

Gradient Methods for Optimizing Metaparameters in the Knowledge Distillation Problem

M. Gorpinich^{1*}, O. Yu. Bakhteev^{2**}, and V. V. Strijov^{2***}

¹*Moscow Institute of Physics and Technology, Dolgoprudnyi, Moscow oblast, 141701 Russia*

²*Dorodnicyn Computing Centre, Russian Academy of Sciences, Moscow, 119333 Russia*
*e-mail: *gorpinich.m@phystech.edu, **bakhteev@phystech.edu, ***strijov@gmail.com*

Received February 17, 2022; revised June 23, 2022; accepted June 29, 2022

Abstract—The paper investigates the distillation problem for deep learning models. Knowledge distillation is a metaparameter optimization problem in which information from a model of a more complex structure, called a teacher model, is transferred to a model of a simpler structure, called a student model. The paper proposes a generalization of the distillation problem for the case of optimization of metaparameters by gradient methods. Metaparameters are the parameters of the distillation optimization problem. The loss function for such a problem is the sum of the classification term and the cross-entropy between the responses of the student model and the teacher model. Assigning optimal metaparameters to the distillation loss function is a computationally difficult task. The properties of the optimization problem are investigated so as to predict the metaparameter update trajectory. An analysis of the trajectory of the gradient optimization of metaparameters is carried out, and their value is predicted using linear functions. The proposed approach is illustrated using a computational experiment on CIFAR-10 and Fashion-MNIST samples as well as on synthetic data.

Keywords: machine learning, knowledge distillation, metaparameter optimization, gradient optimization, metaparameter assignment

DOI: 10.1134/S0005117922100071

1. INTRODUCTION

The paper considers the problem of distillation of deep learning models. Optimizing a deep learning model is computationally challenging [12]. The paper investigates a particular case of an optimization problem called knowledge distillation. It allows using both the training sample and the information contained in pretrained models. Knowledge distillation [5] is the problem of optimizing model parameters that takes into account not only the information contained in the original sample but also the information contained in the teacher model. The teacher model has a high complexity. It contains information about the sample as well as about the distributions of model parameters that will be transferred. The model of a simpler structure, called the student model, is optimized by transferring the knowledge of the teacher model.

The procedure of optimization of metaparameters in the knowledge distillation problem is investigated. Metaparameters are the parameters of the optimization problem. Correct assignment of metaparameters can considerably affect the performance of the final model [11]. Unlike [9, 11], this paper takes into account the difference between hyperparameters, probability parameters of the prior distribution [4], and metaparameters. Despite the number of metaparameter and hyperparameter optimization methods used in deep learning such as random search [2] or models based on the use of probabilistic models [3], many approaches propose sequentially generating a random

Table 1. Complexity of various metaparameter and hyperparameter optimization methods. Here $|\mathbf{w}|$ is the number of model parameters, $|\lambda|$ is the number of metaparameters, r is the number of runs of stochastic optimization methods, and s is the complexity of generation from probabilistic models

Method	Type of optimization method	Complexity
Random search [2]	Stochastic	$O(r \cdot \mathbf{w})$
Based on probabilistic models [3]	Stochastic	$O(r \cdot (\mathbf{w} + s))$
Greedy gradient [8]	Gradient	$O(\mathbf{w} \cdot \lambda)$
Greedy gradient with difference approximation [7]	Gradient	$O(\mathbf{w} + \lambda)$

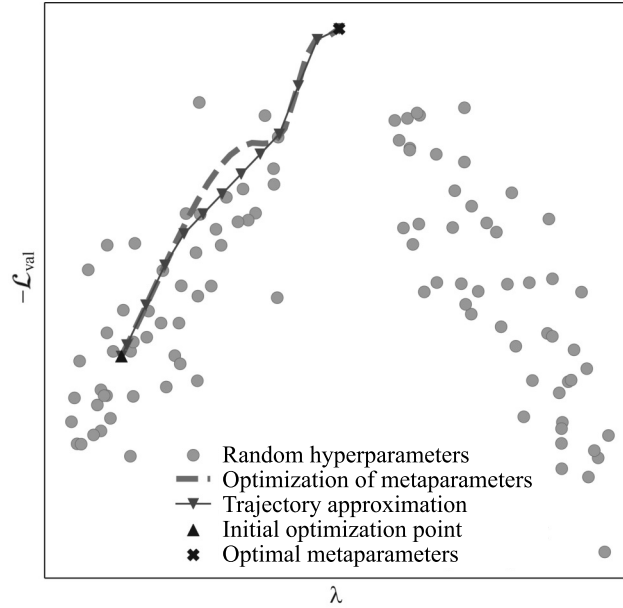


Fig. 1. Scheme of the proposed method: instead of directly optimizing the values of metaparameter λ , it is proposed to approximate the optimization trajectory using linear models to achieve a minimum loss function on the validation part of the sample \mathcal{L}_{val} . Random metaparameters are not minimum points of the function \mathcal{L}_{val} and deliver suboptimal model performance.

value of metaparameters and estimating the quality of the model trained with the given values of the hyperparameters. This approach may not be suitable in the case of training models that require substantial training time. Table 1 lists the values of the complexity of various approaches to optimizing metaparameters. It can be seen that if the parameter optimization takes a long time, approaches that require several runs of optimization are inefficient.

It is proposed to treat the metaparameter optimization problem as a bilevel optimization problem. The first level optimizes the model parameters and the second one, the metaparameters [1, 8, 9]. A greedy gradient method for solving the bilevel problem is described in [8]. Various gradient methods and random search are analyzed in [1]. In the present paper, we analyze an approach to optimizing and predicting metaparameters obtained after applying gradient methods. It can be seen from Table 1 that for large problems, gradient metaparameter optimization methods are preferable. However, even with a greedy metaparameter optimization algorithm with a difference approximation, metaparameter optimization becomes much more computationally demanding, as demonstrated in [7]. To reduce the optimization costs, in this paper we analyze the metaparameter optimization trajectory and predict its value using linear models. This method is illustrated in

Fig. 1. This method is evaluated and compared with other metaparameter optimization methods on image samples from CIFAR-10 [6] and Fashion-MNIST [14] and a synthetic sample.

2. STATEMENT OF THE PROBLEM

We solve a classification problem of the form

$$\mathfrak{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m, \quad \mathbf{x}_i \in \mathbb{R}^n, \quad \mathbf{y}_i \in \mathbb{Y} = \{\mathbf{e}_k | k = 1, \dots, K\},$$

where \mathbf{e}_k is the k th column of the identity matrix and \mathbf{y}_i is a vector with unit in place of class \mathbf{x}_i .

We divide the sample \mathfrak{D} into two subsets, $\mathfrak{D} = \mathfrak{D}_{\text{train}} \sqcup \mathfrak{D}_{\text{val}}$. The subset $\mathfrak{D}_{\text{train}}$ will be used to optimize the model parameters, and the subset $\mathfrak{D}_{\text{val}}$ will be used to optimize the metaparameters.

Consider a teacher model $\mathbf{f}(\mathbf{x})$ that was trained on the sample $\mathfrak{D}_{\text{train}}$. We optimize the student model $\mathbf{g}(\mathbf{x}, \mathbf{w})$, $\mathbf{w} \in \mathbb{R}^s$, by transferring the knowledge of the teacher model. Let us define this problem formally.

Definition 1. Let a function $D : \mathbb{R}^s \rightarrow \mathbb{R}_+$ define the distance between models \mathbf{g} and \mathbf{f} . A D -distillation of the student model is an optimization of the student model parameters that minimizes the function D .

We define a loss function $\mathcal{L}_{\text{train}}$ that takes into account the transfer of knowledge from the model \mathbf{f} to the model \mathbf{g} ,

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

where y_k is the k th component of the response vector and T is the temperature parameter in the distillation problem. The temperature T has the following properties:

1. As $T \rightarrow 0$, we obtain the unit vector $\left\{ e^{\mathbf{g}(\mathbf{x}, \mathbf{w})_k/T} / \sum_{j=1}^K e^{\mathbf{g}(\mathbf{x}, \mathbf{w})_j/T} \right\}_{k=1}^K$.
2. As $T \rightarrow \infty$, we obtain a vector with equal probabilities.

Let us show that the optimization of $\mathcal{L}_{\text{train}}$ is a D -distillation for $\lambda_1 = 0$.

Proposition 1. If $\lambda_1 = 0$, then the optimization of the loss function (1) is a D -distillation with $D = D_{KL}(\sigma(\mathbf{f}(\mathbf{x})/T), \sigma(\mathbf{g}(\mathbf{x}, \mathbf{w})/T))$, where σ is the function softmax $= \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}}$ and D_{KL} is the Kullback–Leibler divergence.

Proof. For $\lambda_1 = 0$, we have

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

We conclude that $\mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\lambda})$ is equal to $D_{KL}(\sigma(\mathbf{f}(\mathbf{x})/T), \sigma(\mathbf{g}(\mathbf{x}, \mathbf{w})/T))$ up to a constant C that does not affect optimization. The constant is the entropy of $\sigma(\mathbf{f}(\mathbf{x})/T)$. The function $D_{KL}(\sigma(\mathbf{f}/T), \sigma(\mathbf{g}/T))$ defines the distance between the logits of the model \mathbf{f} and the model \mathbf{g} . We conclude that the definition of D -distillation is satisfied. ■

Let us define the set $\boldsymbol{\lambda}$ of metaparameters as the vector whose components are the coefficient λ_1 multiplying the terms in $\mathcal{L}_{\text{train}}$ and the temperature T ,

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

Define a bilevel problem

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

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w} \in \mathbb{R}^s} \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\lambda}), \quad (3)$$

where \mathcal{L}_{val} is the validation loss function

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

and the metaparameter T_{val} determines the temperature in the validation loss function. Its value has been chosen manually and is not subject to optimization.

3. GRADIENT OPTIMIZATION OF METAPARAMETERS

One method for optimizing metaparameters is to use gradient methods. Below is a diagram of their application and an approach to optimizing the trajectory of metaparameters.

Definition 2. Let's define an *optimization operator* as an algorithm U that selects the parameter vector \mathbf{w}' of the model using the parameter values \mathbf{w} at the previous step.

Let us optimize 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 parameter vector \mathbf{w} and $\boldsymbol{\lambda}$ is the set of metaparameters.

Let us restate the optimization problem using the definition of the operator U ,

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

We solve the optimization problem (2), (3) using the gradient descent operator,

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

where γ is the gradient descent step length. To optimize the metaparameters, we use the greedy gradient method that depends on the parameter values \mathbf{w} at the previous step alone. At each iteration, we obtain the following value of the metaparameters:

$$\boldsymbol{\lambda}' = \boldsymbol{\lambda} - \gamma_\lambda \nabla_\lambda \mathcal{L}_{\text{val}}(U(\mathbf{w}, \boldsymbol{\lambda}), \boldsymbol{\lambda}) = \boldsymbol{\lambda} - \gamma_\lambda \nabla_\lambda \mathcal{L}_{\text{val}}(\mathbf{w} - \gamma \nabla \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\lambda}), \boldsymbol{\lambda}). \quad (4)$$

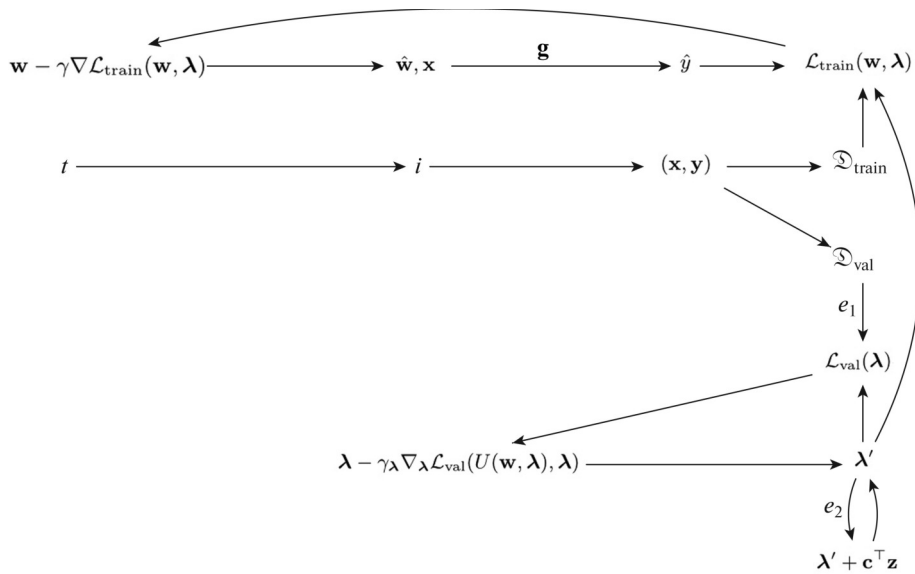


Fig. 2. Scheme of metaparameter optimization.

In this paper, we use a numerical difference approximation for this optimization procedure [7],

$$\begin{aligned} \frac{d\mathcal{L}_{\text{val}}(\mathbf{w}', \boldsymbol{\lambda})}{d\boldsymbol{\lambda}} &= \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\text{val}}(\mathbf{w}', \boldsymbol{\lambda}) - \gamma \nabla_{\boldsymbol{\lambda}, \mathbf{w}'}^2 \mathcal{L}_{\text{val}}(\mathbf{w}', \boldsymbol{\lambda}) \nabla_{\mathbf{w}'} \mathcal{L}_{\text{val}}(\mathbf{w}', \boldsymbol{\lambda}), \\ \nabla_{\boldsymbol{\lambda}, \mathbf{w}'}^2 \mathcal{L}_{\text{val}}(\mathbf{w}', \boldsymbol{\lambda}) \nabla_{\mathbf{w}'} \mathcal{L}_{\text{val}}(\mathbf{w}', \boldsymbol{\lambda}) &\approx \frac{\nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\text{val}}(\mathbf{w}^+, \boldsymbol{\lambda}) - \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\text{val}}(\mathbf{w}^-, \boldsymbol{\lambda})}{2\varepsilon}, \\ \boldsymbol{\lambda}' &\approx \boldsymbol{\lambda} - \gamma \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\text{val}}(\mathbf{w}', \boldsymbol{\lambda}) + \gamma \frac{\nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\text{val}}(\mathbf{w}^+, \boldsymbol{\lambda}) - \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\text{val}}(\mathbf{w}^-, \boldsymbol{\lambda})}{2\varepsilon}, \end{aligned}$$

where $\mathbf{w}' = \mathbf{w} - \gamma \nabla \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\lambda})$, $\mathbf{w}^\pm = \mathbf{w}' \pm \varepsilon \nabla_{\mathbf{w}'} \mathcal{L}_{\text{val}}(\mathbf{w}', \boldsymbol{\lambda})$, and ε is a some given constant.

To further reduce the optimization costs, it is proposed to approximate the metaparameter optimization trajectory. The trajectory is predicted using linear models that are used periodically after a given number of iterations e_1 . After that, the linear model is used to predict the metaparameters over e_2 iterations,

$$\boldsymbol{\lambda}' = \boldsymbol{\lambda} + \mathbf{c}^\top \begin{pmatrix} z \\ 1 \end{pmatrix}, \quad (5)$$

where \mathbf{c} is the parameter vector of the linear model optimized using the least squares method and z is the number of optimization iterations.

The diagram in Fig. 2 describes the resulting optimization method. The model parameters are optimized at the first level of a bilevel optimization problem using the subset $\mathcal{D}_{\text{train}}$ and the loss function $\mathcal{L}_{\text{train}}$. The metaparameters are optimized at the second level using the subset \mathcal{D}_{val} and the loss function \mathcal{L}_{val} . Over e_1 iterations, the metaparameters are optimized using the stochastic gradient descent method. Over e_2 iterations, they are predicted using linear models.

The following theorem proves the well-posedness of the proposed approximation for a simple case where the parameters \mathbf{w} of the model \mathbf{g} have reached the optimum of problem (3), the Hessian $\mathbf{H} = \nabla_{\mathbf{w}}^2 \mathcal{L}_{\text{train}}$ is the identity matrix, and the metaparameters are optimized in the domain where the gradient of the metaparameters can be approximated by a constant. Note that in the general case, these conditions are not met when optimizing deep learning models. It was shown

Algorithm 1 (optimization of metaparameters—algorithm for the proposed method).

require the number e_1 of iterations using gradient optimization

require the number e_2 of iterations with prediction of $\boldsymbol{\lambda}$ by linear models

- 1: **while** there is no convergence **do**
- 2: Optimize $\boldsymbol{\lambda}$ and \mathbf{w} over e_1 iterations by solving a bilevel problem
- 3: **traj** = trajectory($\nabla \boldsymbol{\lambda}$) changes during optimization
- 4: Set $\mathbf{z} = [1, \dots, e_1]^\top$
- 5: Optimize \mathbf{c} by the LSM,

$$\hat{\mathbf{c}} = \arg \min_{\mathbf{c} \in \mathbb{R}^2} \|\mathbf{traj} - \mathbf{z} \cdot c_1 + c_2\|_2^2$$

- 6: Optimize \mathbf{w} and predict $\boldsymbol{\lambda}$ over e_2 iterations using the linear model with parameters \mathbf{c}
 - 7: **end while**
-

in [8, 13] that the use of methods for normalizing intermediate sample representations under the influence of nonlinear functions included in the deep learning model brings the Hessian of the loss function closer to unity. An analysis of the performance of the gradient optimization of metaparameters for the case in which the model parameters have not reached the optimum can be found in [11].

Theorem 1. *If the function $\mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\lambda})$ is smooth and convex and its Hessian $\mathbf{H} = \nabla_{\mathbf{w}}^2 \mathcal{L}_{\text{train}}$ is the identity matrix, $\mathbf{H} = \mathbf{I}$, and also if the parameters \mathbf{w} are equal to \mathbf{w}^* , where \mathbf{w}^* is a point of local minimum for the current value of $\boldsymbol{\lambda}$, then the greedy algorithm (4) finds the optimal solution of the bilevel problem. If there exists a domain $\mathcal{D} \in \mathbb{R}^2$ in the metaparameter space such that the gradient of the metaparameters can be approximated by a constant, then the optimization is linear in the metaparameters.*

Proof. In the paper [11], a formula for $\nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\text{val}} = \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\text{val}}(U(\mathbf{w}, \boldsymbol{\lambda}))$ was obtained for the case in which $\mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\lambda})$ is smooth and convex and a point \mathbf{w}^* of local minimum was found for the current value of $\boldsymbol{\lambda}$,

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

This formula is simplified by eliminating the first term, since the function \mathcal{L}_{val} does not explicitly depend on the metaparameters,

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

If $\nabla_{\mathbf{w}}^2 \mathcal{L}_{\text{train}}$ is equal to the identity matrix, then the greedy algorithm produces the optimal bilevel problem if its step is given by the following formula [8]:

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

We also replace $\nabla_{\mathbf{w}}^2 \mathcal{L}_{\text{train}}$ by the identity matrix.

Let us return to the simplified gradient formula

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

Assume that there exists a domain \mathcal{D} in which $\nabla_{\lambda} \mathcal{L}_{\text{val}}(\lambda)$ is equal to the constant vector

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

Then the optimization step in \mathcal{D} can be represented in the form

$$\lambda_{t+1} = \lambda_t - \gamma_{\lambda} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

and has a form similar to (5). ■

4. COMPUTATIONAL EXPERIMENT

The purpose of the experiment is to evaluate the performance of the proposed distillation method and analyze the resulting models and their metaparameters. The method is evaluated on a synthetic sample as well as CIFAR-10 and Fashion-MNIST samples. Two types of experiments were carried out on the CIFAR-10 sample—on the entire sample, $|\mathcal{D}_{\text{train}}| = 50\,000$, and on the reduced training sample, $|\mathcal{D}_{\text{train}}| = 12\,800$.

The following metaparameter optimization methods were analyzed:

1. Optimization with no distillation.
2. Optimization with random initialization of metaparameters. Metaparameters are generated from the uniform distribution

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

3. Optimization with “naïve” metaparameter assignment,

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

4. Gradient optimization.
5. The proposed method with $e_1 = e_2 = 10$.
6. Optimization using a probabilistic model. For this type of optimization, we used the Hyperopt library [3], which implements optimization using the Parzen window method. For this method, five runs were performed before the final prediction of metaparameters.

The entire training set \mathcal{D} was used for methods 1–3. For methods 4–6, the sample was divided into training, validation, and control $\mathcal{D} = \mathcal{D}_{\text{train}} \sqcup \mathcal{D}_{\text{val}} \sqcup \mathcal{D}_{\text{test}}$.

The “accuracy” metric was used as an external performance criterion,

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

For all experiments, the initial values of the metaparameters were generated as follows:

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

Ten runs were carried out for each experiment, and then the results were averaged. The code for the experiment is available at [15].

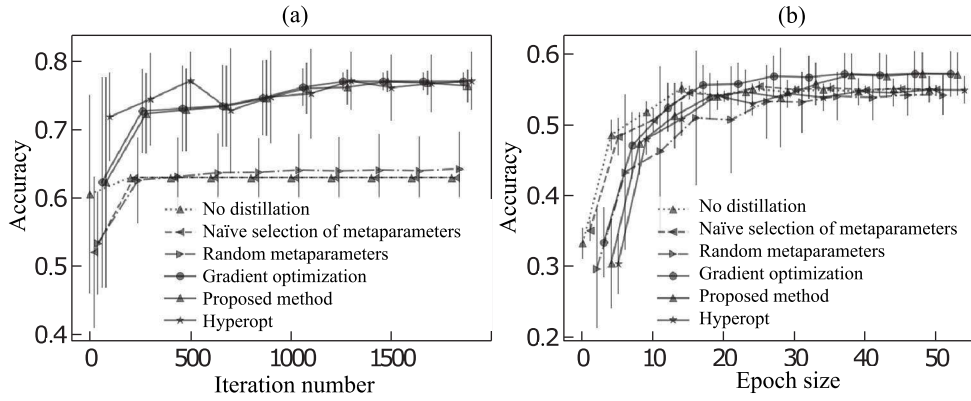


Fig. 3. Model accuracy on samples: (a) synthetic, (b) reduced CIFAR-10. Here and below, the points are slightly shifted with respect to the abscissa axis for better readability of the graphs.

Table 2. Experiment results. The numbers in parentheses are the maximum accuracy values obtained in particular experiments

Method	Synthetic sample	Fashion-MNIST	Reduced CIFAR-10	CIFAR-10
No distillation	0.63 (0.63)	0.87 (0.88)	0.55 (0.56)	0.65 (0.66)
Naïve 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 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)	0.65 (0.69)
Proposed method	0.76 (0.78)	0.88 (0.89)	0.57	0.70 (0.72)

The final results are presented in Table 2. The dependence of accuracy on the iteration number on a synthetic sample and a reduced version of CIFAR-10 is shown in Fig. 3.

4.1. Experiment on a Synthetic Sample

To evaluate the method obtained, an experiment was conducted on a synthetic sample,

$$\mathfrak{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m, \quad x_{ij} \in \mathcal{N}(0, 1), \quad j = 1, 2,$$

$$x_{i3} = [\text{sgn}(x_{i1}) + \text{sgn}(x_{i2}) > 0], \quad y_i = \text{sgn}(x_{i1} \cdot x_{i2} + \delta),$$

where $\delta \in \mathcal{N}(0, 0.5)$ is the noise. The sample size of the student model is much smaller than the sample size of the teacher model and $\mathfrak{D}_{\text{train}}$. To correctly demonstrate the proposed method in this experiment, the sample was divided into three parts: a training sample for the teacher model consisting of 200 objects, a training sample for the student model consisting of 15 objects, and a validation set, which is also a test set, $\mathfrak{D}_{\text{val}} = \mathfrak{D}_{\text{test}}$. It also consists of 200 objects. The sample visualization is depicted in Fig. 4. The teacher model was trained over 20 000 iterations using the stochastic gradient descent method with step length 10^{-2} . A modified feature space was used to train it,

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

This modification does not allow the teacher model to accurately predict the training sample. In this case, to train the student model, it is preferable to use only the distillation term, $\lambda_1 = 0$. The

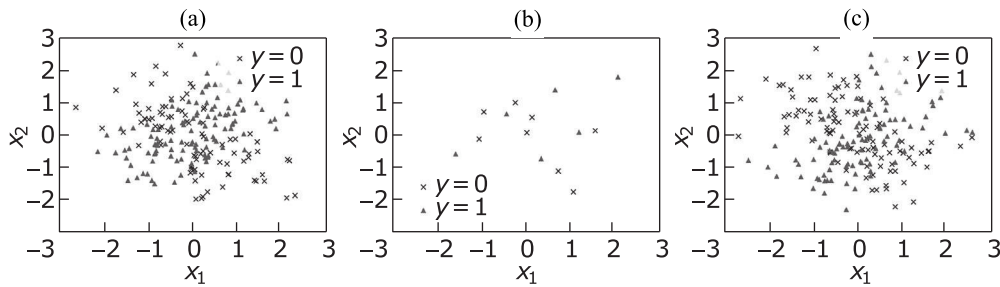


Fig. 4. Sample visualization for (a) teacher model, (b) student model, and (c) test sample.

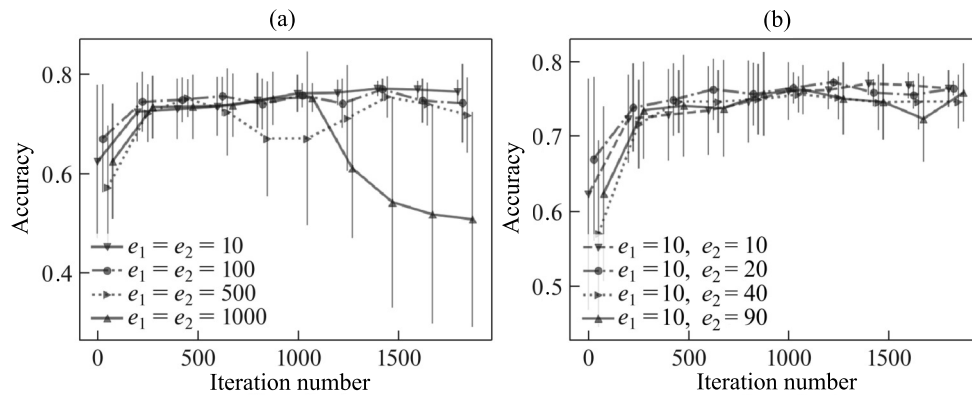


Fig. 5. Model accuracy with e_1 and e_2 values: (a) $e_1 = e_2$, (b) a selection of e_2 with $e_1 = 10$.

student model was trained over 2000 iterations using the stochastic gradient descent method with step length 1.0 and $T_{\text{val}} = 0.1$.

A series of experiments was carried out to determine the best values of e_1 and e_2 . Figure 5a shows the accuracy graph for various e_1 with e_2 equal to 10. Figure 5b shows the accuracy for various values of e_2 . It can be seen that as e_1 and e_2 increase, the quality of approximation of the metaparameter update trajectory decreases.

Figure 3a shows the accuracy of the model for various methods. The best results were obtained for the optimized values of the metaparameters and the proposed method. It can be seen how well the proposed method approximates the optimization of metaparameters in this experiment.

4.2. Experiments on CIFAR-10 and Fashion-MNIST Samples

Both samples were split in proportion 9:1 for training and validation. The stochastic gradient descent method with initial step length 1.0 was used to optimize the model parameters. The step length was multiplied by 0.5 every 10 epochs. The value of T_{val} was set to 1.0.

For the experiment on the CIFAR-10 sample, a pretrained ResNet model in [10] was used as a teacher model. A CNN model with three convolutional layers and two fully connected layers was used as a student model.

For the reduced sample experiments, the step length for metaparameter optimization was 0.25 and the model was trained for 50 epochs. For the experiment on the full sample, the step length 0.1 was used. The model was trained for 100 epochs.

For the experiment on the Fashion-MNIST sample, we used architectures of the student and teacher models similar to the architectures in the experiment on the CIFAR-10 sample. The step length 0.1 was used to optimize the metaparameters, and the model was trained for 50 epochs.

It can be seen from the results in Table 2 that the proposed method and the gradient methods give a high accuracy value. However, the disadvantage of gradient methods is that they get stuck at local minimum points, which results in a much higher variance of results than for other methods. This effect can be seen in Fig. 3 and in Table 2.

5. CONCLUSIONS

The problem of optimizing the parameters of a deep learning model was investigated. A generalization of distillation methods was proposed that consists in gradient optimization of metaparameters. The model parameters are optimized at the first level, and at the second level, metaparameters that specify the type of the optimization problem are optimized. A method has been proposed that reduces the computational complexity of optimizing metaparameters for gradient optimization. The properties of the optimization problem and methods for predicting the trajectory of optimizing the model metaparameters were studied. The model metaparameters are the parameters of the optimization problem of distillation. The proposed generalization has made it possible to distill the model with better performance characteristics and in a smaller number of optimization iterations. This approach is illustrated using a computational experiment on CIFAR-10 and Fashion-MNIST samples and on a synthetic sample. The computational experiment has shown the efficiency of gradient optimization for the problem of choosing the metaparameters of the distillation loss function. The possibility of approximating the optimization trajectory of metaparameters by a locally linear model is analyzed. Further, it is planned to study the optimization problem and analyze the quality of approximation to the optimization trajectory of metaparameters by more complicated predictive models.

FUNDING

This work was supported by K.V. Rudakov's Academic Scholarship and by the Russian Foundation for Basic Research, project no. 20-07-00990.

REFERENCES

1. Bakhteev, O.Y. and Strijov, V.V., Comprehensive analysis of gradient-based hyperparameter optimization algorithms, *Ann. Oper. Res.*, 2020, vol. 289, no. 1, pp. 51–65.
2. Bergstra, J. and Bengio, Y., Random search for hyper-parameter optimization, *Mach. Learn. Res.*, 2012, vol. 13, no. 2.
3. Bergstra, J., Yamins, D., and Cox, D., Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, *Int. Conf. Mach. Learn.* (2013), pp. 115–123.
4. Bishop, C.M., *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 2006.
5. Hinton, G.E., Vinyals, O., and Dean, J., Distilling the knowledge in a neural network, *CoRR*, 2015, vol. abs/1503.02531. [arXiv:abs/1503.02531](https://arxiv.org/abs/1503.02531).
6. Krizhevsky, A. et al., *Learning Multiple Layers of Features from Tiny Images*, 2009.
7. Liu, H., Simonyan, K., and Yang, Y., *Darts: Differentiable Architecture Search*, 2018. [arXiv:1806.09055](https://arxiv.org/abs/1806.09055).
8. Luketina, J., Berglund, M., Greff, K., and Raiko, T., Scalable gradient-based tuning of continuous regularization hyperparameters, *CoRR*, 2015, vol. abs/1511.06727. [arXiv:1511.06727](https://arxiv.org/abs/1511.06727).
9. Maclaurin, D., Duvenaud, D., and Adams, R.P., Gradient-based hyperparameter optimization through reversible learning, *CoRR*, 2015, vol. abs/1502.03492. [arXiv:1502.03492](https://arxiv.org/abs/1502.03492).
10. Passalis, N., Tzelepi, M., and Tefas, A., Heterogeneous knowledge distillation using information flow modeling, *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, (2020).

11. Pedregosa, F., Hyperparameter optimization with approximate gradient, *CoRR*, 2016, vol. abs/1602.02355. [arXiv:abs/1602.02355](https://arxiv.org/abs/1602.02355).
12. Rasley, J., Rajbhandari, S., Ruwase, O., and He, Y., Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters, *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery & Data Min.* (2020), pp. 3505–3506.
13. Vatanen, T., Raiko, T., Valpola, H., and LeCun, Y., Pushing stochastic gradient towards second-order methods—backpropagation learning with transformations in nonlinearities, in *Int. Conf. Neural Inf. Process.*, Berlin–Heidelberg: Springer, 2013, pp. 442–449.
14. Xiao, H., Rasul, K., and Vollgraf, R., Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, *CoRR*, 2017, vol. abs/1708.07747. [arXiv:278abs/1708.07747](https://arxiv.org/abs/1708.07747).
15. Computational experiment code.
<https://github.com/Intelligent-Systems-Phystech/MetaOptDistillation>.
Accessed June 14, 2022.

This paper was recommended for publication by A.A. Lazarev, a member of the Editorial Board