbor. Euclidean distance is also called L2 distance, it compute the the value of straight line distance between two objects. In our project, given vectors x and y, it take a square root of the sum of squared differences in their elements.This method measures the distance between each pair of samples. Leavers out some of the information in the dataset, but dramatically reduce the table length of pairwise distance.

Our proposed label propagation is similar to this approach and we explore different similarity functions. There are two distance metrics are explored:

- Cosine distance:

$$\frac{x_1 \cdot x_2}{max(||x_1||_2 \cdot ||x_2||_2, \epsilon)}$$

- Pairwise Euclidean distance:

$$(\sum_{i=1}^{n} |y_i - x_i|)^2)^{1/2}$$

The results of using different distance metrics in Matching Netowork model shows on Table 1.

Table 1: Similarity Function Comparison (5-way 1-shot)

Model	Cosine	Euclidean
Omniglot (Matching Net)	97.46%	82.03%
ASL (Matching Net)	69.53%	64.84%

From above table, we can conclude that Cosine distance is outperform than Pairwise Euclidean distance in Matching Network model. In other word, Euclidean distance converge faster.

4.5 Loss Computation

The goal of this step is to compute the classification loss between the prediction of the embedding of support set and query set through label propagation and the ground-truth. The cross-entropy function is used to determine the loss and is given by:

$$L(\theta) = -\frac{1}{n}\sum_{i}^{n}[\,y_i\log(p_i) + (1 - y_i)\log(1 - p_i)] = -\frac{1}{n}\sum_{i}^{n}\sum_{j}^{m} y_{ij}\log(p_{ij})$$

where y_i is the ground-truth label of input image and p_i is the predicted label. The index j in y_{ij} indicates the $j_t h$ predicted label from the label propagation.

4.6 Results

Table 2: Accuracies on Omniglot Data sets

Model	5-way 1-shot	5-way 5-shot
Matching Netorks (Vinyals et al.)	98.1%	98.9%
Matching Netorks (Reproduced)	95.12%	98.63%
Label Propagation (Ours)	23.6%	25.7%

4.6.1 Reproduced Matching Network Model

We evaluate our reproduced Matching Networks Model on Omniglot data set and the result is very close to the original paper proposed by Vinyals et al. Experimental results are shown in Table 2 and Figure 7. There is only 3% difference in 5-way 1-shot accuracy and same performance in 5-way 5-shot accuracy.

Figure 7: Validation Accuracy and Loss on Omniglot Data Set (5-way 1 shot)

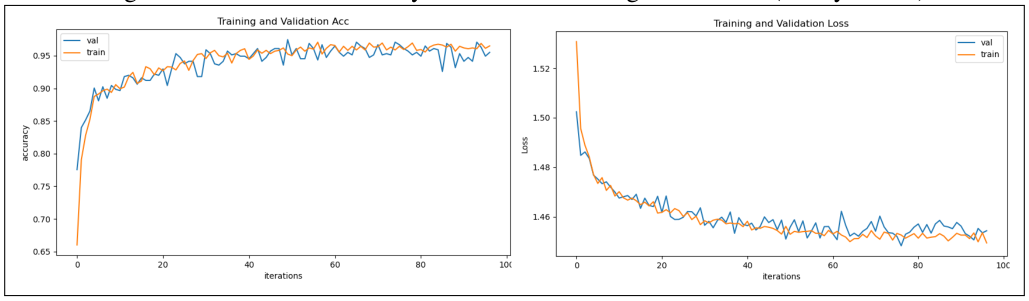
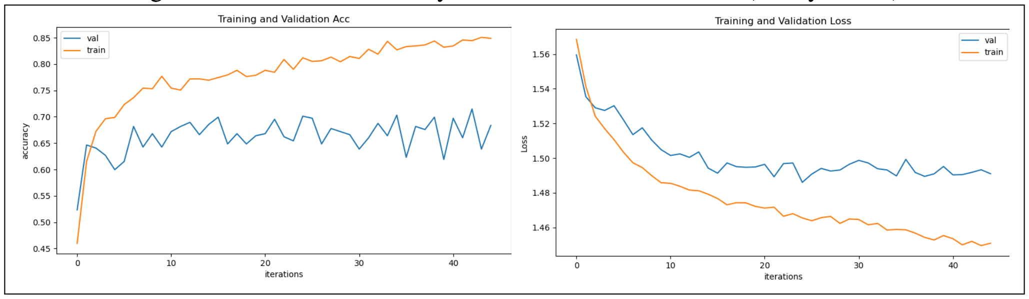


Table 3: Accuracies on ASL Data sets

Model	5-way 1-shot	5-way 5-shot
Matching Netorks (Reproduced)	69.53%	74.22%

From Table 3, the reproduced Matching Networks model shows 69.53% and 74.22% accuracies in 5-way 1-shot and 5-way 5-way tasks. The 5-shot task provide 5 images samples for each class in the support set while 1-shot task provide only 1 sample for each class. As we expected, the 5-shot learning is always outperform than 1-shot learning model.

Figure 8: Validation Accuracy and Loss on ASL Data Set (5-way 1 shot)



4.6.2 Label Propagation on Few-shot Learning

We successfully apply the idea of Label Propagation to the Mathing Network model. However, the results of our proposed label propagation method are disappointed, the accuracies are only 23.6% in 5-way 1-shot and 25.7% in 5-way 5-shot tasks.

We assume the reason that cause this result is that:

- The imput of LSTM is not just embedding features.
- Minor details in the networks still wait to modified.
- The whole idea might not suitable for image classification.

The Matching Networks model uses LSTM to obtain the full context embedding of support set images and query images. The input of LSTM is given by a simple CNN model which is used to extract image features. We think the output LSTM could result in low accuracy when we compute the distance of the full context embedding from LSTM in Label Propagation step. We believe LSTM is more appropriate for Language Modeling field and it could be remove from the model when the task is image classification. Furthermore, it worth to attempt more few-shot learning models with LP method. Due to time limitation and GPU resource, we are not able to explore more other few-shot learning models, but we believe our method could perform better.

5 Conclusion

We reproduce the Matching Network model, analyze and test different distance metrics and apply Label Propagation idea to the few-shot learning model. The project aims to improve the performance of the matching network. Compared to the relation network, the matching network has a good memory of samples feature and reduces the amount of the trainable parameters.

The similarity would be two methods all labeled unlabeled samples based on the neighbors. However, instead of KNN that only counts on each class's numbers in neighbor, LP measures the relationship levels. Replace KNN with LP is the improved method in our project because LP works better on related samples instead of only focusing on quantities of the nearest samples.

The overall performance of our project is shown on the section 4.7, but we still believe that there is still room to continue to rise. Although the results are not good, we successfully apply the idea of Label Propagation to the Matching Network model.

The future of the project will be:

- Introduce sparse coding method in the project to improve the performance on sparse samples distribution.
- Apply the network in language translation.
- Continue improve the network by adjusting the parameter set.

Acknowledgement The project is completed based on the hard work of this team, and further thanks to professor Yuhong Guo, who led us to start this project.

References

[1] A. Iscen, G. Tolias, Y. Avrithis and O. Chum. (2019). Label Propagation for Deep Semi-supervised Learning, IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

[2] B. M. Lake, R. Salakhutdinov, J. Gross, and J. B. Tenenbaum. One shot learning of simple visual concepts. In CogSci, 2011.

[3] F Sung and Y Yang and L Zhang and T Xiang and Philip H. S. Torr and Timothy M. Hospedales. Learning to Compare: Relation Network for Few-Shot Learning. 2018

[4] G Koch, R Zemel, and R Salakhutdinov. Siamese neural networks for one-shot image recognition. In ICML Deep Learning workshop, 2015

[5] J Snell and K Swersky and R S. Zemel. Prototypical Networks for Few-shot Learning. 2017

[6] O., Vinyals, C., Blundell, T., Lillicrap, D., Wierstra, et al. Matching networks for one shot learning. In Advances in Neural Information Processing Systems, pages 3630–3638, 2016.

[7] R. C. Staudemeyer and E. Rothstein Morris, "Understanding LSTM - a tutorial into long short-term memory recurrent neural networks", arXiv:1909.09586, 2019

[8] S Hochreiter and J Schmidhuber. Long short-term memory. Neural computation, 1997.

[9] Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. Hwang, and Y Yang. 2019. Learning to propopagate labels: Transductive propagation network for few-shot learning. In Proceedings of the International Conference on Learning Representations.

[10] A Santoro, S Bartunov, M Botvinick, D Wierstra, and T Lillicrap. Meta-learning with memory-augmented neural networks. In ICML, 2016.

[11] D Rasband. ASL Alphabet Test. url: `https://www.kaggle.com/danrasband/asl-alphabet-test` (accessed: 12.12.2021).