Metaparameter optimization in knowledge distillation

No Author Given

No Institute Given

Abstract. This paper investigates the deep learning knowledge distillation problem. Knowledge distillation is a model parameter optimization problem that allows transferring information contained in the model with high complexity, called teacher, to the simpler one, called student. The paper proposes a generalization of the knowledge distillation optimization problem to optimize metaparameters by gradient descent. Metaparameters are the parameters of knowledge distillation optimization problem. The loss function is a sum of the classification term and cross-entropy between answers of the student model and teacher model. An assignment of optimal metaparameters for the distillation loss function is a computationally expensive task. We investigate the properties of optimization problem and methods to optimize and predict the regularization path of metaparameters. We analyze the trajectory of the metaparamets gradient-based optimization and approximate them using linear functions. We evaluate the proposed method on the CIFAR-10 and Fashion-MNIST datasets and synthetic data.

Keywords: Machine learning · Knowledge distillation · Metaparameter optimization · Gradient-based optimization · Metaparameter selection.

1 Introduction

The paper investigates the knowledge distillation problem for deep learning models. The deep learning model optimization is a challenging task that requires high computational resources [13]. The paper investigates a special case of deep learning model optimization called knowledge distillation. It enables us to use the training dataset and information from already trained models.

We investigate the metaparameter optimization procedure for the knowledge distillation problem. *Knowledge distillation* [5] is a model parameter optimization problem that takes into account not only information contained in the initial dataset but also the information contained in the *teacher model.* The teacher model is a model with high complexity. It contains the information about the dataset and the model parameter distributions to be transferred. The model of the simpler structure called the *student model* is optimized by transferring the knowledge of the teacher model. In [10] the approach to knowledge distillation is proposed that allows to transfer knowledge to the student model, which architecture is different from the one of the teacher model.

One of the challenges in deep learning model optimization tasks is a metaparamer assignment problem. *Metaparameters* are parameters of optimization problem. The correct metaparameter assignment can change the resulting model performance drastically [12]. Opposite to [12, 9], in this paper we distinguish *hyperparamters*, the probabilistic parameters of the prior distribution [4] and metaparamers. Despite an amount of metaparameter and hyperparameter optimization methods available for deep learning, such as random search [2] or models based on using probabilistic models [3], many of them propose to optimize the model metaparameters and hyperparameters sequentially. They propose to sequentially sample a random metaparameter instance and evaluate the performance of the model trained using these metaparameter values. Such an approach can be inconvenient when dealing with models that require a significant time for training. The table containing metaparameter optimization complexities is shown in Table 1.

Table 1: Complexity for different metaparameter and hyperparameter optimization methods. Here $|\mathbf{w}|$ is an amount of model parameters, $|\boldsymbol{\lambda}|$ is an amount of metaparameters, r is a number of optimization run for stochastic optimization methods, s is a complexity of sampling from probabilistc model.

Method	**Optimization method type**	**Complexity**
Random Search [2]	Stochastic	$O(r \cdot \|\mathbf{w}\|)$
Probabilistic model-based [3]	Stochastic	$O(r \cdot \|\mathbf{w}\| + s)$
Greedy gradient-based [8]	Gradient-based	$O(\|\mathbf{w}\| \cdot \|\boldsymbol{\lambda}\|)$
Greedy gradient-based with finite difference approximation [7]	Gradient-based	$O(\|\mathbf{w}\| + \|\boldsymbol{\lambda}\|)$

For the metaparameter optimization, we employ a bi-level optimization problem. Model parameters are optimized on the first level and the metaparameters are optimized on the second level. This approach is described in [8, 1, 9]. The greedy gradient-based method for the bi-level optimization problem is described in [8]. In [1] different gradient-based algorithms and random search are analyzed. This paper analyzes the approach to optimizing and predicting metaparameters obtained after applying gradient-based methods. It can be seen from the Table 1 that for the large-scale tasks the gradient-based methods of metaparameter optimization are more preferable. Nevertheless, the hyperparameter and metaparameters optimization still increases the model optimization time drastically [7]. In order to decrease the optimization cost, we analyze the metaparameter optimization trajectory and predict it using linear model. The Fig. 1 visualizes this approach. We evaluate our method and compare it with other metaparameter

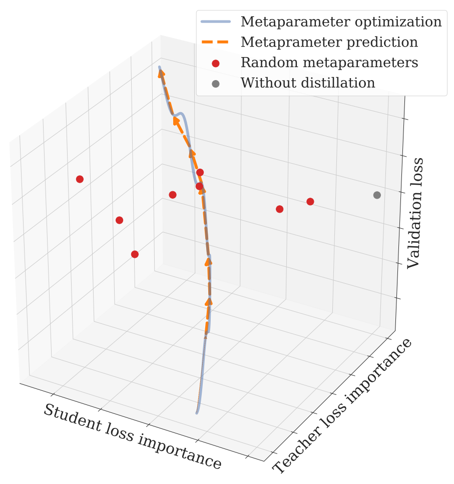

Fig. 1: A principle idea of the proposed method. The metaparameters control the final loss of the model. Instead of optimize the metaparameters straightforwardly we analyze the optimization trajectory and predict it using linear model.

optimization methods on the image datasets CIFAR-10 [6], Fashion-MNIST [14], and synthetic dataset.

Our contributions are:

1) we compare two approaches for the metaparameter optimization for the knowledge distillation task: either stochastic and gradient-based;
2) we propose a modification of the gradient-based optimization to reduce the computational cost of the optimization;
3) we give a theoretical analysis for the proposed method and evaluate its performance for the knowledge distillation.

2 Problem statement

There is given a dataset for K-classification problem:

$$\mathfrak{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m},\ \mathbf{x}_i \in \mathbb{R}^n,\ y_i \in \mathbb{Y} = \{1, \dots, K\},$$

where y_i is a class of $\mathbf{x}_i$.

Split the dataset $\mathfrak{D}$: $\mathfrak{D} = \mathfrak{D}_{\text{train}} \sqcup \mathfrak{D}_{\text{val}}$. The subset $\mathfrak{D}_{\text{train}}$ is used for model parameter optimization, the subset $\mathfrak{D}_{\text{val}}$ for metaparameter optimization.

Given a teacher model $\mathbf{f}$, which was trained on the dataset $\mathfrak{D}_{\text{train}}$. Optimize a student model $\mathbf{g}$ to transfer information obtained from the teacher. The following definition gives a formal statement for this problem.

Definition 1. *Let function $D : \mathbb{R}^s \to \mathbb{R}_+$ defines the distance between the student model $\mathbf{g}$ and the teacher model $\mathbf{f}$. D-distillation of a student model is a student model parameter optimization problem that minimizes the function D.*

Construct the $\mathcal{L}_{\text{train}}$ loss function that takes into account the knowledge distillation from model $\mathbf{f}$ to model $\mathbf{g}$:

$$\begin{aligned}\mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\lambda}) = -\lambda_1 \sum_{(\mathbf{x},y)\in\mathfrak{D}_{\text{train}}} &\underbrace{\sum_{k=1}^{K} y^k \log \frac{e^{\mathbf{g}(\mathbf{x},\mathbf{w})_k}}{\sum\limits_{j=1}^{K} e^{\mathbf{g}(\mathbf{x},\mathbf{w})_j}}}_{\text{classification term}} \\ &- (1-\lambda_1) \sum_{(\mathbf{x},y)\in\mathfrak{D}_{\text{train}}} \underbrace{\sum_{k=1}^{K} \frac{e^{\mathbf{f}(\mathbf{x})_k/T}}{\sum\limits_{j=1}^{K} e^{\mathbf{f}(\mathbf{x})_j/T}} \log \frac{e^{\mathbf{g}(\mathbf{x},\mathbf{w})_k/T}}{\sum\limits_{j=1}^{K} e^{\mathbf{g}(\mathbf{x},\mathbf{w})_j/T}}}_{\text{distillation term}},\end{aligned}$$

where y_k is the k-th component of the target vector, T is a temperature in the distillation problem. The temperature T has the following properties:

1) when $T \to 0$ we obtain a vector that has one class with unit probability;
2) when $T \to \infty$ we obtain the classes with equal probability.

Lemma 1. *When $\lambda_1 = 0$ we minimize loss function which is a D-distillation with $D = D_{KL}\left(\sigma\left(\mathbf{f}/T\right), \sigma\left(\mathbf{g}/T\right)\right)$, where σ is a softmax function.*

Proof. When $\lambda_1 = 0$ we have:

$$\begin{aligned}\mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\lambda}) = \sum_{(\mathbf{x},y)\in\mathfrak{D}_{\text{train}}} \sum_{k=1}^{K} \frac{e^{\mathbf{f}(\mathbf{x})_k/T}}{\sum\limits_{j=1}^{K} e^{\mathbf{f}(\mathbf{x})_j/T}} \log \frac{e^{\mathbf{g}(\mathbf{x},\mathbf{w})_k/T}}{\sum\limits_{j=1}^{K} e^{\mathbf{g}(\mathbf{x},\mathbf{w})_j/T}} \\ = D_{KL}\left(\sigma(\mathbf{f}(\mathbf{x})/T), \sigma(\mathbf{g}(\mathbf{x},\mathbf{w})/T)\right).\end{aligned}$$

The function $D_{KL}\left(\sigma\left(\mathbf{f}/T\right), \sigma\left(\mathbf{g}/T\right)\right)$ defines the distance between logits of model $\mathbf{f}$ and model $\mathbf{g}$. Therefore, the definition of the D-distillation is satisfied. □

Define the set of metaparameters $\boldsymbol{\lambda}$ as a vector which components are coefficients before terms of $\mathcal{L}_{\text{train}}$ and temperature T:

$$\boldsymbol{\lambda} = [\lambda_1, T].$$

Define the bi-level optimization problem:

$$\hat{\boldsymbol{\lambda}} = \arg\min_{\boldsymbol{\lambda} \in \mathbb{R}^2} \mathcal{L}_{\text{val}}(\hat{\mathbf{w}}, \boldsymbol{\lambda}), \tag{1}$$

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w} \in \mathbb{R}^s} \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\lambda}), \tag{2}$$

where $\mathcal{L}_{\text{val}}$ is a validation loss function:

$$\mathcal{L}_{\text{val}}(\mathbf{w}, \boldsymbol{\lambda}) = -\sum_{(\mathbf{x},y)\in\mathfrak{D}_{\text{val}}} \sum_{k=1}^{K} y^k \log \frac{e^{\mathbf{g}(\mathbf{x},\mathbf{w})_k/T_{\text{val}}}}{\sum_{j=1}^{K} e^{\mathbf{g}(\mathbf{x},\mathbf{w})_j/T_{\text{val}}}},$$

the metaparameter T_{val} controls the temperature for the validation loss. It is set manually and is not subject to optimization.

3 Gradient-based metaparameter optimization

One of the methods to optimize metaparameters is to use a gradient-based method. Below we describe the scheme of its usage and an approach to approximate the trajectory of the optimized metaparameters.

Definition 2. *Define an optimization operator as an algorithm U that selects the model parameter vector $\mathbf{w}'$ using the parameters on the previous step $\mathbf{w}$.*

Optimize the parameters $\mathbf{w}$ using η optimization steps:

$$\hat{\mathbf{w}} = U \circ U \circ \dots \circ U(\mathbf{w}_0, \boldsymbol{\lambda}) = U^{\eta}(\mathbf{w}_0, \boldsymbol{\lambda}),$$

where $\mathbf{w}_0$ is the initial value of the model parameter vector $\mathbf{w}$, $\boldsymbol{\lambda}$ is the set of metaparameters.

Redefine the optimization problem using the operator U:

$$\hat{\boldsymbol{\lambda}} = \arg\min_{\boldsymbol{\lambda} \in \mathbb{R}^2} \mathcal{L}_{\text{val}}\big(U^{\eta}(\mathbf{w}_0, \boldsymbol{\lambda})\big).$$

Solve optimization problem (1) and (2) with gradient descent operator:

$$U(\mathbf{w}, \boldsymbol{\lambda}) = \mathbf{w} - \gamma \nabla \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\lambda}),$$

where γ is a learning rate. For the metaparameter optimization we use the greedy gradient-based method, which depends only on the model parameter value $\mathbf{w}$ on the previous step. On every iteration we obtain the following metaparameter value:

$$\boldsymbol{\lambda}' = \boldsymbol{\lambda} - \gamma_{\boldsymbol{\lambda}} \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\text{val}}(U(\mathbf{w}, \boldsymbol{\lambda}), \boldsymbol{\lambda}) = \boldsymbol{\lambda} - \gamma_{\boldsymbol{\lambda}} \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\text{val}}(\mathbf{w} - \gamma \nabla \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\lambda}), \boldsymbol{\lambda}). \tag{3}$$

In this paper we use a numerical difference approximation for this optimization procedure [7]. For further optimization cost reduction we propose to approximate the metaparameter optimization path. The path is predicted with linear models that are used periodically after a certain number of iterations e_1. After that the linear model is used to predict metaparameters for e_2 iterations:

$$\boldsymbol{\lambda}' = \boldsymbol{\lambda} + \mathbf{c}^\top \begin{pmatrix} z \\ 1 \end{pmatrix}, \quad (4)$$

where $\mathbf{c}$ is the vector of parameters optimized using least squares method, z is a number of optimization iteration. The algorithm of the proposed method is show in Fig. 2.

Algorithm 1 Metaparameter optimization

```
Require: number $e_1$ of iterations to use gradient-based optimization
Require: number $e_2$ of iterations to use linear predictions for metaparameters $\boldsymbol{\lambda}$
1: while not converged do
2:     Optimize $\boldsymbol{\lambda}$ and $\mathbf{w}$ for $e_1$ iterations solving a bi-level optimization problem
3:     traj =trajectory of $(\nabla\boldsymbol{\lambda})$ changes during optimization;
4:     Set $\mathbf{z} = [1, \dots, e_1]^\mathsf{T}$
5:     Optimize $\mathbf{c}$ using least square method:
           $\hat{\mathbf{c}} = \arg\min_{\mathbf{c}\in\mathbb{R}^2} ||\mathbf{traj} - \mathbf{z} \cdot c_1 + c_2||_2^2$
6:     Optimize $\mathbf{w}$ and predict $\boldsymbol{\lambda}$ for $e_2$ iterations using linear model with parameters $\mathbf{c}$.
7: end while
```

Fig. 2: An algorithm for the proposed method.

The diagram on Fig. 3 presents resulting optimization method. Model parameters are optimized on the first level of a bi-level optimization problem using $\mathfrak{D}_{\text{train}}$ subset and $\mathcal{L}_{\text{train}}$ function. Metaparameters are optimized in the second level using $\mathfrak{D}_{\text{val}}$ subset and $\mathcal{L}_{\text{val}}$ function. During e_1 iterations metaparameters are optimized using SGD and during e_2 iterations they are predicted using linear model. Then the obtained metaparameter values are used for model parameter optimization.
The following theorem proves the correctness of the approximation by a linear model.

Theorem 1. *If function $\mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\lambda})$ is smooth and convex and its Hessian $\mathbf{H} = \nabla^2_{\mathbf{w}}\mathcal{L}_{\text{train}}$ is invertible and can be well approximated by identity, $\mathbf{H} \approx \mathbf{I}$, then greedy algorithm* (3) *finds optimum solution of the bi-level problem* (1)*. If there is a domain $\mathcal{D} \in \mathbb{R}^2$ in metaparameter space where the gradient of metaparameters can be well approximated by a constant, then the optimization is linear w.r.t the metaparameters.*

$$\begin{tikzcd}
\mathbf{w} - \gamma\nabla\mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\lambda}) \arrow[r] & \hat{\mathbf{w}}, \mathbf{x} \arrow[r, "\mathbf{g}"] & \hat{y} \arrow[r] & \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\lambda}) \arrow[lll, bend right] \\
t \arrow[r] & i \arrow[r] & (\mathbf{x}, \mathbf{y}) \arrow[r] \arrow[dr] & \mathfrak{D}_{\text{train}} \arrow[u] \\
 & & & \mathfrak{D}_{\text{val}} \arrow[d, "e_1"'] \\
 & & & \mathcal{L}_{\text{val}}(\boldsymbol{\lambda}) \arrow[dll] \\
 & \boldsymbol{\lambda} - \gamma_{\boldsymbol{\lambda}}\nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\text{val}}(U(\mathbf{w}, \boldsymbol{\lambda}), \boldsymbol{\lambda}) \arrow[rr] & & \boldsymbol{\lambda}' \arrow[u] \arrow[uuuu, bend right] \arrow[d, bend right, "e_2"'] \\
 & & & \boldsymbol{\lambda}' + \mathbf{c}^{\top}\mathbf{z} \arrow[u, bend right]
\end{tikzcd}$$

Fig. 3: The diagram for the metaparameters optimization.

Proof. In work [12] it was derived a formula for $\nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\text{val}} = \nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\text{val}}(U(\mathbf{w}, \boldsymbol{\lambda}))$ in case when $\mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\lambda})$ is smooth and convex and $\mathbf{H}$ is invertible:

$$\nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\text{val}}(\boldsymbol{\lambda}) = \nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\text{val}} - (\nabla^2_{\mathbf{w},\boldsymbol{\lambda}}\mathcal{L}_{\text{train}})^{\top}(\nabla^2_{\mathbf{w}}\mathcal{L}_{\text{train}})^{-1}\nabla_{\mathbf{w}}\mathcal{L}_{\text{val}}.$$

It can be simplified by excluding the first term because function $\mathcal{L}_{\text{val}}$ does not depend on metaparameters explicitly:

$$\nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\text{val}}(\boldsymbol{\lambda}) = -(\nabla^2_{\mathbf{w},\boldsymbol{\lambda}}\mathcal{L}_{\text{train}})^{\top}(\nabla^2_{\mathbf{w}}\mathcal{L}_{\text{train}})^{-1}\nabla_{\mathbf{w}}\mathcal{L}_{\text{val}}.$$

If $\nabla^2_{\mathbf{w}}\mathcal{L}_{\text{train}}$ can be well approximated by identity then greedy algorithm gives optimum to the bi-level problem if its step has the following formula [8]:

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \eta_1(\nabla^2_{\mathbf{w},\boldsymbol{\lambda}}\mathcal{L}_{\text{train}})^{\top}\nabla_{\mathbf{w}}\mathcal{L}_{\text{val}}.$$

We also exclude $\nabla^2_{\mathbf{w}}\mathcal{L}_{\text{train}}$ by replacing it with identity.
Return to the simplified formula of the gradient:

$$\nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\text{val}}(\boldsymbol{\lambda}) = -(\nabla^2_{\mathbf{w},\boldsymbol{\lambda}}\mathcal{L}_{\text{train}})^{\top}\nabla_{\mathbf{w}}\mathcal{L}_{\text{val}}.$$

Suppose that there is a domain $\mathcal{D}$ where $\nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\text{val}}(\boldsymbol{\lambda})$ can be approximated by a constant vector

$$\nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\text{val}}(\boldsymbol{\lambda}) \approx \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}.$$

Then on $\mathcal{D}$ the optimization step can be represented as

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t - \gamma_{\boldsymbol{\lambda}} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix},$$

which has a linear form similar to (4). $\square$

4 Experiments

The purpose of the experiment is to evaluate the performance of the proposed distillation method and to analyze the resulting models and their metaparameters. The method is evaluated using the synthetic dataset, Fashion-MNIST dataset, and CIFAR-10 dataset. [1] For the CIFAR-10, we conducted two experiments: on the whole dataset and the reduced training subset, $|\mathfrak{D}_{\text{train}}| = 12800$.

We analyzed the following metparameter optimization methods:

1) optimization without distillation;
2) optimization with randomly initialized metaparameter values. The metaparameters were sampled from uniform distribution

$$\lambda_1 \sim \mathcal{U}(0;1), \quad T \sim \mathcal{U}(0.1, 10).$$

3) optimization with "naive" metaparameter assignment: setting

$$\lambda_1 = 0.5, T = 1;$$

4) gradient-based optimization;
5) proposed method with $e_1 = e_2 = 10$.

We used a full training dataset $\mathfrak{D}$ for the methods 1-3. For all the experiments except the experiment on the full CIFAR-10 dataset we also used metaparameter optimization with probablistic model. As a such optimization we used hyperopt library [3] providing a Parzen estimator-based metparameter optimization. For this method we used 5 run before the final metaparameter prediction.

We used accuracy as an external criterion:

$$\text{accuracy} = \frac{1}{m}\sum_{i=1}^{m}[\mathbf{g}(\mathbf{x}_i, \mathbf{w}) = y_i],$$

For all the experiment we sample initial metaparameter values in the following way:

$$\lambda_1 \sim \mathcal{U}(0, 1), \quad \log_{10} T \sim \mathcal{U}(-1, 1).$$

All the experiments were ran 10 times, the results were averaged.

The overall results are displayed in Table 2. The dependency of the accuracy on the epoch number for the synthetic dataset and reduced CIFAR-10 dataset is shown in Fig. 4.

[1] The source code for the computational experiment will be available in the camera-ready version of the paper.

Table 2: Experiment results. The numbers in brackets correspond to the maximum values in the experiment.

Method	Synthetic dataset	Fashion-MNIST	Reduced CIFAR-10	CIFAR-10
Without distillation	0.63 (0.63)	0.87 (0.88)	0.55 (0.56)	0.65 (0.66)
Naive metaparameters	0.63 (0.63)	0.87 (0.88)	0.55 (0.56)	0.66 (0.67)
Random metaparameters	0.64 (0.72)	0.79 (0.88)	0.54 (0.57)	0.64 (0.67)
Gradient-based optimization	**0.77** (0.78)	**0.88** (0.89)	**0.57** (0.61)	**0.70** (0.72)
Hyperopt	**0.77** (0.78)	0.87 (0.88)	0.55 (0.58)	-
Proposed	0.76 (0.78)	**0.88** (0.89)	**0.57**	**0.70** (0.72)

4.1 Experiment on synthetic dataset

For the evaluation of the method we conducted a computational experiment using a synthetic dataset:

$$\mathfrak{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}, \quad x_{ij} \in \mathcal{N}(0,1),\ j = 1,2, \quad x_{i3} = [\text{sign}(x_{i1}) + \text{sign}(x_{i2}) > 0],$$
$$y_i = \text{sign}(x_{i1} \cdot x_{i2} + \delta),$$

where $\delta \in \mathcal{N}(0, 0.5)$ is a noise. The size of the student model dataset is much smaller than the size of the teacher model dataset and $\mathfrak{D}_{\text{train}}$. In order to show the proof of concept for the proposed method, in this experiment, we split the dataset into 3 parts: the training dataset for the teacher, consisting of 200 objects, the training dataset for the student, consisting of 15 objects, and the validation part, which equals to the test dataset, $\mathfrak{D}_{\text{val}} = \mathfrak{D}_{\text{test}}$. It also consists of 200 objects. The visualization of the synthesized dataset is shown in Fig. 5. The teacher was trained for 20000 iterations SGD with a learning rate equal to 10^{-2}. For its training, we used modified feature space:

$$x_{i3} = [\text{sign}(x_{i1}) + \text{sign}(x_{i2}) + 0.1 > 0].$$

The purpose of such modification is to prevent the teacher from nearly perfect fitting on the training dataset. In this case, learning without classification term, $\lambda_1 = 0$, would be preferable for the student. The student model was trained for 2000 iterations with SGD learning rate equal to 1.0 and $T_{\text{val}} = 0.1$.

There were conducted series of experiments to determine the best e_1 and the best e_2. Fig. 6.a plots the model accuracy for different e_1 with e_2 set to 10. Fig. 6.b plots the model accuracy for different e_2. As we can see, with increasing e_1 and e_2 the approximation quality of metaparameter update trajectory decreases.

Fig. 4.a plots model accuracy for different methods. The best results were obtained with optimized metaparameters and the proposed method. As we can

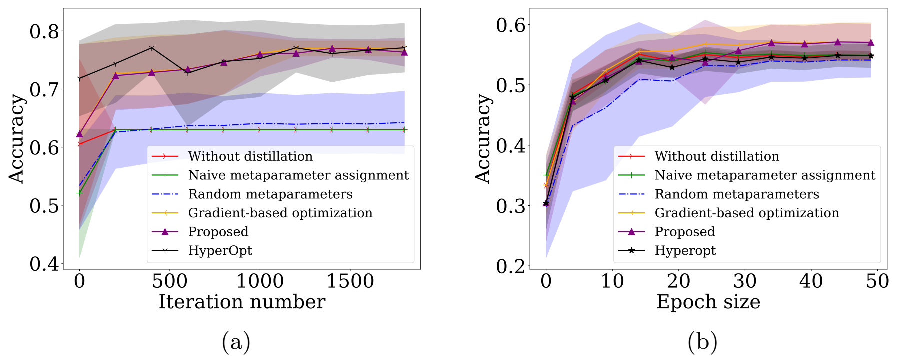

Fig. 4: Model accuracy for different datasets: a) syntetic dataset, b) CIFAR-10 reduced dataset.

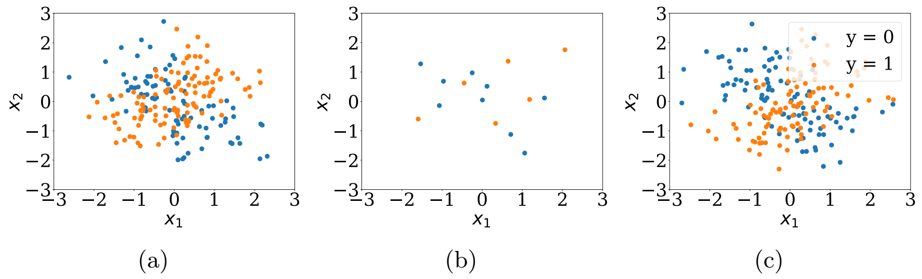

Fig. 5: Visualization of a) teacher model dataset; b) student model dataset; c) test part

see, the proposed method gives a good approximation for the metaparameter optimization for this experiment.

4.2 Experiments on CIFAR-10 and Fashion-MNIST datasets

We split both the dataset in the ratio 9:1 for the training and validation. For the model parameter optimization, we used stochastic gradient descent with initial learning set to 1.0. The learning rate was multiplied by 0.5 every 10 epochs. We set T_{val} equal to 1.0.

For the experiments on CIFAR-10 we used the pretrained ResNet model from [11] as a teacher. As a student, we used CNN model with three convolutional layers and two fully connected layers. For the experiments on the reduced dataset, we used the learning rate for the metaparameter optimization equal to 0.25 and trained the model for 50 epochs. For the experiments on the full dataset, we used the learning rate for the metaparameters set to 0.1. We trained the model for 100 epochs.

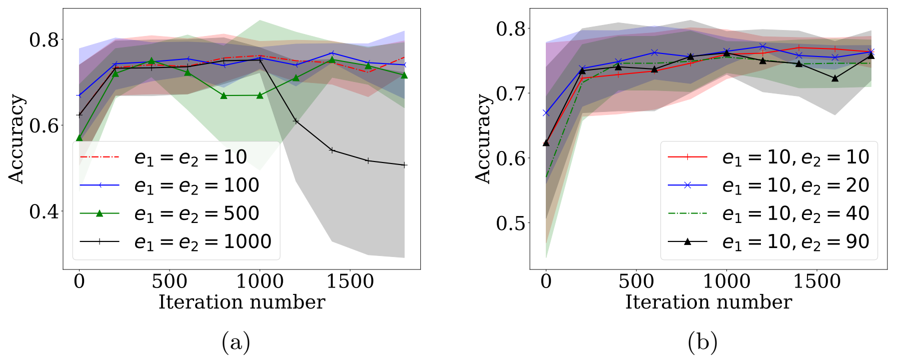

Fig. 6: Model accuracy with e_1 and e_2 values: a) $e_1 = e_2$; b) variation of e_2 with $e_1 = 10$.

For the experiments on Fashion-MNIST, we used teacher and student model architectures similar to those in CIFAR-10 experiments. We used the learning rate for the metaparameter optimization equal to 0.1 and trained the model for 50 epochs.

As we can see from the results from the Table 2, the both the proposed method and gradient-based method give quite competitive results. The probabilistic model-based method shows a similar performance. Thus, the gradient-based methods are preferable because they have similar performance but require fewer optimization iterations. At the same time, the gradient-based methods suffer from getting stuck in local minima, so the variance of their results is much higher than that of other methods. This can be seen from the Fig. 4 and the Table 2.

5 Conclusion

The parameter optimization problem for deep learning models was analyzed. The generalization of knowledge distillation method was proposed that uses gradient-based metaparameter optimization. Model parameters are optimized on the first level and metaparameters on the second. We proposed a method to reduce metaparameter optimization cost for gradient-based optimization. The proposed method was evaluated in the numerical experiment on the Fashion-MNIST, CIFAR-10 datasets and on the synthetic dataset. The computational experiment showed the effectiveness of gradient-based optimization for selecting of metaparameters of the distillation loss function. The possibility of optimization path approximation using linear models was analyzed. In future, we are planning to investigate more complicated predictive models for metaparameter optimization.

References

1. Bakhteev, O.Y., Strijov, V.V.: Comprehensive analysis of gradient-based hyperparameter optimization algorithms. Ann. Oper. Res **289**(1), 51–65 (2020)
2. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of machine learning research **13**(2) (2012)
3. Bergstra, J., Yamins, D., Cox, D.: Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In: International conference on machine learning. pp. 115–123. PMLR (2013)
4. Bishop, C.M.: Pattern recognition and machine learning (information science and statistics) (2006)
5. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. CoRR **abs/1503.02531** (2015), http://arxiv.org/abs/1503.02531
6. Krizhevsky, A., et al.: Learning multiple layers of features from tiny images (2009)
7. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)
8. Luketina, J., Berglund, M., Greff, K., Raiko, T.: Scalable gradient-based tuning of continuous regularization hyperparameters. CoRR **abs/1511.06727** (2015), http://arxiv.org/abs/1511.06727
9. Maclaurin, D., Duvenaud, D., Adams, R.P.: Gradient-based hyperparameter optimization through reversible learning. CoRR **abs/1502.03492** (2015), http://arxiv.org/abs/1502.03492
10. Passalis, N., Tzelepi, M., Tefas, A.: Heterogeneous knowledge distillation using information flow modeling. In: CVPR. pp. 2336–2345. IEEE (2020), https://ieeexplore.ieee.org/xpl/conhome/9142308/proceeding
11. Passalis, N., Tzelepi, M., Tefas, A.: Heterogeneous knowledge distillation using information flow modeling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)
12. Pedregosa, F.: Hyperparameter optimization with approximate gradient. CoRR **abs/1602.02355** (2016), http://arxiv.org/abs/1602.02355
13. Rasley, J., Rajbhandari, S., Ruwase, O., He, Y.: Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 3505–3506 (2020)
14. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. CoRR **abs/1708.07747** (2017), http://arxiv.org/abs/1708.07747