



Deep Learning Model Selection With Parametric Complexity Control

Olga Grebenkova^{1,2}^a, Oleg Bakhteev^{1,3}^b and Vadim Strijov³^c

¹*Moscow Institute of Physics and Technology (MIPT), Russia*

²*Skolkovo Institute of Science and Technology (Skoltech), Russia*

³*FRC CSC RAS, Russia*

Keywords: Model Complexity Control, Hypernetworks, Variational Model Optimization, Bayesian Inference.

Abstract: The paper is devoted to deep learning model complexity. It is estimated by Bayesian inference and based on a computational budget. The idea of the proposed method is to represent deep learning model parameters in the form of hypernetwork output. A hypernetwork is a supplementary model which generates parameters of the selected model. This paper considers the minimum description length from a Bayesian point of view. We introduce prior distributions of deep learning model parameters to control the model complexity. The paper analyzes and compares three types of regularization to define the parameter distribution. It infers and generalizes the model evidence as a criterion that depends on the required model complexity. Finally, it analyzes this method in the computational experiments on the Wine, MNIST, and CIFAR-10 datasets.


1 INTRODUCTION


The paper considers the problem of a deep learning model selection. A deep learning model is a superposition of differentiable functions with respect to parameters. In the paper, we study the problem of model selection based on its complexity. We consider the model complexity as a value assigned during model fine-tuning depending on the desired model performance or size. Since the deep learning model selection procedure is computationally expensive (Zhmoginov et al., 2022), we propose to optimize not a distinct model but a family of models at once. We parameterize it by a desired model complexity.


To deal with the problem of model complexity control we propose to represent the parameters of the model in the form of a hypernetwork. A hypernetwork is a function, which generates the parameters of the desired model (Ha et al., 2016). In other words, a hypernetwork is a mapping from a value responsible for the complexity of the desired model to a set of its parameters. Opposite to (Ha et al., 2016), where the hypernetwork was used to simplify the model parameters representation, we consider a hypernetwork as a mapping from the only one value. Another variant

of hypernetworks usage was presented in (Lorraine and Duvenaud, 2018), where the authors investigated hypernetworks' feasibility to predict best model hyperparameters. Opposite to (Zhmoginov et al., 2022) where the complex deep learning model was used as a hypernetwork, we focus on simple hypernetwork models. We concentrate more on their statistical properties than on final performance of the obtained models.

This paper uses the Bayesian approach to model selection. We introduce probabilistic assumptions about the distribution of deep learning model parameters (Graves, 2011; Bakhteev and Strijov, 2018). We propose to generalize the evidence to control the model complexity. To demonstrate that we gather models of different complexity using optimized hypernetworks, we employ the model pruning methods (Graves, 2011; Han et al., 2015). This paper investigates a simple case when the model parameters are assumed to be distributed with a Gaussian distribution (Graves, 2011). In order to evaluate the ability of hypernetwork to generate model parameters we compare two probabilistic loss functions. These functions are optimized using the variational Bayesian approach (Graves, 2011; Bakhteev and Strijov, 2018). We also investigate a deterministic case when the model parameters are optimized straightforwardly with l_2 -regularization. Both of these approaches, probabilistic or deterministic, are success-

^a <https://orcid.org/0000-0002-1169-5405>

^b <https://orcid.org/0000-0002-6497-3667>

^c <https://orcid.org/0000-0002-2194-8859>

fully used for the model compression (Graves, 2011; Han et al., 2015) and are further developed for more sophisticated pruning techniques (Jiang et al., 2019; Louizos et al., 2017). The resulting hypernetworks generate both simple and complex models depending on the required model properties.

The Figure 1 shows an example of the resulting accuracy surface for the models with different complexity. Along one axis we plot the model complexity, along two others the number of deleted model's parameters and accuracy of the model. As we can see models with greater complexity have greater accuracy at the beginning of pruning procedure. But they have significant decrease during it. At the same time models with small complexity are more robust.

Our contributions are:

1. we propose a method of deep learning model optimization with complexity control. Instead of optimizing a model with some predefined hyperparameter value that controls the model complexity, we propose to optimize a family of models. This family is defined using a mapping that generates model parameters based on the desired model complexity;
2. we investigate two forms of model loss functions that are based on the evidence lower bound. We compare them with a simple deterministic model optimization with l_2 regularization and analyze their properties for our optimization method;
3. we give some brief theoretical justification for the proposed method and empirically evaluate its performance for the deep learning model selection.
4. To demonstrate the proposed idea we carry our computational experiments on MNIST (LeCun and Cortes, 2010), Wine (Blake, 1998) and CIFAR-10 (Krizhevsky et al.,) datasets.

2 PROBLEM STATEMENT

Consider the classification problem. In this paper, we research to what extent it is possible to control the model complexity at the inference step. For this reason, we introduce a method of model selection using hypernetworks, a parametric mapping from a complexity value to a set of model parameters. At the training step, we consider complexity value as a random number. During the model's fine tuning, this value can be assigned for the optimal computational budget. Below we introduce the details of the approach.

There is given a dataset: $\mathcal{D} = \{\mathbf{x}_i, y_i\} \quad i = 1, \dots, m$, where $\mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \{1, \dots, Y\}$, Y is a num-

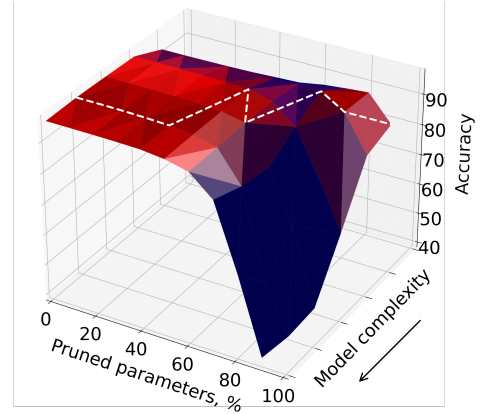


Figure 1: An example of hypernetwork accuracy surface: significant complexity regularization implies models with lower accuracy and higher robustness under pruning. Surface color vary from dark blue to dark red and shows represent the accuracy relative to other models with the same number of model parameters. The colors of the white line marks the most optimal models for different complexity values.

ber of classes. The model is a differential function $\mathbf{f}(\mathbf{x}, \mathbf{w}) : \mathbb{R}^m \times \mathbb{R}^n \rightarrow \mathbb{R}^Y$, where $\mathbf{w} \in \mathbb{R}^n$ is space of the model parameters. Introduce a prior distribution of the parameter vector in \mathbb{R}^n :

$$p(\mathbf{w}|\alpha_{pr}) \sim \mathcal{N}(0, \alpha_{pr}\mathbf{I}), \quad \alpha_{pr} > 0. \quad (1)$$

Although the parameter α_{pr} of the prior distribution can be optimized (Graves, 2011; Bishop, 2006), we suppose that it is fixed during the model optimization (Graves, 2011; Atanov et al., 2019). We use a diagonal matrix $\alpha_{pr}\mathbf{I}$ as the covariance matrix for distributions (Graves, 2011) to simplify the optimization procedure. Then $p(\mathbf{w}|\mathcal{D}, \alpha_{pr}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w}|\alpha_{pr})}{p(\mathcal{D})}$ is the posterior distribution of the parameters vector \mathbf{w} with the given dataset \mathcal{D} and the log-likelihood function $\log p(\mathcal{D}|\mathbf{w}) = \sum_{(\mathbf{x}, y) \in \mathcal{D}} \log p(y|\mathbf{x}, \mathbf{w})$. It depends on the model \mathbf{f} and its parameters \mathbf{w} . To get the posterior distribution $p(\mathbf{w}|\mathcal{D}, \alpha_{pr})$ one must calculate an evidence integral:

$$p(\mathcal{D}|\alpha_{pr}) = \int_{\mathbf{w} \in \mathbb{R}^n} p(\mathcal{D}|\mathbf{w})p(\mathbf{w}|\alpha_{pr})d\mathbf{w}. \quad (2)$$

Since the integral (2) is intractable, we use the variational approach. Suppose that a parametric variational distribution is given: $q(\mathbf{w}|\theta) \sim \mathcal{N}(\mathbf{m}, \mathbf{A}_{ps}^{-1})$, $\mathbf{A}_{ps}^{-1} = \mathbf{diag}(\alpha_{ps})$, where $\theta = (\mathbf{m}, \mathbf{A}_{ps}^{-1})$ are the mean vector and the covariance matrix approximating unknown posterior distribution $p(\mathbf{w}|\mathcal{D}, \alpha_{pr})$. Estimate the logarithm of the integral (2) (Bishop, 2006) :

$$\log p(\mathcal{D}|\alpha_{pr}) \geq -D_{\text{KL}}(q(\mathbf{w}|\theta)||p(\mathbf{w}|\alpha_{pr})) + \mathbb{E}_{q(\mathbf{w}|\theta)} \log p(\mathcal{D}|\mathbf{w}). \quad (3)$$

The first term in (3) is the difference between a posterior and a prior distribution of parameters. It sets the complexity of the parameter distribution based on prior assumptions (1). The Kullback-Leibler divergence determines it. This term controls the divergence between the prior and the variational distribution and thus can be interpreted as a complexity regularization term (Graves, 2011). The second term in formula (3) is the expectation of the likelihood $\log p(\mathcal{D}|\mathbf{w})$.

Define the problem of model parameters optimization by the generalized evidence function \mathcal{L} . It can be defined in different ways. In this paper we compare two variants:

$$\mathcal{L}_1(\lambda) = -\lambda D_{\text{KL}}(q(\mathbf{w}|\theta)||p(\mathbf{w}|\alpha_{pr})) + \mathbb{E}_{q(\mathbf{w}|\theta)} \log p(\mathcal{D}|\mathbf{w}); \quad (4)$$

$$\mathcal{L}_2(\lambda) = -D_{\text{KL}}(q(\mathbf{w}|\theta)||p(\mathbf{w}|\frac{1}{\lambda}\alpha_{pr})) + \mathbb{E}_{q(\mathbf{w}|\theta)} \log p(\mathcal{D}|\mathbf{w}). \quad (5)$$

The first expression (4) controls the prior distribution importance multiplying it by the value λ . This function is generalization of the evidence lower bound, but formally it does not proceed from the evidence expression. Below we prove a statement establishing the connection between this function and the evidence in asymptotics. The second expression (5) controls the importance of the prior multiplying the covariance matrix by $\frac{1}{\lambda}$. The intuition behind this expression is the larger λ the closer \mathbf{w} is to zero, and the more important the regularization is for optimization.

In this paper we compare these loss functions with a simple deterministic loss with l_2 -regularization, see the paper (Han et al., 2015),

$$\mathcal{L}_3(\lambda) = -\lambda \|\mathbf{w}\|^2 + \log p(\mathcal{D}|\mathbf{w}). \quad (6)$$

The following theorem establishes a relation between the expressions presented above.

Theorem 1. *The following relations are true for the presented loss functions (4),(5),(6):*

1. *Let the vector α_{ps} have small enough norm such that we can approximate $q(\mathbf{w}|\theta)$ with the Dirac delta function $\delta(\mu)$. Then for the fixed non-optimized vector α_{ps} the optimization of these expressions is equivalent: $\mathcal{L}_1(\lambda) \approx \mathcal{L}_3(\frac{\lambda}{2\alpha_{pr}})$.*
2. *Let the vector α_{ps} have small enough norm such that we can approximate $q(\mathbf{w}|\theta)$ with the Dirac delta function $\delta(\mu)$. Then for the fixed non-optimized vector α_{ps} the optimization of these expressions is equivalent: $\mathcal{L}_2(\lambda) \approx \mathcal{L}_3(\frac{\lambda}{2\alpha_{pr}})$.*

3. *Let $m = \frac{m_0}{\lambda}, m_0 \in \mathbb{N}, m \gg 0, m_0 \gg 0$. Then the function (4) converges almost surely to the evidence lower bound (3) for the random sample $\hat{\mathcal{D}}, |\hat{\mathcal{D}}| = m_0$ with $m_0 \rightarrow \infty$.*

Proof. Let's prove the first statement. For a small enough norm of the vector α_{ps} we get $q(\mathbf{w}|\theta) \sim \delta(\mu)$, where δ is the Dirac delta function. Then $\mathcal{L}_1(\lambda) \approx -\lambda D_{\text{KL}}(q(\mathbf{w}|\theta)||p(\mathbf{w}|\alpha_{pr})) + \log p(\mathcal{D}|\mu)$.

Then we get the following expression up to a constant: $-\lambda D_{\text{KL}}(q(\mathbf{w}|\theta)||p(\mathbf{w}|\alpha_{pr})) = -\frac{1}{2\alpha_{pr}} \lambda \mu^T \mu + C$, where C is a constant, which does not depend on the optimized parameters μ of the variational distribution q . By leaving only the term related to the gradient we get the expression: $\mathcal{L}_1(\lambda) \approx \log p(\mathcal{D}|\mu) - \frac{\lambda}{2\alpha_{pr}} \|\mu\|^2$, which equals to $\mathcal{L}_3(\frac{\lambda}{2\alpha_{pr}})$. The proof for statement 2 is analogous to the proof above.

For the proof of the third statement, consider the function $\frac{1}{m} \mathcal{L}_1(\lambda)$. Using the Strong Law of large numbers we get:

$$\frac{1}{m} \mathcal{L}_1(\lambda) \xrightarrow{\text{a.s.}} -\frac{\lambda}{m} D_{\text{KL}}(q(\mathbf{w}|\theta)||p(\mathbf{w}|\alpha_{pr})) + \mathbb{E}_{\mathbf{x}, \mathbf{y}} \mathbb{E}_{q(\mathbf{w}|\theta)} \log p(\mathbf{y}|\mathbf{w}, \mathbf{x}), \quad (7)$$

where $\mathbb{E}_{\mathbf{x}, \mathbf{y}}$ is an expectation over objects of the general population corresponding to the dataset \mathcal{D} . Similarly consider the evidence lower bound for the random sample $\hat{\mathcal{D}}, |\hat{\mathcal{D}}| = m_0$, divided by m_0 :

$$-\frac{1}{m_0} D_{\text{KL}}(q(\mathbf{w}|\theta)||p(\mathbf{w}|\alpha_{pr})) + \frac{1}{m_0} \mathbb{E}_{q(\mathbf{w}|\theta)} \log p(\hat{\mathcal{D}}|\mathbf{w}) \xrightarrow{\text{a.s.}} -\frac{1}{m_0} D_{\text{KL}}(q(\mathbf{w}|\theta)||p(\mathbf{w}|\alpha_{pr})) + \mathbb{E}_{\mathbf{x}, \mathbf{y}} \mathbb{E}_{q(\mathbf{w}|\theta)} \log p(\mathbf{y}|\mathbf{w}, \mathbf{x}).$$

The last expression equals to (7) as required to prove. \square

The first and the second statements from the theorem establish a relationship between two loss functions (4),(5) based on probabilistic assumptions and non-probabilistic loss function with l_2 -regularization (6) for the case when the vector α_{ps} corresponding to the variational covariance \mathbf{A}_{ps}^{-1} is sufficiently small. Although these two probabilistic-based expressions are equivalent for this especial case, in general they differ:

$$\mathcal{L}_1(\lambda) - \mathcal{L}_2(\lambda) \propto -(\lambda + 1) \log \det \mathbf{A}_{ps}^{-1}. \quad (8)$$

This difference gives us a different interpretation of λ in these two loss functions: whenever in (4) the value λ monotonically controls the influence of the prior, there is no monotonic dependency between a regularization term D_{KL} and λ in (5). This leads us to different results when varying value λ .

The third statement of the theorem shows that the expression from (4) can be considered as a correct probabilistic approach for variational parameters optimization, where λ controls the dataset size for the evidence lower bound. Both of the expressions (4),(5) can be considered as correct loss functions based on probabilistic assumptions with regularization that controls the importance of prior distribution. However, only the first expression allows us to control the prior importance straightforwardly using the value λ .

Introduce the set of values for the complexity value $\lambda \in \Lambda \subset \mathbb{R}^+$. We want to find a mapping $\mathfrak{G} : \Lambda \rightarrow \mathbb{R}^n$ so that for the arbitrary complexity value $\lambda \in \Lambda$ the model parameters would give the maximum for the following functions:

$$\mathfrak{G}_1(\lambda) = \arg \max_{\theta} (\mathbb{E}_{q(\mathbf{w}|\theta)} \log p(\mathcal{D}|\mathbf{w}) - \lambda D_{KL}(q(\mathbf{w}|\theta) || p(\mathbf{w}|\alpha_{pr}))), (9)$$

$$\mathfrak{G}_2(\lambda) = \arg \max_{\theta} (\mathbb{E}_{q(\mathbf{w}|\theta)} \log p(\mathcal{D}|\mathbf{w}) - D_{KL}(q(\mathbf{w}|\theta) || p(\mathbf{w}|\frac{1}{\lambda}\alpha_{pr}))), (10)$$

$$\mathfrak{G}_3(\lambda) = \arg \max_{\mathbf{w} \in \mathbb{R}^n} (\log p(\mathcal{D}|\mathbf{w}) - \lambda \|\mathbf{w}\|^2). (11)$$

The presented mappings correspond to the optimized functions (4),(5),(6).

3 HYPERNETWORKS FOR THE MODEL COMPLEXITY CONTROL

Solving the optimization problem (9) for an arbitrary value $\lambda \in \Lambda$ is a computationally challenging task. We propose to use a hypernetwork to solve it. This allows us to control the model complexity not during the training step but at the inference step or fine-tune the model for the desired complexity in one-shot manner.

Introduce the set of parameters Λ , which control the complexity of the model. Hypernetwork is a parametric mapping from the set Λ to the set of model parameters: $\mathbf{G} : \Lambda \times \mathbb{R}^u \rightarrow \mathbb{R}^n$, where \mathbb{R}^u is the set of valid hypernetwork parameters. In our work we use the following linear mapping:

$$\mathbf{G}_{\text{linear}}(\lambda) = \lambda \mathbf{b}_1 + \mathbf{b}_2, (12)$$

where $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^n$ are the vectors, which are do not depend on λ .

A natural extension of such linear mapping is the piece-wise linear one:

$$\mathbf{G}_{\text{piecewise}}(\lambda) = \sum_{i=0}^{N-1} \mathbf{F}(t_i, t_{i+1}, \lambda), (13)$$

$$\mathbf{F}(t_i, t_{i+1}, \lambda) = \begin{cases} \mathbf{b}(t_i) + \frac{\mathbf{b}(t_{i+1}) - \mathbf{b}(t_i)}{t_{i+1} - t_i} (\lambda - t_i), & t_i \leq \lambda \leq t_{i+1}, \\ 0, & \text{otherwise,} \end{cases}$$

where $\mathbf{b} \in \mathbb{R}^n : [0, 1] \rightarrow \mathbb{R}^n$; $t_i \in [0, 1]$, N is a number of regions, where this function is linear.

Algorithm 1: The algorithm of the hypernetwork training.

Require: hypernetwork \mathbf{G} , desired model \mathbf{f} , loss function \mathcal{L} , training dataset \mathcal{D}

- 1: **for** every batch $\hat{\mathcal{D}}$ of the dataset \mathcal{D} **do**
 - 2: sample $\log \lambda_{\text{sample}} \sim P(\lambda)$
 - 3: obtain \mathbf{w} from $\mathbf{G}(\lambda_{\text{sample}})$
 - 4: compute $\mathcal{L}(\mathbf{f}(\mathbf{w}, \hat{\mathcal{D}}))$
 - 5: backpropagate and update hypernetwork \mathbf{G}
 - 6: **end for**
 - 7: **return** trained hypernetwork \mathbf{G}
-

Algorithm 2: The algorithm of the hypernetwork inference.

Require: trained hypernetwork \mathbf{G} , desired model \mathbf{f} , testing dataset \mathcal{D} , desired complexity λ_{desired} , criteria for removing of the parameters \mathbf{g}

- 1: obtain \mathbf{w} from $\mathbf{G}(\lambda_{\text{desired}})$
 - 2: compute accuracy for model $\mathbf{f}(\mathbf{w}, \mathcal{D})$
 - 3: use criteria \mathbf{g} to find the most uninformative parameters $\hat{\mathbf{w}}$
 - 4: update weights $\mathbf{w} = \mathbf{w} \setminus \hat{\mathbf{w}}$
 - 5: **return** accuracy for different percent of deleted parameters
-

To approximate the optimization problems (9), (10), (11) we propose to optimize the parameters $\mathbf{U} \in \mathbb{R}^u$ of the hypernetwork \mathbf{G} by randomly generated values of the complexity value $\lambda \in \Lambda$:

$$\mathbb{E}_{\lambda \sim P(\lambda)} (\log p(\mathcal{D}|\mathbf{w}) - \lambda D_{KL}(q(\mathbf{w}|\theta) || p(\mathbf{w}|\alpha_{pr}))) \rightarrow \max_{\mathbf{U} \in \mathbb{R}^u} (14)$$

$$\mathbb{E}_{\lambda \sim P(\lambda)} (\log p(\mathcal{D}|\mathbf{w}) - D_{KL}(q(\mathbf{w}|\theta) || p(\mathbf{w}|\frac{1}{\lambda}\alpha_{pr}))) \rightarrow \max_{\mathbf{U} \in \mathbb{R}^u} (15)$$

$$\mathbb{E}_{\lambda \sim P(\lambda)} (\log p(\mathcal{D}|\mathbf{w}) - \lambda \|\mathbf{w}\|^2) \rightarrow \max_{\mathbf{U} \in \mathbb{R}^u}, (16)$$

where $P(\lambda)$ is prior distribution on the set Λ . In this paper we use log-uniform distribution as the prior distribution: $\log \lambda \sim \mathcal{U}[L, R]$, where values L, R are given in the experiments section. This allows us to significantly vary the desired model complexity during training. Note, that in this paper we consider λ only as a value that should be tuned at the inference step and not expected to be inferred in strictly Bayesian way. The algorithm of the hypernetwork training is shown in Algorithm 1. The scheme of training procedure is presented in Fig. 2. After the inference step for a single hypernetwork \mathbf{G} we can obtain parameters for models of different complexity, which already have high accuracy results without fine-tuning.

(16) can be considered as an analogue of objective function from (Lorraine and Duvenaud, 2018). We will treat it as a baseline model.

3.1 Model Pruning

As it was mentioned before, deep learning models have an excessive number of parameters. So one of the ways to compare models, obtained by different approaches, is to prune them and look at their performance at the same pruning level. Therefore the parameters of each model are pruned after optimization using the approach described in (Graves, 2011). The algorithm of hypernetwork inference is presented in Algorithm 2. As the criterion for removing the parameters we use the relative density of the model (Graves, 2011):

$$\mathbf{g}_{\text{var}}(w_i) \propto \exp\left(-\frac{\mu_i^2}{2\sigma_i^2}\right), \quad (17)$$

where μ_i, σ_i are the i -th components of the mean vector \mathbf{m} and the covariance matrix $\mathbf{A}_{\text{ps}}^{-1}$ of learned variational distribution. We also consider simplified criterion, which can be applied without probabilistic assumptions (Han et al., 2015):

$$\mathbf{g}_{\text{simple}}(w_i) \propto \exp(-w_i^2). \quad (18)$$

The proposed method is based on the assumption that hypernetwork \mathbf{G} approximates the models optimized with different values for the complexity value λ not only in terms of performance but also in statistical properties. This allows us to tune and prune model's parameters derived from hypernetwork similarly to usual model's parameters. The following theorem confirms this assumption for the simple case of a compact domain containing the minimum of the model for all complexity values.

Theorem 2. *Let the following conditions be satisfied:*

1. *there is a given model $\mathbf{f}(\mathbf{w})$ and an continuous loss function \mathcal{L} ;*

2. *there is a compact region $\mathbb{U} \in \mathbb{R}^n$ that contains only one minimum $\mathcal{L}(\mathbf{w}^*(\lambda)) \in \mathbb{U}$, $\mathcal{L}(\mathbf{w}^*(\lambda)) < \infty$ for every $\lambda \in \Lambda$;*
3. *there is a sequence of model parameters $\mathcal{L}(\mathbf{w}_n(\lambda)) \neq \mathcal{L}(\mathbf{w}^*(\lambda)) \quad \forall n \quad \mathcal{L}(\mathbf{w}_n(\lambda)) \in \mathbb{U}$ such that $\mathbb{E}_{\lambda \sim P(\lambda)} \mathcal{L}(\mathbf{w}_n(\lambda)) \xrightarrow{n \rightarrow \infty} \max$.*

Then the continuous function for the sequence of model parameters converge in distribution to the value for minimum $\mathbf{g}(\mathbf{w}_n(\lambda)) \xrightarrow{P} \mathbf{g}(\mathbf{w}^(\lambda))$. Under \mathbf{g} we can consider the criterion for removing the parameters.*

Proof. By definition $\mathbf{w}^*(\lambda)$ gives the maximum for loss function \mathcal{L} . So due to the third condition $\mathbb{E}_{\lambda \sim P(\lambda)} \mathcal{L}(\mathbf{w}_n(\lambda)) \xrightarrow{n \rightarrow \infty} \mathbb{E}_{\lambda \sim P(\lambda)} \mathcal{L}(\mathbf{w}^*(\lambda))$. Then from linearity of expected value $\mathbb{E}_{\lambda \sim P(\lambda)} |\mathcal{L}(\mathbf{w}^*(\lambda)) - \mathcal{L}(\mathbf{w}_n(\lambda))| \xrightarrow{n \rightarrow \infty} 0$, which means that the value of optimisation function for sequence of model parameters converge in mean to the value for minimum

$\mathcal{L}(\mathbf{w}_n(\lambda)) \xrightarrow{L^1} \mathcal{L}(\mathbf{w}^*(\lambda))$. We can show that the argument \mathbf{w}_n of the function \mathcal{L} converges to \mathbf{w}^* in mean. Suppose that this fact is not true, then $\exists \varepsilon > 0 : \forall i \quad \exists j > i : \mathbb{E}_{\lambda \sim P(\lambda)} |\mathbf{w}_j(\lambda) - \mathbf{w}^*(\lambda)| > \varepsilon$.

Let δ be the maximum value of the function \mathcal{L} for $\mathbf{w}_j(\lambda)$ from the region \mathbb{U} so that $\mathbb{E}_{\lambda \sim P(\lambda)} |\mathbf{w}_j(\lambda) - \mathbf{w}^*(\lambda)| > \varepsilon$. Note that $\delta < \mathcal{L}(\mathbf{w}^*(\lambda))$. Then there exists an infinite subsequence of parameters that $\mathcal{L}(\mathbf{w}_j(\lambda)) \leq \delta < \mathcal{L}(\mathbf{w}^*(\lambda))$. Since $\mathbb{E}_{\lambda \sim P(\lambda)} |\mathcal{L}(\mathbf{w}^*(\lambda)) - \mathcal{L}(\mathbf{w}_n(\lambda))| \xrightarrow{n \rightarrow \infty} 0$, we got a contradiction. So $\mathbf{w}_n(\lambda) \xrightarrow{L^1} \mathbf{w}^*(\lambda)$, and correspondingly $\mathbf{w}_n(\lambda) \xrightarrow{P} \mathbf{w}^*(\lambda)$.

We use a continuous mapping theorem, that states that if a function $\mathbf{g} : S \rightarrow S'$ has the set of discontinuity points $D_{\mathbf{g}}$ such that $P[X \in D_{\mathbf{g}}] = 0$, then: $X_n \xrightarrow{d} X \Rightarrow \mathbf{g}(X_n) \xrightarrow{d} \mathbf{g}(X)$. Using the fact that $\mathbf{g}(\mathbf{w}_n)$ satisfies all conditions of this theorem we get $\mathbf{g}(\mathbf{w}_n(\lambda)) \xrightarrow{P} \mathbf{g}(\mathbf{w}^*(\lambda))$. \square

4 EXPERIMENTS

To analyze the properties of the optimization problems (14), (15), (16) and the proposed methods for implementation of a hypernetwork (12), (13), we carried out a toy experiment on Wine dataset (Blake, 1998) and experiments on the MNIST dataset of handwritten digits (LeCun and Cortes, 2010) and the CIFAR-10 dataset (Krizhevsky et al.,) of tiny images.¹

¹The source code is available at <https://github.com/intsystems/VarHyperNet>

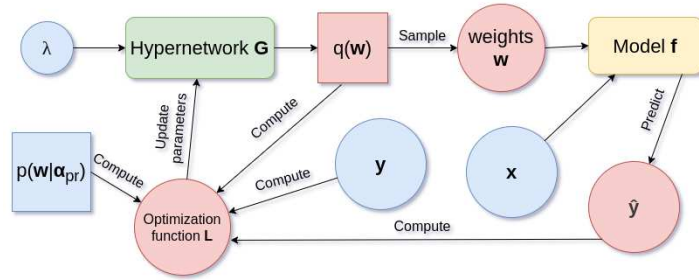


Figure 2: The diagram of the hypernetwork training. All the given variables are marked in blue. All the variables to optimize are marked in red.

For all the experiments we considered model classification accuracy as a quality criterion. We used ADAM optimizer with a learning rate $5 \cdot 10^{-4}$. As logarithm of the variance for the variational distribution at initialization α_{ps} we used -3.0 ; as the prior variance we use $\alpha_{pr} = 1.0$. For each of the models we carried out 5 runs, the results were averaged.

4.1 Preserving of Statistical Properties

For the first experiment, we used the Wine dataset, consisting of 178 objects categorized into 3 classes. Our main goal of this experiment was to demonstrate that the hypernetworks can preserve the statistical properties of the approximated model. For this experiment, we split the dataset into 142 objects for the train and 36 objects for the test. We used variational linear model (9) as a basic classification model optimized directly without hypernetwork. We used two types of hypernetworks to approximate this model: variational linear hypernetwork (12) and variational piecewise-linear hypernetwork (13) with $N=5$ piecewise-linear regions.

We used optimization with minibatch size set to 1. We trained every model for 200 epochs. We used $\Lambda = \in [10^2; 10^6]$. This set was designed to consider models with different performances: from slightly regularized models with accuracy $\approx 95\%$ to overregularized models with accuracy $\approx 53\%$.

Since our goal was not to obtain the highest accuracy using hypernetwork, but to obtain performance and parameter distribution similar to the linear model, we tracked the difference in the accuracy between the hypernetwork and directly optimized model and the difference between their distribution. For this difference we used the symmetrized KL-divergence: $\bar{D}_{KL}(q_1, q_2) = D_{KL}(q_1, q_2) + D_{KL}(q_2, q_1)$, where q_1 is a variational distribution from hypernetwork, q_2 is a variational distribution from the directly optimized model. After the hypernetwork training, we also fine-tuned the obtained models for one epoch with fixed λ . We hypothesize that if the hypernetwork approx-

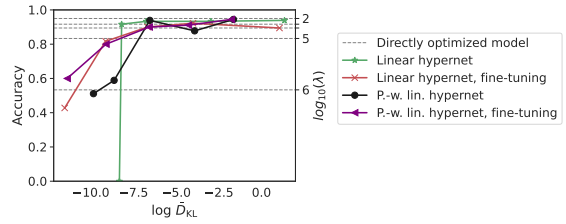


Figure 3: The results for the toy dataset for the directly optimized model (9), linear hypernetwork (12) and piecewise-linear hypernetwork (13) (P-w. lin. hypernet). Each line corresponds to the models performance obtained from hypernetwork for different $\lambda \in \{10^2, 10^3, 10^4, 10^5, 10^6\}$.

imates the statistical properties of the directly optimized model well, after fine-tuning it will get accuracy closer to the directly optimized model, and its \bar{D}_{KL} will also decrease.

The results are shown in Figure 3. The gray lines correspond to the accuracy values for different $\lambda \in \{10^2, 10^3, 10^4, 10^5, 10^6\}$ obtained by directly optimized models. The x-axis corresponds to the logarithm of \bar{D}_{KL} , therefore the perfect approximation of the directly optimized model should be represented by a line with points corresponding to the gray lines on the y-axis and very low values on the x-axis. As we can see, the linear hypernetwork poorly approximates the directly optimized model in comparison to a more complex piecewise-linear hypernetwork that illustrates the result of Theorem 2: the better model can approximate the directly optimized model in terms of optimization, the better it preserves its statistical properties. After fine-tuning the piecewise-linear hypernetwork also improved its performance everywhere except $\lambda = 10^6$, where it got a better \bar{D}_{KL} result, but worse accuracy. Note that training the model (9) from scratch with only one epoch gave us accuracy from 0 to 61% for different λ . This shows that the hypernetworks really contained parameter distribution close to the parameter distribution of the directly optimized model, the fine-tuning step only increased its performance, but not fully retrained the distribution parameters. It also gives us a real scenario for the hypernetwork usage: to store one set of parameters for

models with different complexity to tune them to the desired complexity on demand.

4.2 MNIST and CIFAR-10: Experimental Settings

The main goals of these experiments is to demonstrate the availability of the hypernetworks to generate the deep learning model parameters with the condition on the complexity value λ . As we obtain the parameters for the desired model we prune it to check how many informative ones have each of the models depending on the complexity value λ . This experiment allows us to compare properties of models which parameters were obtained from hypernetwork with properties of directly optimized ones.

For both the experiment we trained our models for 50 epochs. The minibatch size is set to 256. The following implementations were compared:

- (a) variational neural network (9);
- (b) network with covariance reparametrization (10);
- (c) base network (11);
- (d) variational linear hypernetwork (12);
- (e) network with covariance; reparametrization (10) with linear hypernetwork (12);
- (f) base network (11) (Lorraine and Duvenaud, 2018) with linear hypernetwork (12);
- (g) variational piecewise-linear hypernetwork (13), $N = 5$;
- (h) network with covariance reparametrization (10) with piecewise-linear hypernetwork (13), $N = 5$;
- (i) base network (11) (Lorraine and Duvenaud, 2018) with piecewise-linear hypernetwork (13), $N = 5$.

We launched the neural network training for different values of the complexity value $\lambda \in \Lambda$. The parameters of each model were pruned after the optimization using the \mathbf{g}_{var} criterion (17). For the implementations (c), (f), (i) we used the simplified criterion $\mathbf{g}_{\text{simple}}$ (18).

4.3 MNIST Experiment Results

For the MNIST dataset we used a neural network consisting of two layers with 50 and 10 neurons, where the second layer contains the softmax function. Parameters L, R for uniform distribution were set to -3 and 3 correspondingly.

Fig. 4a shows how the accuracy changes when parameters were pruned for variational neural network (9). The graph shows that the variational

method allows to remove $\approx 60\%$ parameters for $\lambda \in \{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1\}$ and $\approx 80\%$ parameters for $\lambda = 10^2$ without significant loss of classification accuracy. If we delete more parameters, the accuracy for all values decreases. For large values of $\lambda > 10^2$ we obtain an oversimplified model. It contains a small number of informative parameters. Thus, removing of them for a given value of λ has little effect on the classification accuracy. However, the initial accuracy is low.

Fig. 4d shows how the classification accuracy changes for the model with covariance reparametrization (10). Fig. 4g shows how the classification accuracy changes for the base network (11). The classification accuracy of these two models hardly changed, but the networks with the variational approach were more robust to parameter deletion.

Fig. 4b, e, h shows how the classification accuracy changes when parameters are removed by the specified method for models with the linear hypernetworks. As can be seen from the graph, the average classification accuracy for all values of $\lambda \in \Lambda$, increased. The deviation from the mean also increased for the big percents of deleted parameters. At the same time, for all values of $\lambda \in \Lambda$, a more stable models were obtained: the classification accuracy less depends on the removal of parameters.

Fig. 4c, f, i shows how the classification accuracy changes when parameters were removed by the specified method for a model with the piecewise-linear approximation. Models with the piecewise-linear hypernetwork showed similar behaviour to models that were trained directly during pruning. Moreover, for all values of $\lambda \in \Lambda$, a more stable models were obtained. All results are presented in the Table 1 and on Fig. 5, where results for all λ were averaged.

4.4 CIFAR-10 Experiment Results

For the CIFAR-10 dataset, we used CNN-based architecture with convolutional layers of size (3,48), (48, 96), (96, 192), (192, 256), ReLU activation, and feed-forward layer in the end. Parameters L, R for uniform distribution were set to -2 and 0 correspondingly.

It can be seen from the Fig. 6a that the variational method also allowed removing $\approx 60\%$ parameters for $\lambda = 0.01, 0.1$, in contrast to the base model Fig. 6d, where the classification accuracy dropped significantly when 40 percent of the parameters were removed.

The network with covariance reparametrization (10) showed poor results for CIFAR-10. They are presented on the Fig. 8. The poor results can be mainly explained by the speciality of (5) for the mod-

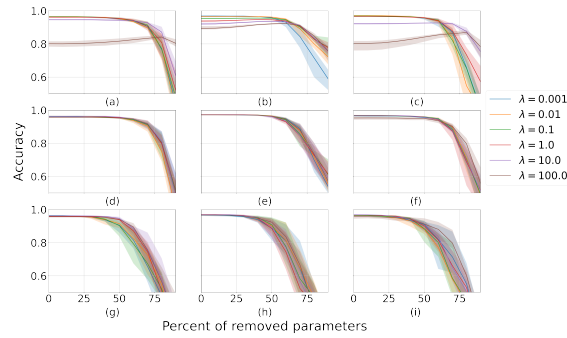


Figure 4: The dependence graph of the classification accuracy on the percentage of removed parameters on MNIST dataset for: (a) variational neural network (9), (b) variational linear hypernetwork (12), (c) variational piecewise-linear hypernetwork (13); (d) network with covariance reparametrization (10), (e) network with covariance reparametrization (10) with linear hypernetwork (12), (f) network with covariance reparametrization (10) with piecewise-linear hypernetwork (13); (g) base network (11), (h) base network (11) with linear hypernetwork (12), (i) base network (11) with piecewise-linear hypernetwork (13).

Table 1: Accuracy after pruning for MNIST dataset.

Implementation/ Percent of deleted parameters	0%	10%	30%	50%	70%	90%
Variational network	0.9676	0.9678	0.9661	0.9602	0.9350	0.8280
Network with covariance reparametrization	0.9667	0.9668	0.9665	0.9605	0.9388	0.6208
Base net	0.9662	0.9659	0.9630	0.9563	0.8613	0.4917
Variational linear hypernetwork	0.9703	0.9700	0.9699	0.9652	0.9182	0.8393
Network with covariance reparametrization with linear hypernetwork	0.9752	0.9749	0.9743	0.9698	0.9198	0.7039
Base network with linear hypernetwork	0.9723	0.9719	0.9687	0.9527	0.8119	0.3470
Variational piecewise-linear hypernetwork	0.9736	0.9733	0.9712	0.9621	0.9280	0.8229
Network with covariance reparametrization with piecewise-linear hypernetwork	0.9706	0.9707	0.9701	0.9630	0.9186	0.6545
Base network with piecewise-linear hypernetwork	0.9710	0.9699	0.9656	0.9474	0.8774	0.3807

els with a large number of parameters, which is also confirmed by (8). We see that while the λ parameter monotonously controls the influence of the prior distribution $p(\mathbf{w}|\alpha_{pr})$ in (4), there is no such monotonicity for (5), therefore the calibration of the parameter for the such a model is a more difficult task and the scale for the λ parameter can drastically differ for (5) and (4),(6).

Fig. 6b, e shows graphs for variational (9) and base (11) models with a linear hypernetwork (12). As we can see, the classification accuracy improved

for all $\lambda \in \Lambda$ and the model's robustness to parameter deletion increased.

The same results(Fig. 6c,f) were reached with piece-wise implementation of hypernetwork (13). In addition, the piece-wise hypernetwork better approximated the behaviour of directly trained models.

All the results for CIFAR-10 dataset are presented in Table 2 and Fig. 7. The experiments show that the variational (4) and the base (6) loss functions give great and interpreted results. Despite the good result on MNIST dataset, function with covariance

Table 2: Accuracy after pruning for CIFAR-10 dataset.

Implementation/ Percent of deleted parameters	0%	10%	30%	50%	70%	90%
Variational network	0.8612	0.8614	0.8615	0.8508	0.8048	0.4577
Base net	0.8852	0.8839	0.8728	0.8191	0.5683	0.1582
Variational linear hypernetwork	0.8719	0.8719	0.8691	0.8520	0.8189	0.6107
Base network with linear hypernetwork	0.8984	0.8984	0.8919	0.8683	0.7565	0.1656
Variational piecewise-linear hypernetwork	0.8720	0.8715	0.8703	0.8561	0.8207	0.5173
Base network with piecewise-linear hypernetwork	0.8879	0.8868	0.8752	0.8321	0.5146	0.1354

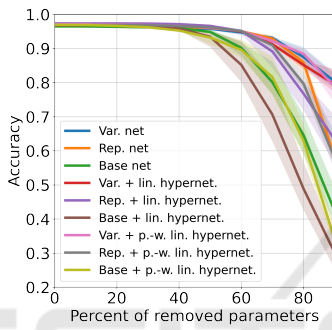


Figure 5: The dependence graph of the classification accuracy on the percentage of removed parameters for all models on MNIST dataset.

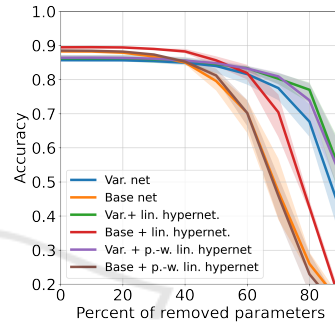


Figure 7: The dependence graph of the classification accuracy on the percentage of removed parameters for all models on CIFAR-10 dataset.

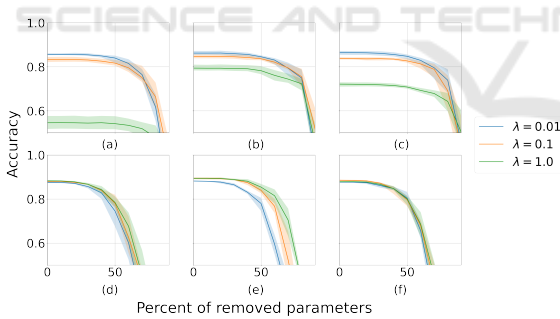


Figure 6: The dependence graph of the classification accuracy on the percentage of removed parameters on CIFAR-10 dataset for: (a) variational neural network (9), (b) variational linear hypernetwork (12), (c) variational piecewise-linear hypernetwork (13); (d) base network (11), (e) base network (11) with linear hypernetwork (12), (f) base network (11) with piecewise-linear hypernetwork (13).

reparametrization (5) requires more accurate tuning for different models and data, that is why it is not suitable in many cases. In addition, experiments show that we can obtain a hypernetwork that precisely approximates original network. This result supports the Theorem 2.

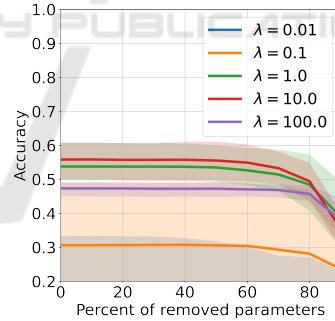


Figure 8: The dependence graph of the classification accuracy on the percentage of removed parameters for network with covariance reparametrization (10) on CIFAR-10 dataset.

5 CONCLUSION

This paper investigated the problem of deep learning model complexity control at the inference. To control the model complexity, we introduced probabilistic assumptions about the distribution of parameters of the deep learning model. The paper analyzed three forms of regularization to control the model parameter dis-

tribution. It generalized the model evidence as a criterion that depends on the required model complexity. The proposed method was based on the representation of deep learning model parameters in the form of hypernetwork output. We analyzed this method in the computational experiments on the Wine, MNIST and CIFAR-10 datasets. The results showed that models with hypernetworks have the same properties as models trained directly but use less computational resources. Furthermore, these models are more stable in terms of deleting parameters and can be easily adjust to computational restrictions. In future, we are going to research other variants of hypernetwork implementation and advanced methods of controlling model's complexity. Besides, it is still a question how to choose the complexity parameter λ for new dataset. We plan to investigate it in future research.

REFERENCES

- Atanov, A., Ashukha, A., Struminsky, K., Vetrov, D., and Welling, M. (2019). The deep weight prior. In *International Conference on Learning Representations*.
- Bakhteev, O. and Strijov, V. (2018). Deep learning model selection of suboptimal complexity. *Automation and Remote Control*, 79:1474–1488.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Blake, C. (1998). Uci repository of machine learning databases. <http://www.ics.uci.edu/mllearn/MLRepository.html>.
- Graves, A. (2011). Practical variational inference for neural networks. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F. C. N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 2348–2356.
- Ha, D., Dai, A. M., and Le, Q. V. (2016). Hypernetworks. *CoRR*, abs/1609.09106.
- Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both weights and connections for efficient neural network. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.
- Jiang, T., Yang, X., Shi, Y., and Wang, H. (2019). Layer-wise deep neural network pruning via iteratively reweighted optimization. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5606–5610.
- Krizhevsky, A., Nair, V., and Hinton, G. (-). Cifar-10 (canadian institute for advanced research).
- LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database. <http://yann.lecun.com/exdb/mnist/>.
- Lorraine, J. and Duvenaud, D. (2018). Stochastic hyperparameter optimization through hypernetworks. *CoRR*, abs/1802.09419.
- Louizos, C., Ullrich, K., and Welling, M. (2017). Bayesian compression for deep learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Zhmoginov, A., Sandler, M., and Vladymyrov, M. (2022). Hypertransformer: Model generation for supervised and semi-supervised few-shot learning. *arXiv preprint arXiv:2201.04182*.