Meta Networks

Tsendasuren Munkhdalai 1  Hong Yu 1

Abstract

Deep neural networks have been successfully applied in applications with a large amount of labeled data. However, there are major drawbacks of the neural networks that are related to rapid generalization with small data and continual learning of new concepts without forgetting. We present a novel meta learning method, Meta Networks (MetaNet), that acquires a meta-level knowledge across tasks and shifts its inductive bias via fast parameterization for the rapid generalization. When tested on the standard one-shot learning benchmarks, our MetaNet models achieved near human-level accuracy. We demonstrated several appealing properties of MetaNet relating to generalization and continual learning.

1. Introduction

Deep neural networks have shown great success in several application domains with a large amount of labeled data available. However, the availability of such large data from which training signals can be extracted have been prerequisite. Furthermore the standard deep neural networks lack an ability to incrementally learn new concepts on the fly, without forgetting the old knowledge.

In contrast, humans can rapidly learn and generalize from one or two examples of the same concept. Humans are also very good at incremental (i.e. continual) learning. These abilities are mostly explained by the meta learning (i.e. learning to learn) process in the brain (Harlow, 1949).

Previous work on meta learning has formulated the problem as a two-level learning, a slow learning of a meta-level model performing across tasks and a rapid learning of a base-level model acting within each task (Mitchell et al., 1993; Vilalta & Drissi, 2002). The goal of the meta-level learner is to acquire generic knowledge of different tasks. The knowledge can then be transferred to the base-level learner to provide generalization in the context of a single task. The base and the meta-level models can be framed in a single learner (Schmidhuber, 1993) or in separate learners (Bengio et al., 1990; Hochreiter et al., 2001).

In this work we introduce a meta learning model called MetaNet (for Meta Networks) that supports meta-level continual learning by allowing neural networks to learn and to generalize new task or new concept from a single example on the fly. The overall architecture of MetaNet is shown in Figure 1. MetaNet consists of two main learning components, a base learner and a meta learner, and is equipped with an external memory. Learning occurs at two levels in separate spaces (i.e. meta space and task space). The base learner performs in the input task space whereas the meta learner operates in a task-agnostic meta space. By operating in the abstract meta space, the meta learner supports continual learning and performs meta knowledge acquisition across different tasks. To this end, the base learner first analyzes the input task. It then provides the meta learner with a meta-level information that explains the base learner’s status in the current task space. Based on the meta information, the meta learner rapidly parameterizes both itself and the base learner so that the MetaNet model can recognize the new concepts of the input task. Specifically the training weights of MetaNet evolves at different timescales: standard slow weights updated through a learning algorithm (i.e. backpropagation), task-level fast weights
updated within the scope of each task, and example-level fast weights updated for a specific input example. Finally MetaNet equipped with the external memory allows for such rapid learning and generalization.

Under the MetaNet framework, it is important to define types of the meta information which can be derived from the base learner. While other types of the meta information are also applicable, here we used the gradient of the base learner as meta information for an end-to-end training of MetaNet via backpropagation. We extensively studied its performance and characteristics on one-shot learning problems, under varying settings. Our proposed method not only improves the state-of-the-art results on standard benchmarks but also shows some interesting properties relating to generalization and continual learning.

2. Related Work

Our work connects different threads of research in order to allow neural networks for rapid learning and generalization. The rapid learning and generalization refers to a one-shot learning scenario where a learner is introduced to a sequence of tasks, each with a single or few labeled examples of many classes. A key challenge in this setting is that the classes or concepts vary across the tasks. Due to this, one-shot learning problems have been widely addressed by generative models and metric learning methods. One notable success is reported by a probabilistic programming approach (Lake et al., 2015). They used specific knowledge of how pen strokes are composed to produce characters of different alphabets. Koch (2015) applied Siamese Networks to perform one-shot classification. Recently, Vinyals et al. (2016) unified the training and testing of a one-shot learner under the same procedure and developed an end-to-end differentiable nearest neighbor method for one-shot learning. Santoro et al. (2016) proposed a memory-based approach and trained Neural Turing Machines (Graves et al., 2014) for one-shot learning, although the meta-learner and the one-shot learner in this work are not separable explicitly. Their training procedure adapted the work of Hochreiter et al. (2001) in which they used LSTMs as the meta-level model. More recently an LSTM-based one-shot optimizer was proposed (Ravi & Larochell, 2017). By taking in the loss, the gradient and the parameters of the base learner, the meta optimizer was trained to update the parameters for one-shot classification.

There is a line of work on building meta optimizers (Hochreiter et al., 2001; Andrychowicz et al., 2016; Li & Malik, 2017). As the main interest here is to train an optimization algorithm within the meta learning framework, the previous research have mainly focused on tasks with large datasets. In contrast, with the absent of large datasets, we emphasize the difficulties to optimize a neural network with a large number of parameters to generalize with limited examples of new concept. Our work instead proposes a novel rapid parameterization approach by employing the meta information. By following the success of the previous work (Mitchell et al., 1993; Younger et al., 1999; Andrychowicz et al., 2016; Ravi & Larochell, 2017), we studied the meta information in the form of the loss gradient of neural nets. The ideas of fast weights and generating parameters for one neural network with another have been explored individually in separate contexts. Hinton & Plaut (1987) suggested the usage of fast weights for rapid learning. Ba et al. (2016) recently used fast weights to replace soft attention mechanism. Fast weights have also been used to implement recurrent nets (Schmidhuber, 1992). Those usages of fast weights are well motivated by the fact that synapses have dynamics at many different time-scales.

The approach proposed by Gomez & Schmidhuber (2005) is more closely related to our work. They used recurrent nets to generate fast weights for a single-layer network controller. De Brabandere et al. (2016) used one network to generate slow filter weights for a convolutional neural net whereas more recently David Ha & Le (2017) generated slow weights for recurrent nets. Our MetaNet generates fast weights at two time-scales by operating in meta space. To integrate the fast weights with the slow weights, we propose a novel layer augmentation approach.

Finally, we note that our MetaNet equipped with an external memory can be seen as a memory augmented neural network (MANN). MANNs have shown promising results on a range of tasks starting from small programming problems (Graves et al., 2014) to large-scale language tasks (Weston et al., 2015; Munkhdalai & Yu, 2017).

3. Meta Networks

The model consists of two main learning modules (Figure 1). The meta learner that operates across tasks is responsible for fast weight parameterization. These fast weights shifts the inductive bias of a neural network and are integrated into the base learner and the meta learner. Particularly the base learner parameterized by the fast weights is now able to generalize well for the new task. We propose a novel layer augmentation method to integrate the standard slow weights and the task or example specific fast weights in a neural net.

3.1. Base Learner

The base learner denoted as b is a function or a neural net that maps an input example x (i.e. image) to an output y (i.e. label). However, unlike standard neural net b is parameterized by slow weights W and example-level fast weights W^*. The slow weights are updated via a learning algorithm
Meta Networks

![Figure 2. A layer augmented MLP](image)

During training whereas the fast weights are generated for every input.

By using some sort of meta information the base learner should be able to explain its setting in the current task space in order for the meta learner to generate the fast weights. We obtain such meta information in the form of the gradient by first performing a forward pass and then backpropagating as follows:

\[
\mathcal{L}_i = \text{loss}_b(b(W, x_i), y_i) \tag{1}
\]

\[
\nabla_i = \nabla_W \mathcal{L}_i \tag{2}
\]

where \(\mathcal{L}_i\) is the loss for input and label pair \(\{x_i, y_i\}_{i=1}^N\). \(N\) is the number of support examples in the task set which usually contains a single instance per class in one-shot setup. \(\nabla_i\) is the loss gradient with respect to parameters \(W\) and is our meta information. Note that the loss function \(\text{loss}_b\) is generic and for one-shot classification we use cross-entropy loss. Now the meta learner takes in the gradient information and generates the fast parameters \(W^*_i\). We will describe this later.

Assuming that the fast weights \(W^*_j\) for input \(x_j\) is defined, the base learner performs the one-shot classification as:

\[
P(y_j|x_j, W, W^*_j) = b(W, W^*_j, x_j) \tag{3}
\]

where \(\{x_j\}_{j=1}^L\) is an example drawn from the current task set. To train MetaNet, we use another \(L\) number of labeled examples other than the support set and minimize the cross-entropy loss.

3.2. Meta Learner

The meta learner processes the meta information along with the task support set and generates the fast weights.

Particularly it transforms the loss gradient \(\{\nabla_i\}_{i=1}^N\) to a set of fast weights \(\{W^*_i\}_{i=1}^N\):

\[
W^*_i = m(Z, \nabla_i) \tag{4}
\]

where \(m\) is a neural network with parameter \(Z\). The fast weights are then stored in a memory \(M = \{W^*_i\}_{i=1}^N\). The memory \(M\) is indexed with task dependent representations \(R = \{r_i\}_{i=1}^N\) of the support set \(\{x_i\}_{i=1}^N\), obtained by a dynamic key embedding function \(u\). The embedding function \(u\) is a neural net parameterized by slow weights \(Q\) and task-level fast weights \(Q^*\):

\[
r_i = u(Q, Q^*, x_i) \tag{5}
\]

where the parameters \(Q\) and \(Q^*\) are integrated using the layer augmentation method. Note that the fast weights \(Q^*\) are generated per task basis to perform a task-specific embedding of input. Therefore \(Q^*\) is defined as:

\[
\mathcal{L}_t = \text{loss}_a(u(Q, x_t), y_t) \tag{6}
\]

\[
\nabla_t = \nabla_Q \mathcal{L}_t \tag{7}
\]

\[
Q^* = m_{LSTM}(G, \nabla_i) \tag{8}
\]

where \(m_{LSTM}\) is LSTM and \(G\) is the LSTM parameters. First, we sample \(T\) examples (possibly \(T = N\)) \(\{x_t, y_t\}_{t=1}^T\) from the support set and obtain the loss gradient as meta information. Then LSTM observes the gradient corresponding to each sampled example and summarizes into the task specific parameter. Thus the order of LSTM input does not matter and alternatively we can take summation or average of the gradients and use a MLP. However, in our preliminary experiment we observed that the latter results in a poor convergence. In a similar way defined in the Equation 6-8, we can also parameterize the base learner with task-level fast weights. We will report an ablation experiment on different variation of MetaNet in Section 4.

The loss, \(\text{loss}_a\) does not need to align with the base learner’s loss, but it should be able to capture a good representation learning objective. We use cross-entropy loss when the support set has only a single example per class. However, when there are more than one examples per class available, contrastive loss (Chopra et al., 2005) is a natural choice for \(\text{loss}_a\) since both positive and negative samples can be formed. In this case, we randomly draw \(T\) number of pairs to observe the gradients and the loss is

\[
\mathcal{L}_t = \text{loss}_a(u(Q, x_{1,t}), u(Q, x_{2,t}), l_t) \tag{9}
\]

where

\[
l_t = \begin{cases} 1, & \text{if } y_{1,t} = y_{2,t} \\ 0, & \text{otherwise} \end{cases} \tag{10}
\]
Once the parameters are stored in the memory $M$ and the memory index $R$ is constructed, the meta learner parameterizes the base learner with the fast weights $W_j^*$. First it embeds the input $x_j$ in the task space:

$$r_j = u(Q, Q^*, x_j)$$

and then reads the memory with soft attention:

$$a_j = \text{attention}(R, r_j)$$

$$W_j^* = \text{softmax}(a_j)^\top M$$

where attention calculates similarity between the memory index and the input embedding and we use cosine similarity as attention.

The meta learner fully parameterize its embedding function and the base learner and thus the shape of $W_j^*$ and $Q_j^*$ are same as $W_j$ and $Q$. The embedding function $u$ can be seen as a dynamic embedding network that changes its parameters within the scope of the current input task. The training parameters of MetaNet $\theta$ consists of the slow weights $W$ and $Q$ and the meta weights $Z$ and $G$ (i.e. $\theta = \{W, Q, Z, G\}$) and jointly updated via a training algorithm such as backpropagation to minimize the cross-entropy loss (Equation 3). Note that $\text{loss}_s$ and $\text{loss}_u$ in Equations 1 and 6 are calculated to obtain the gradient by using the support set during both training and testing for the rapid parameterization and these losses are not used for $\theta$ parameter updates.

### 3.3. Layer Augmentation

A slow weight layer in the base learner is extended with its corresponding fast weights for rapid generalization. An example of the layer augmentation approach applied to a MLP with single hidden layer is shown in Figure 2. Input to each layer is first transformed by both slow and fast weights and then it is passed through a non-linearity (i.e. ReLU) resulting in two separate activation vectors. Finally the activation vectors are aggregated by an element-wise vector addition. For the last softmax layer, we first aggregate two transformed inputs and then normalize for classification output.

Overall the layer augmentation building block is similar to that of ResNet architecture (He et al., 2016). However, ResNet has an identity transformation of a preceding activation (i.e. shortcut connection) and a non-linearity after the element-wise additive aggregation while our layer augmented net has a fast weight mapping which captures the input task prior and a non-linearity before the aggregation. Intuitively, the fast and slow weights in the layer augmented neural net can be seen as feature detectors operating in two distinct numeric domains. The application of the non-linearity maps them into the same domain, which is $[0, \infty)$ in the case of ReLU so that the activations can be aggregated and processed further. Our aggregation function here is element-wise sum.

### 4. Results

We carried out one-shot classification experiments on two different image datasets: Omniglot and MNIST. The Omniglot dataset consists of images across 1623 classes with only 20 images per class, from 50 different alphabets (Lake et al., 2015). It also comes with a standard split of 30 training and 20 evaluation alphabets. Following (Santoro et al., 2016), we augmented the data set through 90, 180 and 270 degrees rotations. The images are resized to 28 x 28 pixels for computational efficiency. We used the MNIST images as an out-of-domain data throughout our experiments. MNIST was also downscaled to the same size.

#### 4.1. Training Details

To train and test MetaNet on one-shot learning, we adapted the training procedure introduced by Vinyals et al. (2016). First we split the Omniglot data into training and test sets consisting of two disjoint classes. We then formulate a series of tasks from the training set. Each task has a support set of $N$ images and $N$ different classes, resulting an N-way one-shot classification problem. In addition to the support set, we also include $L$ number of labeled examples in each task set to update the parameters $\theta$ during training. For testing, we follow the same procedure to form a set of test tasks from the disjoint classes. However, now MetaNet assigns class labels to $L$ examples based only on the labeled support set of each test task.

For the one-shot benchmarks reported in the following sections, we used a CNN as base learner $b$. This CNN has 5 convolutional layers, each of which is a 3 x 3 convolution with 64 filters, followed by a ReLU non-linearity and a 2 x 2 max-pooling and additional two layers of a fully connected (FC) layer and a softmax layer. Another CNN with the same architecture is used define the dynamic embedding function $u$ and we take the output of the FC layer as the task dependent representation $r$. For the networks $m_{\text{LSTM}}$ and $m$, we used a single-layer LSTM with 20 hidden units and a three-layer MLP with 20 hidden units and ReLU non-linearity. As in Andrychowicz et al. (2016), the parameters $G$ and $Z$ of $m_{\text{LSTM}}$ and $m$ are shared across the coordinates of the gradients $\nabla$ and the gradients are normalized using the same preprocessing rule (with $p = 7$) before it is given to the meta learner. The MetaNet parameters $\theta$ is optimized with ADAM. The initial learning rate was set to $10^{-3}$. The model parameters $\theta$ were randomly initialized from the uniform distribution over [-0.1, 0.1).
### 4.2. One-shot Learning Test

In this section we will report three different groups of benchmark experiments: Omniglot previous split, MNIST as out-of-domain data and Omniglot standard split.

#### 4.2.1. OMNIGLOT PREVIOUS SPLIT

Following the previous setup Vinyals et al. (2016), we split the Omniglot classes into 1200 and 423 classes for training and testing. We performed 5, 10, 15 and 20-way one-shot classification and compared our performance against the state-of-the-art results. We also studied three variations of MetaNet as an ablation experiment in order to show how fast parameterization affects the network dynamics.

In Table 1, we compared the performance of our models with all published models (as baselines). The first group of methods are the previously published models. The next group is MetaNet variations. MetaNet is the main architecture described in Section 3. MetaNet- is a variant without task-level fast weights $Q^*$ in the embedding function $u$ whereas MetaNet+ has additional task-level weights for the base learner in addition to $W^*$. Our MetaNet model improves the previous best results by 0.5% to 2% accuracy. As the number of classes increases (moving from 5-way to 20-way classification), overall the performance of the one-shot learners decreases. MetaNet’s performance drop is relatively small and it is around 2% while the drop for the other models ranges from 3% to 15%. As a result, our model shows an absolute improvement of 2% on 20-way one-shot task.

Comparing different MetaNet variations, the additional task-level weights in the base learner did not seem to help and even this model (MetaNet+) has shown a decreasing result. MetaNet- however performed surprisingly well but still falls behind the MetaNet model as it lacks the dynamic embedding function. This performance gap even increases when we test them on the following out-of-the domain setting.

#### 4.2.2. MNIST AS OUT-OF-DOMAIN DATA

We treated MNIST images as a separate domain data. Particularly a model is trained on the Omniglot training set and evaluated on the MNIST test set in 10-way one-shot learning setup. We hypothesize that models with a high dynamic should perform well on this task.

In Figure 3, we plotted the results of this experiment. MetaNet- achieved 71.6% accuracy which was 0.6% and 3.2% lower than the other variants with fast weights. This is not surprising since MetaNet without dynamic embedding function lacks an ability to adapt its parameters to MNIST image representations. The standard MetaNet model achieved 74.8% while MetaNet+ obtained 72.3% while Matching Net (Vinyals et al., 2016) reported 72.0% accuracy in this setup. Again we did not observe improvement with MetaNet+ model here. The standard MetaNet model has shown the best performance by effectively adapting its bias to the new data distribution.

#### 4.2.3. OMNIGLOT STANDARD SPLIT

Omniglot data comes with a standard split of 30 training alphabets with 964 classes and 20 evaluation alphabets with 659 classes. We trained and tested only the standard MetaNet model in this setup. In order to best match the evaluation protocol of Lake et al. (2015), we form 400 tasks

### Table 1. One-shot accuracy on Omniglot previous split

| Model | 5-way | 10-way | 15-way | 20-way |
|-------|-------|--------|--------|--------|
| Pixel kNN (Kaiser et al., 2017) | 41.7 | -      | -      | 26.7   |
| Siamese Net (Koch, 2015) | 97.3 | -      | -      | 88.1   |
| MANN (Santoro et al., 2016) | 82.8 | -      | -      | 93.8   |
| Matching Nets (Vinyals et al., 2016) | 98.1 | -      | -      | 95.0   |
| Siamese Net with Memory (Kaiser et al., 2017) | 98.4 | 98.32 | 96.68  | 97.0   |
| MetaNet- | 98.4  | 98.32 | 96.68  | 96.13  |
| MetaNet | 98.95 | 98.67 | 97.11  | 97.0   |
| MetaNet+ | 98.45 | 97.05 | 96.48  | 95.08  |

Figure 3. MNIST 10-way one-shot classification results.
Table 2. One-shot accuracy on Omniglot standard split

| Model                                           | 5-way | 10-way | 15-way | 20-way |
|------------------------------------------------|-------|--------|--------|--------|
| Human performance (Lake et al., 2015)          | -     | -      | -      | 95.5   |
| Pixel kNN (Lake et al., 2013)                  | -     | -      | -      | 21.7   |
| Affine model (Lake et al., 2013)               | -     | -      | -      | 81.8   |
| Deep Boltzmann Machines (Lake et al., 2013)    | -     | -      | -      | 62.0   |
| Hierarchical Bayesian Program Learning (Lake et al., 2015) | -     | -      | -      | 96.7   |
| Siamese Net (Koch, 2015)                       | -     | -      | -      | 92.0   |
| MetaNet                                         | 98.45 | 97.32  | 96.4   | 95.92  |

In Table 2, we listed the MetaNet results along with the previous models and human performance. Our MetaNet outperformed the human performance by a slight margin, but underperformed the probabilistic programming approach. However, the performance gap is rather small between these top three baselines. In addition while the probabilistic programming performs slightly better than MetaNet, our model does not rely on any extra prior knowledge about how characters and strokes are composed. Comparing the results on two Omniglot splits in Tables 1 and 2, MetaNet showed decreasing performances on the standard split. The later setup seems to be slightly difficult as the number of classes in the training set is less (1200 vs 964) and test classes are bigger (423 vs 659).

4.3. Generalization Test

We conducted a set of experiments to test the generalization of MetaNet from multiple aspects. The first experiment is to test whether a MetaNet model skilled on N-way one-shot task could generalize to another K-way task (where $N \neq K$) without actually training on the second task. The second experiment is set to see if a meta learner trained for rapid parameterization of a base learner $b_{\text{train}}$ could parameterize other base learner $b_{\text{eval}}$ during evaluation. The last group of experiments we performed in the scope of MetaNet generalization test is to show that MetaNet supports a meta-level continual learning.

Table 3. Accuracy of MetaNet trained on N-way and tested on K-way one-shot tasks

| Train | 5-way | 10-way | 15-way | 20-way |
|-------|-------|--------|--------|--------|
| 5-way | 98.95 | 96.4   | 93.6   | 93.07  |
| 10-way| 99.25 | 96.87  | 96.95  | 96.21  |
| 15-way| 99.35 | 98.17  | 97.11  | 96.36  |
| 20-way| 99.55 | 98.87  | 97.41  | 97.0   |

4.3.1. N-way Training and K-way Testing

We first trained MetaNet on N-way one-shot classification task and then tested it on K-way one-shot tasks. The number of training and test classes are varied (i.e. $N \neq K$). To handle this, we inserted a softmax layer into the base learner during evaluation and then augmented it with the fast weights generated by the meta learner. If the meta learner is generic enough, it should be able to parameterize the new softmax layer on the fly. The new layer weights remained fixed since no parameter update was performed for this layer. The K-way test tasks were formed from the 423 unseen classes in the test set.

The MetaNet models were trained on one of 5, 10, 15 and 20-way one-shot tasks and evaluated on the rest. In Table 3, we summarized the results. As a comparison we also included some results from Table 1, which reports accuracy of N-way train and test setting. The MetaNet model trained on 5-way tasks obtained 93.07% of 20-way test accuracy which is still a closer match to Matching Network and higher than Siamese Net trained 20-way tasks. An interesting finding is that when $N$ is smaller than $K$, i.e. the model is trained on easier tasks than test ones, we observe a decreasing performance. Conversely the models skilled on harder tasks (i.e. $N > K$) achieved increasing performances when tested on the easier tasks and the performance is even higher than the ones that were applied to the tasks with the same level difficulty (i.e. $N = K$). For example, the model skilled on 20-way classification improved the 5-way one-shot baseline by 0.6% showing a ceiling performance in this setting. We also conducted a preliminary experiment on more extreme test-time classification. MetaNet trained on 10-way task achieved around 65% on 100-way one-shot classification task.

This flexibly in MetaNet is crucial because one-shot learning usually involves an online concept identification scenario. By looking at the results, we can empirically obtain a performance lower or upper bound. Particularly the test performance obtained on the tasks with the same level difficulty that the model was skilled on can be used as a performance lower or an upper bound depending on a scenario under which the model will be deployed in the future. For
4.3.2. RAPID PARAMETERIZATION OF FIXED WEIGHT BASE LEARNER

We replaced the entire base learner with a new CNN during evaluation. The slow weights of this network remained fixed. The fast weights are generated by the meta learner that is trained to parameterize the old base learner and used to augmented the fixed slow weights.

We tested a small and a big CNN for the base learner. The small CNN has 32 filters and the big one has 128 filters. These CNNs consist of the same number of layers as the base learner (described in Section 4.1). In Figure 4, the test performances of these CNNs are compared. The base learner (target CNN) optimized along within the model performed better than the fixed weight CNNs. The performance difference between these models is large in earlier training iterations. However, as the meta learner sees more one-shot learning trials, their test accuracies of the base learners are to match. This results show that MetaNet effectively learns to parameterize a neural net with fixed weights.

4.3.3. META-LEVEL CONTINUAL LEARNING

MetaNet operates in two spaces: input problem space and meta (gradient) space. If the meta space is problem independent, MetaNet should support meta-level continual learning or life-long learning. This experiment is set to examine this in the case of the loss gradient.

Following the previous work on catastrophic forgetting in neural networks (Srivastava et al., 2013; Goodfellow et al., 2014; Kirkpatrick et al., 2016), we formulated two problems in a sequential manner. We first trained and tested the model on the Omniglot sets and then we switched and continued training on the MNIST data. After training on a number of MNIST one-shot tasks, we re-evaluated the model on the same Omniglot test set and compare the two performances. A decreasing performance indicates that the meta weights $Z$ and $G$ of the neural nets $m$ and $m^{LSTM}$ are prone to catastrophic forgetting and MetaNet therefore does not support the continual learning. An increasing performance means MetaNet supporting a reverse transfer learning and the continual learning.

We allocated separate parameters for the weights $W$ and $Q$ when we switched the problems so the only meta weights were updated. We used two three-layer MLPs with 64 hidden units as the embedding function and the base learner. The MNIST image and classes were augmented by randomly permuting the pixels. We created 50 different random shuffles and thus the training set for the second one-shot problem consisted of 500 classes. We conducted multiple runs and increased the MNIST training trials by multiples of 400 (i.e. 400, 800, 1200...) in each run giving more time to MetaNet to adapt its meta weights on the second problem so MetaNet may forget the knowledge about Omniglot. Each run was repeated five times and we reported the average statistics. For every run, the network and the optimizer were reinitialized and the training started from scratch.

In Figure 5, we plotted the accuracy difference between two Omniglot test performances obtained before and after training on MNIST task. The performance improvement (y-axis) after skilled on the MNIST tasks ranges from -1.7% to 1.24% depending on the training time (x-axis). The positive values indicate that the training on the second problem automatically improves the performance of the earlier task exhibiting the reverse transfer. MetaNet performs the reverse transfer. At the same time, it skilled on MNIST one-shot classification. The MNIST training accu-
racy already reached over 72% after seeing 2400 MNIST trials. However, the reverse transfer happens up to a certain point (2400 trials). As we gave more time to train on MNIST, the meta weights started to forget the Omniglot information and from 2800 trials the Omniglot test performance dropped. Nevertheless, the biggest drop we observed by running the model up to 7600 trials, at which point the MNIST training accuracy reached over 90%, was only 1.7%.

5. Discussion and Future Work

One-shot learning in combination with meta learning framework can be a useful approach to address certain neural network drawbacks relating to the rapid generalization with small data and the continual learning. We present a novel meta learning model, Meta Networks (MetaNet), that performs a generic knowledge acquisition in a meta space and shifts the parameters and the inductive bias via fast parameterization for the rapid generalization.

Under the MetaNet framework, an important consideration is the type of meta information that can be extracted from the model operating on a new task. We require that the meta information should be generic and problem independent. It should also be expressive enough to explain the model setting in the current task space. We explored the loss gradient in this work. As shown in the results, using the gradient as meta information seems to be a promising direction. We evaluated the MetaNet model on the image one-shot classification problems. MetaNet obtained the state-of-the-art results on several benchmarks and has shown to be very flexible in the online setups. It supported continual learning up to a certain point. It must be interesting to challenge a similar setup on two very different tasks, such as image classification versus text classification. We leave this for the future evaluation. It might of a long term interest to come up with a new type of meta information that is more robust and expressive. One could take inspiration from the meta learning process in the brain and ask whether the brain operates on some kind of meta information to generalize across tasks and acquire new skills.

The rapid parameterization approach presented here has been shown to be a good alternative to the direct optimization methods that learn to update network parameters for one-shot generalization. However, a problem this approach poses is the integration of slow and fast weights. As a solution to this, we presented a simple layer augmentation method. Although the layer augmentation worked reasonably well, this method is questionable when a neural net has many types of parameters operating in multiple different time-scales. An example of this is that the base learner equipped with three types of weights (slow, example-specific, and task-level weights) integrated under the layer augmentation paradigm could not perform as well as a simpler one. Therefore, a potential extension would be to train MetaNet so it can discover its own augmentation schema for efficiency.

In this work, we applied the fast parameterization approach to CNNs and MLPs. The fast parameterization of sequential models such as RNNs have yet to be explored and this would have many useful applications of sequence modeling and language-based conversational tasks.

It was observed that neural nets with the fixed slow weight can perform well for new task inputs when augmented with the fast weights. When the slow weights are updated during training, it encodes the domain bias resulting in even better performance. However, one could expect a higher performance from the fixed weight network when aiming for one-shot generalization across very distinct domains.

Finally, important components of MetaNet are the dynamic key embedding function and the external memory. Indeed, similar modules can also be found in the previous models such as Matching Net and memory based Siamese Net. The differences are that MetaNet learns to parameterize this embedding network as a response to the new task inputs yielding a dynamic attention and the MetaNet memory temporary stores neural net weights generated.

References

Andrychowicz, Marcin, Denil, Misha, Gomez, Sergio, Hoffman, Matthew W, Pfau, David, Schaul, Tom, and de Freitas, Nando. Learning to learn by gradient descent. In Advances in Neural Information Processing Systems, pp. 3981–3989, 2016.

Ba, Jimmy, Hinton, Geoffrey E, Mnih, Volodymyr, Leibo, Joel Z, and Ionescu, Catalin. Using fast weights to attend to the recent past. In Advances In Neural Information Processing Systems, pp. 4331–4339, 2016.

Bengio, Yoshua, Bengio, Samy, and Cloutier, Jocelyn. Learning a synaptic learning rule. Université de Montréal, Département d’informatique et de recherche opérationnelle, 1990.

Chopra, Sumit, Hadsell, Raia, and LeCun, Yann. Learning a similarity metric discriminatively, with application to face verification. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pp. 539–546. IEEE, 2005.

David Ha, Andrew Dai and Le, Quoc V. Hypernetworks. In ICLR 2017, 2017.

De Brabandere, Bert, Jia, Xu, Tuytelaars, Tinne, and Van Gool, Luc. Dynamic filter networks. In Neural Information Processing Systems (NIPS), 2016.
Meta Networks

Gomez, Faustino and Schmidhuber, Jürgen. Evolving modular fast-weight networks for control. In *International Conference on Artificial Neural Networks*, pp. 383–389. Springer, 2005.

Goodfellow, Ian J, Mirza, Mehdi, Xiao, Da, Courville, Aaron, and Bengio, Yoshua. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *ICLR 2014*, 2014.

Graves, Alex, Wayne, Greg, and Danihelka, Ivo. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Harlow, Harry F. The formation of learning sets. *Psychological review*, 56(1):51, 1949.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hinton, Geoffrey E and Plaut, David C. Using fast weights to deblur old memories. In *Proceedings of the ninth annual conference of the Cognitive Science Society*, pp. 177–186, 1987.

Hochreiter, Sepp, Younger, A Steven, and Conwell, Peter R. Learning to learn by using gradient descent. In *International Conference on Artificial Neural Networks*, pp. 87–94. Springer, 2001.

Kaiser, Lukasz, Nachum, Ofir, Roy, Aurko, and Bengio, Samy. Learning to remember rare events. In *ICLR 2017*, 2017.

Kirkpatrick, James, Pascanu, Razvan, Rabinowitz, Neil, Veness, Joel, Desjardins, Guillaume, Rusu, Andrei A, Milan, Kieran, Quan, John, Ramalho, Tiago, Grabska-Barwinska, Agnieszka, et al. Overcoming catastrophic forgetting in neural networks. *arXiv preprint arXiv:1612.00796*, 2016.

Koch, Gregory. *Siamese neural networks for one-shot image recognition*. PhD thesis, University of Toronto, 2015.

Lake, Brenden M, Salakhutdinov, Ruslan R, and Tenenbaum, Josh. One-shot learning by inverting a compositional causal process. In *Advances in neural information processing systems*, pp. 2526–2534, 2013.

Lake, Brenden M, Salakhutdinov, Ruslan, and Tenenbaum, Joshua B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Langley, P. Crafting papers on machine learning. In Langley, Pat (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.