Few-Shot learning Based Image Classification

Saghar Irandoust
Department of Computer Science
sirandou@sfu.ca

Abolfazl Eskandarpour
Department of Engineering Science
eskandar@sfu.ca

Mohammadhadi Mohandes
Department of Engineering Science
mmohande@sfu.ca

Sedigheh Razmpour
Department of Engineering Science
srazmpou@sfu.ca

Abstract

Few-shot learning has become of great significance to generate models that generalize from few examples. In this work, we use a baseline model -a conceptually simple, flexible, and general framework for few-shot learning- where a classifier must learn to recognise new classes given only few examples from each. We employed a method, called the Relation Network (RN). Relation Network uses meta-learning to tackle the problem of few-shot classification. Instead of learning the classes themselves, meta-learning *learns to learn a metric* to compare a few images within episodes, each simulating the few-shot setting. Once trained, it can classify images of new classes by computing relation scores between image features and the few examples of each new class. We show that using transfer learning in this context can help improve the accuracy of the baseline, as very strong feature extractors like ResNet work much better than CNNs trained with very few data in the setting of Few-shot learning problem. All our code and implementation is on Github[1].

1 Introduction

Great success has been observed in visual recognition tasks such as image classification through deep learning methods. Using large numbers of labeled data in such supervised learning methods, limits their ability to recognize new classes or rare categories where numerous tested images may never exist. On the contrary, having very little supervision like few shot learning seems to work efficiently for humans in recognizing objects. Motivated by this, there has been an interest in machine few shot learning. Few-shot learning aims to recognize new classes having seen only a few labeled examples [1]. Current techniques to few-shot learning is usually to split training into Meta–learning phases where transferable knowledge is learned [2]. In the framework that we use (meta-learning) [3], we learn how to discriminate between any group of given classes called the *support set*. Our baseline method learns how to extract metrics from the support set and use it for relation comparison between the support set and a given test image [1]. Then we take this approach further by using ResNet-18 as our feature extractor.

2 Previous Works

There has been many different approaches to tackling the few-shot problem. Besides significant progress, the complexity of network designs, meta-learning algorithms, and differences in implementation details makes it difficult for a fair comparison [2]. In the following, you can see repre-

[1]https://github.com/sirandou/meta-learning-few-shot-classification

sentative few shot learning algorithms that overcome the data efficiency issues; initialization based, metric learning based and hallucination based methods.

Initialization based method solves the few-shot learning problem by "learning to fine-tune". One method is to learn *model initialization* (weights of the network) so with a small number of labeled examples and limited gradient update steps (iterations), classifiers for new classes can be learned [3]. The strategy is to search for the weight configuration of a given neural network such that it can be effectively fine-tuned on a sparse data problem within a few gradient-descent update steps [2]. Another approach learns an optimizer. For example [4], Goes further in meta-learning and includes the LSTM-based method to replace SGD optimizer. While these initialization based methods are able to achieve adaption for novel classes with small number of training sets, they suffer from the need to fine-tune the target.

Metric learning based approach solves the few-shot classification by "learning to compare". If this approach is successful to compare the similarity of two images, it can classify an unknown and unseen input image with the labeled examples [5]. Meta learning based approach make continuous detection on metric to few labeled examples to learn a comparison model. Examples of metric learning are CNN-based relation module [1] which aims to learn a distance metric to compare the target image ta few labeled images, ridge regression [6], and graph neural network [7]. [2] compares the performance of three distance metric learning methods and show that a simple baseline method with a distance-based classifier (without training over a collection of tasks/episodes as in meta-learning) achieves competitive performance with respect to other complicated algorithms.

Delusion based methods address data shortage by "learning to augment". This class of methods learns a generator from data in the base classes and use the learned generator to delude new class of data for data augmentation. One type of generator aims at transferring appearance variations exhibited in the base classes. [8] uses generators to transfer variance in base class data to novel classes. [9] uses GAN models to transfer the style. Another type of generators does not explicitly specify what to transfer, but directly integrate the generator into a meta-learning algorithm for improving the classification accuracy [10].

3 Proposed Approach

While some works focus on the learning of the transferable embedding and pre-define a fixed metric such as Euclidean, in this paper our approach is based on *meta-learning* [2]. In this framework, instead of learning how different classes look like, we learn *how to discriminate* between any group of given classes, called the *support set*.

3.1 Dataset

We consider the task of *few-shot classifier learning*. We have two datasets: a training dataset and a test dataset. The training and the test dataset have no overlap in their label spaces. In our work, we used *mini*ImageNet as datasets in both baseline and proposed architecture. This dataset consists 60,000 color images. There are 100 classes each having 600 examples in them and it's derived from imagenet dataset. It is originally divided to 64, 20 and 16 classes for training, validation and testing, respectively. In this work, we used 20 classes set for training, and the 16 classes set for testing.

3.2 Relation Network

In both training and test phases, data is represented in 2 sets. The *support set* and the *query set*. The support set contains K labeled examples for each of C unique classes (*C-way K-shot*). The goal in the test phase is to recognize the labels of the query set using information from the support set.

Episodes: An effective way for exploiting training set in few-shot learning is episode based learning. In the training phase, in each episode we select C classes with K samples from each class randomly to make our support set (1).

$$S = \{(x_i, x_j)\}_{i=1}^{m}(m = K \times C) \tag{1}$$

We select our query set (2) from the remaining data in the same C classes.

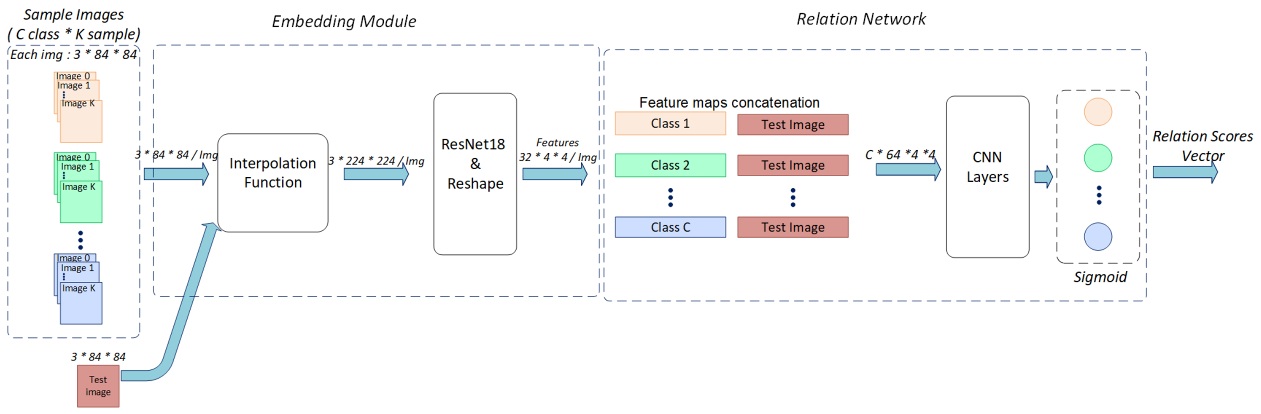


Figure 1: Relation Network architecture for C-way K-shot

$$Q = \{(x_i, x_j)\}_{j=1}^{n} \tag{2}$$

Therefore, each episode mimics the Few-shot scenario. In each training episode we update our model's parameters based on the evaluation of the model's performance on the training query set.

Networks: The system general architecture is illustrated in Fig. 1. It is the combination of two parts, embedding module and relation module. The embedding module is used to extract features from both support and query sets. The feature map (f_φ) of each sample from the query and the mean feature map of each class from the support set are combined and fed into the relation module. The relation module (g_Φ), produces a scalar in range of 0 to 1 representing the similarity between x_i (i^{th} sample) and x_j (j^{th} class in query), which is called relation score.

$$r_{(i,j)} = g_\Phi(C(f_\varphi(x_i), f_\varphi(x_j))) \tag{3}$$

The number of relation scores for one query is always C in both one-shot or few-shot setting.

Objective function: We use mean square error (MSE) loss (4) to train our model.

$$\Phi, \varphi \leftarrow argmin_{\Phi,\varphi} \sum_{i=1}^{m} \sum_{j=1}^{n} (r_{i,j} - 1(y_i == y_j))^2 \tag{4}$$

Matched pairs have similarity 1 and the mismatched pair have similarity 0.

3.3 ResNet

Residual neural networks (ResNets) are introduced to solve vanishing gradient problem in deep neural networks as the layers of networks increase [11]. ResNets apply identity mapping using residual blocks by utilizing skip connections, or shortcuts to jump over some layers Fig. 2. In fact, the activation unit from a layer could be fed directly to a deeper layer of the network. As a result, by skipping some layers which is followed by decreasing the impact of vanishing gradient, the network gradually restores the skipped layers as it learns the features. During later training, when all layers are expanded, it will stay closer to the manifold and thus learns faster.

In this project we use ResNet-18 (18 layers deep) which is a convolutional neural network and it is trained on more than a million images from the ImageNet database. In this regard, ResNet-18 is very useful and efficient in image classification and can help us to classify images into 1000 object categories. In addition, Resnet is a powerful feature extractor and its output from embedding module leads to better final feature concatenation and accuracy.

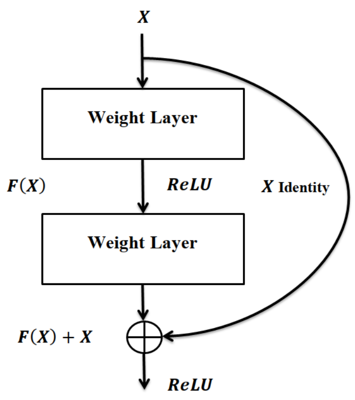


Figure 2: Residual learning: a building block

4 Experimental Results

4.1 Network Specifics and Training

In our structure we finetuned ResNet-18 removing its last fully connected layer, and used it as our embedding module. We normalized and interpolated input images in order to change their dimension from $82 * 82$ and feed them into ResNet. We also reshaped the output feature vector into a feature tensor suitable to be fed to the relation module.

The relation module is a 2 layer CNN network, using sigmoid as its last activation function. There are BatchNorm and MaxPooling layers after each CNN layer that help overcome overfitting and help reduce the dimensionality respectively. The detailed architecture is illustrated in Fig. 3.

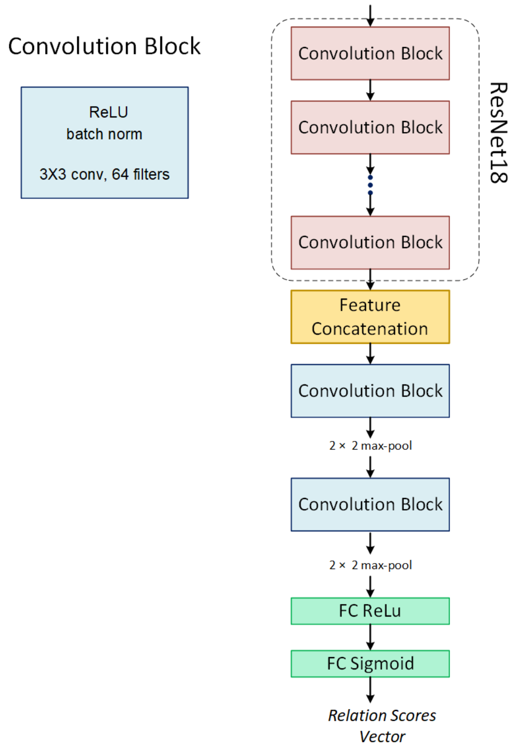


Figure 3: Network Architecture Layers.

The baseline model [1] uses a 4 layer CNN encoder as its embedding module. After implementing both the baseline paper and our model, we trained the 2 networks in both models for 1000 episodes, and the plot of training loss per episode can be seen in Fig. 4 and Fig. 5. We can see that our model has been able to obtain lower training loss than the baseline. In addition, it seems the baseline stops improving after a few episodes. While our model still seems to be improving even after 1000 episodes. At the end, the final accuracy of both systems under the same settings are mentioned in Fig. 6. Dataset miniImageNet [12] is used in both systems' implementation.

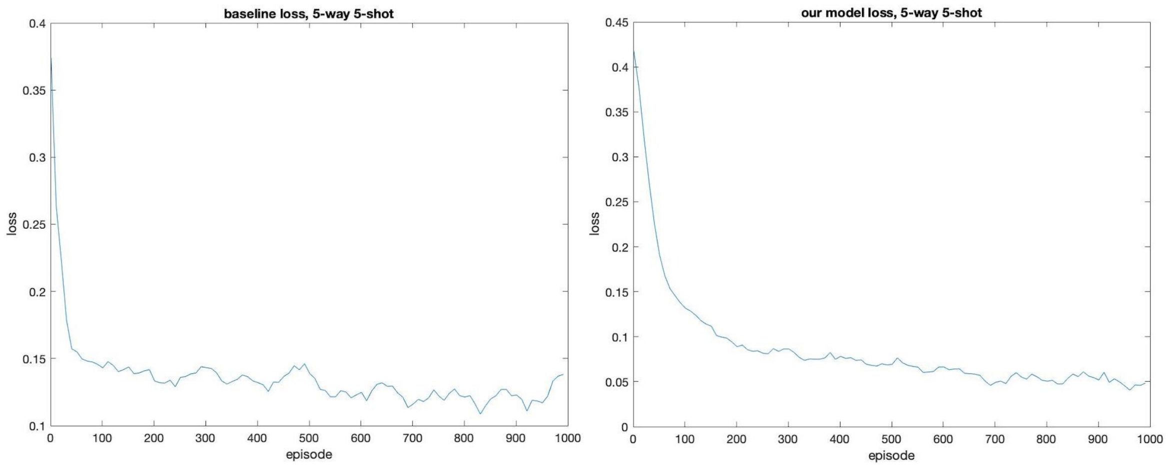


Figure 4: 5-way 5-shot Training Loss

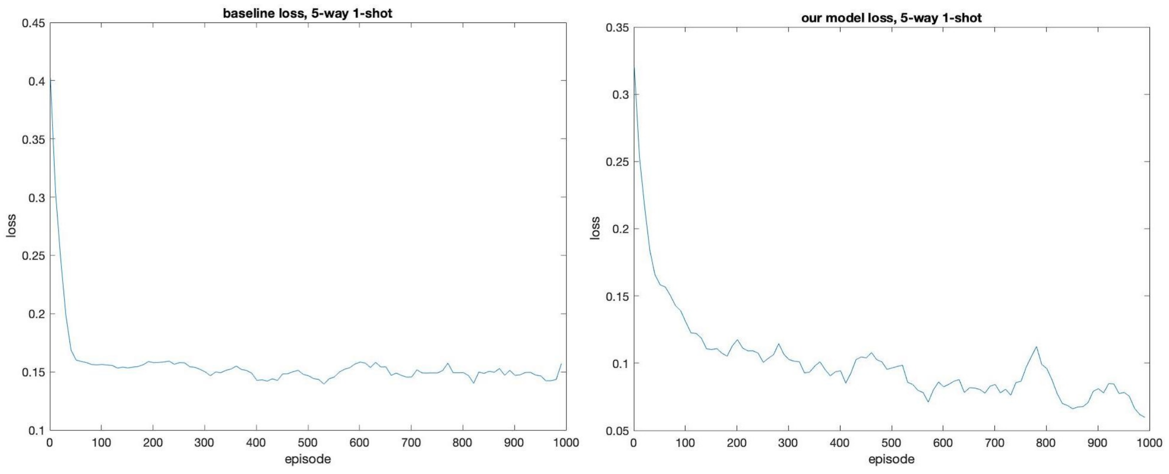
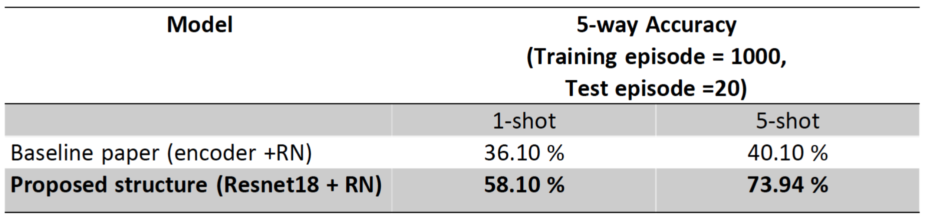


Figure 5: 5-way 1-shot Training Loss



Model	5-way Accuracy (Training episode = 1000, Test episode =20)	
	1-shot	5-shot
Baseline paper (encoder +RN)	36.10 %	40.10 %
Proposed structure (Resnet18 + RN)	**58.10 %**	**73.94 %**

Figure 6: Baseline VS our Accuracy

4.2 Qualitative Results

In Fig. 7, we can see our model's behaviour on a completely random test task. This task is an example of the 5-way 5-shot problem. Given are 25 images belonging to 5 classes and their labels, along with some unlabeled images. The interesting part is that our networks have not seen any of these classes in training phase, but are able to find the relationship among corresponding image samples anyways. The accuracy in this example is 76%, which is close to the mean value of 1000 test episode's accuracies, 73.94%.

By testing the trained networks on the training dataset, we can see in Fig. 8 that the accuracy is 96% and only 1 label is predicted incorrectly. However this experiment is only done out of curiosity and the main purpose of this model is to perform well with unseen datasets.

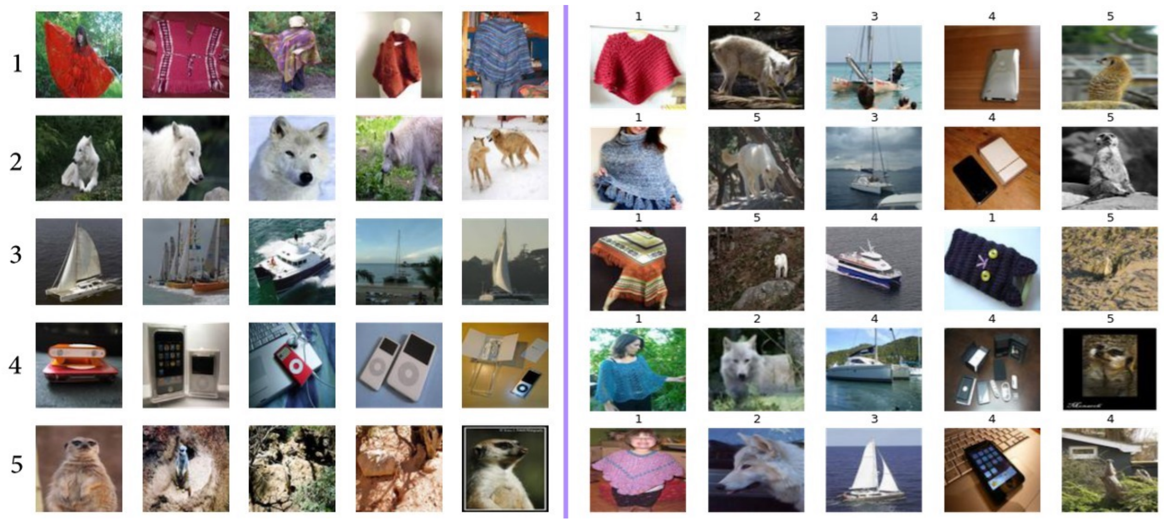


Figure 7: 5-way 5-shot Results on the test dataset. On the left we can see a random support set sampled from the test dataset, each row representing 1 class. On the right we can see our query set and their predicted labels. 76% of the labels are predicted correctly.

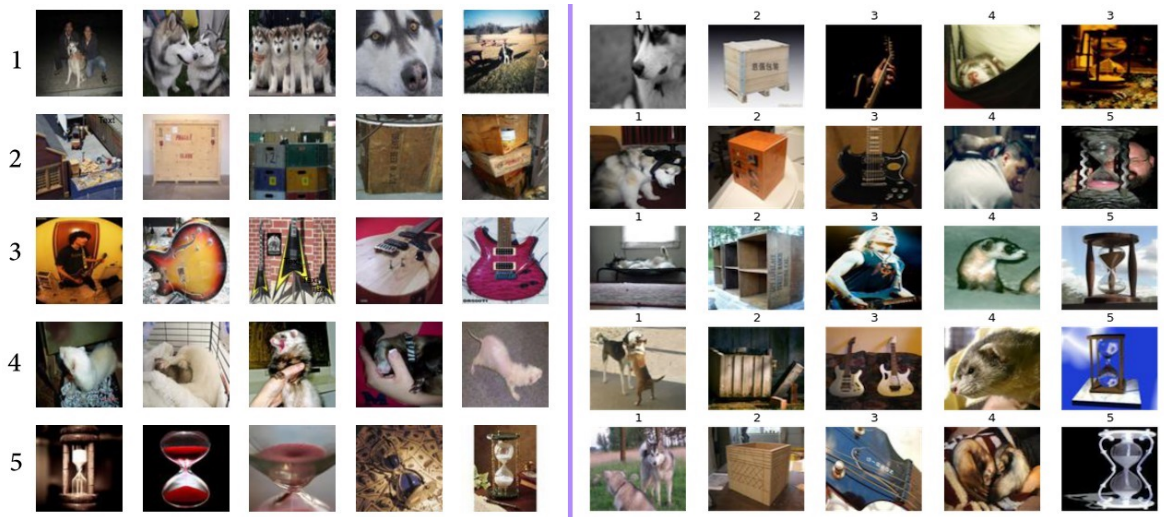


Figure 8: 5-way 5-shot Results on the training dataset. On the left we can see a random support set sampled from the train dataset, so this time our class labels are not new to our networks. On the right we can see our query set and their predicted labels. 96% of the labels are predicted correctly.

5 Conclusion

We implemented a platform for few-shot learning, which exploits meta-learning for better accuracy in test set. We both implemented baseline [1] and also our proposed architecture. As it is illustrated and compared in the table, the proposed architecture has better performance in the equal situation. ResNet-18 is a powerful feature extractor which leads to better final feature concatenation and accuracy.

Contributions

- **Saghar Irandoust**: Literature study, preparation of codes and experiments (implementing the baseline and our model, training our model on Google Colab, generating results, creating github repository), participated in writing the report.
- **Sedigheh Razmpour**: Reading few-shot papers (baseline and another related), reading implementation codes of baseline paper, preparing the poster, generating shapes, participated in preparing the report.
- **Abolfazl Eskandarpour**: Helped in preparing the miniimagenet dataset, training the baseline model on Google Colab and generating results, helped in writing the report.
- **Mohammadhadi Mohandes**: Writing the Abstract, Introduction and Related work parts of report and the poster. Reading the baseline paper and another few-shot learning paper. Helping to generate shapes and preparing the poster.

References

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

[2] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

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

[4] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[5] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[6] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018.

[7] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.

[8] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027, 2017.

[9] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

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

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.