Imitation networks: Few-shot learning of neural networks from scratch

Akisato Kimura[1,2], **Zoubin Ghahramani**[2,3], **Koh Takeuchi**[1], **Tomoharu Iwata**[1] **and Naonori Ueda**[1]

[1] NTT Communication Science Laboratories, Japan.
[2] Department of Engineering, University of Cambridge, United Kingdom. [3] Uber AI Labs, USA.
akisato@ieee.org, zoubin@eng.cam.ac.uk, koh.t@acm.org, {iwata.tomoharu, ueda.naonori}@lab.ntt.co.jp

Abstract

In this paper, we propose imitation networks, a simple but effective method for training neural networks with a limited amount of training data. Our approach inherits the idea of knowledge distillation that transfers knowledge from a deep or wide reference model to a shallow or narrow target model. The proposed method employs this idea to mimic predictions of reference estimators that are much more robust against overfitting than the network we want to train. Different from almost all the previous work for knowledge distillation that requires a large amount of labeled training data, the proposed method requires only a small amount of training data. Instead, we introduce pseudo training examples that are optimized as a part of model parameters. Experimental results for several benchmark datasets demonstrate that the proposed method outperformed all the other baselines, such as naive training of the target model and standard knowledge distillation.

1 Introduction

The current state-of-the-art in machine learning heavily relies on the supervised learning of deep neural networks with large-scale datasets. However, constructing such large-scale datasets generally requires painstaking effort, and in many real-world applications, only a limited number of training examples can be obtained. Deep neural networks easily overfit the training data, especially when only a small amount of training data is available. Several techniques for alleviating overfitting in deep learning have been proposed so far, e.g. semi-supervised learning [Goodfellow *et al.*, 2014], transfer learning [Caruana, 1994] and few-shot learning [Koch *et al.*, 2015]. However, all the above approaches require either a large number of unsupervised training examples or a pre-trained model trained with a large amount of supervised training data, and thus learning neural networks from only a few examples remains a key challenge.

Meanwhile, several other estimators such as support vector machines (SVMs) and Gaussian processes (GPs) can ease the adverse effect of overfitting by making use of Bayesian principle or maximum margin. The universal approximator the-
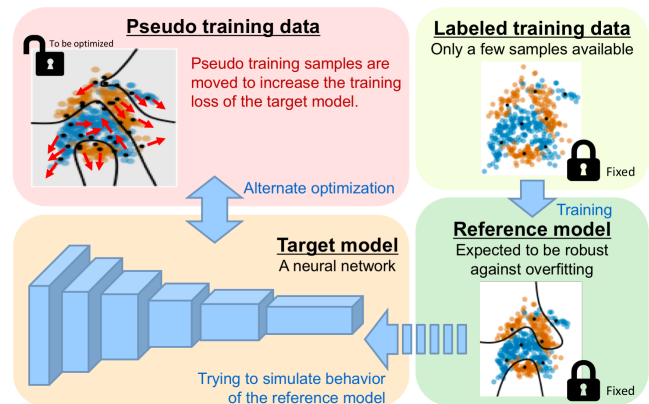

Figure 1: Basic idea of the proposed method

orem [Hornik *et al.*, 1989] guarantees that an infinitely wide neural network with at least one hidden layer can represent any Lipschitz continuous function to an arbitrary degree of accuracy. This theorem implies that there exists a neural network that well imitates the behavior of other estimators, while keeping a great representation power of neural networks.

In this paper, we propose a novel method for training neural networks with a small amount of supervised training data. Figure 1 shows the basic idea of our proposed method named *imitation networks*. The proposed method first trains a reference model with a given small amount of supervised training data, and then transfers knowledge from the reference model to a target neural network model in a similar manner to knowledge distillation [Buciluǎ *et al.*, 2006; Ba and Caruana, 2014; Hinton *et al.*, 2015]. Although any types of black-box estimators can be applied as a reference model in our method in principle, we particularly select GPs as the reference model that are expected to be robust against overfitting. Different from almost all the previous work for knowledge distillation that employs a large number of supervised training examples, our proposed method requires only a few supervised training examples for knowledge transfer. To augment training examples, we introduce inducing points [Snelson and Ghahramani, 2006] that are pseudo training examples helping make the model training tractable or much easier. In the original inducing point methods used for scal-

able GP inference, both inducing points and model parameters are updated to increase the objective function that is actually a lower bound (ELBO) on the marginalized likelihood. In our proposed method, however, parameters of the target model are updated to *decrease* a training loss, but pseudo training examples are updated to *increase* the training loss. By doing this, we can move pseudo training examples toward areas where the current target model has not been well trained. We also introduce fidelity weighting [Dehghani *et al.*, 2017] for controlling the contribution of each pseudo training example based on uncertainty in predictions obtained from reference models. This enables us to eliminate unreliable pseudo examples from the model training.

Summarizing the above discussion, our main contributions of this paper are the followings:

1. Presenting a novel framework for few-shot training of neural networks based on knowledge transfer.

2. Applying the idea of inducing points into the training of neural networks, which can be optimized in almost the same way as model parameters of neural networks.

2 Related work

Techniques for avoiding overfitting: Several techniques have already been proposed for alleviating overfitting in neural network training. Widely used techniques include data augmentation, model regularization, early stopping, dropout and weight decay, all of which are implemented into almost all the deep learning libraries. Semi-supervised learning is also a major approach when a large amount of unsupervised training examples are available. Recently, generative models such as generative adversarial networks (GAN) [Goodfellow *et al.*, 2014] have become one of the most popular approaches for semi-supervised learning as they can disentangle the supervised information from many other latent factors of variation in a principled way [Kingma *et al.*, 2014]. Transfer learning [Caruana, 1994] is also widely used for neural network training when a base model trained with a large amount of training data for related tasks is available. One-shot or few-shot learning [Koch *et al.*, 2015; Bertinetto *et al.*, 2016; Vinyals *et al.*, 2016] is a problem of learning a classifier from only a few supervised examples per class. As with transfer learning, few-shot learning relies on reference models pre-trained with a large amount of labeled training examples whose class labels are different from target class labels, and captures the characteristics of the target classes from relationships among other reference classes and a single target example. Compared with those approaches, our method (1) requires no additional training examples unlike semi-supervised learning, and (2) achieves few-shot learning *from scratch*, meaning that the reference model is trained with a few training examples, and neither additional (supervised / unsupervised) examples nor reference models trained with a large amount of supervised training examples are required.

Knowledge distillation: Knowledge distillation is a class of techniques for training a shallow and/or narrow network, and it is also called model distillation or model compression. More specifically, knowledge distillation learns a small student network by transferring knowledge from large teacher networks, mainly for implementing the network onto devices with limited computational power. [Buciluǎ *et al.*, 2006] pioneered this approach, and [Ba and Caruana, 2014] extended this idea to deep learning. [Hinton *et al.*, 2015] generalized the previous methods by introducing a new metric between the output distribution of teacher and student predictions. Other previous researches demonstrates that knowledge can be transferred from either intermediate layers of a teacher model [Romero *et al.*, 2015; Yim *et al.*, 2017], from multiple teacher models [You *et al.*, 2017], mismatched unsupervised stimuli [Kulkarni *et al.*, 2017] or examples reconstructed from meta-data [Gontijo Lopes *et al.*, 2017]. Recent papers [Papernot *et al.*, 2016b; Furlanello *et al.*, 2017; Tarvainen and Valpola, 2017] showed that the knowledge distillation framework is applicable to learn a student network from a teacher network with completely the same structure with the student network. Meanwhile, our method (1) inherits the idea of knowledge distillation, (2) exploits an arbitrary black-box estimator as a teacher, in contrast to standard knowledge distillation [1], and (3) does not require a large amount of real training data, and instead employs pseudo training data that is optimized during the model training.

Inducing point methods: Inducing point methods [Snelson and Ghahramani, 2006] have been originally developed for scalable inference of GP regression and classification. Inducing points are pseudo training examples that can help the model training tractable or much easier. [Quinonero Candela and Rasmussen, 2006] implies that initial inducing points can be selected from supervised training examples. [Titsias, 2009] suggests a variational approach which provides an objective function for optimizing inducing points. In this method, both inducing points and covariate function parameters are updated to increase the objective function that is actually the evidence lower bound (ELBO) on the marginalized likelihood. [Hensman *et al.*, 2013; Hensman *et al.*, 2015] introduced stochastic variational inference for improving scalability, where inducing points are regarded as latent variables and marginalized out in the model inference. Meanwhile, our method (1) first introduces inducing points into supervised neural network training, and (2) updates pseudo training examples to *increase* a training loss but updates model parameters to *decrease* the loss, in an adversarial manner.

3 Model optimization

3.1 Knowledge distillation

Before introducing the framework of our proposed method, we first describe knowledge distillation that is the source of our main idea. Knowledge distillation is a family of methods that transfer the generalization ability of a pre-trained reference model $g(\boldsymbol{x}; \boldsymbol{\theta}_g)$ to a target model $f(\boldsymbol{x}; \boldsymbol{\theta}_f)$, where $\boldsymbol{x}$ is an input example, and $\boldsymbol{\theta}_g$ and $\boldsymbol{\theta}_f$ are model parameters of the reference and target models, respectively. In the following, we will omit model parameters $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_g$ for simplicity except we explicitly state. Recent knowledge distillation methods are mainly focused on the training of a shallow or narrow

---

[1]However, this has already been acknowledged in the context of black-box adversarial attacks [Papernot *et al.*, 2016a]

neural network as the target model with the help of reference models that are much deeper or wider neural networks, by minimizing the following distillation loss L_{dis} with respect to the target $f(\cdot)$:

$$L_{\mathrm{dis}}(\boldsymbol{X}^{\mathrm{L}}, \boldsymbol{Y}^{\mathrm{L}}) = \frac{\lambda_1}{N_{\mathrm{L}}} \sum_{n=1}^{N_{\mathrm{L}}} D_1(\boldsymbol{y}_n^{\mathrm{L}}, f(\boldsymbol{x}_n^{\mathrm{L}}))$$
$$+ \frac{\lambda_2}{N_{\mathrm{L}}} \sum_{n=1}^{N_{\mathrm{L}}} D_2(g(\boldsymbol{x}_n^{\mathrm{L}}), f(\boldsymbol{x}_n^{\mathrm{L}})), \quad (1)$$

where $\boldsymbol{X}^{\mathrm{L}} = \{\boldsymbol{x}_1^{\mathrm{L}}, \ldots, \boldsymbol{x}_{N_{\mathrm{L}}}^{\mathrm{L}}\}$ is a set of supervised training examples, $\boldsymbol{Y}^{\mathrm{L}} = \{\boldsymbol{y}_1^{\mathrm{L}}, \ldots, \boldsymbol{y}_{N_{\mathrm{L}}}^{\mathrm{L}}\}$ a set of the corresponding supervisors, $D_1(\boldsymbol{y}, \hat{\boldsymbol{y}})$ is a supervised loss that compares a supervisor $\boldsymbol{y}$ and a prediction $\hat{\boldsymbol{y}}$, and $D_2(\hat{\boldsymbol{y}}_1, \hat{\boldsymbol{y}}_2)$ is an unsupervised loss that computes a loss between two different predictions $\hat{\boldsymbol{y}}_1$ and $\hat{\boldsymbol{y}}_2$. Knowledge distillation usually deals with classification problems, where each supervisor $\boldsymbol{y}_n^{\mathrm{L}}$ is represented by a one-hot vector, however, it can also be directly applied to regression or other machine learning problems by replacing the supervisors into the shapes appropriate for the problem to be solved.

3.2 Proposed loss function

As shown in Eq. (1), almost all the formulations of the previous knowledge distillation work employ supervised training examples $(\boldsymbol{X}^{\mathrm{L}}, \boldsymbol{Y}^{\mathrm{L}})$ to train a target model $f(\cdot)$. However, you may notice that (1) minimizing the distillation loss with a limited number of supervised training examples causes overfitting, and (2) real supervisions $\boldsymbol{Y}^{\mathrm{L}}$ are not required for computing the unsupervised loss. Based on this observation, our proposed method called *imitation networks* newly introduces pseudo training examples $\boldsymbol{X}^{\mathrm{P}} = (\boldsymbol{x}_1^{\mathrm{P}}, \ldots, \boldsymbol{x}_{N_{\mathrm{P}}}^{\mathrm{P}})$ to train a target model $f(\cdot)$ from a reference model $g(\cdot)$, by minimizing the following loss function L_{imi} that we call *the imitation loss*:

$$L_{\mathrm{imi}}(\boldsymbol{X}^{\mathrm{L}}, \boldsymbol{Y}^{\mathrm{L}}, \boldsymbol{X}^{\mathrm{P}}) = \frac{\lambda_1}{N_{\mathrm{L}}} \sum_{n=1}^{N_{\mathrm{L}}} D_1(\boldsymbol{y}_n^{\mathrm{L}}, f(\boldsymbol{x}_n^{\mathrm{L}}))$$
$$+ \frac{\lambda_2}{N_{\mathrm{P}}} \sum_{n=1}^{N_{\mathrm{P}}} D_2(g(\boldsymbol{x}_n^{\mathrm{P}}), f(\boldsymbol{x}_n^{\mathrm{P}})). \quad (2)$$

In the same way as knowledge distillation, a reference model $g(\cdot)$ is first pre-trained with supervised training examples $(\boldsymbol{X}^{\mathrm{L}}, \boldsymbol{Y}^{\mathrm{L}})$. However, we have to note that only a few number of supervised examples are available in our problem setting. A reference model $g(\cdot)$ can be a single estimator or an ensemble of multiple estimators. We can build multiple estimators from a single model by changing hyper-parameters, such as a variance and a length scale in RBF kernels for GPs. When introducing an ensemble of multiple estimators, we choose to average their predictions in a similar manner to the previous work [Hinton *et al.*, 2015].

The above formulation indicates that the target model tries to imitate predictions of the reference model for pseudo examples and at the same time it tries to return predictions for supervised examples as accurately as possible. Therefore, the target model can inherit two different properties coming from the reference model and neural networks.

3.3 Fidelity weighting

We note that all the pseudo training examples are not always useful for knowledge transfer. Examples yielding unreliable predictions from the reference model are more likely harmful and should be discarded in the model training. For this purpose, we introduce fidelity weighting [Dehghani *et al.*, 2017], which adaptively weighs training examples based on uncertainty of predictions obtained from the reference model With the introduction of fidelity weighting, the imitation loss Eq. (2) can be replaced by the following equation.

$$L_{\mathrm{imi}}(\boldsymbol{X}^{\mathrm{L}}, \boldsymbol{Y}^{\mathrm{L}}, \boldsymbol{X}^{\mathrm{P}}) = \frac{\lambda_1}{N_{\mathrm{L}}} \sum_{n=1}^{N_{\mathrm{L}}} D_1(\boldsymbol{y}_n^{\mathrm{L}}, f(\boldsymbol{x}_n^{\mathrm{L}}))$$
$$+ \frac{1}{N_{\mathrm{P}}} \sum_{n=1}^{N_{\mathrm{P}}} \lambda_2(g, \boldsymbol{x}_n^{\mathrm{P}}) D_2(g(\boldsymbol{x}_n^{\mathrm{P}}), f(\boldsymbol{x}_n^{\mathrm{P}})). \quad (3)$$

where $\lambda_2(g, \boldsymbol{x}_n^{\mathrm{P}})$ is a new example-wise weight for a pseudo example, which is computed from uncertainty $\sigma_g(\boldsymbol{x}_n^{\mathrm{P}})$ of the reference prediction $g(\boldsymbol{x}_n^{\mathrm{P}})$. More specifically, the weight $\lambda_2(g, \boldsymbol{x}_n^{\mathrm{P}})$ is computed as follows:

$$\lambda_2(g, \boldsymbol{x}_n^{\mathrm{P}}) = \hat{\lambda}_2 \exp(-\overline{\lambda}_2 \sigma_g(\boldsymbol{x}_n^{\mathrm{P}})/\overline{\sigma}_g), \quad (4)$$

where $\hat{\lambda}_2$ is an upper bound of weights, $\overline{\sigma}_2$ is a mean uncertainty over all the pseudo examples and $\overline{\lambda}_2$ is a weight for the mean uncertainty.

Fidelity weighting is valid for various kinds of reference estimators, unless Bayesian treatments cannot be applied to the reference estimators in principle. For example, the original work [Dehghani *et al.*, 2017] utilized GP classifiers that are one of the most popular Bayesian estimators. Bayesian variants of SVMs [Zhang and Jordan, 2006] and Bayesian neural networks [Gal and Ghahramani, 2016] can also be employed for this purpose.

When transferring knowledge useful for improving performance, the selection of pseudo examples plays a significant role. We will describe this step in the following section.

4 Pseudo example optimization

4.1 Inducing point method

As described in Section 2, the inducing point method has been originally developed for scalable inference of GP regression and classification, and in this method both inducing points and model parameters are updated to increase the objective function. On the other hand, in our proposed method, updates of model parameters and pseudo training examples are going off in completely different directions. Namely, model parameters are updated to *decrease* the imitation loss, meanwhile pseudo training examples are updated to *increase* the loss. By doing this, we can move pseudo training examples towards areas where the current target model has not been well trained.

Our proposed technique for updating pseudo examples is inspired by adversarial training [Szegedy *et al.*, 2015], which increases the robustness of neural networks to adversarial examples formed by applying small but intentionally worst-case perturbations to examples from the dataset. The

most popular technique for generating an adversarial example $\boldsymbol{x}_{\mathrm{AT}}(\boldsymbol{x}^{\mathrm{L}}, \boldsymbol{y}^{\mathrm{L}})$ from a given supervised example $(\boldsymbol{x}^{\mathrm{L}}, \boldsymbol{y}^{\mathrm{L}})$ is a fast gradient sign method [Goodfellow *et al.*, 2015].

$$\boldsymbol{x}_{\mathrm{AT}}(\boldsymbol{x}^{\mathrm{L}}, \boldsymbol{y}^{\mathrm{L}}) = \boldsymbol{x}^{\mathrm{L}} + \epsilon \operatorname{sign}\{\nabla_{\boldsymbol{x}^{\mathrm{L}}} D_1(\boldsymbol{y}^{\mathrm{L}}, f(\boldsymbol{x}^{\mathrm{L}}))\}, \quad (5)$$

where $\nabla_{\boldsymbol{x}}$ is a partial differentiation with respect to $\boldsymbol{x}$ and $\epsilon \geq 0$ is a small constant.

You may notice that the above adversarial example $\boldsymbol{x}_{\mathrm{AT}}(\boldsymbol{x}, \boldsymbol{y})$ can be seen as a stochastic update of the original example $\boldsymbol{x}$ to *increase* the loss and a small constant ϵ can be seen as a learning rate, if the gradient sign part $\operatorname{sign}\{\nabla_{\boldsymbol{x}} D_1(\boldsymbol{y}, f(\boldsymbol{x}))\}$ is replaced by a standard stochastic gradient $\nabla_{\boldsymbol{x}} D_1(\boldsymbol{y}, f(\boldsymbol{x}))$. Also, the ground-truth supervisor $\boldsymbol{y}$ can be removed from the update, since every pseudo training example $\boldsymbol{x}^{\mathrm{P}}$ has a soft supervision $g(\boldsymbol{x}^{\mathrm{P}})$ instead. Based on the above discussion, the update of a pseudo training example $\boldsymbol{x}^{\mathrm{P}}$ can be obtained as

$$\boldsymbol{x}_{\mathrm{imi}}(\boldsymbol{x}^{\mathrm{P}}) = \boldsymbol{x}^{\mathrm{P}} + \epsilon \nabla_{\boldsymbol{x}^{\mathrm{P}}} D_2(g(\boldsymbol{x}^{\mathrm{P}}), f(\boldsymbol{x}^{\mathrm{P}})). \quad (6)$$

Instead of the direct use of this stochastic gradient update, we can introduce the recent advances of stochastic optimization such as Adam [Kingma and Ba, 2015] and Nesterov accelerated Adam (Nadam) [Dozat, 2016] for faster optimization. In addition, the stochastic gradient part can be replaced by other types of adversarial examples, such as natural [Zhao *et al.*, 2017] and spatially transformed adversary [Xiao *et al.*, 2018].

We can build an adversarial training procedure that alternatively updates parameters $\boldsymbol{\theta}_f$ of the target model to *decrease* the imitation loss and pseudo examples $\boldsymbol{X}^{\mathrm{P}}$ to *increase* the loss. However, such an adversarial training procedure is well known to be unstable [Salimans *et al.*, 2016]. Instead, we adopt another approach that augments pseudo examples. We employ a fixed set $\boldsymbol{X}^{\mathrm{P}(t)}$ of pseudo examples for the t-th step of model training, and update another set $\boldsymbol{X}^{\mathrm{P}(t+1)}$ of pseudo examples that is originally a carbon copy of the current set $\boldsymbol{X}^{\mathrm{P}(t)}$. After the convergence of the t-th training step, the current set $\boldsymbol{X}^{\mathrm{P}(t)}$ is merged into the next set $\boldsymbol{X}^{\mathrm{P}(t+1)}$. Locking in pseudo examples for a while will make the model training more stable, and augmenting pseudo examples whose predictions are distant from those of the reference model makes the target model to be closer to the reference model.

4.2 Algorithm

Summarizing the above discussion, the training procedure of our proposed imitation networks is the following:

1. A reference model $g(\cdot)$ is trained with (a few) labeled training examples $(\boldsymbol{X}^{\mathrm{L}}, \boldsymbol{Y}^{\mathrm{L}})$.

2. An initial set $\boldsymbol{X}^{\mathrm{P}(0)}$ of pseudo training examples is generated, a carbon copy $\boldsymbol{X}^{\mathrm{P}(1)}$ of the initial set $\boldsymbol{X}^{\mathrm{P}(0)}$ is created and the index t representing the current training step is set to 0.

3. Model parameters $\boldsymbol{\theta}_f$ of a target model $f(\cdot)$ are updated by using the current pseudo examples $\boldsymbol{X}^{\mathrm{P}(t)}$ and their outputs of the reference model $g(\cdot)$ so as to decrease the imitation loss shown in Eq. (3).

4. The next set $\boldsymbol{X}^{\mathrm{P}(t+1)}$ of pseudo examples is updated with the current target model $f(\cdot)$ by Eq. (6).

5. Repeat 3.-4. until convergence.

6. A carbon copy $\boldsymbol{X}^{\mathrm{P}(t+2)}$ of the next set $\boldsymbol{X}^{\mathrm{P}(t+1)}$ is created for further updates, the current pseudo examples $\boldsymbol{X}^{\mathrm{P}(t)}$ are included into the next pseudo examples $\boldsymbol{X}^{\mathrm{P}(t+1)}$, namely $\boldsymbol{X}^{\mathrm{P}(t+1)} \leftarrow \boldsymbol{X}^{\mathrm{P}(t)} \cup \boldsymbol{X}^{\mathrm{P}(t+1)}$, and the time index t is incremented as $t \leftarrow t + 1$.

7. Repeat 3.-6. until convergence.

5 Experiments

5.1 Qualitative analysis

To check the behavior of the proposed method qualitatively, we first turned to the Banana dataset [2] [Hensman *et al.*, 2015] for binary classification with 400 two-dimensional examples. We embedded each of two-dimensional examples into a 100-dimensional vector space by a linear transformation, where a transformation matrix was randomly determined in advance, and we selected five supervised training examples per class randomly. The selected supervised examples and other unused examples are shown in Figure 2 a), where crosses are supervised training examples, dots are unused examples in the dataset and a color corresponds to its class label. We employed a GP classifier with an RBF kernel as reference models, and built an ensemble of multiple reference models trained with different initial kernel parameters. We used GPflow [Matthews *et al.*, 2017] with variational Gaussian approximation [Opper and Archambeau, 2009] and L-BFGS-B optimization [Zhu *et al.*, 1997] for GP inference, and utilized mean predictions as pseudo supervisions. The target model we selected was a 7-layer fully-connected neural network where each of the intermediate layers has 1000 units. We used Nadam [Dozat, 2016] for model parameter optimization and Adam [Kingma and Ba, 2015] for pseudo example optimization, and their initial learning rates were set to 0.001 and 0.05, respectively. The batch size was set to 100, and the number of training epochs was set to 200 in total.

First, we examined an extreme case where 2500 pseudo examples are densely distributed onto whole the feature space. Although this is not realistic especially for high-dimensional spaces, it is useful to check whether our proposed method can imitate the behavior of the reference model precisely. Figure 2 b) shows its distribution, where green dots are pseudo examples. In this case, we did not optimize pseudo examples in this case, and we did not employ fidelity weighting shown in 3.3, which means that we utilized the original imitation loss shown in Eq. (2). We utilized Kullback-Leibler divergence for the soft losses D_2 and ignored the hard loss D_1 (i.e. $\lambda_1 = 0$). Figure 2 c) and d) show decision boundaries of the reference and target models, respectively. This result indicates that our method imitated the predictions of the reference model almost completely, as expected.

---

[2] The data can be obtained from `https://github.com/GPflow/GPflow/tree/master/doc/source/notebooks/data`
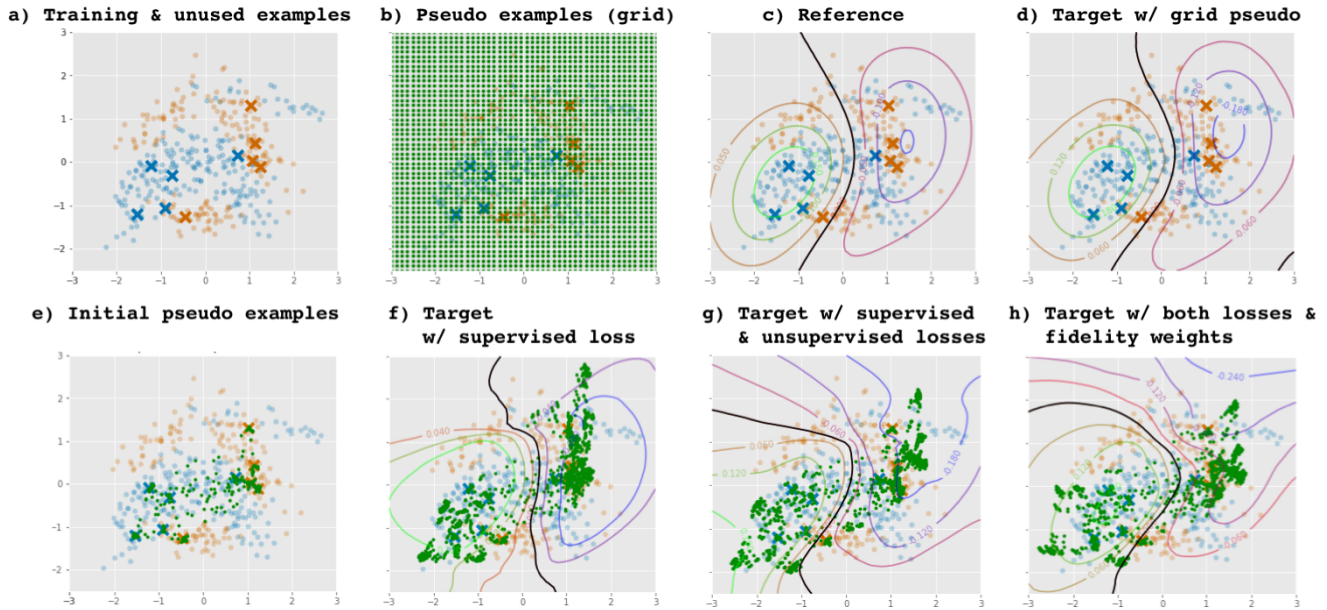
Figure 2: Visualizations of predictions by the reference and target models, where orange and blue crosses = supervised training examples, orange and blue dots = other unused examples in the dataset, green dots = pseudo training examples, and black lines = classification boundaries. From the top left to the bottom right, a) supervised and unused examples, b) pseudo training examples densely distributed onto whole the feature space as an extreme case, c) a reference model trained with supervised training examples, d) a target model trained with densely distributed pseudo examples, e) 1000 pseudo training examples generated from supervised examples as seeds, f) a target model trained with generated pseudo examples, g) a target model trained with supervised and pseudo examples. h) a target model trained with supervised and pseudo examples plus fidelity weighting.

Next, we considered a more realistic setup, where 250 initial pseudo examples are generated by a mixture of interpolation and Gaussian augmentation of supervised training examples, and augmented them to 1000 with our technique shown in Section 4.1, where the number of pseudo training examples is much smaller than the first setup. Thus, we had 4 training steps each of which had 50 training epochs and employed 250, 500, 750 and 1000 pseudo examples, respectively. We utilized hinge loss for the hard loss D_1 and Kullback-Leibler divergence for the soft losses D_2, and the weights λ_1 and $\{var\lambda_2$ were set to 1.0. Figure 2 e) shows an initial distribution of pseudo training examples, and f), g) and h) show decision boundaries and optimized pseudo examples of the target models trained with only soft supervisions, with soft and hard supervisions, and supervisions plus fidelity weighting, respectively. The results indicate that the target model well imitated the reference model even with sparsely distributed pseudo examples. Also, Figure 2 g) and h) indicate that the introduction of hard supervisions yielded a better classification performance for supervised examples (see the orange cross at the center bottom) while preserving the decision boundary as much as possible, but fidelity weighting did not affect the boundary so much. A distribution of optimized pseudo examples partly describes properties of our proposed method. Optimized pseudo examples were distributed over whole the feature space so as to decrease the difference between the target and reference models.

Table 1: Classification performances for MNIST (top) and fashion MNIST (bottom) datasets, where "imitation", "opt" and "fidelity" stand for the proposed method, pseudo example optimization, and fidelity weighting, respectively

Methods / #labeled	10	20	50	100	200
NN	37.9	46.0	66.0	78.3	86.7
GP	39.9	51.6	64.6	73.2	80.0
Imitation	43.5	51.2	67.7	78.1	86.1
Imitation, opt	44.1	53.7	70.0	79.5	**86.7**
Imitation, opt, fidelity	**44.1**	**53.9**	**70.4**	**80.0**	86.6

Methods / #labeled	10	20	50	100	200
NN	39.3	47.9	58.3	64.9	71.3
GP	44.6	52.4	59.9	65.7	71.4
Imitation	43.6	50.9	60.0	67.3	**72.5**
Imitation, opt	41.2	49.7	60.1	67.3	72.2
Imitation, opt, fidelity	**44.8**	**52.7**	**62.1**	**68.0**	**72.5**

5.2 Quantitative evaluations

Next, we quantitatively evaluated classification performances of the proposed method for several benchmark datasets. We used MNIST [Le Cun *et al.*, 1998] and fashion MNIST [Xiao *et al.*, 2017] as the datasets for this experiment. We again employed a GP classifier as a reference model, where all the setups related to the reference models were the same as Section 5.1. The target model was a 3-layer CNN for

Table 2: Distillation performances for MNIST (top) and fashion MNIST (bottom) datasets.

Methods / #labeled	10	20	50	100	200
NN (7-layer)	44.9	58.2	77.7	86.5	92.4
NN (3-layer)	37.9	46.0	66.0	78.3	86.7
Imitation	42.9	54.5	73.7	83.8	90.3
Imitation, opt	**43.1**	55.2	**74.2**	**84.1**	**90.5**
Imitation, opt, fidelity	**43.1**	**55.4**	**74.2**	**84.1**	**90.5**

Methods / #labeled	10	20	50	100	200
NN (7-layer)	40.3	48.5	61.4	68.7	73.7
NN (3-layer)	39.3	47.9	58.3	64.9	71.3
Imitation	**44.6**	**51.8**	**62.6**	**69.5**	**74.3**
Imitation, opt	44.1	51.7	62.3	69.2	74.0
Imitation, opt, fidelity	44.1	51.6	62.3	68.5	73.6

both of the datasets, where each convolution layer group has a convolution layer (3×3 kernel with 2 strides and $16 \times (floor(l/2) + 1)$ channels for the l-th layer group), batch normalization, PReLU and dropout (rate= 0.5) and 2 fully-connected layers (with batch normalization, PReLU and dropout after the first FC layer) are placed afterward. We again used Nadam for model parameter optimization and Adam for pseudo example optimization, and both of the initial learning rates were set to 0.02. We prepared $1.25K$ initial pseudo examples by interpolating two different supervised training examples, and augmented them to $10K$ by employing our technique shown in Section 4.1. Thus, whole the training process contained 8 training steps and each training step had 25 training epochs. The weight λ_1 for the hard loss was set to 1.0, and the mean weight $\bar{\lambda}_2$ for the soft loss was decreased 100 to 1 as the number of supervised training examples increased. We trained the target model with 1 to 20 supervised training examples per class randomly selected from the 50K training examples in each of the datasets. Then, we tested the trained model with the 10K test examples, and compared classification accuracy averaged over 20 different selection of supervised training examples. All the other experimental conditions were the same as the second case in Section 5.1. We compared the performance of the following 5 training strategies, namely (1) the reference estimators, (2) a naive training of the target neural network, (3) our method with only the imitation loss (no pseudo example optimization and fidelity weighting), (4) our method with the imitation loss and pseudo example optimization (no fidelity weighting), and (5) our method with the imitation loss, pseudo example optimization and fidelity weighting.

Table 1 shows the experimental result. The result indicates that our proposed method outperformed a naive training of the target neural network model. The result also indicates that a naive training of the target neural network outperformed the reference models for MNIST dataset. Even with this undesirable setting, our proposed method transferring knowledge of a rather weak reference model to the large target model was surprisingly superior or comparable to the naive training. In addition, our proposed method surprisingly outperformed

the reference models, since the supervised loss improved the classification performance. Our proposed method reasonably worked well even without pseudo example optimization, however, the introduction of pseudo example optimization improved the classification performance, especially for MNIST dataset. Although fidelity weighting also improved the performance in some cases, the contribution was minor compared with pseudo example optimization that is our main contribution of this paper.

Our proposed method can be directly applied to standard knowledge distillation setups without any modifications, where the reference model is a deep or wide network and the target model is a shallow or narrow network. As a sub-task, we also tested the proposed method in this knowledge distillation setup. We selected a 7-layer CNN as the reference model and a 3-layer CNN as the target model for both of the datasets, where the detailed model configurations were the same as the first experiment. We employ Bayesian dropout method [Gal and Ghahramani, 2016] for computing prediction uncertainty from the reference neural network. Table 2 shows the experimental result. The result indicates that our proposed method greatly improved the classification performance of the shallow target model as expected. The result also demonstrates that the proposed method surprisingly outperformed the reference deep model, mainly due to the use of pseudo supervisors as implied in the previous work [Furlanello *et al.*, 2017]. Meanwhile, pseudo example optimization and fidelity weighting might not be necessarily effective for standard knowledge distillation, however, they did not have large negative impacts on the classification performance.

6 Conclusion

In this paper, we have proposed imitation networks, a simple but effective method for training neural networks with a limited number of training examples. Our proposed method employed a reference model that is more robust against overfitting, and built a target neural network so as to imitate the behavior of the reference model trained with the limited training examples. Different from the previous work for knowledge distillation that employs a large amount of supervised training examples, our method required only a reference model trained with a few supervised examples. Instead, we introduced pseudo examples and optimized them through the process of target model training. Qualitative and quantitative experimental results demonstrated that our proposed method effectively reduces the risk of target model overfitting caused by a few amount of training data.

Since our proposed framework shown in Section 3.2 is generic, it can be directly applied to other combinations of reference and target models. For example, it enables us to train a deep target network with a pre-trained shallow reference network trained on a small amount of data, which is the inverse of knowledge distillation. This may provide a new way of pre-training deep neural networks. On the other hand, our method for optimizing pseudo training examples is rather specific, and sophisticated management of pseudo examples remains largely to be investigated.

References

[Ba and Caruana, 2014] J. Ba and R. Caruana. Do deep nets really need to be deep? In *Proc. NIPS2014*.

[Bertinetto *et al.*, 2016] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *Proc. NIPS2016*.

[Buciluǎ *et al.*, 2006] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proc. KDD2006*.

[Caruana, 1994] R. Caruana. Learning many related tasks at the same time with backpropagation. In *Proc. NIPS1994*.

[Dehghani *et al.*, 2017] M. Dehghani, A. Mehrjou, S. Gouws, J. Kamps, and B. Schölkopf. Fidelity-weighted learning. *arXiv pre-prints*, November 2017.

[Dozat, 2016] T. Dozat. Incorporating Nesterov momentum into Adam. In *Proc. ICLR Workshop*, 2016.

[Furlanello *et al.*, 2017] T. Furlanello, Z. C. L., L. Itti, and A. Anandkumar. Born again neural networks. In *NIPS Workshop on Meta Learning*, 2017.

[Gal and Ghahramani, 2016] Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proc. ICML2016*.

[Gontijo Lopes *et al.*, 2017] R. Gontijo Lopes, S. Fenu, and T. Starner. Data-Free Knowledge Distillation for Deep Neural Networks. *arXiv pre-prints*, October 2017.

[Goodfellow *et al.*, 2014] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proc. NIPS2014*.

[Goodfellow *et al.*, 2015] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *Proc. ICLR2015*.

[Hensman *et al.*, 2013] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Proc. UAI2013*.

[Hensman *et al.*, 2015] J. Hensman, A. Matthews, and Z. Ghahramani. Scalable Variational Gaussian Process Classification. In *Proc. AISTATS2015*.

[Hinton *et al.*, 2015] G. E. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[Hornik *et al.*, 1989] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[Kingma and Ba, 2015] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR2015*.

[Kingma *et al.*, 2014] D. P Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *NIPS2014*.

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

[Kulkarni *et al.*, 2017] M. Kulkarni, K. Patil, and S. S. Karande. Knowledge distillation using unlabeled mismatched images. *arXiv pre-prints*, March 2017.

[Le Cun *et al.*, 1998] Y. Le Cun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. the IEEE*, 86(11):2278–2324, 1998.

[Matthews *et al.*, 2017] A. G. de G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman. GPflow: A Gaussian process library using TensorFlow. *JMLR*, 18(40):1–6, 2017.

[Opper and Archambeau, 2009] M. Opper and C. Archambeau. The variational gaussian approximation revisited. *Neural Computation*, pages 786–792, 2009.

[Papernot *et al.*, 2016a] N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in Machine Learning: From Phenomena to Black-Box Attacks using Adversarial Samples. *arXiv pre-prints*, May 2016.

[Papernot *et al.*, 2016b] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proc. IEEE Symposium on Security and Privacy (SP2016)*.

[Quinonero Candela and Rasmussen, 2006] J. Quinonero Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *JMLR*, 6:1939–1959, 2006.

[Romero *et al.*, 2015] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. In *Proc. ICLR2015*.

[Salimans *et al.*, 2016] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen. Improved techniques for training GANs. In *Proc. NIPS2016*.

[Snelson and Ghahramani, 2006] E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Proc. NIPS2006*.

[Szegedy *et al.*, 2015] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proc. CZVPR2015*.

[Tarvainen and Valpola, 2017] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proc. NIPS2017*.

[Titsias, 2009] M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Proc. AISTATS2009*.

[Vinyals *et al.*, 2016] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *Proc. NIPS2016*.

[Xiao *et al.*, 2017] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv pre-prints*, August 2017.

[Xiao *et al.*, 2018] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song. Spatially Transformed Adversarial Examples. *arXiv pre-prints*, January 2018.

[Yim *et al.*, 2017] J. Yim, D. Joo, J. Bae, and J. Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proc. CVPR2017*.

[You *et al.*, 2017] S. You, C. Xu, C. Xu, and D. Tao. Learning from multiple teacher networks. In *Proc. KDD2017*.

[Zhang and Jordan, 2006] Z. Zhang and M. Jordan. Bayesian multicategory support vector machines. In *Proc. UAI2006*.

[Zhao *et al.*, 2017] Z. Zhao, D. Dua, and S. Singh. Generating Natural Adversarial Examples. *arXiv pre-prints*, October 2017.

[Zhu *et al.*, 1997] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, December 1997.