Computer Science
&
Data Science

Capstone Report - Fall 2022

Learning to Learn: A Simple Approach to Few Shot Classification

Jade Zhou,
Leyi Guo

supervised by
Li Guo

Preface

In the area of machine learning, a large number of samples is always needed for machines to be trained on to differentiate between various categories. In contrast to these traditional machine learning approaches, few-shot learning is introduced in cases where available data is insufficient. In few-shot learning, the model typically learns base categories first and then generalizes the data to novel classes, in this way, it can perform well with only a few samples.
This paper is aimed at readers who might be interested in the field of Few-shot Learning. After reading this paper, our readers will have a basic understanding of several popular models today, particularly of the Embedding Adaptation model. As two bachelors from the Data Science and Computer Science department in New York University Shanghai, we hope that our practice and analysis will be helpful and inspire you.

Acknowledgements

We would like to express our gratitude to our supervisor, Li Guo, who guided us throughout this project.

Abstract

Inspired by humans' ability to learn quickly, scientists dive into the field of meta learning, building models that can learn how to learn. Similar to meta learning, Few-Shot Learning is useful for dealing with the challenges of limited data. One challenging part of Few-Shot Learning is to increase the model's efficiency of extracting learning techniques from information of previously SEEN classes. In this paper, we propose several improvements to the Embedding Adaptation model to further improve the accuracy of the model by allowing it to extract associations between base classes. Specifically, we changed the transformer layer of the FEAT model to include query images in the transformer, which will not only study the difference between labeled samples, but also compare between query and each support set. Our method improves the accuracy of the model by approximately 1%.

Keywords

Few-Shot Learning; Embedding Adaptation; FEAT; Self-Attention

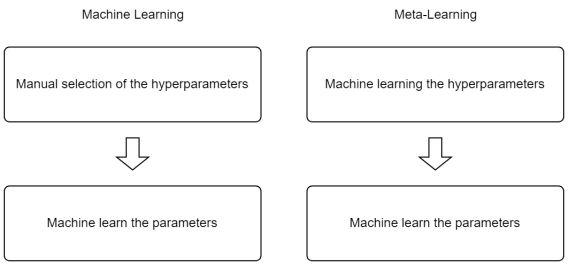
Contents

1 Introduction

Machine learning models have made huge progress in visual recognition [1]. In most cases, a large dataset is required for training the models to predict well. However, in practical applications, we often encounter the problem of insufficient samples. Take the task of recognizing different animal images as an example, there might be some rare species with very few available pictures. In such a situation, the performance of existing deep learning models for image recognition will be significantly limited [2]. Few-shot learning aims to deal with the problem of learning with limited data, which mimics humans' ability to recognize new concepts with high accuracy by learning only a few examples [3]. It is thus of great interest to learn a classifier which is able to adapt to learning new classes that have not been seen in the past, given only a few examples of each class.

2 Related Work

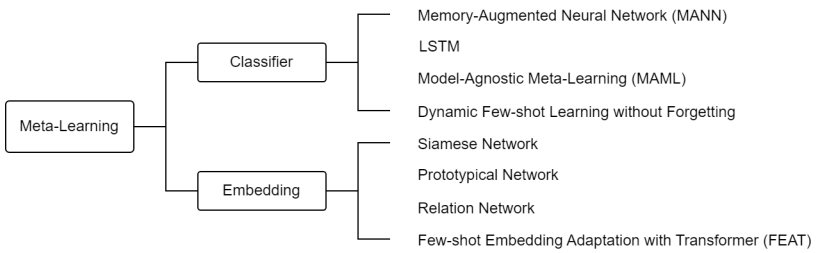
Figure 1: Differences between machine learning and meta-learning



The most common method for few-shot learning is meta-learning. Motivated by human's actions when learning new classes, meta-learning focuses on learning how to learn new concepts efficiently from previous experiences, which will lead to a better performance when learning other unseen classes in the future [4]. For example, a classic machine learner learns how to classify cats and lions from thousands of pictures from each class. However, a meta-learner will not look at the pictures of cats and lions directly. Instead, the meta-learner learns how to classify animals by pre-training on thousands of animal classification tasks in advance. Therefore, when presented with a few pictures of cats and lions, the meta-learner could quickly adapt to distinguish each new class by making full use of the learned knowledge from previous tasks. Training a meta-learning involves learning at two levels, within and across tasks. At the first layer, the learner learns how to rapidly finish one task. At the second layer, the meta-learner learns across multiple tasks, in order to perform well in any new tasks given [5].

Based on the idea of meta-learning, researchers have come up with different methods to improve the learning process, which fall broadly into two categories. The first is to control how a classifier should be constructed while another category focuses on learning an embedding that is applicable to all tasks [6]. We will give a brief summary of important models from both categories.

Figure 2: Different approaches in meta-learning



Classifier

One important direction in the field of few-shot learning is optimizing the classifier, which has a series of beautiful models [6].

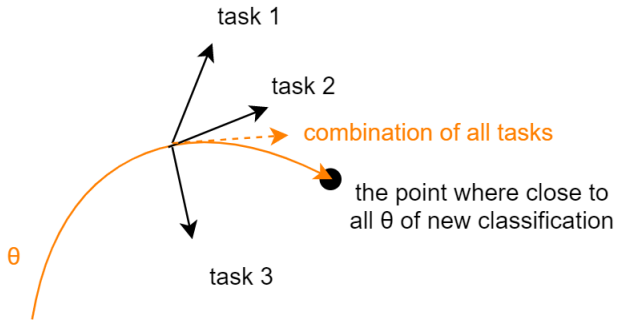
Inspired by human brain, the work of Santoro et al. proposed a memory-augmented neural network (MANN) based on the idea of neural turing machines in 2016, which uses a new method for using external memory based on LSTM [7]. Since MANN is capable of meta-learning, which refers to rapid learning from sparse data, they design the task structure to incorporate meta-knowledge. It carries both short-term and long-term memory as humans do, and achieves high accuracy in classification after only a few samples. For the memory writer, they use LRUA (Least Recently Used Access) to write extracted features. MANN is trained to learn how to store and retrieve related memories for each classification task.

After MANN, several works made full use of the idea of memory from the past. LSTM (Long short-term memory) is a concept mentioned a lot in various works. It is an artificial neural network and the units are composed of a cell, an input gate, an output gate and a forget gate. It remembers values from an input sequence and adjusts weights according to its memory. The work of Andrychowicz et al. uses an LSTM to train a neural network [8]; however, they are interested in learning a general optimization algorithm to train neural networks for large-scale classification, whereas Ravi et al. were specifically interested in the few-shot learning problem [9]. Their article proposes an LSTM based meta-learner model to optimize a learner neural network classifier in few-shot learning, which addresses the problems with gradient-based optimization

model by using meta-learning. The authors mention training a meta-learner LSTM to learn how to provide optimal updates to the parameters for a deep neural network using its states, thus learning the optimal initial weights of the classifier and offering good starting points for new classification tasks.

LSTM has also made great contributions to the problem, however, it has the huge weakness of temporality by nature. Based on Ravi's work, Finn et al. proposed Model-Agnostic Meta-Learning (MAML) [10]. The unique feature of MAML is that it is almost unlimited and can be applied to any deep learning model, the only requirement is that the model is trained using gradient descent. The core idea of MAML is to train the initialization parameters of the model. With a starting point that is considerably close to all endpoints, the model can converge quickly on a new task with a few rounds of gradient descent using a small number of samples. MAML has another advantage that previously mentioned models lack. MAML can be applied to any deep learning model without adding a large number of learning parameters.

Figure 3: The basic principle of MAML



In order to solve the problem of overfitting on new classes that appeared in the predecessor model, Gidaris and Komodakis proposed two technological innovations in 2018 [11]. In the learning process of the classifier, the authors propose an additional component, the Few-shot classification-weight generator, which generates a corresponding classification weight vector for each new class by accepting a small number of samples from some new classes. Its main feature is that, in order to construct new classification-weight vectors, it explicitly exploits previously acquired knowledge about the visual world by adding an attention mechanism to the classification-weight vectors of the basic classes. This attention mechanism significantly improves the performance of the recognition of new categories, especially when only one training example is available to learn them. The second method is to transform the classifier into a cosine similarity

function based on the feature vectors and the classification weights. This implementation not only solves this problem, but also learns features in the new class better than the dot-product-based classifier.

Embedding

Another approach to few shot classification focuses on embedding [6]. This approach assumes that if the model has a great enough embedding that captures all necessarily representations of input data, then it only needs simple classifiers to do the classification job. This embedding will help avoid the danger of overfitting on a small number of labeled instances from the new classes.

In 2015, Koch et al. introduced Siamese Neural Networks for One-shot Image Recognition [12]. The authors propose the combination use of twin neural networks and rank of the similarity between the inputs. Siamese networks are conjoined neural networks which conjoin is achieved by sharing weights. The Affine distortions method is used additionally to increase the amount of data when processing the data. Affine distortions method is a small deformation of the existing samples in the dataset they used.

After the Siamese Neural Network, Snell et al. proposed another approach by arguing that each class can be represented by a specific point in some high-dimensional space in 2017 [3]. The Prototypical Network maps the sample data of each class into a space and extracts their mean to represent it as the primitive of the class. Euclidean distance is used to calculate the distance. By training, the instances of the same class have the closest distance to the prototype representation of the class, and the distances to the prototype of other classes are farther after mapping by the model.

One year later, in 2018, Relation Network was proposed by Sung et al which avoids fine-tuning and complex recurrent neural network architectures, making it simpler and more efficient than Prototypical Network, especially in one-shot classification tasks [13]. This article introduced a two-branch Relation Network for tackling the problem. Relation network focuses on learning an embedding and a transferable deep non-linear distance metric for comparing the relation between images. By expressing the inductive bias of a deeper solution, they make it easier to learn a generalisable solution to the problem. Despite its simplicity and small computational cost, the RN model failed to perform well on 5-shot problems due to its over-simplified solution to obtain the feature information.

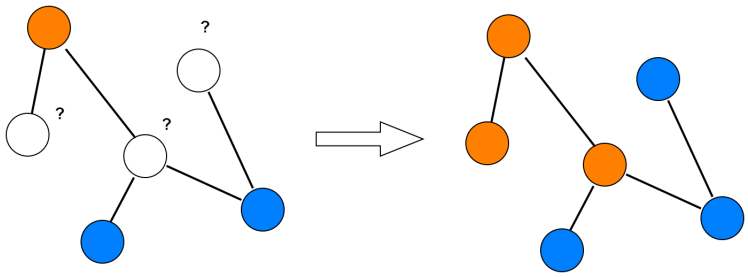
Most few-shot visual recognition models usually learn a discriminative instance embedding

model on the seen categories. However, the problem occurs as they share a common embedding space for all the unseen categories. Ye et al. introduces the model few-shot embedding adaptation with transformer (FEAT), which deals with the problem [6]. The authors suggest that such approaches fail to make the visual features of unseen samples discriminative as for the seen classes, so they propose an adaptation strategy which generates customized embedding spaces for various target tasks using a set-to-set function Transformer. They also applied a self-attention mechanism to extracted features. The idea of making task-specific embeddings significantly improves the performance on generalized few-shot classification of both seen and unseen classes.

3 Solution

Graph Convolutional Networks

Figure 4: GCN



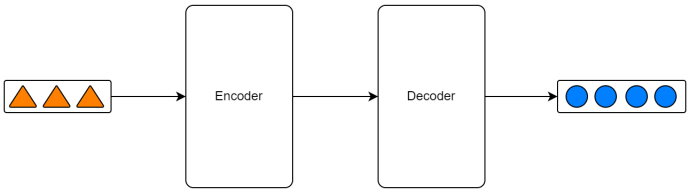
A Graph Convolutional Network (GCN) is a type of neural network that is designed to operate on graph-structured data. GCNs are a generalization of convolutional neural networks, which are typically used for image data, to graph data.

GCNs use a graph convolution operation to aggregate information from neighbors in the graph and propagate it throughout the network. This allows GCNs to capture the underlying structure of the data and make use of it in the learning process.

Firstly the model constructs the degree matrix A to represent the similarity between instances in a set. If two instances come from the same class, then we set the corresponding element in A to 1, otherwise to 0. Based on A, we build the "normalized" adjacency matrix.

We chose to use the result of applying GCN to a 5-way 5-shot task on MiniImageNet with ResNet backbone as the baseline of our approach, with the test accuracy at 0.8185.

Figure 5: attention



Attention and Transformer

In artificial neural networks, attention is a technique designed to mimic human attention. This mechanism enhances some parts of the input data so that the model focuses more attention on the important part. It allows the model to focus on various parts of the input and weight the contributions of different parts of the input when making a prediction.

Attention mechanisms can be broadly divided into two categories: self-attention and external attention. Self-attention, also known as intra-attention, is a mechanism where the model calculates attention weights for each part of the input using the input itself. External attention, also known as inter-attention, is a mechanism where the model calculates attention weights using an external source of information.

Attention is frequently employed in natural language processing. The ordinary substitution of words in the target language for their counterparts in the original language does not work because the order of words changes when moving from one language to another. To solve this problem, a recurrent neural network with Encoder-Decoder structure was created. It first reads all the words in the sentence to be translated through an Encoder recurrent neural network to obtain an intermediate hidden layer containing all the information of the original language, and then feeds the intermediate hidden layer state into the Decoder network and outputs the translation result word by word. Regardless of the order of the key words in the input, the Encoder-Decoder network can handle the output position and form of these words well since they are all packed into the middle layer together and input to the back-end network.

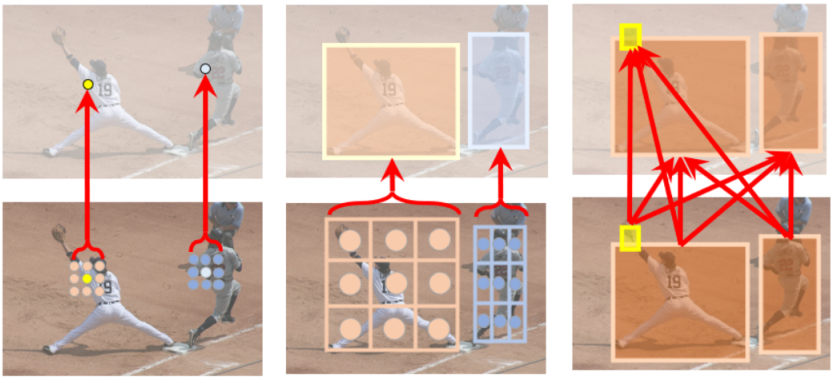
Attention was originally published in 2015 for improving the Seq2Seq model; later it was found that Attention was not limited to the Seq2Seq model, but could be used on top of all RNNs. Then later, it was found that there was no need to use RNNs at all, and Attention alone worked better instead, and the Transformer model was proposed [14].

Take machine translation as an example, the difference between Transformer and self-embedding is that after adding the data sequence to inputs embedding, we need to add positional encoding

to the word vector of each word. For example, "I like cats" and "cats like me" have completely different meanings. It is important to get information about the position of the word in the sentence. But Transformer is entirely based on self-Attention, and self-attention is not able to get word position information. If we don't add position encoding, even if we disrupt the position of words in a sentence, each word can still calculate the attention value with other words. It is equivalent to a powerful bag-of-words model that has no effect on the results. In the field of computer vision, it is equivalent to having the tail growing on top of the head but still being recognized as a cat. So we need to add positional encoding to each word vector when we input.

Like RNNs, Transformers take sequential data as input. But unlike RNNs, Transformers process the entire input at once. The attention mechanism provides context for any position in the input sequence, which allows for parallel computing thus reducing training time. The graph below shows a great example on applying transformer to computer vision [15].

Figure 6: Transformer in computer vision

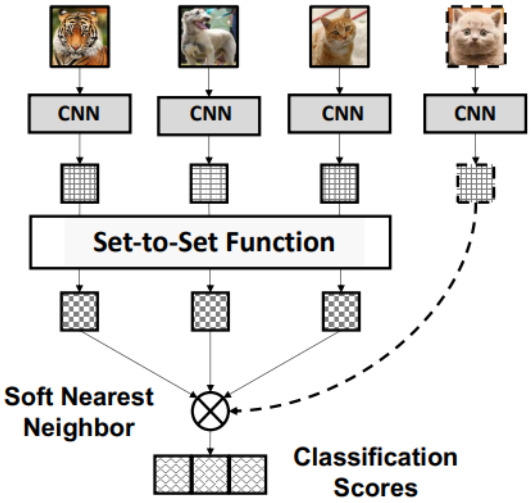


In contrast to the non-attention model, the encoder-decoder now has the opportunity to learn whether each word is derived from some other word and to look for correlations between words. This inspires us to use a similar mechanism that allows the machine to learn the relationships between different categories of pictures. Attention mechanism occurs between Target elements and all elements in Source, while Self-Attention, as the name implies, refers to the Attention mechanism that occurs between elements within Source or between elements within Target, which can also be understood as the Attention mechanism in the special case of Target equal to Source.

FEAT

Transformer architecture is applied to implement a set-to-set function. In particular, the model employs the self-attention mechanism mentioned above to transform each instance embedding

Figure 7: FEAT

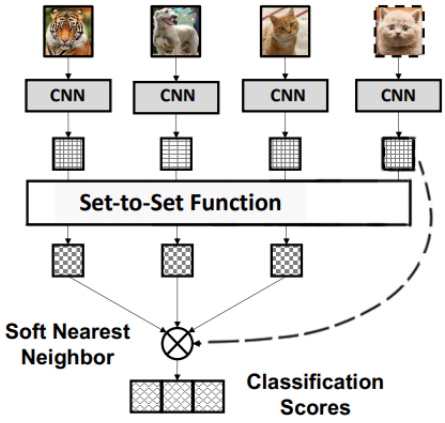


with consideration to its contextual instances. It naturally satisfies the desired properties of set-to-set function because it outputs refined instance embeddings and is permutation invariant. We denote it as Few-Shot Embedding Adaptation with Transformer (FEAT).

The test accuracy of applying FEAT to 5-way 5-shot tasks on MiniImageNet with ResNet backbone is 0.8185, slightly improved compared with GCN.

semi-FEAT

Figure 8: semi-FEAT



In semi-feat, the model not only takes the support sets, but also the query images into the self-attention step as well. The input to set-to-set function is a combination of both support and query, for the transformer layer will now have the ability to adjust to the query images and better capture the context in queries and all support sets.

Algorithm 1 Algorithm for semi-FEAT

Require: Seen class set S
  for all iteration = 1,...,MaxIteration **do**
    Sample N-way M-shot (D-train, D-test) from S
    Compute the embedding of all pictures
    for all item in test set **do**
      Compute set function T by applying transformer to the concatenation of all centered support set and the whole query vector
      Predict label for query images
      Compute the loss of y-hat of each test and the label
    end for
    Compute gradient decent of E and T
    Update E and T use SGD
  end for
  return Embedding function E and set function T.

The test accuracy of applying semi-FEAT to 5-way 5-shot tasks on MiniImageNet with ResNet backbone is 0.8201, slightly improved compared with FEAT. But the learning accuracy is higher than normal FEAT which we will further discuss in the next session.

semi-proto-FEAT

In this approach, the classifier of the model is implemented with an additional Prototypical Networks, which uses prototypes to classify new, unseen examples.

In the PN model, the prototypes are computed by taking the mean of the embeddings of a set of the support set for each class. The prototypes are then used to classify query samples by computing the distance or similarity between the prototypes and the query set examples and using the distances or similarities as weights to compute a weighted sum of the prototypes.

The test accuracy of applying semi-proto-FEAT to a 5-way 5-shot task on MiniImageNet with ResNet backbone is 0.7525, which is not satisfying as semi-FEAT.

multi-FEAT

In our approach, we modified the transformer in semi-feat. In semi-feat, the transformer takes all query images all at once regardless of their original class. We split the query sets into single pictures and apply it to the self-attention mechanism, so that the model can learn to extract the vital features of each input image and improved its accuracy of making correct predictions.

The test accuracy of applying our approach to 5-way 5-shot tasks on MiniImageNet with ResNet backbone is 0.8193.

Algorithm 2 Algorithm for multi-FEAT

Require: Seen class set S
  for all iteration = 1,...,MaxIteration **do**
    Sample N-way M-shot (D-train, D-test) from S
    Compute the embedding of all pictures
    for all item in test set **do**
      for all item in query **do**
        Compute set function T by applying transformer to the concatenation of all centered support set and one query image
        Predict label for the query image
        Compute the loss of y-hat of each test and the label
      end for
      Compute the average loss
    end for
    Compute gradient decent of E and T
    Update E and T use SGD
  end for
  return Embedding function E and set function T.

4 Results and Discussion

	1-Shot 5-Way		5-Shot 5-Way	
Embedding adaptation	ConvNet	ResNet	ConvNet	ResNet
GCN	0.5325	0.6450	0.7059	0.8165
FEAT	0.5515	0.6678	0.7161	0.8185

Table 1: Test accuracy on the two embedding adaptation

Embedding	Accuracy
GCN	0.8165
FEAT	0.8185
semi-FEAT	0.8201
semi-proto-FEAT	0.7525
multi-FEAT	0.8253

Table 2: Test accuracy on 5-way 5-shot ResNet backbone MiniImagenet

Our experiments were made on the dataset MiniImagenet, and include comparing between FEAT models with different backbones, evaluation shots and embeddings respectively. We found that 5-shot classification outperforms 1-shot classification by more than 10%, and the result is shown in figure 11. We also concluded that the backbone of ResNet has better performance than ConvNet 12.

In addition, we also experimented on semi-proto-FEAT, which applies the semi-supervised learning technique using an unlabeled pool set and the training accuracy is displayed in figure 10, which doesn't give an ideal result. Therefore, considering all the factors, we then apply our

multi-FEAT using 5-way 5-shot and the backbone of ResNet.

As shown in figure 9, combining both Semi-FEAT and FEAT models, our approach has a more stable fluctuation in accuracy as the number of epochs increases, and also has a slightly higher accuracy during the training.

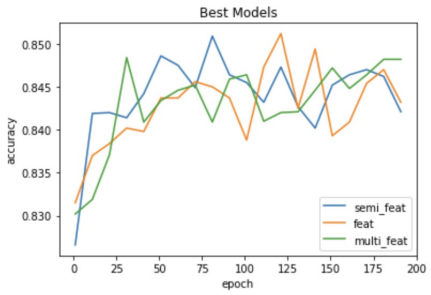


Figure 9: Training Accuracy on 5-way 5-shot ResNet backbone

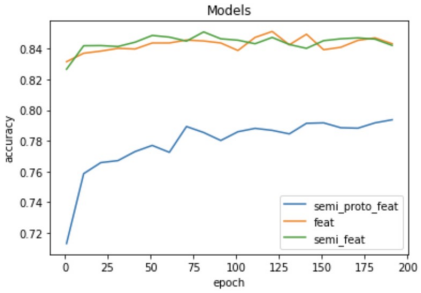


Figure 10: Training Accuracy on 5-way 5-shot ResNet backbone

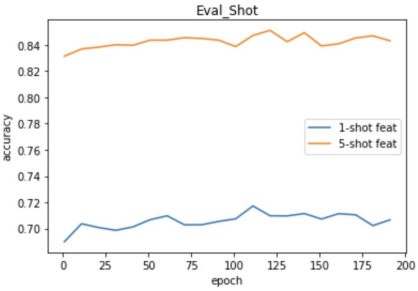


Figure 11: Training Accuracy on 5-way 1-shot & 5-way 5-shot

5 Discussion

The results of our study indicate that our approach of modifying the existing FEAT model is effective at improving the performance. In our experiments, our method outperforms FEAT in

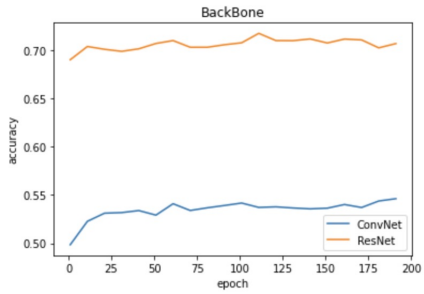


Figure 12: Training Accuracy on ResNet/ConvNet backbones

both training and testing, achieving an average improvement of 1% in accuracy. Our method could be helpful to the readers as it combines two learning algorithms with outstanding performance, and further makes improvements.

While our results are encouraging, there are several limitations to our study that should be considered. First, the experiments were conducted on only one dataset MiniImagenet, and it is not clear how well the algorithm will generalize to other datasets like TieredImageNet, CUB, etc. Further research will be needed to evaluate the algorithm on a wider range of tasks and data.

Secondly, the inner for loop significantly affected the calculation speed which forced us to decrease the number of queries in each batch. This can be improved by grouping the queries and find a prototype representation before input to the transformer in the learning process, similar as what we do in the the support set.

Additionally, our study focused on a specific type of few-shot learning problem, and it is not clear how well the algorithm will perform on other types of tasks. Future work could explore the use of the algorithm on a broader range of problems to better understand its capabilities.

Overall, our study provides evidence that the new learning algorithm is effective at improving performance on few-shot learning tasks. While there are limitations to our study, the results suggest that this approach has the potential to advance the field of few-shot learning and open up new possibilities for machine learning applications.

6 Conclusion

In conclusion, we improved the accuracy of the FEAT model by including the query pictures in the transformer. In this case, the transformer will not only capture the key features between support pictures, but also find the possible relationship between query images and each class which helps increase the accuracy.

Currently our model lack experiments on various situations which can be further researched by applying it on datasets like TieredImageNet, CUB, etc. We also propose that the embedding of the model could be learned with a refined self-attention mechanism instead of the soft nearest neighbor, as it will reduce the number of parameters in the model and increase the calculation speed.

7 Future Improvements

For our future improvements, there are several areas that we plan to focus on in future research, including:

1. Applying our approach to more challenging tasks. While our model has shown good performance on the MiniImagenet dataset we have tested it on so far, there is still a long way to go in terms of achieving strong performance on more challenging tasks, such as those involving large-scale or high-dimensional data. We plan to explore ways to scale our approach to these tasks and achieve better performance.

2. Improving our feature representation and extraction. Since our algorithm involves an inner loop of iterating over each query image, the speed of training the model is significantly affected. In the future, we plan to experiment with different feature representations and learning methods in order to find the best way to capture the underlying structure of the data, such as by grouping the queries and find a prototype representation before input to the transformer in the learning process.

3. Further improving our learning algorithm. Currently our embedding learns with a soft nearest neighbour objective, and we propose that we can replace it by a refined self-attention mechanism to better fit our model.

References

[1] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," 2019. [Online]. Available: https://arxiv.org/abs/1904.04232

[2] Y.-H. H. Tsai and R. Salakhutdinov, "Improving one-shot learning through fusing side information," 2017. [Online]. Available: https://arxiv.org/abs/1710.08347

[3] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," 2017. [Online]. Available: https://arxiv.org/abs/1703.05175

[4] A. Parnami and M. Lee, "Learning from few examples: A summary of approaches to few-shot learning," 2022. [Online]. Available: https://arxiv.org/abs/2203.04291

[5] C. Giraud-Carrier, R. Vilalta, and P. Brazdil, "Introduction to the special issue on meta-learning," *Machine Learning*, vol. 54, no. 3, p. 187–193, 2004.

[6] H.-J. Ye, H. Hu, D.-C. Zhan, and F. Sha, "Few-shot learning via embedding adaptation with set-to-set functions," 2018. [Online]. Available: https://arxiv.org/abs/1812.03664

[7] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "One-shot learning with memory-augmented neural networks," 2016. [Online]. Available: https://arxiv.org/abs/1605.06065

[8] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," 2016. [Online]. Available: https://arxiv.org/abs/1606.04474

[9] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *ICLR*, 2017.

[10] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," 2017. [Online]. Available: https://arxiv.org/abs/1703.03400

[11] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," 2018. [Online]. Available: https://arxiv.org/abs/1804.09458

[12] G. R. Koch, "[pdf] siamese neural networks for one-shot image recognition: Semantic scholar," 2015. [Online]. Available: https://www.semanticscholar.org/paper/Siamese-Neural-Networks-for-One-Shot-Image-Koch/f216444d4f2959b4520c61d20003fa30a199670a

[13] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," 2017. [Online]. Available: https://arxiv.org/abs/1711.06025

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[15] "Why is transformer so popular in computer vision," https://www.msra.cn/zh-cn/news/features/cv-transformer, accessed: 2022-12-14.