Merging Neural Network Models Based on Their Geometric Representation

Sergey Eremeev[1, a) [0000-0001-8482-1479]], Denis Pankratov[1, b) [0009-0008-6799-2936]]

[1] Murom Institute of Vladimir State University, Russia
a) sv-eremeev@yandex.ru, b) denis_pankratov2000@mail.ru

Abstract. With a sharp increase in the number of different neural network models, there is a need to effectively combine them to solve various tasks, the largest number of which is observed in image processing. In this regard, the integration of models is one of the fundamental problems in the field of artificial intelligence. This study aims to develop a method for integrating neural network models based on the analysis of geometric structures formed by neurons in the form of hyperplanes dividing the feature space. The intersections of hyperplanes corresponding to the neurons of the pre-trained models are analyzed. The structure of the network layers and the weight matrices of the integrated model are formed automatically based on the intersection information, which allows you to create a new model from the family of initial models without additional training. A formal description of the proposed method is given. Experiments on test and real data for a two-dimensional feature space are demonstrated, in which the possibilities of merging pre-trained models into a new integrated model without training are geometrically illustrated.

Keywords: integration of neural network models, neurons, hyperplanes, intersection of straight lines, geometric representation of neural network models

1 Introduction

Artificial intelligence currently has widespread adoption. Models are being developed for various fields of human activity. Its use is particularly active for image processing. Based on existing software libraries and open datasets, creating a model for a specific subject area is greatly simplified. There are thousands of different models all over the world that perform certain functions and solve specific tasks in various fields, for example, recognizing parts in production, identifying a person's face when boarding a train, etc. The growth of solved practical tasks will only increase in the near future.

However, the global artificial intelligence community is facing a number of challenges. The presence of thousands of models poses new challenges for researchers in terms of integrating already trained neural network models [1]. But creating a new model requires significant computing and time resources.

By integrating neural network models, we will understand the creation of a new model that has the capabilities of ready-made trained models [8]. The availability of

effective ways to combine models opens up new opportunities for combining their functional properties, i.e. inheritance and skill expansion. Having ready-made models for solving individual tasks, for example, in robotics, it is possible to significantly expand the skills of robotic devices during integration. However, there are certain obstacles to obtaining such integration [11]. Often, in order to achieve a high-quality result during integration, additional training and recalculation of neural network coefficients are required. And this, in turn, requires significant financial costs.

Thus, the problem of integrating already trained neural network models and developing new effective methods of merging them, allowing to obtain an integrated model with minimal costs and high quality and accuracy of the new neural network, is currently urgent.

2 Review

At the moment, there are already quite a lot of different approaches for integrating neural network models. The simplest approaches are based on arithmetic averaging of weight matrices. For this purpose, several models of the same architecture are used, which were trained on similar data. In [10], it is shown how averaging the weight coefficients of different models allows you achieving high accuracy in image classification. The authors of [7] use stochastic averaging of weights, which improves the generalization of models and requires practically no computational costs. To increase reliability, ensemble learning is used, in which the models trained on different data are also averaged [12]. Despite the obvious simplicity and minimal computational costs, accuracy decreases in the case of too heterogeneous models.

Methods with multitasking learning are actively developing [2], in which several tasks are simultaneously solved using one common model. This type includes a model that contains common layers for training in its architecture, and then divides the network structure into several parts to solve a specific problem. Such architectures increase efficiency in solving different tasks because there is a common part of the architecture for data training. But creating such architectures is a rather complicated process and it is not possible to find a common neural network architecture for training for all tasks.

Approaches based on vector arithmetic of problems change weights through vector operations [6]. The weight matrix of the W_M model of the new M model is constructed using arithmetic operations between the weight matrices W_{M_1}, W_{M_2} and W_{base} of the pre-trained M_1 and M_2 models, which share a common basic M_{base} model:

$$W_M = W_{base} + \alpha(W_{M_1} - W_{base}) + \beta(W_{M_2} - W_{base}),$$

where α, β are empirically selected numerical parameters that allow you to control the new model.

In continuation of the considered approach, good results in terms of model integration were achieved in one of the recent works [1], which is based on an evolutionary algorithm. The algorithm proposed by the authors optimizes model merging using two spaces: a parametric space for combining the weights of pre-

trained models and a data flow space for optimizing the route of input information during its computational passage through the model. With such an evolutionary merger, a significant reduction in the cost of calculating the weights of the new model and obtaining additional features is achieved.

Fusion through decomposition assumes that the weight matrices of the pre-trained models are approximated by low-rank matrices using singular value decomposition, after which the bases and singular values are combined to form a new model. Low-rank approximation methods are being improved to compress deep neural networks [5], and tensor decomposition is being applied [9].

To create a compact and fast model, knowledge distillation is used, resulting in the process of transferring knowledge from a large model to a smaller one. An overview of distillation methods is given in [4].

In methods based on sparse masking, models have overlapping areas of specialization [3]. A binary mask is generated for each model, which determines the most important weights. Next, the weights are integrated only into those positions where the masks are active.

Thus, the existing model integration approaches are aimed at reducing computational resources during training.

3 Method

Let $X = \{(x_{1j}, x_{2j}, \dots, x_{nj})\}_{j=1}^{m}$ be the set of features from m points in an n dimensional space. Also, as initial data, we will consider k pre-trained models $M_1, M_2, \dots, M_k$ which solve their specific problem on the set X, forming the output data $Y_1 = M_1(X), Y_2 = M_2(X), \dots, Y_k = M_k(X)$. Let's take binary classification as the tasks that the models solve. Then the output of the model will produce one of two values, i.e. $Y_i \in \{0,1\}$ $(i = 1,2, \dots, k)$.

We introduce a model M that will integrate classification results from an arbitrary combination of models $M_1, M_2, \dots, M_k$, denoting it as follows:

$$M = M_{\alpha_1} \cap M_{\alpha_2} \cap \dots \cap M_{\alpha_p},$$

where $\alpha_1, \alpha_2, \dots, \alpha_p \in \{1,2, \dots, k\}$ $(\alpha_i \neq \alpha_j)$ are the indices of the initial pre-trained models, p is the number of models for integration, the sign $\cap$ indicates the intersection of the results of the problems solved by the models.

Let's set the task to obtain a model M without additional training, relying only on information from the pre-trained models $M_1, M_2, \dots, M_k$.

In the future, without reducing generality, we will show the basic idea of the proposed method for integrating two models, each of which receives a set of features from two parameters as input. This will illustrate how the method works on a plane.

Let two models be given M_1 and M_2, pre-trained on the input data set $X = \{(x_{1j}, x_{2j})\}_{j=1}^{m}$. We will consider the M_1 and M_2 models, each of which contains fully connected network layers.

Let us now consider the geometric representation of the M_1 and M_2 models. We will assume that each neuron of the network corresponds to a hyperplane that divides

the multidimensional feature space. In our case, for two features, it will be a straight line dividing the plane.

As a result of the intersection of the structures formed by the lines, we will get a new one or more structures. This structure will include lines from two models at once, forming a set:

$$S = \left\{L_{\alpha_1}, L_{\alpha_2}, \ldots, L_{\alpha_p}, L'_{\beta_1}, L'_{\beta_2}, \ldots, L'_{\beta_t}\right\},$$

where $\alpha_1, \alpha_2, \ldots, \alpha_p$ and $\beta_1, \beta_2, \ldots, \beta_t$ are the indices of the lines $L_{\alpha_1}, L_{\alpha_2}, \ldots, L_{\alpha_p}, L'_{\beta_1}, L'_{\beta_2}, \ldots, L'_{\beta_t}$ for the models M_1 and M_2, respectively, p and t are the number of lines for the M_1 and M_2 models that are present at the intersection.

Now, knowing the set S, it is possible to form the network architecture for the new model M. To do this, the number of neurons in the layer will correspond to the capacity of the set S, and the values of the neurons will be inherited from the parameters of the lines in S. At the same time, training for the M model is not required, which is a key feature in the proposed method.

The proposed method for integrating models trained on the same feature space and solving their specific tasks with a different network architecture consists of the following basic steps:

Step1. Extraction of weight matrices of pre-trained models to form equation parameters of the corresponding hyperplanes.

Step2. Construction of bounding hyperplanes and creation of multidimensional spatial structures for each initial model.

Step3. Search for intersections of spatial structures to create the desired solution area.

Step4. Filtering hyperplanes from all source models and including in the result only those that consist of structures corresponding to the solution area found in step 3.

Step5. Formation of a neural network for an integrated model, the values of the weight matrix of which are inherited from the hyperplane parameters obtained in step 4.

4 Results

4.1 Merging models based on test data

Let us now conduct a study of the proposed method. To do this, consider the following experiment. Figure 1a shows the initial training data as a two-dimensional set of features. Let's assume that two models have already been trained on this data. The first model A solves the classification problem “Hitting the two right squares” (blue squares), and the second model B is “Hitting the two left squares” (red squares). The straight lines obtained from the neurons of the models and dividing the feature space are shown in Figure 1b,c.

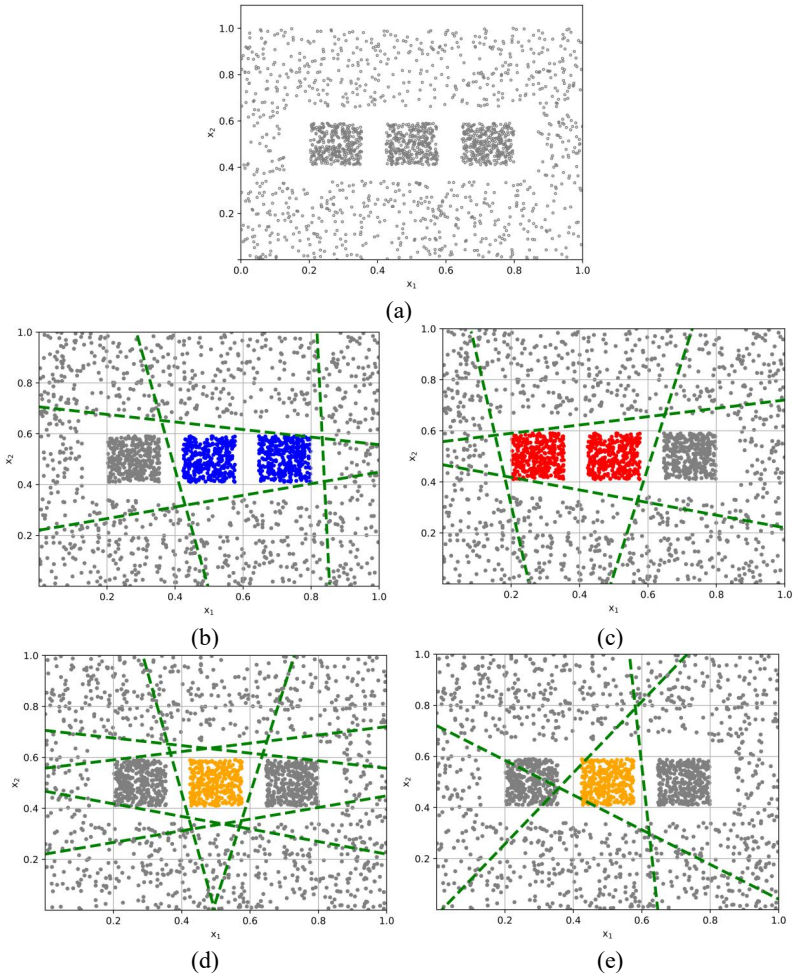

Fig. 1. Initial data and experimental results on test data: (a) initial data, (b) problem solving by model A, (c) problem solving by model B, (d) integration of solutions without training by model M, (e) integration with full training

Next, we will create an automatically integrated model M that will solve the classification problem “Hitting the middle square.” This problem can be solved in a standard way by training a model to classify data on a mean square. But we will do something else and take advantage of the fact that the middle square is a partial solution to the original problems, and the necessary solution is the intersection of solutions to the original problems. Let's define the straight lines that are present at the intersection. We inherit information for the neurons of the future model from this set of lines in the form of equation parameters. The result of the intersection is shown in Figure 1d.

Note that the required solution limits 6 straight lines, which means that this number of neurons will be in the model and this number differs from what was in the original

models. Note that the newly trained model uses only 3 bounding lines (Fig. 1e), although 6 neurons were specified.

The initial neural network models are a fully connected architecture with a single hidden layer and were trained with a classification accuracy of 99%. The new model showed a classification accuracy of 99.8%. With the classical training method, the accuracy was 93%, and the number of epochs was 500. The training time on a sample of 2000 points was 13 seconds. The model, created automatically, demonstrated a high accuracy rate in classification and the time to create its structure with the values of neurons was less than 1 second.

4.2 Merging models based on real data

The experiment used a dataset with earthquake information from the ANSS Comprehensive Earthquake Catalog (ComCat)[1], which is one of the world's largest databases of seismic events. The catalog combines information from national and regional seismological networks and contains information on recorded earthquakes around the world. The main purpose of the catalog is to store and disseminate information about earthquake parameters, including the coordinates of the epicenter, the depth of the source, the magnitude, the time of occurrence of the event and additional characteristics. The data is regularly updated as information from seismic stations becomes available and event parameters are updated.

The dataset volume is 1796 lines and includes 22 features, of which 2 features were selected for visualization on the plane: the depth and magnitude of the event. These two parameters are directly related to the measurement quality. The depth affects the propagation of seismic waves and the accuracy of determining the event, and the magnitude is related to the strength of the earthquake. During training, these parameters were normalized.

At first, two separate classification tasks were set and solved based on information about depth and magnitude. The first of them based on Model A was to search for well-predicted earthquakes. The root-mean-square (RMS) target parameter was used for this. This parameter indicates to what extent the observed arrival time of the signal corresponds to the predicted arrival time for a given location. The lower the value, the better the data match. In experiments, it was assumed that if RMS does not exceed 0.2, then it will be considered Low RMS, otherwise Normal RMS.

The second classification task based on Model B allowed us to determine a large gap between stations adjacent in azimuth based on the Gap target parameter. Earthquake epicenters, in which the azimuth gap exceeds 180 degrees, usually have a large error in determining the location and depth of occurrence. In this case, the Gap parameter will be considered Big gap, otherwise Normal gap.

The purpose of integration in the experiment is to build a model M based on trained models A and B that takes into account both aspects of quality: measurement accuracy (RMS) and geometric coverage of stations (Gap) in order to identify events with complex characteristics.

---

[1] https://earthquake.usgs.gov/data/comcat/

The geometric representation of the solution to each classification problem and their merging is shown in Figure 2.

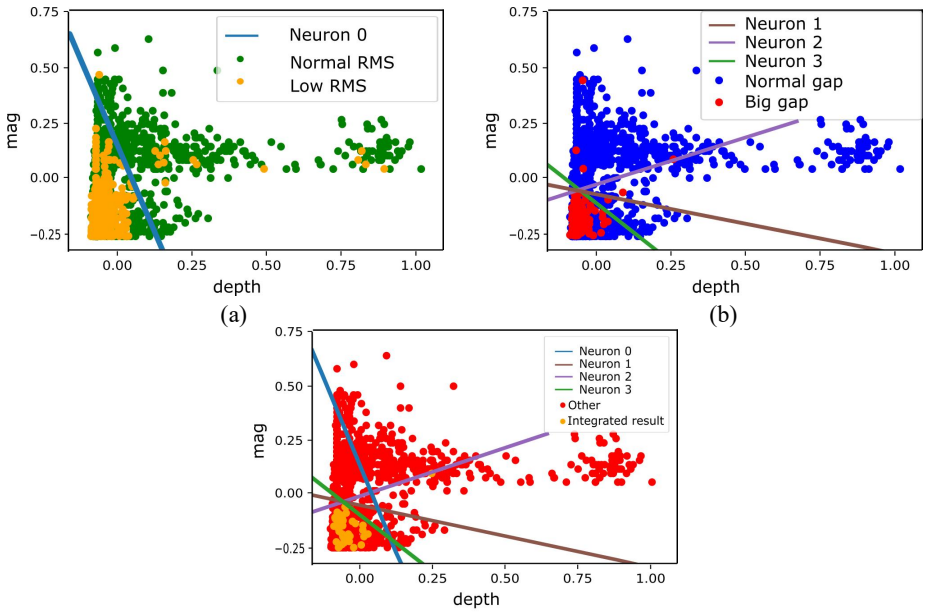

Fig. 2. Experimental results on real data: (a) problem solving by model A, (b) problem solving by model B, (c) integration of solutions without training by model M

To solve the first and second classification problems, an architecture with a fully connected neural network with one hidden layer (1 and 3 neurons, respectively) and one neuron on the output layer was used. For the integrated model, the neural network architecture was formed automatically based on geometric information from two pre-trained models and contains 4 neurons on a hidden layer.

Table 1 shows the numerical characteristics of the models.

Table 1. Numerical characteristics of accuracy indicators of classification models for individual tasks and their merging

Indicator	Model A	Model B	Merging A and B (model M)	Full training
accuracy	0.91	0.74	0.97	0.97
f1	0.95	0.85	0.98	0.98

Conclusions

The study presents a method for integrating pre-trained models without additional training of a new model. All information about the network structure and the number of neurons in the layers is inherited according to certain rules from the original models. This is achieved by using the geometric representation of neural network models and analyzing geometric structures that are formed using hyperplanes in accordance with information about neurons. During integration, a new model is formed automatically, using information from only those hyperplanes of the pre-trained models that participate in the intersection.

The proposed method allows you to integrate two or more initial models. Examples of using the method for models that solve problems with different network structures on the same set of source data are shown. The integrated model allows you to solve a more complex task that includes the functionality of several models at once. Due to the fact that the weight matrix inherits values from pre-trained models, the computational costs of creating a new model with new functional properties are significantly reduced.

Experiments have been conducted to solve classification problems on a two-dimensional set of test and real source data in order to graphically show the operation and capabilities of the method on a plane. In future research, it is planned to scale the application of the proposed method on multidimensional tabular data and images.

Acknowledgements. This study was supported by the Russian Science Foundation, project no. 26-21-20099, https://rscf.ru/project/26-21-20099/.

References

1. Akiba, T., Shing, M., Tang, Y., Sun, Q.: Evolutionary optimization of model merging recipes. Nature Machine Intelligence 7, 195–204 (2025). https://doi.org/10.1038/s42256-024-00975-8
2. Crawshaw, M.: Multi-task learning with deep neural networks: a survey. arXiv preprint arXiv:2009.09796 (2020). https://doi.org/10.48550/arXiv.2009.09796
3. Fedus, W., Zoph, B., Shazeer, N.: Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. Journal of Machine Learning Research 23 (2022). https://doi.org/10.48550/arXiv.2101.03961
4. Gou, J., Yu, B., Maybank, S.J., Tao, D.: Knowledge distillation: a survey. International Journal of Computer Vision 129, 1789–1819 (2021). https://doi.org/10.48550/arXiv.2006.05525
5. Idelbayev, Y., Carreira-Perpinan, M.: Low-rank compression of neural nets: learning the rank of each layer. In: CVPR 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020), pp. 8049–8059. https://doi.org/10.1109/CVPR42600.2020.00807
6. Ilharco, G., Ribeiro, M.T., Wortsman, M., Gururangan, S.: Editing models with task arithmetic. arXiv preprint arXiv:2212.04089 (2023). https://doi.org/10.48550/arXiv.2212.04089

7. Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D.: Averaging weights leads to wider optima and better generalization. In: 34th Conference on Uncertainty in Artificial Intelligence (2018), pp. 876–885. https://doi.org/10.48550/arXiv.1803.05407
8. Matena, M.S., Raffel, C.A.: Merging models with fisher-weighted averaging. Advances in Neural Information Processing Systems 35, 17703–17716 (2022). https://doi.org/10.48550/arXiv.2111.09832
9. Oseledets, I.: Tensor-train decomposition. SIAM Journal on Scientific Computing 33, 2295–2317 (2011). https://doi.org/10.1137/090752286
10. Wortsman, M., Ilharco, G., Gadre, S., Roelofs, R.: Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. arXiv preprint arXiv:2203.05482 (2023). https://doi.org/10.48550/arXiv.2203.05482
11. Yang, Y., Lv, H., Chen, N.: A Survey on ensemble learning under the era of deep learning. Artificial Intelligence Review 56, 5545–5589 (2023). https://doi.org/10.1007/s10462-022-10283-5
12. Zhou, K., Yang, Y., Qiao, Y., Xiang, T.: Domain adaptive ensemble learning. IEEE Transactions on Image Processing 30, 8008–8018 (2021). https://doi.org/10.1109/TIP.2021.3112012