Deep Neural Network Compression with Knowledge Distillation Using Cross-Layer Matrix, KL Divergence and Offline Ensemble

Hsing-Hung Chou* and Ching-Te Chiu*† and Yi-Ping Liao†
* Institute of Communications Engineering, National Tsing Hua University, Hsinchu, Taiwan
† Institute of Computer Science, National Tsing Hua University, Hsinchu, Taiwan

***Abstract*— Knowledge Distillation is one approach in Deep Neural Networks (DNN) to compress huge parameters and high level of computation associated with a teacher model to a smaller student model. Therefore, the smaller model can be deployed in embedded systems. Most of Knowledge Distillations transfer information at the last stage of the DNN model. We propose an efficient compression method that can be split into three parts. First, we propose a cross-layer Gramian matrix to extract more features from the teacher's model. Second, we adopt Kullback Leibler (KL) Divergence in an offline deep mutual learning (DML) environment to make the student model find a wider robust minimum. Finally, we propose the use of offline ensemble pre-trained teachers to teach a student model. With ResNet-32 as the teacher's model and ResNet-8 as the student's model, experimental results showed that Top-1 accuracy increased by 4.38% with a 6.11x compression rate and 5.27x computation rate.**

***Index Terms*—Deep Convolutional Model Compression, Knowledge Distillation, Transfer Learning**

I. Introduction

Recently, deep learning is becoming a mainstream technology owing to the availability of high computation GPGPU and the ability to process massive data. Many state-of-the-art performances have been achieved with deep learning computer vision workloads like image classification[1], object detection[2], and semantic segmentation[3]. However, when neural networks get deeper and wider, the computation of deep neural network (DNN) models become more expensive. There are five approaches to achieve a compact yet accurate model: frugal architecture, pruning, matrix decomposition, quantization, and specialist knowledge distillation (KD). Knowledge distillation is a concept that a large DNN trained for a given task teaches shallower student neural network (S-DNN) on the same task.

We combine three ways of deep neural network compression with knowledge distillation. The first method is Gramian matrix. Gatys et al. [4] represent texture information of the input image with Gramian matrix since Gramian matrix is composed by computing the inner product of features vectors. We use Gramian matrix crossing one to three layers and combine them into one. Second method is KL divergence. We approach the probability distribution of the T-DNN by training S-DNN. Lastly, there are originally two branches of Knowledge distillation approaches. One is the conventional approach, referred to as offline methods[5][6][7][8][9][10], which training the teacher neural network (T-DNN) first, then use the S-DNN to mimic the pre-trained T-DNN. Although this two-phase approach incurs considerable computation time, we get a better performing S-DNN. Sometimes, this S-DNN is better performing than the pre-trained T-DNN, because the T-DNN is already pre-trained and has more layers than the S-DNN, while the S-DNN has a better initial weight. The other approach is referred to as online methods [11][12][13], which start both as scratch models that train together. This one-phase approach expects to train a better model than when only training the S-DNN model. Compared with the offline methods, these online methods do not need to train a T-DNN first, so there is less training time. We adopt the offline approach as our last method.

II. Proposed Architecture

The core idea of knowledge distillation is how to define the vital information, then transfer the knowledge from the T-DNN to the S-DNN. The overall architecture for our three proposed compression methods are shown in Fig. 1. First, we propose cross-layer Gramian matrix to extract more features by the flow of solution procedure (FSP) method in the yellow part. Second, we adopt the KL Divergence in the offline environment to make S-DNN find a wider robust minimum in the brown part. Finally, we propose the use of offline ensemble pre-trained T-DNN to teach a S-DNN by using stochastic mean in the red part. The details are shown as follows.

A. Cross-Layer Matrix

1) Proposed Distilled Knowledge:

Yim et al [5] proposed "FSP" by using Gramian matrix to mimic the generated features of the T-DNN, which can be a hard constraint for the S-DNN. We propose to let teacher and student produce Gramian matrix respectively. Then use KD Loss to compute their losses in order to decrease their losses' difference. However, the features crossing different layers are not utilized, so we generate more Gramian matrices by crossing more than one layer. The number of cross matrices

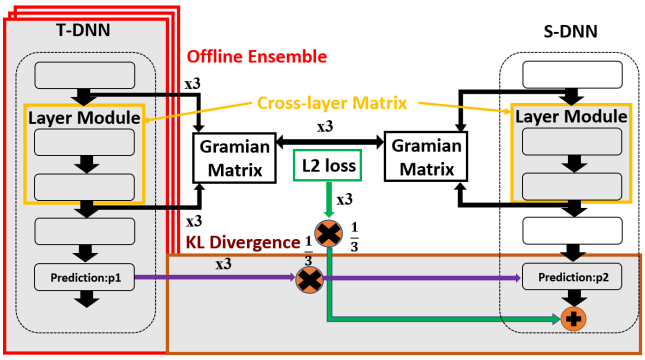

Figure 1: Overall architecture of our proposed methods.

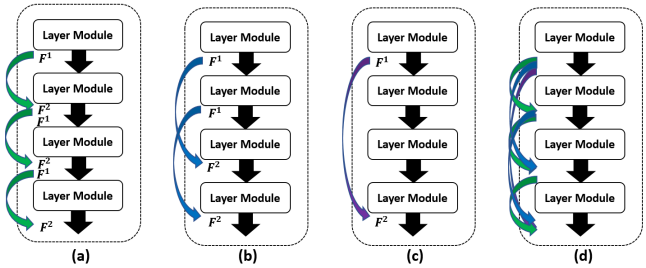

Figure 2: (a) Cross one layer. (b) Cross two layers. (c) Cross three layers. (d) Our proposed.

we add as loss function depend on how many layer modules are in DNN model. With more Gramian matrices in loss function, it would make the S-DNN get better performance. Fig. 2(a)(b)(c) show the different crossing schemes of passing features from different layer modules. Our proposed method is to combine the Gramian matrices crossing one layer, two layers, and three layers as shown in Fig. 2(d) as knowledge.

Based on FSP [5], the Gramian matrix can be defined by two output feature maps. We propose the Gramian matrix as the knowledge to transfer. The Gramian matrix $G \in \mathbb{R}^{m \times n}$ is generated by the features from two layers. One output feature map is defined as $F^1 \in \mathbb{R}^{h \times w \times m}$, where h, w, represents the height and width of output feature maps and m represents the number of output channels. The other output feature map is defined as $F^2 \in \mathbb{R}^{h \times w \times n}$. Then, the Gramian matrix $G \in \mathbb{R}^{m \times n}$ is calculated by (1)

$$G_{i,j}(x;W) = \sum_{s=1}^{h}\sum_{t=1}^{w} \frac{F^1_{s,t,i}(x;W) \times F^2_{s,t,j}(x;W)}{h \times w} \quad (1)$$

where s, t represent the index of the height and width of output feature maps, i, j represent the points of cross-one-layer results, x represent the input image and W the weights of the network model. Unlike FSP [5], we select several points not only from cross-one module layer but also from cross-more-than-one module layer to generate more Gramian matrices as shown (2) and (3).

$$G_{i,q}(x;W) = \sum_{s=1}^{h}\sum_{t=1}^{w} \frac{F^1_{s,t,i}(x;W) \times F^2_{s,t,q}(x;W)}{h \times w} \quad (2)$$

$$G_{i,r}(x;W) = \sum_{s=1}^{h}\sum_{t=1}^{w} \frac{F^1_{s,t,i}(x;W) \times F^2_{s,t,r}(x;W)}{h \times w} \quad (3)$$

where i, q represent the points of cross-two-layer results as shown in Fig. 2(b) and i, r represent the points of cross-three-layer results as shown in Fig. 2(c).

2) KD Loss for The Gramian Matrix:

As discussed previously, the T-DNN will teach S-DNN the solution of question by using the Gramian matrix. We assume that there are B Gramian teacher matrices G^T_b, $b = 1, ..., B$, which are generated by the T-DNN, and B Gramian student matrices G^S_b, $b = 1, ..., B$, which are generated by the S-DNN. The B Gramian matrices includes cross-one-layer, cross-two-layer and cross-three-layer Gramian matrices defined in Eq. (1)(2)(3). Next, each pair of Gramian matrices will be calculated as the cost function by using the squared L2-norm. The cost function of knowledge distillation $L_{KD}(W_t; W_s)$ is defined as (4):

$$L_{KD}(W_t; W_s) = \frac{1}{B}\sum_{x}\sum_{b=1}^{B} \lambda_b \times ||G^T_b(x;W_t) - G^S_b(x;W_s)||^2_2 \quad (4)$$

where λ_b represents the weight for each KD loss and B represents the numbers of Gramian matrices. Because our proposed method adds more Gramian matrices by creating the cross matrices, we initially set all KD losses with the same weight, and see if there is any other knowledge we could extract from more than one layer. As a result, the values of λ_b are identical in our experiments.

B. KL Divergence

We propose using Kullback Leibler (KL) Divergence, which was used in DML [11], as our second-order loss function. In contrast to the online method[11] with two-direction learning, our offline method is only used in one direction from T-DNN to S-DNN. Given D as the data examples $X = \{x_n\}^D_{n=1}$ from C classes, we represent the corresponding label set as $Y = \{y_n\}^C_{n=1}$. The probability of class C for data example x_n is given by a neural network θ_1 and computed as

$$p^C_1(x_n) = \frac{exp(z^C_1)}{\sum^C_{c=1} exp(z^c_1)} \quad (5)$$

where $p^C_1(x_n)$ represents the probability distribution of θ_1 and the logit z^C_1 is the output of the "softmax" layer in θ_1. As a result, the formulation of KL Divergence can be computed as

$$L_{KL}(p_T||p_S) = \sum_{d=1}^{D}\sum_{c=1}^{C} p^c_T(x_d) log \frac{p^c_T(x_d)}{p^c_S(x_d)} \quad (6)$$

where p_T and p_S are the probability distribution of teacher and student model respectively. We believe that student model can get full knowledge from teacher model by having distribution similar to teacher's .

C. Offline Ensemble

The original method of FSP [5] is discussed with one T-DNN to transfer one S-DNN. Compared with FSP [5], we propose using offline ensemble pre-trained teachers to generate the stochastic mean and improve the image classification result. The cost functions of knowledge distillation and KL Divergence are defined as

$$L_{KD}^{Ensemble} = \frac{1}{K}\sum_{k=1}^{K} L_{KD,k} \tag{7}$$

$$L_{KL}^{Ensemble} = \frac{1}{K}\sum_{k=1}^{K} L_{KL}(p_k||p_S) \tag{8}$$

where $L_{KD}^{Ensemble}$ represents the loss function of offline ensemble knowledge distillation, $L_{KL}^{Ensemble}$ represents the loss function of offline ensemble KL Divergence, K represents the numbers of pre-trained teacher models (K=3). We believe that the offline ensemble pre-trained teacher models with the same architecture, but the different weights will transfer knowledge to student model by using the stochastic mean.

D. Overall Loss Function

We had already proposed L_{KD}, L_{KL} and stochastic mean for our method. Hence, the overall loss function $L_{total}(\theta_1)$ for training S-DNN is shown as (9)

$$L_{total}(\theta_1) = L_{CE}(\theta_1) + \frac{1}{K}\sum_{k=1}^{K} L_{KL}(p_k||p_s) + \frac{1}{K}\sum_{k=1}^{K} L_{KD,k} \tag{9}$$

with the objective function of multi-class image classification $L_{CE}(\theta_1)$ to train the network θ_1 is defined as the cross entropy error between the predicted values and the correct labels:

$$L_{CE}(\theta_1) = -\sum_{d=1}^{D}\sum_{c=1}^{C} I(y_d, c) log(p_1^c(x_d)) \tag{10}$$

with an indicator function I defined as

$$I(y_d, c) = \begin{cases} 1, & y_d = c \\ 0, & y_d \neq c \end{cases} \tag{11}$$

To prevent $L_{KD,k}$ larger than $L_{CE}(\theta_1)$ from inducing gradient explosion, we will adopt gradient clipping [14] to limit the gradient of knowledge distillation$\nabla(\theta_1)_{KD}^{clipped}$ during training procedure as shown in Eq. (12):

$$\nabla(\theta_1)_{KD}^{clipped} = \begin{cases} \beta \times \nabla(\theta_1)_{KD}, & \nabla(\theta_1)_{KD} < \nabla(\theta_1)_{CE} \\ \nabla(\theta_1)_{KD}, & otherwise \end{cases} \tag{12}$$

$$\beta = \frac{1}{1 + exp(-\tau + p)} \tag{13}$$

$$\tau = \frac{||\nabla(\theta_1)_{CE}||_2}{||\nabla(\theta_1)_{KD}||_2} \tag{14}$$

where β is a sigmoid function. In Eq. (13), p means the current epoch of training. Furthermore, the L2-norm ratios are the L_{CE} and $L_{KD,k}$ in Eq. (14). Hence, the rich knowledge distilled from T-DNN can be transferred knowledge S-DNN without worrying about gradient explosion.

III. Experimental Results

In this section, we will evaluate our proposed compression method with two datasets and three different models. The two datasets are the familiar CIFAR-100 [15] and the rich collection of images, ImageNet64*64 [16]. Additionally, there are two models, VGG and ResNet, training and testing on CIFAR-100 and one model named MobileNet, training and testing on ImageNet64*64 for image classification.

A. Environment and Datasets

Our proposed method is implemented in TensorFlow [17] with Python 3.5 interference on the computers (CPU: Intel® Core^{TM} i7-7800X @ 3.5 GHZ, main memory: 32 GB DRAM, GPU: NVIDIA GEFORCE ® GTX 1080).

The CIFAR-100 dataset consists of 60,000 images with a size of 32×32, divided as 50,000 training data and 10,000 test data, and 100 classes. We used random shift, random rotation and horizontal flip as data augmentations. Our proposed method was tested under the same conditions as FSP [5], and for increasing the dependability of the testing results, we ran the experiments three times and took the average as the final experimental results. We take VGG and ResNet as the DNN to prove that our proposed method works. The T-DNN and S-DNN models are shown in Fig. 3.

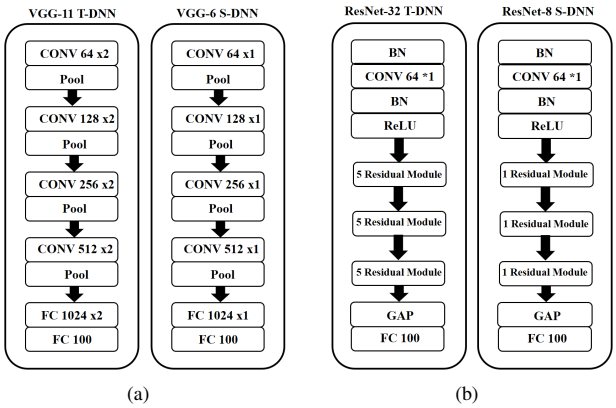

Figure 3: T-DNN and S-DNN of the VGG and ResNet models. T-DNN: VGG-11 and ResNet-32. S-DNN: VGG-6 and ResNet-8.

The ImageNet64*64 dataset consists of about 1.2 million images with a size of 64×64, divided with about 1.2 million training data and 50,000 test data, and 1000 classes. We used the same data augmentations as same with CIFAR-100 and the experiments were run three times and took the average as the final result. On ImageNet64*64, we defined MobileNet-16 as T-DNN and MobileNet-9 as S-DNN as shown in Fig. 4. We only considered MobileNet due to hardware limitation. This heuristic method needs more memory space due to our third proposed method "Offline Ensemble", the computer will face

a memory overflow with bigger models. As a result, we pick the smaller model MobileNet for ImageNet dataset to have a faster experiment to prove that our heuristic method had better Top1-accuracy than competitors.

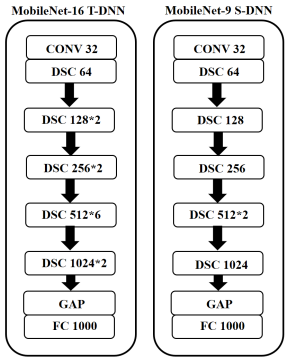

Figure 4: T-DNN and S-DNN of the MobileNet models. T-DNN: MobileNet-16. S-DNN: MobileNet-9.

On CIFAR-100, the training procedure for networks was considered by FSP [5] and SSKD_SVD [9]. We set the batch size to 128 and the training epochs to 200 during training, optimized the procedure by stochastic gradient descent (SGD) [18], and adopted Nesterov accelerated gradient [19]. The initial learning rate was set to 10^{-2} and the momentum was set to 0.9. The decay parameter was set to 10^{-4}. The learning rate was reduced to 0.1 per 50 epochs. Additionally, we set the batch size to 64 during training, training epochs to 40, and the learning rate was reduced to 0.1 per 10 epochs for ImageNet64*64.

B. Comparison with Other Work

With the same compression on S-DNN, it can be seen that our proposed method got the state-of-the-art results on VGG and ResNet models compared with the competitors [6][5][9][10]. Yim et al. [5] exploited flow between layers computed as the inner product of feature maps between layers. Another method for training small networks is distillation [6] which uses a larger network to teach a smaller network. In [9], authors improve [5] with Singular Value Decomposition (SVD). For Feature Knowledge Distillation (FKD) on [10], authors propose distance-wise and angle-wise distillation, penalizing structural differences in relations. As we can see in Table I, the result of our proposed method achieves a 66.67% Top-1 accuracy with a 2.08x compression rate and 3.5x computation rate. Additionally, the result of our proposed method achieves a 68.45% Top-1 accuracy with a 6.11x compression rate and 5.27x computation rate as shown in Table II. Furthermore, we can see in Table III that the result of our proposed method achieves a 49.86% Top-1 accuracy with a 1.59x compression rate and 2.05x computation rate. Compared to other methods, our proposed method has better performance Top-1 accuracy for all the three different network models.

Table I: Computation, parameters, and average Top-1 accuracy comparison with VGG-11 and VGG-6 on CIFAR-100. T-DNN: VGG-11, S-DNN: VGG-6.

	FLOPSs [M]	Param [M]	Exp1.	Exp2.	Exp3.	Average
T-DNN	212.8	7.93	66.01%	65.75%	66.54%	66.10%
S-DNN	60.71	3.8	63.06%	62.79%	63.45%	63.10%
Hinton [6]	60.71	3.8	64.89%	64.81%	64.84%	64.84%
FSP [5]	60.71	3.8	64.51%	64.18%	64.53%	64.40%
SSKD_SVD [9]	60.71	3.8	66.51%	66.57%	66.50%	66.52%
RKD [10]	60.71	3.8	62.41%	62.38%	62.33%	62.37%
Our Method	60.71	3.8	66.62%	66.60%	66.78%	**66.67%**

Table II: Computation, parameters, and average Top-1 accuracy comparison with ResNet-32 and ResNet-8 on CIFAR-100. T-DNN: ResNet-32, S-DNN: ResNet-8.

	FLOPSs [M]	Param [M]	Exp1.	Exp2.	Exp3.	Average
T-DNN	1101.7	7.4	68.80%	69.43%	68.79%	69.00%
S-DNN	191.79	1.21	64.34%	63.68%	64.20%	64.07%
Hinton [6]	191.79	1.21	67.83%	67.97%	67.72%	67.84%
FSP [5]	191.79	1.21	65.30%	65.48%	65.65%	65.47%
SSKD_SVD [9]	191.79	1.21	65.96%	66.32%	66.11%	66.13%
RKD [10]	191.79	1.21	66.84%	66.78%	66.05%	66.55%
Our Method	191.79	1.21	68.53%	68.41%	68.41%	**68.45%**

Table III: Computation, parameters, and average Top-1 accuracy comparison with MobileNet-16 and MobileNet-9 on ImageNet64*64. T-DNN: MobileNet-16, S-DNN: MobileNet-9.

	FLOPSs [M]	Param [M]	Exp1.	Exp2.	Exp3.	Average
T-DNN	117.59	2.97	53.48%	53.33%	53.72%	53.51%
S-DNN	57.32	1.86	45.73%	45.86%	45.89%	45.82%
Hinton [6]	57.32	1.86	49.23%	49.10%	49.12%	49.15%
FSP [5]	57.32	1.86	46.38%	46.15%	45.65%	46.39%
SSKD_SVD [9]	57.32	1.86	45.61%	44.91%	44.97%	45.16%
RKD [10]	57.32	1.86	48.98%	48.41%	48.41%	48.79%
Our Method	57.32	1.86	49.90%	49.80%	49.90%	**49.86%**

IV. Conclusion

Deep Neural Networks (DNN) have solved many tasks, including image classification , object detection, and semantic segmentation. However, when there are huge parameters and high level of computation associated with a DNN model, it becomes difficult to deploy on mobile devices. In this paper, we propose a method named Deep Neural Network Compression with Knowledge Distillation Using Cross-Layer Matrix, KL Divergence and Offline Ensemble. The experimental results achieve better accuracies than state-of-art approaches.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer*

Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009, pp. 248–255.

[2] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[4] A. S. Ecker L. A. Gatys and M. Bethge, "A neural algorithm of artistic style," in *arXiv preprint arXiv:1508.06576*, 2015, pp. 1,2.

[5] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4133–4141.

[6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[7] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[8] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens, "Net2net: Accelerating learning via knowledge transfer," *arXiv preprint arXiv:1511.05641*, 2015.

[9] Seung Hyun Lee, Dae Ha Kim, and Byung Cheol Song, "Self-supervised knowledge distillation using singular value decomposition," in *European Conference on Computer Vision*. Springer, 2018, pp. 339–354.

[10] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho, "Relational knowledge distillation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3967–3976.

[11] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu, "Deep mutual learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4320–4328.

[12] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E. Dahl, and Geoffrey E. Hinton, "Large scale distributed neural network training through online distillation," *arXiv:1804.03235*, 2018.

[13] Xu Lan, Xiatian Zhu, and Shaogang Gong, "Knowledge distillation by on-the-fly native ensemble," *arXiv preprint arXiv:1806.04606*, 2018.

[14] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks," *International Conference On Machine Learning*, 2012.

[15] Alex Krizhevsky, Geoffrey Hinton, et al., "Learning multiple layers of features from tiny images," Tech. Rep., Citeseer, 2009.

[16] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter, "A downsampled variant of imagenet as an alternative to the cifar datasets," *arXiv preprint arXiv:1707.08819*, 2017.

[17] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al., "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[18] Jack Kiefer, Jacob Wolfowitz, et al., "Stochastic estimation of the maximum of a regression function," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, 1952.

[19] Yurii Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2)," in *Doklady AN USSR*, 1983, vol. 269, pp. 543–547.