COMP 4901J Project Final Report: One-shot Pokemon Classification

Xiang Li, Qixu Chen, Xinran Zhao
{xlide, qchenax, xzhaoar}@connect.ust.hk

Abstract

We aim to solve the Kaggle One-shot Pokemon Classification problem where we classify pokemon images "in the wild" under the condition that for each class of Pokemon, only one labeled image is given. We adapted and improved the simple and flexible Relation Network for few-shot learning. Our network learned to learn how to embed the Pokemon images as well as how to compare their embeddings based on a deep distance metric. We propose several adjustments on both the embedding network and distance metric network to enable more robust representation and metric to be learned. With the same settings, our model out-perform the original relation network on our task. Experiments and an ablation study are done to show that the adaptions make a more effective model for few-shot image classification on Pokemon images.

1. Introduction

Deep learning based Computer Vision algorithms has achieved great performance on image classification problems. However, it has long been a problem that these models usually require a huge amount of human labelled images. Image classification algorithms, for example, usually require thousands of labels for each classes. Introducing new classes would be costly in both time and efficiency, especially when the class has only limited amount of data (e.g. some deep sea animals). In comparison, humans could learn thousands of object classes starting from a very young age of life. Children show great ability in generalizing their abilities in recognition and classification on new concepts with only few instance given.

The problem to learn to generalize the classification to unseen classes with only few examples given is know as few-shot learning. One-shot learning refers to the machine learning problem where we only have one labeled example for training, as defined in [2]. One-shot learning keeps gaining popularity in recent years because of its intrinsic similarity to the human cognitive process, where the general knowledge learned previously is transferred to the one-shot task. Contemporary approaches working on this problem focus on how to learn the meta knowledge of recognizing a image hence classify different classes of images. These works include forming good initialization [3] or good optimization [10]. Another promising work, Relation Network [13], does not require complex adaption on the optimization strategies. It focuses on storing the meta information in a well designed neural network.

Specifically, the Relation Network approach divide the task into two modules: embedding module, also known as the embedding network, and relation module, also known as the distance metric network. The former one learns to convert the images into hidden representations. And the relation module learns a non-linear comparison function to compute the similarity between the query image and support image(s) for a specific unseen class. Finally, the similarity would be compared and the query would be classified to the one with highest similarity score given by relation module.

With the understandings about the pipeline, in this work, we focus on making the Relation Network more robust through three aspects: task-specific data argumentation, a better embedding module and a relation module with more information involved in each layer.

In this project, we tackle the one-shot image classification problem on the Kaggle Pokemon images dataset [15]. Pokemon creatures are significantly different in color and shape. They also live in different environment, which add the difficulty of the problem. Given hundreds of Pokemons only appear once or twice in the original cartoon, a few-shot classification is natural for the Pokemon classification problem. Also, Pokemon trainers are required to classify the Pokemon creatures they met in the wild, given only a few images collected by the Pokemon Gym. This adds the significance of few-shot Pokemon classification task in an empirical setting.

To solve this, we experiment and make adaptations on the Relation Network architecture. We replace the conventional CNN backbone for embedding module with a ResNet [4], which we believe to be more capable in capturing features. As we notice that the backgrounds of the Pokemon

images vary a lot. We also perform saliency detection on the input support images and add the resulted saliency map as an additional channel other than original RGB. Finally, in the relation module, we concatenate the hidden representation from embedding modules to the later layers to allow feature reusing and more direct supervision (gradient propagation) to the embedding module. We also experiment on using separated neural network for the module for query image and support image for the unseen classes, under the hypothesis that the embedding for query and support could need different feature weights.

Overall our contribution is provide a robust framework based on the existing state-of-the-art few-shot learning method, Relation Network. The framework shows great improvement comparing with the baseline Relation Network on our novel dataset aiming at classifying Pokemon. Ablation study will be performed in later section to show the worked ideas and the failed ones as well.

2. Related Work

In the one-shot classification setting, the model is given abundant training data from the base classes, and the model aim to classify novel classes with only one supporting labeled data. Much efforts have been done from different ways to tackle with the problem with finding representative and robust model with only a very small amount of data. In the following, we discuss about two main categories of few-shot learning algorithms: meta-learning based and distance metric learning based.

2.1. Meta-learning based methods

Methods to be discussed in this section tackle the few-shot learning problem by building a set of new mechanisms to train the neural network to allow it learn with only limited number of labelled examples. One approach is to learn a good initialization of the parameters of a network so that it could learn to classify novel class by fine-tune with only a few gradient steps [3]; [9]; [11]. Other work focus on learning a new optimizer with memory based on LSTM or external memory [10]; [8]. These meta learning based algorithms could achieve state-of-the art results on few-shot learning problem. But experiments also show that these methods could not deal with the case where the base classes and novel classes have very different domains [1].

2.2. Distance Metric Learning based methods

Methods to be discussed in this section regard the few-shot learning problem as a problem to find the most similar novel class for a test image. In these methods, the models are first trained to represent the images with a hidden representation and then compare the distance of these representations as a metric to do comparison. The distance metrics are given as cosine similarity [14]; Euclidean distance [12]; and CNN based relation module to learn a neural network based distance metric [13]. These methods are easier to explain and apply to different settings are few-shot learning problem. And the Relation Network using a CNN based relation module over-perform all the other. In this report, we will implement and adapt the Relation Network to fit our one-shot Pokemon classification problem.

3. Problem Statement

3.1. Dataset

The Kaggle dataset of RGB Pokemon images [15] includes more than 800 classes of Pokemon, each with 1 or 2 labelled support image. In addition, 3000 unseen test images are presented, as shown in Figure 1.

Figure 1. Examples of one training image and one test image in the Pokemon dataset

Pokemon creatures vary a lot in terms of shape and color, which contributes to the significance of this problem. Also, particular to this dataset, the support images have white or black background; while in the test images, Pokemon creatures are put into different backgrounds, which motivates our data preprocessing approach with saliency detection. The saliency map detected is added as an extra channel for the image.

3.2. One-shot Classification

Our problem setting is an instance of *C-way one-shot problem*, where we have to do the classification among C classes with only one labeled image for each class. The training and testing for the one-shot learning problem is different than the regular deep learning problems. We adopted the episodic training method proposed by [14].

The dataset was separated into *meta-training* classes and *meta-testing* classes, each containing a number of classes with one support image, the label of which is available to the network, and multiple query images, whose labels are unavailable to the network.

In each training iteration, an episode is created by randomly selecting C classes from the *meta-training* classes with one support image from each of the C classes acting as the *support set* $S = \{(x_i, y_i)\}_{i=1}^{m} (m = K \times C)$. A fraction of the remainder of the query images from those C classes classes act as the *query set* $Q = \{(x_j, y_j)\}_{j=1}^{n}$. The

support set and the query set will be fed into our networks and eventually produce a numeric value between 0 to 1 representing the similarities between each pair of query image x_j and support image x_i, which is called relation score. In our C-way one-shot classification task, C relation scores $r_{i,j}$ are generated between one of the test image and the support set images and we apply softmax. We use mean-square-loss function to train our model, and our objective function is:

$$\phi, \varphi = \arg\min_{\phi,\varphi} \sum_{i=1}^{m} \sum_{j=1}^{n} (r_{i,j} - 1(y_i = y_j))^2$$

where ϕ , φ are the set of parameters in the embedding module and relation module of our network respectively.

Namely, we do not just use the labeled support images to training a classifier, instead, we imitate the test time scenario and input a set of support images and a set of test images, using the support images only as reference for the specific class and learn the meta-knowledge about classification.

4. Technical Approach

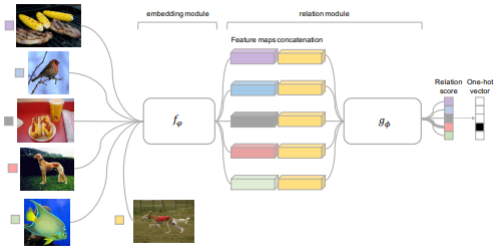

Figure 2. Illustration of the original relation network [13] architecture

We extend the idea of the relation network [13], which is the current state-of-the-art for few-shot classification task on various datasets. The original structure of the relation network is shown in Figure 2. The support and test images are first projected into an embedding space by a convolutional neural network, which we refer to as the *embedding module* or the *embedding network*. Then, the embedding vector of each test image is concatenated pairwise with the embedding vector of the support image, and passed through a convolutional *relation module*, also known as *distance metric network*, that serves as the classifier, producing the classification result after the final softmax function. The following adjustments were made to the network and the complete network structure is shown in figure 4.

4.1. Separated Embedding Network

Inspired by the idea to use different neural network for heterogeneous input, here we also want to use two different instances of the embedding network for the support image embedding and query image embedding. We hypothesize that a good embedding for support and query could show different focus. In terms of data, the support images have blank backgrounds while the query images have diverse backgrounds. In term of the usage of the embedding, a good embedding of support image should be representative and more regularized, while a good embedding of test image should show as much feature and uniqueness as possible.

4.2. Saliency detection

The background varies drastically in different test images. Since the relation network essentially performs a pairwise similarity comparison, the background clutter could potentially affect the classification accuracy. Therefore, we perform saliency detection on the test images to obtain the foreground object. The pre-trained saliency detection model outputs the class-agnostic foreground mask, requiring no mask label from our dataset. The output of the saliency detection is illustrated in Figure 3.

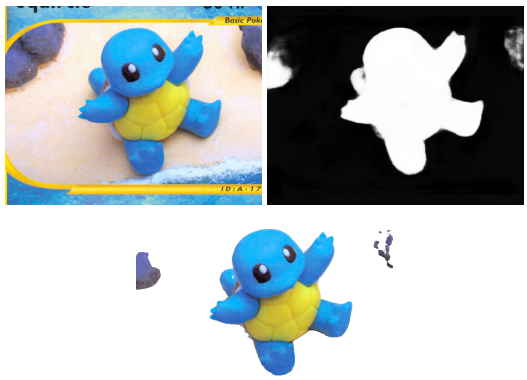

Figure 3. Example of the original image, saliency mask, and the output foreground image

We adopted a saliency detection model based on [5]. We used the pre-trained weights as a transfer learning approach for detecting the foreground Pokemon in the images. The result is visually feasible for test images with simple background or the test images where the contrast between the Pokemon and the background is strong. However when the Pokemon blend in with the background the model produces wrong detection results, sometimes completely missing the Pokemon.

In order to alleviate the adverse effect of the saliency detection's incorrect output, we do not simply extract the foreground and feed it to the model. Instead, we concatenate the

saliency map channel-wise to the original input image. In this way, the network could learn to use the information in the saliency map without compromising the original image. Concatenating the output foreground image instead of the saliency map, or feeding in the foreground image directly are also tried, but they were not chosen as the final approach because they produce worse classification accuracy.

4.3. Cross Layer Linkage

In the ResNet Module [4], the cross layer linkage, or the addition of previous layers on later layers could show two kinds of benefits. First, the loss would be easier to be propagated to the early layers. Second, the intermediate information from former layer could also be useful for later prediction. In our problem, though we do not suffer from vanishing gradients, links between the embedding module and the relation module are hypothesized to be beneficial for similar reasons. First, the loss information might be easier to reach initial layers, so that the embedding would perform more task-specifically, i.e. more focus on create a embedding for better object classification. Second, direct embedding results could also be useful for further layer in the relation module. So instead of using the hidden representation provided by the embedding module only on the first layer of relation module, we could keep the hidden layer to restore as much information as possible for the later classification.

More specifically, we first pool the hidden representation from embedding module to different size suiting the size of the layers in the relation module. We then concatenate these pooled embedding of the images to each layer of the relation module to ensure that the gradient could be backpropogated into the embedding module directly on each stage and the embedding layer could be considered, without too many interference in each layer of the prediction in the relation module.

4.4. Choice of Depth of the Embedding Module

Relation Network provides us a simple and flexible framework for few-shot classification. Since the ultimate goal is to find the similarity between the images, it is crucial that how much information is restored during the conversion from the original image space to the embedding space. To capture a good representation of the image suitable for the final classification, the choice of embedding module is important. The choices could be made from the variations of architectures as well as the complexity or depth of the architecture. The choice should be task specific since the parts that make the difference between different classes vary from task to task. For example, for small input and easy tasks, the embedding module should not be too deep since direct information could be lost. This motivates us to experiment on the architecture and depth of the embedding module.

Specifically, we tried ResNet-18, ResNet-50, Resnet-101 and DenseNet as a replacement of the original shallow Convolutional Neural Network. These more sophisticated network backbones are believed to show higher efficiency in prediction and gradient back-propagation. The result of these experiments is that ResNet-18 is the best choice for our task. This indicates that for our task the features should be more direct and simple.

5. Experiments and Results

5.1. Ablation Study

To better understand our final model and examine the effect of each adjustments, we performed an ablation study for it on the Pokemon one-shot dataset[15]. The meta train-test split was 3:1 with 600 classes of Pokemon images for meta-training and 200 classes for meta-testing. In meta-training, all the networks are trained for 50000 random selected episodes from the meta-training set, each with 5 classes and 10 images(1 support image and 1 query image for each class), and all models have converged. The models were trained using the atom optimizer with learning rate 1e-3 and the learning rate is halved every 10000 episodes. In meta-testing, we randomly select 100 episodes in the meta-test set to get an average classification accuracy. We repeat the meta-testing for 10 times to get a further averaged accuracy, which is reported in Table 1.

Method	Average Accuracy
Baseline	55.60%
Fine-tuned baseline	75.04%
Split encoder	70.18%
Saliency	73.78%
Concatenation	75.22%
Resnet-18 encoder	75.50%
Final model	**77.71%**

Table 1. Ablation study of our Pokemon classification network in 5-way 1-shot settings

We directly adapted the Relation Network as is as our baseline. Since the baseline method is designed for the Omniglot dataset[7], where each image is a 28×28 monochrome image, some adjustment needed to be made before it is suitable for our task. We changed the input dimension to 224×224 in RGB and added convolution blocks to compensate for the increase in input dimension. The result is presented as "Fine-tuned baseline" in Table 1 and it is much improved compared to the original baseline.

The result of using different network for the encoding of the support and query images is presented in the table as "Split encoder". The overall performance suffered, potentially suggesting that not sharing encoder weights for support and query images could make the support and query image be embedded into different spaces and increase the

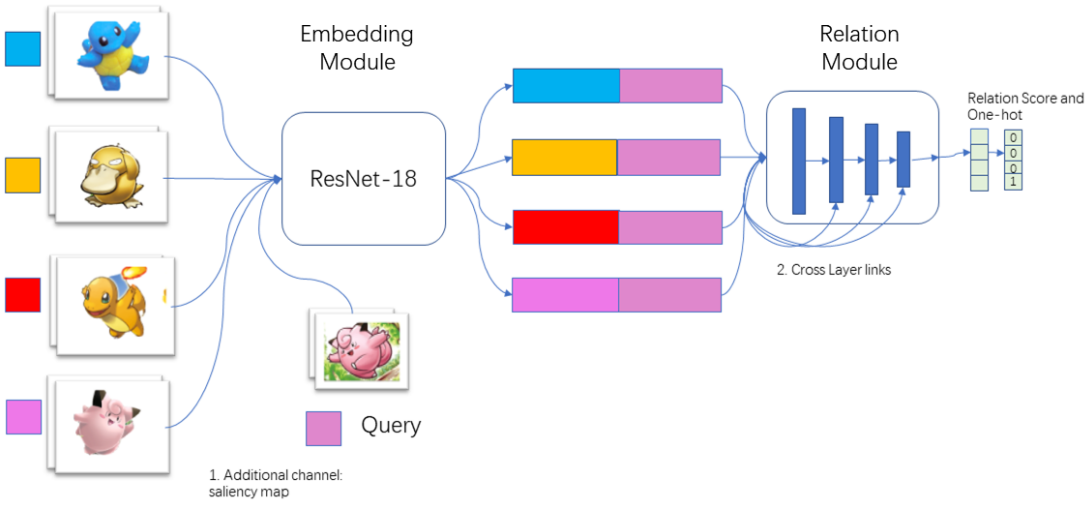

Figure 4. Illustration of the overall architecture

difficulty of the similarity comparison of the distance metric network.

The result of adding the saliency map channel is presented as "Saliency". It was observed that the network could make use of the information in the saliency map, however, since the saliency detection did not always accurately find the Pokemon, it could mislead the network sometimes. In the experiments, we could observe that using saliency detection increase the variance of meta-testing, suggesting that sometimes saliency detection helped and sometimes it had an adverse effect on the classification. After averaging, the overall result dropped as compared to the fine-tuned baseline.

The result of concatenating the pooled embedding vector to each stage of the distance metric network is presented as "Concatenation". The averaged result slightly improved, which may show that the distance metric network is learned better when the original embedding vectors are available to all different stages.

The result of changing the encoder network backbone is presented as "Resnet-18 encoder". Different backbones was tried including deeper Resnet[4] and Densenet[6]. However, the Resnet-18 performed the best. The reason might be that the embedding learned by the deeper network is too complex for the simple classification task, and may provide extra noise and confusion to the distance metric network.

Our final model is based on the "Fine-tuned baseline" and included "Saliency", "Concatenation" and "Resnet-18 encoder". The "split encoder" was discard due to its negative effect to the results. Our final model achieved a 2.67% improvement on the already very high fine-tuned baseline. Although the individual adjustments did not increase the accuracy by a large margin, they achieved a much larger improvement when combined together, this may show that the Resnet-18 could make better use of the information in saliency detection, and such a better embedding vector is reinforced at every stage of the distance metric network because of the concatenation.

5.2. Failure Case

We have also extracted and examined the failure cases of our network.

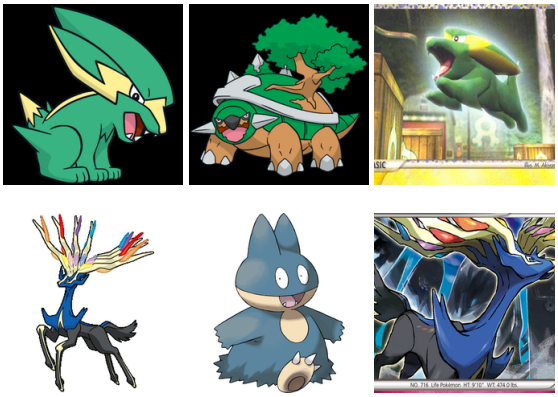

Figure 5. Failure cases. Left images are the correct support images, middle are the support images wrongly classified to, and the right are the query images

There are 2 main reasons for the classification failures. Firstly, some cases could be inherently difficult. The query image could visually similar to multiple support images, which is shown in the first row of Figure 5. However, this is still better than the query image being visually different from all of the support images. Secondly, the saliency detection could make mistakes. In the example on the second

row of Figure 5, the image was wrongly classified because of the false positive in the saliency detection. There were also cases before the saliency detection is added, where the classification is misguided by the background objects, but those cases are very difficult to find after the saliency detection is adopted.

6. Future Work

In the future, we could hand-craft some features and concatenate them into the embedding vector to provide clearer information to the distance metric network for comparing the similarity between two images. We could also improve the performance by collecting more support images and averaging their embedding vectors, although this is different from our one-shot classification task.

Also, considering currently the saliency detection sometimes makes mistake, one future improvement is that we could manually add segmentation mask label to the query images and train(or fine-tune) the saliency detection network as we train the relation network. Then we could obtain a task-specific saliency detection network that could hopefully perform better in the domain of pokemon images.

Further developing this idea, when training the saliency detection network on the domain of Pokemon images, we could provide the corresponding support image to the saliency detection network as extra information. This is very similar to a semantic segmentation task guided by a few images, i.e. few shot semantic segmentation. In the future, we could consider adding relation network to a semantic segmentation network, and solve the task of few-shot semantic segmentation on the Pokemon dataset with segmentation mask label.

7. Conclusion

In conclusion, we clearly defined the one-shot Pokemon classification problem and got a comprehensive understanding on our dataset. Based on the the idea of Relation Network [13], we proposed adjustments and augmentation to the network structure and the dataset to enhance the performance, including the saliency detection, the integration of ResNet block, the separated encoder network and the multi-stage concatenation of embedding vector to the distance metric network. We carried out extensive experiments and performed an ablation study, also examining the failure cases of the network. Finally, we propose the future improvements and a new task on our dataset.

During the project, we have learned a lot about one-shot learning and deep learning in general. Firstly, one-shot learning is hard. If the one support is not representative enough of all the images(or entities) in the class, it is very difficult to proceed to make the classification dicision. Another thing we have learned is that, for a convolution neural network, deeper is not always better. The performance is dependent on the input images and the specific tasks. Also, although certain information has been present(e.g. saliency, embedding vector), instead of hoping that the network could learn these information by itself, it could be beneficial to explicitly provide these information to the network, for example, when we concat the saliency map to the input and concat the embedding vector to multiple stages of the distance metric network. Finally, in the relation network framework, a good result is the combination of a good embedding and a good distance metric. Therefore, we could improve the performance by improving the embedding network, changing its structure and enhancing its input, while also improving the distance metric network, reinforcing the embedding vector at multiple stage of the similarity comparison.

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. *CoRR*, abs/1904.04232, 2019. 2

[2] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):594–611, Apr. 2006. 1

[3] C. Finn, K. Xu, and S. Levine. Probabilistic model-agnostic meta-learning, 2018. 1, 2

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. 1, 4, 5

[5] Q. Hou, M.-M. Cheng, X.-W. Hu, A. Borji, Z. Tu, and P. Torr. Deeply supervised salient object detection with short connections. 2016. 3

[6] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks, 2016. 5

[7] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. 4

[8] T. Munkhdalai and H. Yu. Meta networks, 2017. 2

[9] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms, 2018. 2

[10] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *In International Conference on Learning Representations (ICLR)*, 2017. 1, 2

[11] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell. Meta-learning with latent embedding optimization, 2018. 2

[12] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning, 2017. 2

[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. 1, 2, 3, 6

[14] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning, 2016. 2

[15] A. Yin. One-shot-pokemon images: Colorful and fun dataset for one shot learning problem, gotta recognize them all. https://www.kaggle.com/aaronyin/oneshotpokemon, 2018. 1, 2, 4