as a manuscript

NATIONAL RESEARCH
UNIVERSITY

Mikhail Figurnov

PROBABILISTIC METHOD OF ADAPTIVE COMPUTATION TIME IN NEURAL NETWORKS

SUMMARY
of a dissertation to obtain the degree of
Doctor of Philosophy in Computer Science HSE

Moscow — 2019

The PhD Dissertation was prepared at National Research University Higher School of Economics.

Academic Supervisor: Dmitry P. Vetrov, Candidate of Sciences, Research Professor, National Research University Higher School of Economics.

Topic of the thesis

In this work we present a probabilistic method for spatial adaptation of computation time in convolutional neural network, a popular computer vision model. This method improves computational efficiency and interpretability.

Actuality of the work. In recent years the amount of data collected worldwide is rapidly growing. This increases the importance of machine learning methods that allow to automatically extract patterns from data. In machine learning tasks it is assumed that the real world objects are described by *features*. There also exists a *training set* obtained from the general set of objects. In supervised learning problem the true *labels* for the training set objects are also known, and the problem is to restore the dependence of the labels from the features. The *quality* of the obtained solution is usually estimated by accuracy, the fraction of correctly determined labels on the test set. An alternative to this approach is unsupervised learning where the training set consists only of the object features. The goal of unsupervised learning is to obtain a compact and informative description of objects that can later be used, for instance, in supervised learning on a smaller labeled set [1].

A common way of solving the aforementioned machine learning problems is probabilistic modeling. For supervised learning case, the probabilistic model defines a distribution of the labels given the observed features. For unsupervised learning, a common approach is to introduce latent (unobserved) variables that define the factors of variation of the data. The parameters of the probabilistic model are fitted using the maximum likelihood method using the training set and gradient-based optimization methods. In many latent variable models the likelihood cannot be computed analytically. In this case variational methods are often employed.

The success of machine learning methods crucially depends on the informativeness of the feature representation. Some of the most complex objects to represent by features are high-dimensional unstructured objects: images, sounds, texts, graphs, etc. The volume of such data rapidly grows due to the spread of internet and social networks. By the beginning of the 2010s, methods based on expert knowledge of subject areas were developed for such data. For example, in image processing SIFT [2] and HOG [3] features were widely used; in sound processing MFCC [4] features were popular. Unfortunately, the informativeness of such features remained unsatisfactory for solving real-world problems. The lack of obvious ways to improve those features led to stagnation in the quality of the methods [5; 6].

In the last five years deep learning has become the most effective way of working with high-dimensional unstructured data [7]. Deep learning suggests using multi-layer (deep) feature representations of objects defined by neural networks with dozens or even hundreds of layers. The architecture of the neural network is determined based on the properties of the data. For example, convolutional neural networks (CNNs) [8] are often used for image processing, while recurrent neural networks (RNNs) [9] are widely employed for sound and text recognition. The last layer of the neural network usually corresponds to an answer to the problem being solved, e.g. a probability distribution for the labels. The parameters of the model, numbering up to billions [10], are trained using stochastic gradient-based optimization methods that maximize the likelihood of the probabilistic model. Thus, deep learning considers parametric models chosen based on the properties of the data and relatively simple training methods.

The key reasons for success of deep learning are creation of very large *labeled* training sets such as ImageNet [6] and development of new computing technology, most notably the graphic processing units (GPUs). In 2012 a team from Toronto successfully trained a CNN for image classification problem [11]. They were able to dramatically improve the quality of solution compared to all previous approaches, none of which used neural networks. After that CNNs became a crucial element of computer vision systems. The adoption of CNNs allowed to significantly advance scene understanding (pattern recognition) problems such as image classification, object identification, object detection and semantic segmentation. Furthermore, it turned out that improvement of quality can be achieved by increasing the amount of computation, first of all by increasing the depth (number of layers) of CNN. For instance, the abovementioned CNN from the year 2012 consists of 8 layers, while a residual network proposed in 2015 has 152 layers [12].

Despite the breakthrough in quality of solution, the CNN model has several downsides:

1. CNNs have huge computational cost which is mostly caused by convolutional layers that take up over 80% of the computation time. Modern CNNs use tens of billions floating point operations to process a single image. Such computational demands limit applicability of CNNs in many scenarios, including real-time video processing, as well as deployment on devices without powerful GPUs and in devices where the power supply is limited.

2. CNNs are hard to interpret. The complex structure of the models, large number of parameters and computation mean that the classical model analysis techniques are not applicable to CNNs. Because of this reason, it is problematic to use CNNs in high cost of error scenarios where the decisions of the system need to be validated by an expert. There exist several methods for interpretation of previously trained CNNs [13; 14]. An important problem is development of more interpretable CNNs.

In the thesis we approach these problems by using a hypothesis that CNNs are *spatially redundant*, meaning that application of some of the layers in some of the spatial positions is not necessary to obtain a high quality solution. Therefore, a method that allows skipping convolutional layers in some spatial positions would improve the speed-quality trade-off for CNN. Moreover, if the skipped spatial positions are chosen based on an object, the obtained computation time maps improve the interpretability of CNNs: the regions that have more computation are more important for the problem at hand. This mechanism is similar to how biological vision systems spend more time on the important regions of the presented image [15].

The spatial adaptivity of computation time can be seen as an attention model. Currently existing attention models for CNNs have significant drawbacks. Glimpse-based attention models [16–19] cannot be applied to many classes of problems (object detection, image segmentation, image generation); soft spatial attention models [20; 21] do not allow reducing the amount of computation; hard attention models [20; 22] are tuned using REINFORCE method [23] that makes training significantly harder.

The goal of this work is to develop a method that improves the speed-quality trade-off in CNNs.

To achieve this goal the following **problems** are solved in this thesis:

1. Perforated convolutional layer is proposed that allows to spatially adjust and lower the amount of computation.
2. Adaptive computation time method [24], that was previously proposed for RNNs, is applied to spatial adjustment of the depth (number of layers) of a CNN for a given object.
3. A probabilistic model that adjusts the spatial depth of CNNs is proposed, as well as a method to train this model.

Key results and conclusions

The novelty of this work is that for the first time the following points are shown:

1. Reduction of spatial redundancy of intermediate representations of a network allows to increase the speed of CNN.
2. Spatial adaptation of the depth (number of layers) of a CNN based on the object improves the relationship between the speed and quality of CNN and also improves the interpretability of the model.
3. Variation of the depth of a CNN can be performed by a probabilistic model with latent variables.

Theoretical and practical significance. The obtained results widen the scope of applicability of CNNs by improving the speed-quality trade-off and improving the interpretability.

Methodology and research methods. This work uses the methodology of deep learning; the toolkit of probabilistic modeling; Python, CUDA, MATLAB programming languages; NumPy, MatConvNet, TensorFlow frameworks.

Reliability of the obtained results is ensured by a detailed description of the methods and algorithms, proofs of theorems, as well as description of experiments and release of the source code which facilitates reproducibility.

Main provisions for defense:

1. Method for perforation of convolutional networks that allows to spatially adjust the amount of computation in a CNN.
2. Method for spatially adaptive computation time for adjustment of the depth (number of layers) of a CNN based on the object and the spatial position.
3. Probabilistic latent variable model for adaptation of CNN depth, as well as a method of stochastic variational optimization for training of this model.
4. Experimental evaluation of the proposed methods, including a comparison with analogous solutions.

Personal contribution into the main provisions for defense. The results are personally obtained by the author. In the works on the topic of the thesis the author proposed the key scientific ideas, implemented and performed the experiments, wrote the papers. The results of subsection 4.4 of the paper "PerforatedCNNs: Acceleration through Elimination of Redundant Convolutions" (NIPS 2016) were obtained by Aizhan Ibraimova and not in-

cluded in the thesis. The contribution of other coauthors is review of the code of the experiments, technical help with setup of experiments, discussion of the obtained results, editing of the text of the papers, problem formulation and general supervision of research.

Publications and probation of the work

The aspirant is the main author in all papers on the topic of the thesis.

First-tier publications.

1. *Figurnov M., Ibraimova A., Vetrov D. P., Kohli P.* PerforatedCNNs: Acceleration through Elimination of Redundant Convolutions // Advances in Neural Information Processing Systems 29. 2016. P. 947–955. Rank A* conference, indexed by SCOPUS.
2. *Figurnov M., Collins M. D., Zhu Y., Zhang L., Huang J., Vetrov D., Salakhutdinov R.* Spatially Adaptive Computation Time for Residual Networks // The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. P. 1039–1048. Rank A* conference, indexed by SCOPUS.
3. *Figurnov M., Sobolev A., Vetrov D.* Probabilistic adaptive computation time // Bulletin of the Polish Academy of Sciences: Technical Sciences. 2018. Vol. 66, no. 6. P. 811–820. Journal indexed by Web of Science (Q2) and SCOPUS (Q3).

Other publications.

1. *Figurnov M., Vetrov D. P., Kohli P.* PerforatedCNNs: Acceleration through Elimination of Redundant Convolutions // International Conference on Learning Representations (ICLR) Workshop. 2016.

Reports at conferences and seminars.

1. Seminar of Bayesian methods research group, Moscow, 20 February 2015. Topic: “Acceleration of convolutional neural networks”.
2. Christmas colloquium on computer vision, Skoltech, Moscow, 28 February 2015. Topic: “PerforatedCNNs: Acceleration through Elimination of Redundant Convolutions”.
3. Seminar of Institute of Information Transmission Problems of the Russian Academy of Sciences “Structure models and deep learning”, Moscow, 21 March 2016. Topic: “Acceleration of Convolutional Neural Networks through Elimination of Redundant Convolutions”.

4. “International Conference on Learning Representations 2016”, workshop, San Juan, Puerto Rico, USA, 3 May 2016. Topic: “PerforatedCNNs: Acceleration through Elimination of Redundant Convolutions”.
5. “Conference on Neural Information Processing Systems 2016”, main section, Barcelona, Spain, 7 December 2016. Topic: “PerforatedCNNs: Acceleration through Elimination of Redundant Convolutions”.
6. Seminar of OpenAI, San Francisco, California, USA, 1 March 2017. Topic: “Spatially Adaptive Computation Time for Residual Networks”.
7. Seminar of Bayesian methods research group, Moscow, 10 March 2017. Topic: “Spatially Adaptive Computation Time for Residual Networks”.
8. International summit “Machines Can See”, Moscow, 9 June 2017. Topic: “Spatially Adaptive Computation Time for Residual Networks”.
9. “IEEE Conference on Computer Vision and Pattern Recognition 2017”, main section, Honolulu, Hawaii, USA, 22 July 2017. Topic: “Spatially Adaptive Computation Time for Residual Networks”.
10. Christmas colloquium on computer vision, Skoltech, Moscow, 26 December 2017. Topic: “Spatially Adaptive Computation Time for Residual Networks”.

Volume and structure of the work. The thesis contains an introduction, four chapters and a conclusion. The full volume of the thesis is 116 pages, including 30 figures and 7 tables. The list of references contains 167 items.

Contents of the work

Introduction substantiates the relevance of the research presented in this thesis; formulates the goals and tasks of the works; describes the scientific novelty of the work and the main provisions for defense.

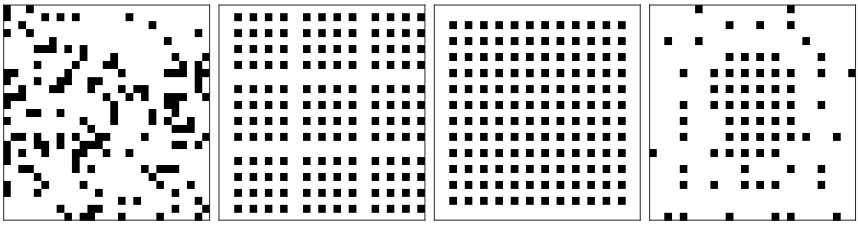
First chapter is an overview that consists of two parts. The first part presents methods of deep learning, particularly convolutional neural networks (CNNs). The problems solved by CNNs are described, including image classification, image segmentation, object detection, etc. The supervised learning problem, for which deep learning methods are currently most effective, is formulated. Stochastic optimization methods and backpropagation algorithm are presented for tuning the parameters (training) of neural networks. Parameter initialization methods are also described, since the initialization significantly

affects the final results due to non-convexity of the objective. Then, the most commonly used neural network layers are described: fully-connected layer, various activation functions, dropout layer, softmax. Next, CNN-specific layers are considered: convolutional layer, pooling layer, batch normalization, etc. A historical reference on the ImageNet image classification contest is provided, as well as examples of convolutional neural architectures that show the best performance in practice: AlexNet, VGG-16, residual network (ResNet). The second part of the chapter considers methods for training random variables parameters. These methods are required for training neural networks with stochastic variables. Several methods are described: REINFORCE method that is applicable to a wide range of probability distributions but suffers from high variance of gradients; reparameterization trick that is only applicable to a small set of continuous random variables but has low variance of gradients; Gumbel-Softmax relaxation that allows training of discrete random variables parameters using reparameterization.

Second chapter proposes CNN perforation method that accelerates CNNs by reducing their spatial redundancy. The name of the method comes from the loop perforation method that accelerates programs by skipping some loop iterations.

First, *perforated convolutional layer* is described. This is a modification of the standard convolutional layer and has the same input, output and weight tensor dimensions. The key hyperparameter of perforated convolutional layer is *perforation mask*, a subset of spatial positions of the convolutional layer's output. *Perforation rate* is the fraction of the spatial positions not in the perforation mask. The output values in the spatial positions from the perforation mask are computed exactly, meaning that they are equal to the corresponding values of the standard convolutional layer. The values in the other spatial positions are interpolated using the value from the nearest position that is computed exactly. Other interpolation methods are possible, such as replacing the missing values with zeros. Perforated convolutional layer is a generalization of the standard convolutional layer. Equivalency is achieved if the perforation masks contains all spatial positions.

Several methods for perforation mask generation (choice of a subset of output spatial positions) are presented, see fig. 1. *Uniform* mask is obtained by an equiprobable choice of positions without replacement. Its disadvantage is that the obtained points often form compact groups, increasing the average distance to the perforation mask points. *Grid* mask is Cartesian product of subsets of positions for each coordinate that are chosen using the pseudorandom



(a) Uniform (b) Grid (c) Structure (d) Impact

Figure 1: Examples of perforation masks.

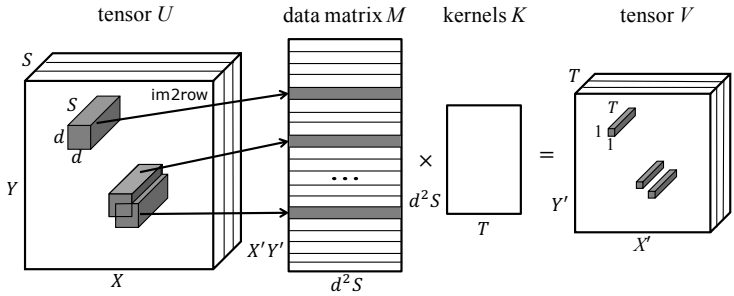


Figure 2: Reduction of convolutional layer computation to matrix multiplication.

integer sequence generation scheme [25]. If the number of positions evenly divides the size of the dimension, the mask forms a regular grid. Otherwise, the grid contains irregularities. *Structure* perforation mask contains positions that are used in the following pooling layer most often. This mask is based on an observation that for some pooling layer settings, e.g. for kernel size 3×3 and stride 2 (such settings are used in Network in Network and AlexNet models), various outputs of a convolutional layers are used different number of times. Finally, *impact* perforation mask considers the relative contribution of the spatial positions to the loss function. Denote by *impact of a position* at the output of a convolutional layer for a given image a first-order Taylor approximation to the absolute change of loss function when the true value in this position is replaced with zero. Impact of all output positions can be efficiently computed using the backpropagation algorithm. *Impact of a spatial position* is defined as a sum of impacts across channels, averaged over objects of the training set. Impact mask contains the spatial positions with the largest values of impact. For ImageNet dataset the impact mask has a center bias since the classified object is usually centered. Also, a grid similar to the one obtained in structure mask is automatically inferred.

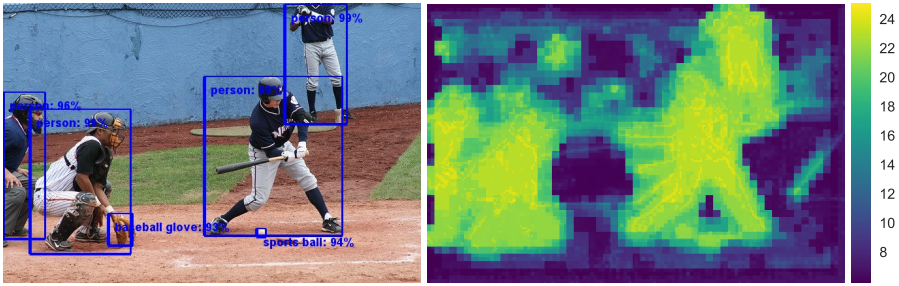


Figure 3: Object detections (left) and computation time map of the proposed method SACT (right) for a validation object of COCO dataset. SACT method uses more computation for object-like regions of the image.

An advantage of the perforated convolutional layer is that it can be effectively implemented: the reduction in computation time can be close to the reduction in the number of operations. To do that, we reduce evaluation of the layer to matrix multiplication. Specifically, subtensors corresponding to the output values from the perforation mask are cropped from the input tensor and placed into rows of the data matrix, fig. 2. Multiplication of the data matrix by the kernel matrix provides the exact values of the convolutional layer in the spatial positions of the perforation mask. Interpolation of the missing values is performed *implicitly* by indexing the read operations of the next layer. Because of this the method also reduces the memory consumption.

The method is experimentally validated on the image classification problem. First, the proposed perforation mask types are compared on the problem of acceleration of a single convolutional layer of AlexNet network (ImageNet dataset). The best-performing masks are grid and impact. Then, using the Network in Network architecture (CIFAR-10 dataset) it is shown that perforation outperforms simple baselines, such as increasing the convolutional layer stride and decreasing the input image resolution, in terms of speed-quality trade-off. Finally, perforation is used for acceleration of the whole AlexNet and VGG-16 networks for ImageNet dataset, illustrating the applicability of perforation to acceleration of large CNNs. The considered networks may be accelerated by a factor of two on CPU and GPU with an error increase of at most 2.6%. The theoretical speedups are usually close to the empirical ones, proving the effectiveness of the proposed implementation of perforated convolutional layer.

Spatially adaptive computation time (SACT) method for residual networks is introduced in **the third chapter**. This method allows to focus the computation on important regions of an image, fig. 3.

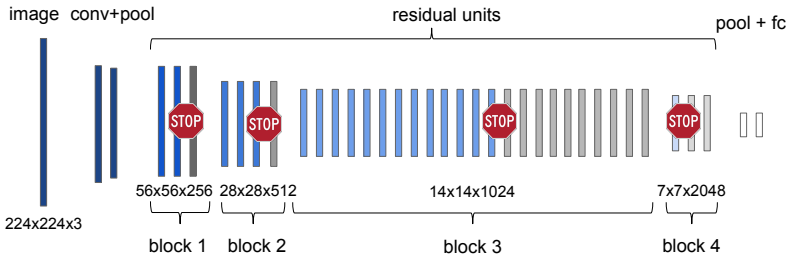


Figure 4: Spatially adaptive computation time method for a residual network ResNet-101.

First, adaptive computation time (ACT) method [24] that has been previously proposed for RNNs is applied to residual networks, a popular CNN architecture. A residual network consists of *residual units*, functions of the form $U^l = U^{l-1} + f(U^{l-1})$, where $f(U^{l-1})$ is a convolutional neural subnetwork called the *residual function*. A sequence of residual units with equal output dimensions is called a *residual block*. In the ACT method, each residual unit additionally outputs a *halting probability*, a number from the $[0, 1]$ range. Residual units and their probabilities are computed sequentially. As soon as the cumulative sum of the halting probabilities reaches one, all the following residual units in the current block are skipped. The halting probability distribution is defined as the computed halting probabilities, where the last value is replaced by *remainder*. The value of remainder is determined from the normalization condition of the probability distribution. The output of the block is defined as weighted average of the residual units outputs, where the weights are the corresponding halting probabilities. Finally, *ponder cost* is the sum of the number of computed residual units and the remainder. Minimization of the ponder cost increases the halting probability of all the modules, except for the last one, which leads to an earlier stopping. Ponder cost is used a regularizer for the original loss function. The described method is applied to every residual block of the network independently with the ponder costs added up. Thus, in every block only the first several modules are executed, fig. 4. We prove that the ACT method generalizes the residual network model.

The idea of the proposed SACT method is that ACT is applied to every spatial position of a block. Then, every position has its own halting probability. The output of the block in a spatial position is defined as a weighted average of the residual units outputs of the block in this position. A spatial position of a residual module is called *active* if its cumulative halting probability does not exceed one. It is clear that only the values in the active positions affect the out-

puts of the block, so it is reasonable to only compute these values. The values of the residual function in the inactive positions are imputed by zero, which is equivalent to copying the previous values. A residual unit that is only evaluated in the active positions can be implemented efficiently using perforated convolutional layer proposed in the second chapter, with the missing values imputed by zeros. We show that the SACT method generalizes the ACT method and therefore the residual network.

ACT and SACT methods are experimentally validated by applying to the residual network ResNet-101. *Dead residual unit* problem occurring in ACT and SACT is described: if the model is initialized "incorrectly", the last residual units of the blocks are never used. Several initialization heuristics are proposed to solve this problem.

First, image classification problem is considered (ImageNet dataset). Non-adaptive residual network with a comparable number of floating point operations is used as a baseline for the ACT and SACT methods. When the test-time resolution is increased, which is a standard practice, SACT outperforms ACT and baselines in terms of the trade-off of the quality and number of operations. It is demonstrated that the advantage of SACT persists if the training-time resolution is increased.

Then, object detection problem is considered for the COCO dataset. In this case, high resolution images are used, such as 1000×600, which is significantly larger that the standard 224×224 for ImageNet classification. SACT mechanism allows to reduce the computation time for the low information background. Faster R-CNN [26] object detection method is used, with the feature extraction residual network replaced by SACT. This approach outperforms the baseline of using non-adaptive ResNet for feature extraction in terms of the trade-off between speed and mean average precision (mAP). An example of the resulting detections and computation time map is presented on fig. 3.

Finally, experiments showing that the computation time maps of SACT correlate well with the visual saliency are presented. This is done on a large cat2000 dataset that is obtained by showing images to people and measuring the eye fixation positions. The target map is a smoothed histogram of the positions. SACT models pre-trained on ImageNet and COCO are used, and no fine-tuning on the saliency prediction problem is performed. However, a parametric post-processing of the computation time maps is done that smoothes them and accounts for the center bias of the cat2000 target maps. The post-processed maps outperform a baseline, a centered Gaussian. Thus, SACT automatically

learns to focus the computation on the regions that people consider important, therefore improving the interpretability of CNNs.

Probabilistic adaptive computation time method is proposed in **the fourth chapter**. It is based on a probabilistic model where the discrete latent variables define the number of the executed iterations. ACT method is a heuristic relaxation of the proposed probabilistic model that has a significant downside of having a discontinuous loss function. This means that the ACT method cannot be used jointly with the reparameterization trick that requires a smooth loss function.

At the beginning of the chapter a mathematical framework for stochastic MAP-inference in discriminative probabilistic models is developed. First, we describe the variational optimization method [27; 28] for maximization of a function $f(z)$ of a discrete or continuous variable z. This method is based on the variational bound

$$L(\phi) = \mathop{\mathbb{E}}_{q(z|\phi)} f(z) \leq \mathop{\mathbb{E}}_{q(z|\phi)} \max_z f(z) = \max_z f(z) \tag{1}$$

that is valid for any auxiliary distribution $q(z|\phi)$. The inequality becomes tight when the auxiliary distribution is a delta-function at the argmax of $f(z)$. Assume that the quantity $L(\phi)$ can be computed with an acceptable computational cost. Then, one can maximize $L(\phi)$ using gradient-based optimization methods.

For analytically intractable or too computationally expensive function $L(\phi)$ we propose a new method of *stochastic variational optimization*. For reparameterizable distribution $q(z|\phi)$ we propose to do reparameterization. For discrete distribution $q(z|\phi)$ two options are proposed: using REINFORCE method, or applying Gumbel-Softmax relaxation and training with the reparameterization trick. In any of these cases it becomes possible to compute a stochastic gradient of the loss function.

Consider a discriminative probabilistic model $p(y, z|x) = p(y|x, z)p(z)$, where x is object, y is target label and z is latent variable. Here $p(y|x, z)$ denotes the likelihood of the target label given the object and the latent variable that can be defined, for instance, with a neural network. *MAP inference* problem consists of finding a value of the latent variable z^* that maximizes the posterior distribution $p(z|x, y) = \frac{p(y,z|x)}{p(y|x)}$. To solve this problem one can use variational optimization with an auxiliary distribution $q(z|x, \phi)$ that does not depend on the true label, allowing to use it at the test time.

Then, probabilistic method for adaptive computation time is proposed. *Adaptive computation block* is a computational module that chooses the num-

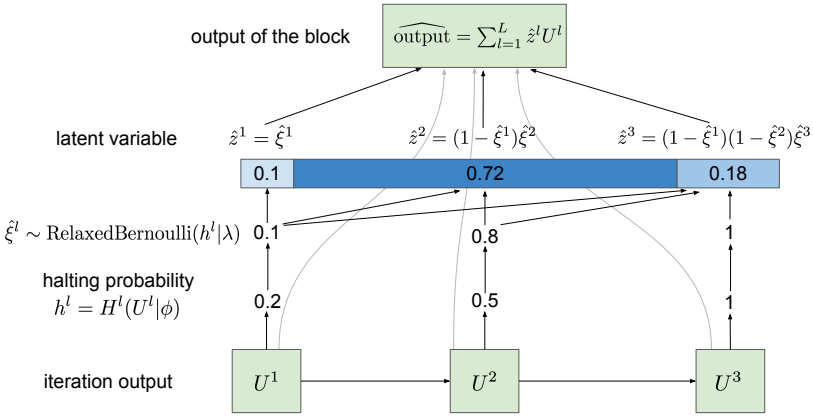


Figure 5: Relaxed adaptive computation block.

ber of iterations depending on the input. The iterations can be, for example, layers of a neural network. It is assumed that the outputs of iterations of a block have the same dimensions. Depending on the specific type of the latent variables, the block can be *discrete*, *thresholded* or *relaxed*. The blocks of different types are compatible, meaning that the parameters of a model trained with one block type can be evaluated with another. After each iteration the *halting probability*, a number in $[0, 1]$ range, is computed and used as parameter for a latent variable. In discrete block a Bernoulli random variable is generated after each iteration. This variable is called a *halting indicator*. If it is equal to one, the computation is stopped. In the thresholded block the computation is terminated when the halting probability exceeds 0.5. The relaxed block is obtained from the discrete one by replacing Bernoulli distribution with relaxed Gumbel-Softmax distribution, fig. 5. In this case, the halting indicator takes values from the $[0, 1]$ range. An output of the relaxed block is weighted average of the iterations outputs, where the weights are obtained by a stick-breaking process over the halting indicators. The model with the relaxed block can be trained using stochastic gradient descent by applying the reparameterization trick.

Assume that a neural network contains several adaptive computation blocks with a latent variable encoding the number of iterations per block. For each latent variable the prior distribution is chosen to be truncated Geometric distribution (the truncation is performed by the maximum number of iterations). Then, stochastic MAP inference is performed for the number of iterations, where the auxiliary distribution is defined by the halting indicators. The final objective has two terms: log-likelihood of the correct answer averaged

over the auxiliary distribution and a linear penalty for the expected number of iterations. This objective is analogous to the one obtained in the ACT model, but with the heuristic ponder cost replaced by the expected number of iterations.

At the end of the section several examples of applying the proposed method to neural network architectures are described. For residual networks a spatially adaptive version is introduced. Each spatial position of a residual network's block is assigned an adaptive computation block. The adaptive computation iterations correspond to residual units. The obtained method is a probabilistic counterpart of SACT. For the recurrent networks case, the construction is similar to the ACT method [24]: an adaptive computation block is used on every timestep to choose the number of network updates.

The experimental validation of the methods is performed on ResNet-32 and ResNet-110 models for CIFAR-10 classification problem. First, it is demonstrated that the parameters of the relaxed model (the model using relaxed adaptive computation blocks) are compatible with discrete and thresholded models. To this end, during training of the relaxed model its parameters are tested in discrete and thresholded models. The loss function, accuracy and the number of operations stay close across models. Then, training of the relaxed model is compared to training of the discrete model using REINFORCE method. The number of latent variables is varied by combining the spatial positions into groups and assigning a single latent variable to each group. It is shown that both training methods perform comparably for the number of latent variables under one hundred. However, when the number of latent variables is increased, REINFORCE does not allow to successfully train the model due to a high variance of gradients. Training with relaxation allows using up to 1344 latent variables. The relaxed model and SACT method have a similar speed-quality trade-off. An advantage of the probabilistic approach is that testing can be performed in a thresholded mode that has an extremely simple implementation without a loss of quality.

The main results of the work are presented in the **conclusion**:

1. A new method of convolutional neural network acceleration is developed. It is based on perforated convolutional layer that allows to spatially vary the amount of computation. It is demonstrated that the perforated convolutional layer can be efficiently implemented on both CPU and GPU. Several types of input-independent perforation masks are proposed and experimentally compared. The developed method allows to make AlexNet and VGG-16 convolutional networks several

times faster. Reducing the spatial redundancy of convolutional neural network's intermediate representations allows to improve the speed-quality trade-off.

2. Adaptive computation time method which has previously been used for recurrent neural networks is applied to residual networks. The obtained method allows to vary the number of layers in residual networks depending on the input object. Spatially adaptive computation time method is developed that allows to choose the number of layers per spatial position. It is proved that this method generalizes the previous one. Perforated convolutional layer with an input-dependent perforation mask is used for an efficient implementation of the method. The spatially-adaptive version is empirically shown to improve the speed-quality trade-off of residual networks. The best results are obtained when processing high-resolution images. It is also shown that the computation time map can be used as a human saliency model.
3. Probabilistic model of adaptive computation time is proposed. This model allows to adapt the number of layers in deep learning models such as convolutional neural networks. A training method for this model is developed. It uses stochastic variational optimization and Gumbel-Softmax discrete random variable relaxation. The original adaptive computation time method is a heuristic relaxation of the proposed model. It is shown that the the proposed method achieves similar results to the adaptive computation time method, but has a simpler implementation. Thus, it is proved that probabilistic models can be used for varying the depth of convolutional neural networks.

Bibliography

1. *Bengio Y.*, *Courville A.*, *Vincent P.* Representation learning: A review and new perspectives // IEEE transactions on pattern analysis and machine intelligence. — 2013. — Vol. 35, no. 8. — P. 1798–1828.
2. *Lowe D. G.* Object recognition from local scale-invariant features // Conference on Computer Vision and Pattern Recognition. — 1999. — Vol. 2. — P. 1150–1157.
3. *Dalal N.*, *Triggs B.* Histograms of oriented gradients for human detection // Conference on Computer Vision and Pattern Recognition. — 2005. — Vol. 1. — P. 886–893.
4. *Murty K. S. R.*, *Yegnanarayana B.* Combining evidence from residual phase and MFCC features for speaker recognition // IEEE signal processing letters. — 2006. — Vol. 13, no. 1. — P. 52–55.
5. *Furui S.* 50 years of progress in speech and speaker recognition research // ECTI Transactions on Computer and Information Technology (ECTI-CIT). — 2005. — Vol. 1, no. 2. — P. 64–74.
6. ImageNet Large Scale Visual Recognition Challenge 2016 (ILSVRC2016) Results / http://image-net.org/challenges/LSVRC/2016/results. — 2016.
7. *LeCun Y.*, *Bengio Y.*, *Hinton G.* Deep learning // Nature. — 2015. — Vol. 521, no. 7553. — P. 436–444.
8. *LeCun Y.*, *Boser B.*, *Denker J. S.*, *Henderson D.*, *Howard R. E.*, *Hubbard W.*, *Jackel L. D.* Backpropagation applied to handwritten zip code recognition // Neural computation. — 1989. — Vol. 1, no. 4. — P. 541–551.
9. *Hochreiter S.*, *Schmidhuber J.* Long short-term memory // Neural computation. — 1997. — Vol. 9, no. 8. — P. 1735–1780.
10. *Shazeer N.*, *Mirhoseini A.*, *Maziarz K.*, *Davis A.*, *Le Q.*, *Hinton G.*, *Dean J.* Outrageously large neural networks: The sparsely-gated mixture-of-experts layer // International Conference on Learning Representations. — 2017.
11. *Krizhevsky A.*, *Sutskever I.*, *Hinton G. E.* Imagenet classification with deep convolutional neural networks // Advances in Neural Information Processing Systems. — 2012.
12. *He K.*, *Zhang X.*, *Ren S.*, *Sun J.* Deep Residual Learning for Image Recognition // Conference on Computer Vision and Pattern Recognition. — 2016.
13. *Yosinski J.*, *Clune J.*, *Nguyen A.*, *Fuchs T.*, *Lipson H.* Understanding neural networks through deep visualization // ICML Deep Learning Workshop. — 2015.
14. *Nguyen A.*, *Dosovitskiy A.*, *Yosinski J.*, *Brox T.*, *Clune J.* Synthesizing the preferred inputs for neurons in neural networks via deep generator networks // Advances in Neural Information Processing Systems. — 2016. — P. 3387–3395.

15. *Rensink R. A.* The dynamic representation of scenes // Visual cognition. — 2000. — Vol. 7, no. 1–3.

16. *Larochelle H.*, *Hinton G. E.* Learning to combine foveal glimpses with a third-order Boltzmann machine // Advances in Neural Information Processing Systems. — 2010.

17. *Mnih V.*, *Heess N.*, *Graves A.*, [et al.]. Recurrent models of visual attention // Advances in Neural Information Processing Systems. — 2014.

18. *Ba J.*, *Mnih V.*, *Kavukcuoglu K.* Multiple object recognition with visual attention // International Conference on Learning Representations. — 2015.

19. *Jaderberg M.*, *Simonyan K.*, *Zisserman A.*, *Kavukcuoglu K.* Spatial transformer networks // Advances in Neural Information Processing Systems. — 2015.

20. *Xu K.*, *Ba J.*, *Kiros R.*, *Cho K.*, *Courville A.*, *Salakhutdinov R.*, *Zemel R. S.*, *Bengio Y.* Show, attend and tell: Neural image caption generation with visual attention // International Conference on Machine Learning. — 2015.

21. *Sharma S.*, *Kiros R.*, *Salakhutdinov R.* Action Recognition using Visual Attention // International Conference on Learning Representations Workshop. — 2016.

22. *Bengio E.*, *Bacon P.-L.*, *Pineau J.*, *Precup D.* Conditional Computation in Neural Networks for faster models // International Conference on Learning Representations Workshop. — 2016.

23. *Williams R. J.* Simple statistical gradient-following algorithms for connectionist reinforcement learning // Machine learning. — 1992.

24. *Graves A.* Adaptive Computation Time for Recurrent Neural Networks // arXiv. — 2016.

25. *Graham B.* Fractional Max-Pooling // arXiv. — 2014.

26. *Ren S.*, *He K.*, *Girshick R.*, *Sun J.* Faster R-CNN: Towards real-time object detection with region proposal networks // Advances in Neural Information Processing Systems. — 2015.

27. *Staines J.*, *Barber D.* Variational Optimization // arXiv. — 2012.

28. *Staines J.*, *Barber D.* Optimization by Variational Bounding // ESANN. — 2013.