The problem of neural networks communication

M Iu Leontev¹², V Iu Islenteva³, A V Mikheev³, K V Sviatov³ and S V Sukhov¹³

¹Kotelnikov Institute of Radio Engineering and Electronics of Russian Academy of Sciences (Ulyanovsk branch), Spasskaya str., 14, Ulyanovsk, Russia, 432011
²Ulyanovsk State University, S.P. Kapitsa Research Institute of Technology (Technological Research Institute), Universitetskaya Naberejnaya 1, Ulyanovsk, Russia, 432000
³Ulyanovsk State Technical University, Severny Venets, 32, Ulyanovsk, Russia, 432027

e-mail: s_sukhov@hotmail.com

Abstract. In spite of the successful application of artificial neural networks (ANNs) for the solution of multiple problems (forecasting, language translation, image classification, voice recognition etc.), ANNs are still autonomous entities incapable of communication or exchange of their knowledge. Meanwhile, the ability to communicate is critical for further development of methods of artificial intelligence. We propose and test several methods of communication and knowledge fusion of ANNs. These methods do not require the presence of the initial training data and use only the internal parameters of ANNs. We propose generative iterative and non-iterative methods of ANNs communication. Noniterative methods show the classification accuracy similar to that provided by an ensemble of ANNs. The accuracy of generative methods is similar to a network trained on the joint dataset.

1. Introduction

The ability to communicate and exchange their knowledge is crucial for artificial neural networks (ANNs) and is necessary for further development of artificial intelligence and self-learning systems. Modern ANNs, however, behave as completely autonomous units incapable of any information exchange. Additional difficulty comes from the fact that ANNs store the acquired knowledge in a distributed way amongst the millions of internal parameters (weights). Such an implicit representation of knowledge complicates its interpretation, interferes with its update, and obstructs its exchange with other neural networks. If joint use of knowledge is needed, there exists two current approaches: 1) training new neural network with a joint dataset; 2) using an ensemble of networks to produce joint prediction [1]. However, there could be situations when both of these approaches become unacceptable: previously used training data may become unavailable; storing and maintaining an ensemble of ANNs may become too resource demanding. Ideally, neural networks should be able to exchange their knowledge without any external help; the result of this exchange would be a single ANN containing the fused knowledge (in the following we denote this as fANN).

The problem of knowledge fusion of several ANNs may be divided into two separate sub-problems: extraction of knowledge from one neural network and its incorporation into another ANN. The problem of knowledge extraction is essential for understanding of the functioning of neural networks. Without this understanding, ANNs cannot be used in such critical applications as
transportation and health care. The problem of new knowledge incorporation into a neural network is closely related to the problem of catastrophic interference: when trained on new piece of data without an access to previous datasets, ANN quickly forgets all the previously learned knowledge [2]. Thus, the development of algorithms for communication and knowledge exchange between neural networks would simultaneously solve other important problems of machine learning. At the same time, one can try to find methods that do not require explicit extraction and embedding of knowledge, but instead operate with internal parameters of networks.

This paper is devoted to the development of methods of knowledge fusion of neural networks that do not lead to the forgetting of previously learned content and do not require previous training data. The need in the integration of knowledge from different ANNs without an access to previous training data may appear in various situations. One example is medical expert systems, which training data could be confidential, but joint knowledge can lead to better diagnosis. Another example is autonomous vehicles, which can benefit from the experience of peers. Previously, the problem of knowledge fusion was solved only for ANNs with similar weights [3] and for Support Vector Machines [4]. We developed both noniterative methods and generative iterative approaches for the solution of knowledge fusion problem for ANNs trained independently. Noniterative methods allow fast knowledge fusion; generative methods provide more accuracy.

Let us note that the problem of knowledge fusion should not be confused with the term “knowledge transfer” (or “transfer learning”) [5]. According to transfer learning concept, the weights of deep network trained on one set of data can be used for training on another dataset from a similar area to speed up the training [5]. Thus, transfer learning is more concerned with the learning of new task rather than maintaining all the knowledge. Examples of transfer learning can be found in such areas as computer vision [6], language processing [7] etc. In contrast to knowledge transfer, knowledge fusion tries to integrate all the knowledge of initial networks into a single ANN. Another difference with knowledge transfer is the absence of training data, which makes the problem of knowledge fusion even more challenging.

2. Problem statement
In its simplest form, the problem of knowledge fusion can be stated as follows. Two ANNs (A and B) with a similar structure are trained on a classification task by some standard method (for example, by backpropagation). Each neural network is trained on its own dataset. After the training is complete, each neural network contains different knowledge. It is necessary to modify the parameters of one of the networks (for example, by changing the weights) in such a way that the resulting network would be able to reproduce the response of both of the initial networks as if it was trained on a joint dataset (figure 1). During fusion of knowledge, we assume that all the previous training data are no longer accessible.

\[ \text{Network } A \bigcup \text{Network } B = \text{Network with fused knowledge } A \bigcup B \]

**Figure 1.** The scheme of knowledge fusion of neural networks. The weights of the original networks A and B are changed in such a way that the final network A \( \bigcup \) B is able to classify the union of original classes. The output layer of the final network may need to be expanded to accommodate the full set of classes.

In experiments performed in this work, we use one-hot encoding in the output layers of ANNs. This encoding assumes activation of one output neuron corresponding to a specific class. Thus, the
output layer of the final network may need to be expanded to have a possibility to encode larger number of classes (figure 1).

For the fusion of knowledge of deep networks, we additionally utilize the concept of transfer learning [5]. A typical architecture of deep networks consists of a convolutional part, which extracts the features from raw data, and a fully connected classifier on top that performs the separation of the features into classes. The process of training of deep networks consisted of several stages. First, the weights of some network trained on large image dataset (for example, ImageNet [8]) were used for initialization of the weights of the convolutional part. The large set of classes of ImageNet database makes the pretrained network an ideal candidate for transfer learning. At the second stage, the bottom convolutional part was supplemented by an untrained classifier, which was later trained on various classification tasks (for example, classification of animals or urban objects). At the third stage, the convolutional part of each network underwent fine tuning leading to the final slight modification of the weights. As a result, we obtained a set of deep networks with different set of weights and containing different knowledge.

Next section describes the methods for the merging the knowledge contained in several networks in more details.

3. The methods of knowledge fusion

3.1. Method of weights summation

The method of weights summation (WS) is based on a probabilistic process of summation of a large number of random variables (weights of ANNs) [9]. The weights of the network with a fused knowledge \( \theta_F \) can be obtained by direct summation of weights of networks \( A \) and \( B \):

\[
\theta_F = \theta_A + \theta_B.
\]

In spite of its simplicity, this algorithm results in a rather high classification accuracy of the final fused network. We propose the following explanation of the efficiency of WS method. For some feature vector \( f_A \) belonging to class \( c_A \) of network \( A \), the presynaptic activity in \( f_{ANN} \) is determined by the following expression:

\[
\theta_F f_A = \theta_A f_A + \theta_B f_A.
\]

In principle, the trained neural networks are robust under the perturbation of the weights. Thus, if the second term at the right-hand side of equation (1) is significantly smaller than the first one, the fused network will still be able to perform correct classification. To ensure the inequality \( \theta_A, i f_A \gg \theta_B, i f_A \) for every node \( i \) (where \( \theta_{(A,B),i} \) are the weights connected to the node \( i \)), several conditions should be met. First, networks \( A \) and \( B \) should be well trained in a sense that they produce noticeable presynaptic activity in the presence of native features and low activity in the presence of foreign ones. Second, the mean of the weights’ probability distribution within one layer should be zero \( \langle \theta_A \rangle = \langle \theta_B \rangle = 0 \).

Indeed, the weights of one layer \( \theta_{A,i,j} \), \( \theta_{B,i,j} \) and components of the feature vector \( f_{i,A} \) may be considered as random variables taken from some distribution. For network \( B \), feature vector \( f_{i,A} \) contains independent random variables. The product of independent random variables \( \langle \theta_B \rangle f_{i,A} \) can be rewritten as

\[
\theta_B, i f_A = \sum_j \theta_{B, i,j} f_{A,j} \approx N_f \langle \theta_{B,i} \rangle f_{i,A},
\]

where \( N_f \) is the number of components in the feature vector and average \( \langle \theta_{B,i} \rangle \) is taken over presynaptic weights of corresponding layer. It follows from equation (2), that the term \( \theta_B, i f_A \) will be close to zero if the expected value of weights distribution is zero \( \langle \theta_B \rangle \approx 0 \). The property \( \langle \theta_B \rangle = 0 \) is valid for common training techniques [10] and is maintained in current research.

In our experiments, the WS algorithm was tested on shallow fully-connected feed-forward networks and on deep convolutional networks. The experiments with fully-connected networks were performed on a set of handwritten digits MNIST [11]. The whole dataset consists of 60000 training images of size 28 by 28 and 10000 test samples corresponding to 10 classes. We arbitrarily divided the whole dataset into two equal sets containing 5 classes each. The intensity of every input image was normalized to be in the range \([0,1]\). Each network \( A \) and \( B \) was trained on data corresponding only to the target classes of particular network. Thus, each network never saw the samples corresponding to
the classes of another network and could not meaningfully classify corresponding images. The training algorithm was implemented in Keras 2.2 with the TensorFlow backend. Both networks A and B had an input layer with 784 nodes corresponding to the number of pixels in input images, one hidden layer with 800 nodes, and output layer with 5 nodes corresponding to the number of classes. All hidden layers had an additional bias term. The nodes in the hidden layer had ReLU activation functions. For the output layer, we used softmax activation; the cross-entropy was used as a loss function. Before training, all the weights were initialized randomly from a normal distribution with zero mean with a standard deviation 0.05. L2 regularization was used during training with $L_2 = 0.001$. Adam iterative method was used for training with all the parameters set to the Keras' default values. In each experiment, a batch size of 200 was used. Early stopping was used while testing the accuracy on a validation set consisting of 12000 samples. We quantify the efficiency of classification of each network through accuracy (percent of correctly classified images). Simulations were repeated 10 times with random initialization of weights and with random subdivision of digits into classes to accumulate proper statistics.

The accuracy of the original networks A and B on partial datasets achieved 0.983±0.002. After the fusion of knowledge by weights summation, the accuracy of fANN turned out to be 0.864±0.035. This number is certainly smaller than the accuracy of a network trained on a joint dataset 0.968±0.002, but still much larger than a chance level.

We also compared the accuracy of fANN to the classification accuracy of an ensemble of networks A and B. The ensemble prediction was chosen on a base of maximum activity among all output neurons of all networks. The accuracy of fANN appeared to be just a little lower than the accuracy of an ensemble 0.900±0.021. However, one should remember that fANN replaces the ensemble of networks with just one neural net.

Experiments with deep networks were performed on separate classes of images of ImageNet project [8] (www.image-net.org). Considering the variety of the objects indexed in the project, one can construct the networks trained to classify either similar or dissimilar classes of objects. In particular, for current experiment, two “indigenous” neural networks were trained to classify 11 African and 11 Australian animals (table 1). The third “urban” neural network was trained to classify 11 objects pertinent to urban life (table 1). Each class had approximately 1500 samples.

As in experiments with MNIST dataset, it was assumed that the original trained networks A and B observed only the objects of their own classes and never saw the objects from the classes of other networks. The aim was to embed the knowledge contained in one neural network into another one in such a way that the resulting network can recognize the objects (e.g., animals or household item) from the previously unknown category even without ever “seeing” the objects from those categories before.

For the architecture of networks A and B, we chose VGG-16 model [12] included in Keras version 2.2 with its weights pre-trained on 1000 ImageNet classes. Those 1000 classes include large variety of objects, but, still, do not contain all the classes used in our experiments.

With the concept of knowledge transfer, the pretrained convolutional part of VGG-16 network was a good starting point for the training on previously unknown objects. In the pretrained VGG-16 network, we kept only its bottom convolutional part that we used as universal feature extractor. On top of this convolutional part, we placed a custom classifier consisting of one hidden layer with 128 nodes and one output layer with 11 neurons corresponding to the number of classes of the networks being fused. Dropout was used for the weights connecting hidden to output nodes of the classifier. ReLU activation function was used for the hidden layer, softmax activation was used for the output layer.

The input to VGG-16 network has to be a fixed-size 224 × 224 RGB image that was cropped out of a center of input raw images. Augmentation of dataset was not used. All the images were preprocessed by scaling the intensity of every image to the range [0,1]; 15% of the images were dedicated to validation and another 15% were used for the test. For the training, we used 64 pictures batch, and the training was performed with Adam optimizer on a computer with GPU support (Ubuntu 16.04 x64, Intel core i3, 6GB RAM, Nvidia GeForce GTX 1050 Ti 4GB).

The training procedure for the initial networks A and B was performed in several steps. First, the bottom convolutional part was frozen, and the top classifier was initialized with random weights and trained with small learning rate ($10^{-5}$) over 4 epochs.
Then we froze the weights of the classifier and unfroze the top block (3 layers) of VGG-16 convolutional part. This block was trained over 2 epochs with the same small learning rate. Such a training procedure resulted in networks $A$ and $B$ different not only in their top fully connected parts, but also in their convolutional layers. As a result, the networks trained to classify African and Australian animals achieved 0.951±0.005 and 0.933±0.004 accuracy, correspondingly. The network trained to classify urban objects achieved 0.955±0.003 accuracy. The main source of classification errors was the mixing between classes with similar features. For example, the African network often misclassified chimpanzees and gorillas. The Australian network, in its turn, misclassified kangaroos and wallabies.

As the weights of the convolutional parts of both networks $A$ and $B$ originated from the same set of pretrained weights, during fusion by WS method, those weights were averaged according to the approach adopted by Smith & Gashler [3]. The weights of the classifiers on the top of networks were obtained from an independent set of weights and thus were added together according to the weights summation method. Experiments demonstrated that the classification accuracy of the network with merged knowledge depends on the similarity of the features of the objects, classified by original networks $A$ and $B$. For example, if both networks $A$ and $B$ were trained to recognize the objects of the same category, then the network with fused knowledge would confuse the objects with similar sets of features. Figure 2 shows the result of knowledge fusion between two networks in the case when they were trained to recognize African and Australian animals. fANN is confident (classification accuracy is of the order of 90%) about animals with unique features, for example, zebras, but confuses similar animals from different continents (for example, emus and ostriches). An average classification accuracy of fANN is 76%. For comparison, the classification accuracy of the same set of classes by an ensemble of networks $A$ and $B$ is 81%.

In the case of very different set of features of the objects recognized by networks $A$ and $B$ (for example, in the case of living and inanimate objects), the classification accuracy of fANN increases. Figure 3 shows the result of knowledge fusion from two networks in a case when one of the networks was trained to recognize African animals and the other was trained to recognize urban objects. An average classification accuracy of fANN was 81%, which is better than in previous example. The classification accuracy by an ensemble of networks was 88%.

3.2. Method of elastic weight consolidation

The second noniterative method that we used for the fusion of knowledge of ANNs was based on an idea of selective modification of weights. If one changes the weights of networks $A$ and $B$ until these weights coincide with additional condition that loss functions $L_A(\theta)$ and $L_B(\theta)$ increase insignificantly

$$\theta_F = \arg \min_{\theta} [L_A(\theta) + L_B(\theta)],$$

then the resulting network would have the properties of both networks $A$ and $B$. 

| Table 1. Classes of objects (synsets) of ImageNet database used for training three neural networks. |
|-----------------------------------------------|
| African animals | Australian animals | Urban objects |
| name | Synset index | name | Synset index | Name | Synset index |
| gazelle | n02423022 | dingo | n02115641 | Laptop | n03642806 |
| African elephant | n02504458 | koala | n01882714 | Airplane | n02691156 |
| zebra | n02391049 | echidna | n01872401 | park bench | n03891251 |
| chimpanzee | n02481823 | wallaby | n01877812 | Desk | n03179701 |
| gorilla | n02480855 | platypus | n01873310 | Telephone | n04401088 |
| hippopotamus | n02398521 | wombat | n01883070 | television set | n04405907 |
| lion | n02129165 | Tasmanian devil | n01884104 | Chair | n0301627 |
| cheetah | n02130308 | flying fox | n02140049 | Train | n04468005 |
| ostrich | n01518878 | cassowary | n01519563 | Automobile | n02958343 |
| rhinoceros | n02391994 | emu | n01519873 | Building | n02913152 |
| giraffe | n02439033 | giant kangaroo | n01877606 | Clock | n03046257 |
Figure 2. Confusion matrix for classification of union set of African and Australian animals after fusion of knowledge of two networks by the method of weights summation. Numbers show the percentage of classified object of a specific category.

The increase of total loss function $L_F(\theta_F) = L_A(\theta_F) + L_B(\theta_F)$ is minimal if we change only less important weights for current task. Following paper [13], we call this method Elastic Weight Consolidation (EWC).

To estimate the change of the loss function, one can decompose it into Taylor’s series with respect to weights. Here is an example of such a decomposition for network $A$:

$$L_A(\theta) \approx L(\theta_A) + \sum_i \frac{\partial L}{\partial \theta_i} \bigg|_{\theta = \theta_A} (\theta_i - \theta_A) + \frac{1}{2} \sum_{i,j} \frac{\partial^2 L}{\partial \theta_i \partial \theta_j} \bigg|_{\theta_i = \theta_A, \theta_j = \theta_A} (\theta_i - \theta_A)(\theta_j - \theta_A).$$

(3)

Here $\theta_A$ are the optimal weights of network $A$ trained on corresponding dataset; the summation in equation (3) is performed over all the weights. As the number of parameters $\theta_A$ reaches millions in modern neural networks, the computation and the storage of the whole matrix of second order partial derivatives (Hessian) $\frac{\partial^2 L}{\partial \theta_i \partial \theta_j}$ is impossible. Following paper [13], we approximated the whole Hessian by its diagonal part:

$$L_A(\theta) \approx \frac{1}{2} \sum_i F_{A,i} (\theta_i - \theta_A)^2,$$

(4)

where $F_{A,i} \equiv \frac{\partial^2 L_A}{\partial \theta_i^2}$. Expressions similar to (3) and (4) should be written for network $B$ as well. Factors $F_{A,i}, F_{B,i}$ in equation (4) express the importance of weights $\theta_A, \theta_B$. Usually, one needs to calculate these coefficients in advance during the training of neural network.
Figure 3. Confusion matrix for classification of African animals and urban objects after merging the knowledge of two deep networks by weights summation method. Numbers show the percentage of classified objects of a specific category.

It is easy to show that the weights of fANN $\theta_F$ that minimize the loss function $L_F$ can be calculated with the following formula

$$\theta_{F,i} = \frac{F_{A,i} \theta_{A,i} + F_{B,i} \theta_{B,i}}{F_{A,i} + F_{B,i}}.$$  (5)

In principle, equation (5) solves the problem of fusion knowledge of two ANNs. However, weights $\theta_A$, $\theta_B$ of independently trained networks are not necessarily located in a vicinity of one another, and decomposition (3) may become insufficient. In attempt to bring closer the values of $\theta_A$ and $\theta_B$, we performed an “alignment” of networks by rearranging hidden neurons. The contribution to the overall loss function $L_F$ from a pairing of a node $k$ of network $A$ and a node $l$ of network $B$ is

$$L_{F,k,l} = \sum_i \left[ F_{A,i,k} (\theta_{F,i,k} - \theta_{A,k,i})^2 + F_{B,i,l} (\theta_{F,i,l} - \theta_{B,l,i})^2 \right] = \sum_i \frac{F_{A,i,k} F_{B,i,l}}{F_{A,i} + F_{B,i}} (\theta_{A,k,i} - \theta_{B,l,i})^2,$$

where the summation is performed over all the weights $i$ connected to the nodes $k$ and $l$. The total loss function $L_F$ can be found as a summation over all the pairs $(k, l)$:

$$L_F = \sum_{(k, l)} L_{F,k,l}.$$
To find the optimal pairing of neurons from networks $A$ and $B$, one needs to solve, so called, Assignment Problem [14]. The coefficients $L^k_l$ compose the cost matrix that serves as input for the Assignment Problem algorithm.

For experiments with EWC method, we used the networks with the same architecture as in WS experiments. Similarly, the experiments were performed on shallow networks with MNIST data and on deep convolutional networks with ImageNet data. In spite of heavier computations, EWC demonstrated performance very similar to the one of weights summation method. This tells us that noniterative methods of knowledge fusion are limited in achieving high classification accuracy. An alternative approach can be developed on the base of generative models.

### 3.3. Generative methods

Recently, neural networks were developed that can generate data of the same type and similar probabilistic distribution $p(\mathbf{x})$ as the initial training data. One of such ANNs are generative-adversarial networks (GANs) [15]. Another type of generative neural networks is variational autoencoder (VAE) [16].

The idea behind the generative knowledge fusion is in the following. The knowledge of the trained classifier is its ability to predict the probability $p(\mathbf{y} = c|\mathbf{x})$ of an object $\mathbf{x}$ belonging to class $c$. If another network somehow learns to reproduce this probability than it acquires the knowledge of the first network. Moreover, original dataset $\{\mathbf{x}\}$ is not necessary; one just need to reproduce probability $p(\mathbf{y} = c|\mathbf{x})$ for any $\mathbf{x}$ sampled from probability $p(\mathbf{x})$. Generative models can provide both samples $\mathbf{x}$ from probability $p(\mathbf{x})$ and soft targets $\tilde{\mathbf{y}}$ representing probabilities of these samples to belong to certain classes.

![Figure 4](image.png)

**Figure 4.** Generative approach to knowledge fusion of ANNs. (a) The architecture of generative network. Initial classifier (placed inside the dashed box) is supplemented by latent variable sampler and decoder; (b) The scheme of knowledge fusion. Artificial samples $\tilde{x}_A, \tilde{x}_B$ and corresponding soft targets $\tilde{y}_A, \tilde{y}_B$ produced by generative parts of networks $A$ and $B$ are used to retrain one of the networks (network $B$).

For our experiments, we used variational autoencoders [16]. The architecture of VAE (figure 4a) was similar to the one used by van de Ven & Tolias [17] for prevention of catastrophic interference in sequential learning. Regular classifier was supplemented by generative part (figure 4a). The generative part includes standard VAE components [16]: sampler from latent distribution $N(\mu, \sigma)$ and decoder (figure 4a). The loss function of the resulting VAE $L_{total}$ consisted of three parts: categorical cross-entropy classification error $L_{class}$, binary cross-entropy reconstruction loss $L_{recon}$, and latent variable regularization term $L_{KL}$ [16,17] (Kullback–Leibler divergence between prior distribution of latent variable and its approximate posterior):

$$L_{total} = L_{class} + L_{recon} + L_{KL}.$$

After the VAE have being trained, it can generate artificial data $\tilde{\mathbf{x}}$ that properly sample probability distribution $p(\mathbf{x})$; the complementing soft targets $\tilde{\mathbf{y}}$ can be obtained as output of classifier for $\tilde{\mathbf{x}}$ acting
as input. The synthetic training dataset \( \{ \tilde{x}, \tilde{y} \} \) now can be used to update the knowledge of another ANN. To prevent catastrophic forgetting, synthetic data should be generated not only from knowledge donor but also from knowledge acceptor (figure 4b).

The experiments with generative approach were performed on MNIST dataset. The classifier of the same structure as the one described in section 3.1 was supplemented by generative part (figure 4a) consisting of two layers with 8 nodes each encoding latent distribution mean \( \mu \) and variance \( \sigma \), one hidden layer with 800 neurons (decoder) and ReLU activation function, and output layer with sigmoid activation function and 784 nodes corresponding to the dimension of the original images.

As in previous experiments, MNIST dataset was arbitrarily divided into two parts containing 5 classes each. Networks A and B were trained over 40 epochs with Adam optimizer to properly classify and generate images of digits. The classification accuracy of networks A and B achieved 0.983±0.004. After the networks A and B have being trained, 60000 artificial samples \( \tilde{x} \) with corresponding soft labels \( \tilde{y} \) were generated from the decoder by supplying it with random variable sampled from standard normal distribution. The examples of generated images are shown in figure 5. As usually happens with images generated by VAEs, the digits appear somewhat blurred.

![Figure 5. Examples of images of digits generated by network A (top) and network B (bottom).](image)

The generated datasets \( \{ \tilde{x}_A, \tilde{y}_A \}, \{ \tilde{x}_B, \tilde{y}_B \} \) were supplied to network B (figure 4b); network B was trained on this combined dataset over 40 epochs. The accuracy of the retrained network B tested on complete MNIST test set was 0.929±0.006. This is considerably higher than the accuracy of noniterative knowledge fusion, higher than the accuracy of an ensemble of networks, and it is getting close to the accuracy of a network trained on joint dataset (0.9698±0.0006). The shortcoming of generative method however is larger network, that should incorporate generative part and longer training time.

4. Conclusion

In this work, we demonstrated the possibility of knowledge fusion of several neural networks without an access to the original training data. The developed methods are applicable both for fully-connected and for convolutional networks. The methods were tested in classification tasks using publicly available datasets.

Noniterative methods (summation of weights and elastic weight consolidation) demonstrate the possibility of fast fusion of knowledge from several neural networks. The classification accuracy of the network with fused knowledge depends on the similarity of features of objects in the fused datasets: the classification accuracy increases when fusing knowledge from different areas. The classification accuracy of fANN obtained with WS and EWC methods is similar to one obtained from an ensemble of networks.

Generative iterative methods are slower than noniterative ones, however, they show higher accuracy in knowledge fusion. The accuracy of fANN obtained with generative methods is similar to the accuracy of the network trained on a joint set of training data. The quality of knowledge fusion is determined by the quality of reconstruction of original data by generative models and can be increased by improving VAE’s generation quality [18] or by using different generative models [19]. In this paper, we presented generative methods only for shallow networks. In the future, we plan to extend the generative approaches for deep convolutional networks.

In this work, we considered examples of knowledge fusion between two ANNs. However, the proposed methods allow simple generalization for several networks and for networks with dissimilar architecture.

In addition to applications in machine learning, development of methods of interaction and information exchange between neural networks will be useful for other scientific areas such as neuro-
engineering (e.g., for development of neuro-prostheses) and for neurobiology for understanding the principles of communication between brain areas.

5. References
[1] Li H, Wang X and Ding S 2018 Artif. Intell. Rev. 49 455-479
[2] McCloskey M and Cohen N J 1989 Catastrophic interference in connectionist networks: The sequential learning problem Psychol. Learn. Motiv. 24 109-165
[3] Parisi G I, Kemker R, Part J L, Kanan C and Wermter S 2019 Neural Netw. 113 54-71
[4] Smith J and Gashler M 2017 Proc. 16th IEEE Int. Conf. on Machine Learning and Applications (New Jersey: IEEE) 731-736
[5] Zhu L, Ikeda K, Pang S, Ban T and Sarrafzadeh A 2018 Neural Netw. 100 25-38
[6] Lu J, Behbood V, Hao P, Zuo H, Xue S and Zhang G 2015 Knowl. Based. Syst. 80 14-23
[7] Nikonorov A V, Petrov M V, Bibikov S A, Kutikova V V, Morozov A A and Kazanskiy N L 2017 Computer Optics 41(6) 875-877 DOI: 10.18287/2412-6179-2017-41-6-875-887
[8] Shin S, Hwang K and Sung W 2016 Generative Knowledge Transfer for Neural Language Models Preprint arXiv:1608.04077
[9] Russakovsky O 2015 Int. J. Comput. Vis. 115 211-252
[10] Akhlaghi M I and Sukhov S V 2018 Neural Process. Lett. 48 257-272
[11] Bellido I and Fiesler E 1993 Proc. Int. Conf. on Artificial Neural Networks (London: Springer) 772-775
[12] Blundell C, Cornebise J, Kavukcuoglu K and Wierstra D 2015 Proc. 32nd Int. Conf. on Machine Learning 37 1613-1622
[13] LeCun Y, Bottou L, Bengio Y and Haffner P 1998 Proc. of the IEEE 11 2278-2324
[14] Simonyan K and Zisserman A 2014 Very Deep Convolutional Networks for Large-Scale Image Recognition Preprint arXiv:1409.1556
[15] Kirkpatrick J 2017 Proc. Natl. Acad. Sci. U. S. A. 28 3521
[16] Taha H A 1976 Operations research: an introduction (New York: Macmillan.)
[17] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y 2014 Proc. 27th Int. Conf. on Neural Information Processing Systems 2 2672-2680
[18] Kingma D P and Welling M 2013 Auto-encoding variational bayes Preprint arXiv:1312.6114
[19] van de Ven G M and Tolias A S 2018 Generative replay with feedback connections as a general strategy for continual learning Preprint arXiv:1809.10635
[20] Hou X, Shen L, Sun K and Qiu G 2017 Proc. IEEE Winter Conf. on Applications of Computer Vision (New Jersey: IEEE) 1133-1141
[21] Shin H, Lee J K, Kim J and Kim J 2017 Proc. 31st Int. Conf. on Neural Information Processing Systems (Red Hook: Curran Associates Inc) 2994-3003
[22] Larsen A B, Sønderby S K, Larochelle H and Winther O 2016 Proc. Int. Conf. on Machine Learning 48 1558-1566

Acknowledgments
The reported study was funded by Russian Foundation for Basic Research and the government of Ulyanovsk region according to the research project N18-47-732006.