EPJ Web of Conferences **173**, 01009 (2018) https://doi.org/10.1051/epjconf/201817301009
Mathematical Modeling and Computational Physics 2017

Two-Stage Approach to Image Classification by Deep Neural Networks

Gennady Ososkov[1,⋆] and *Pavel* Goncharov[2,⋆⋆]

[1] *Laboratory of Information Technologies, Joint Institute for Nuclear Research, 141980 Dubna, Moscow region*
[2] *Sukhoi State Technical University of Gomel, Gomel, Belarus*

Abstract. The paper demonstrates the advantages of the deep learning networks over the ordinary neural networks on their comparative applications to image classifying. An autoassociative neural network is used as a standalone autoencoder for prior extraction of the most informative features of the input data for neural networks to be compared further as classifiers. The main efforts to deal with deep learning networks are spent for a quite painstaking work of optimizing the structures of those networks and their components, as activation functions, weights, as well as the procedures of minimizing their loss function to improve their performances and speed up their learning time. It is also shown that the deep autoencoders develop the remarkable ability for denoising images after being specially trained. Convolutional Neural Networks are also used to solve a quite actual problem of protein genetics on the example of the durum wheat classification. Results of our comparative study demonstrate the undoubted advantage of the deep networks, as well as the denoising power of the autoencoders. In our work we use both GPU and cloud services to speed up the calculations.

1 Introduction

The physicists have accumulated a solid experience on the use of artificial neural networks (ANN) to the solution of many high energy and nuclear physics (HENP) problems [1]. An important role in this experience plays the remarkable privilege to generate training samples of any arbitrary length needed for sufficient ANN training by Monte Carlo simulations of advanced theoretical models. However such comfortable circumstances largely distracted the physicists from the new deep learning neural network (NN) paradigm proposed in the newcoming challenging "Big Data" era asking for intensive data management. Since the problems to be solved are now getting more and more complicated, the ANN architecture needs to become deeper by adding more hidden layers for better parameterization and more adequate to problem modelling, but the ANNs mainly used by physicists are now obsolete and quite rarely can show the needed efficiency.

It is worth mentioning the series of researches devoted to the deep learning concepts published during the last decade [2, 3] in which the authors dealing with neural networks were inspired by the knowledge about the deep hierarchical structure of the mammals visual cortex [4]. There were

⋆e-mail: ososkov@jinr.ru
⋆⋆e-mail: kaliostrogoblin3@gmail.com

proposed two ways of making deeper the neural network hierarchy. The first proposal was to add more hidden layers to a multilayer perceptron with its supervised learning paradigm [5], the second to change this paradigm by including into the neural network a hierarchy of additional layers but with different, unsupervised, learning. The second way implements an analog of the animal visual cortex and was proposed by G. Hinton [2] as a combination of several stacked layers of recurrent NN of Boltzmann machine type realizing unsupervised representations of input data with the last supervised layer completing a classification task. This type of deep networks was named Deep Belief Networks (DBN) referring one to the stochastic NNs and the probability distribution of their neurons. These stacked layers are restricted Boltzmann machines (RBM) [3] that can learn themselves a probability distribution over its set of inputs. The quite slow evolution of RBM layers and Hinton's trick to speed it up by the so-called "contrastive divergence" approach are described in [2, 6]. Nevertheless, due to the notable DBN disadvantages, there is now another type of deep NNs which is much wider applicable for the image classification. These are the Convolutional neural networks (CNN), which feed-forward many layer NN with the neurons in a layer connected only to a small region of the layer before it. The main CNN idea following the vision processing in living organisms is to convert every input image by scanning its small sub-regions and filtering them on a stack of convolutional and pooling layers into a set of elementary sub-images which correspond well to the original image, but are not sensitive to different orientations, scales or lighting. Then being trained by a labeled sample with the back-prop method, CNN obtains all weights of convolutional filters and constructs a library of image elements linked to different image classes (see exhaustive CNN description in [7]).

One more significant deep learning approach is to build out a deep autoassociative neural network of an autoencoder type. Autoencoder algorithms consist of training a neural network to produce an output that is identical to the input, but having fewer nodes in the middle hidden layer than in the input forming a "bottleneck". In this case one has built a tool for compressing data to a smaller number of the most informative features [8].

There are many other types of deep learning ANNs with quite different structures such as the Long Short Term Memory (LSTM), the Gated Recurrent Unit (GRU) networks, the Reinforcement Learning (RL) networks (see, for example, J. Schmidhuber's survey [9]. However we are not going to survey deep learning networks. Our goal is to illustrate the advantages of the deep learning approaches over the ordinary neural network on their comparative applications to image classifying. Besides we demonstrate the power of CNN on the example of its application to an actual problem of protein genetics.

2 Deep Learning study at the JINR

In the Laboratory of Information Technologies of JINR four neural network based programs for the image recognition were developed:

1. An autoassociative NN realizing the nonlinear principal component analysis (NLPCA) intended for the significant compression of the input data;

Three NN involve a special optimization of their structures and parameters as image classifiers:

2. Perceptron with one hidden layer;

3. Deep Belief Network;

4. Deep Neural Network (DNN) with initialization of normalized weights.

A study was recently fulfilled to compare the efficiency of these classifiers by testing them on two well-known datasets:

– the MNIST handwritten digit database [10] (70,000 handwritten digit images);

– the FERET face database [11] (40 people with 10 photos for each).

2.1 Data compression by autoencoders

An autoencoder (AE) accomplishes unsupervised learning for data compression to a smaller number of the most informative features by realizing the nonlinear principal component analysis (NLPCA). It was proposed by M. Kramer [8] as an autoassociative neural network with three hidden layers including the bottleneck layer in the middle. AE training is performed with sigmoidal activations by self-supervised backpropagation minimizing the sum of the squared errors (SSE) as the loss function.

We prefer to use the AE as a standalone program due to the following reasons:

– it gives us the same set of compressed data as the input vectors of the three neural networks to be compared further as classifiers;

– AE realized as an autoassociative neural network needs to optimize its structure, tune parameters of the activation function, choose a proper normalization of input data and obtain a fast convergence of the training process;

– AE provides highly effective data compression, which is getting especially important in the Big Data era.

In our study we considerably modified the Kramer scheme in order to improve its efficiency and speed up the training time by solving convergence problems. Our AE improvement consists in:

1. replacing sigmoidal activations by Leaky Rectified Linear (LReLU) activation [12]

$$f(x) = \begin{cases} x, & \text{if } x \geq 0, \\ \frac{x}{a}, & \text{otherwise,} \end{cases}$$

   with a carefully chosen specific parameter a different for each data set;

2. centering the training data by subtracting mean data value from each sample and then specific input normalization (MinMax was chosen);

3. applying tied weights;

4. stochastic gradient descent (SGD) with Adam optimization [13].

The idea of tied weights means the encode and decode weight matrices are simply transposes of each other, what gives less parameters to be optimized and stored and, therefore, faster training.

We use the Adaptive Moment Estimation (Adam) method for efficient SGD which requires first-order gradients only. Adam computes adaptive learning rates for each parameter and also keeps an exponentially decaying average of the past gradients, similar to the momentum of inertia [13].

2.2 Deep and shallow Neural Networks

Deep belief network. As already mentioned above, the deep belief networks (DBN) were proposed as probabilistic generative models which are composed of multiple layers of stochastic, latent variables [2]. DBN runs a pretraining procedure, using latent restricted Boltzmann machines (RBM) layers with a final layer of variables that performs discriminative fine-tuning of the desired outputs by backpropagating the error derivatives. Since there is a rich literature devoted to the detailed DBN description and for the sake of brevity we shall not enter into the details on how it works, the reader may find them, e.g., in [2, 3].

Deep neural network. As mentioned above, the training time of the DBNs is always too long, but it is possible to use another type of classifiers the convergence of which is much faster, namely Deep Neural Networks (DNN) with the initialization performed by normalized weights [14]

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right], \tag{1}$$

where $U[-a, a]$ is the uniform distribution in the interval $(-a, a)$ and the n is the size of the previous layer (the number of W columns).

The DNN works much faster than DBN, because it avoids the pretraining procedure for the RBM hidden layers.

Shallow neural network.[1] For our comparative study we use also a feed-forward perceptron with one hidden layer and sigmoid activations as shallow neural network estimator. The input of the perceptron is an array of eigenvectors of dimension 64, which is received from the latent representation of the input images of the bottleneck layer of the autoencoder.

2.3 Selection of optimal hyperparameters for the network models

Improving the autoencoder. We need to choose first the optimal number of hidden neurons for the best input data recovery by our autoencoder. After calculating the dependences of the classification efficiency and the training time on different numbers of neurons in the bottleneck layer we conclude that the optimal number is 64. It guarantees 95% classification efficiency, while a larger number of neurons would result in unnecessary model complexity and cause overfitting.

Then we test how the applications of the tied weights and SGD Adam methods [13] would influence the convergence time. Test results are shown in fig. 1. The prefix "Tied/Untied" means the type of network weights: tied or separate weights, respectively. "GD" stands for the gradient descent abbreviation. As we can see in fig. 1, the Adam optimization method significantly reduces the convergence time, while the use of tied weights although do not benefit much for speeding up, reduce twice the number of network parameters what is important.

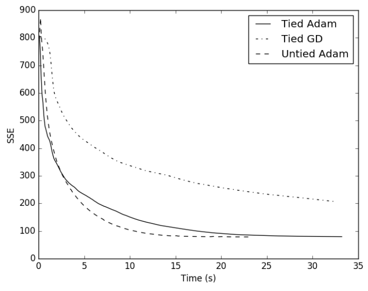

Figure 1. Decrease of the AE loss function in time in terms of the optimization method and the type of weights

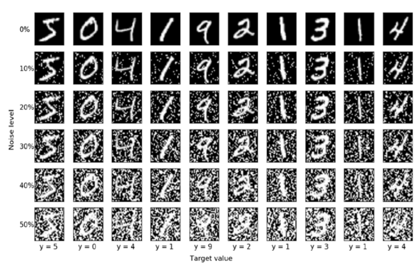

Figure 2. Examples of noisy images with different percentage of noise

[1]The concepts of shallow and deep NN are widely used in the literature (see, for example https://arxiv.org/pdf/1608.03287.pdf).

From similar considerations, we find that the number of neurons in both the mapping-demapping layers should not exceed 256. Eventually, the dimensions of the 5 autoencoder layers are chosen as **(m,256,64,256,m)**, where **m** is the length of the input vector.

In our further work with shallow, deep and deep belief neural networks we also use SGD Adam optimization.

Optimizations of deep classifiers and obtained results. We use grid search cross validation ([15]) for tuning DNN and DBN architectures. The training database is split in two parts: 80% as train set and another 20% as cross-validation set. Then we divide the cross-validation data into 3-fold batches, calculate the score during the fit of an estimator on each fold and take an average score value. After fitting the estimator on a parameter grid, we choose the parameters such as to maximize the cross-validation score, for which the network classification efficiency is selected.

All this gives two hidden layers for both the DNN and DBN while the optimal numbers of the hidden neurons per 1st and 2d layers are 300 and 100 respectively.

2.4 Results of testing by three classifiers

The images of both data sets rescaled by the autoencoder to 64-dimensional eigenvectors were classified by three of our classifiers with results presented in table 1.

The results look quite satisfactory. The deep learning techniques give the 100% classification efficiency on the faces database and the 92–98 percent efficiency on the handwritten digits dataset. Although it is not a problem for the humans to distinguish various faces, the DBN can make better predictions, even in the case of digits, than the humans can.

Table 1. Comparison of the different estimators

	Perceptron	DNN	DBN
MNIST	0.7977	0.9285	0.9839
FERET	0.8750	1.0000	1.0000

2.5 Autoencoder application for image denoising

Generally, the real image examples are not so clean – they can have a lot of noise. We test a remarkable feature of our autoencoder model to recover the input images with different percent of noise imposed on them. A denoising procedure takes noisy data as input and outputs this into data, from which the noise is removed.

Our classifiers are trained on non-corrupted images. The denoising autoencoder network (DAEN) is trained with 20% noise added to input images. Then the classifiers are tested on six datasets with noise levels in the range from 0% to 50% (see fig. 2) compressed previously by DAEN.

As the results in table 2 show, the efficiency keeps well even on the data with 30% noise level. Deep Belief Network performs better classification results again, but under noise increasing the DNN outperforms the DBN. Both deep classifier efficiencies are notably higher than those of the perceptron with one hidden layer.

3 Convolutional Neural Networks for image recognition

The CNN architecture involves a sequence of layers the each layer of which transforms one volume of activations to another through a filter which is a differentiable function.

Table 2. Comparison of classifying efficiency for noisy images in the case of applying LReLU activations and MinMax normalization of input

	FERET			MNIST		
Noise(%)	Perceptron	Deep NN	Deep Belief	Perceptron	Deep NN	Deep Belief
0	0.9166	0.9867	1.0000	0.7449	0.9182	0.9861
10	0.9083	0.9867	0.9833	0.6901	0.8779	0.9644
20	0.9000	0.9750	0.9666	0.6115	0.7832	0.8794
30	0.8750	0.9583	0.9583	0.5001	0.6443	0.6027
40	0.8500	0.9667	0.9499	0.4100	0.5202	0.3863
50	0.8000	0.9583	0.9333	0.3036	0.3973	0.2283

There are three main types of layers to build CNN architectures: the Convolutional layer, the Pooling (subsampling) Layer, and the Fully-Connected Layer (just MLP with backprop). There are also RELU (rectified linear unit) layers performing the max(0,x). Each Layer accepts an input 3D volume (x,y,RGB color) and transforms it to an output 3D volume. To construct all filters of convolutional layers our CNN must be trained by a labeled sample with the back-prop method [7].

We describe now the CNN use for the solution of an actual problem of the protein genetics, namely, the durum wheat classification by the labeled sample of electrophoretic spectra of gliadin (alcohol soluble protein) [16]. Each electrophoretogram becames after its digitalization a densitogram spectrum consisting of about 4000 pixels, as it shown in fig. 3. It can be considered as a simple genetic formula, which allows for a qualified expert to classify any spectrum to its corresponding protein.

The real size of the training sample was 3225 EF-densitograms for 50 sorts preliminarily classified by experts for each of wheat sorts. Having 4000 input values means that one would have a neural network with more than 10 millions of weights or equations to solve by the error back propagation method! There were several attempts to extract the most informative features from input data to reduce the data cardinally [16, 17]. Although it succeeded to reduce the input by more than an order of magnitude, the classification by multilayer perceptrons with one hidden layer failed to classify with an admissible efficiency more than 10 sorts of wheat [17].

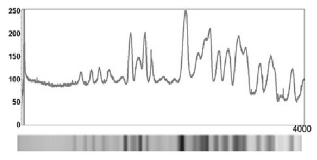

Figure 3. Example of the electrophoretogram strip (below) with its dencitogram (above)

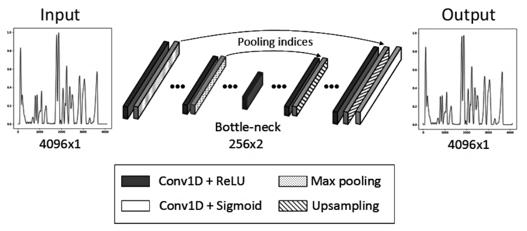

Figure 4. Convolutional Autoencoder scheme

To solve the problem of classifying 50 wheat sorts we proposed to use a two-step algorithm based on data-compression by an autoencoder at the first step followed then at the second step by applying either DNN or convolutional NN. To make a justified choice of AE we elaborate also a convolutional one (CAE) (see fig. 4).

In our study we have had to develop two versions of autoencoders and two versions of classifiers in order to compare them by the higher efficiency criterion with reasonable time consuming. We applied the improvements elaborated above for AE such as the tied weights and stochastic gradient descent (SGD) with Adam optimization. The total number of CAE parameters is 315 what is 5 orders of magnitude less than for DNN and gives a significant gain in the memory need to store AE as well as in training speed. The parameters of the deep and convolutional NN are shown in table 3.

Table 3. Parameters of the deep and convolutional NN

Parameter	CNN	DNN
Model size (MB)	3	66
Total epochs number	150	400
Epoch time (s)	1	0.2
Training time (s)	69	84

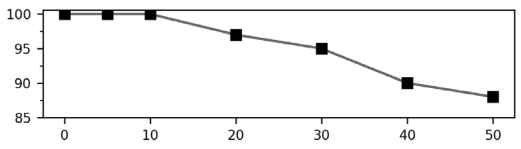

Figure 5. Classification efficiency vs sort numbers (the same for CNN and DNN)

The results of the protein classification efficiency are the same for both the CNN and DNN (fig. 5). One can see the advantages of the convolutional network as compared to DNN. With the same efficiency CNN takes 22 times less memory, while its training time is faster. These results demonstrate the overwhelming superiority of the deep approach in the solution of the protein classification problem.

4 Deep training with GPU acceleration in Cloud

In our study we speed up significantly our deep network calculations by using the multicore computational systems and multiprocessor graphics cards or GPUs via the cloud service facilities provided at the JINR heterogeneous computing cluster HybriLIT [18]. Although our classification models were developed on a notebook Dell Vostro 5480 (CPU: Intel Core i3-4500U @ 1.70 GHz (4 cores); RAM: 4096 MB; Intel HD Graphics 4400) we use HybriLIT cloud sevice and two free libraries to carry out deep training with GPU acceleration in the cloud with Nvidia Tesla K80: TensorFlow [19] and Keras [20].

These services greatly facilitate the deep learning performance. For the solution of a same problem, one training epoch for the convolutional autoencoder takes on the notebook 70–80 sec, but operating in the HybriLIT using TensorFlow and Keras it takes only 1 sec for one training epoch.

5 Conclusion

In our work, we propose a deep learning model for the image classification, which consists of two neural networks: an autoencoder for preliminary input compression and an estimator for classifying images. A comparative study of three neural classifiers: a perceptron with one hidden layer, DBN and DNN was accomplished on images of two famous benchmark data sets MNIST and FERET being previously compressed by the autoencoder.

A comparative study of three neural classifiers: a perceptron with one hidden layer, DBN and DNN was accomplished on images of both data sets being previously compressed by the autoencoder. The comparison of the investigated estimators shows that DBN provides the highest classification efficiency, but DNN accomplishes a faster training time having competitive efficiency. Both the DBN and DNN perform much better than the perceptron with one hidden layer.

It is important to note that the main efforts to deal with the deep learning networks were spent to quite a painstaking work for optimizing the structures of those networks and their components, as activation functions, weights, as well as the procedures of minimizing their loss function to improve their performances. We have to make a special study to find the optimal numbers of layers and hidden neurons, to apply tied weights and SGD Adam method, to choose the most suitable activation functions and weight normalization in order to improve the performance of all used neural nets and speed up their learning time. Additionally we observe such a remarkable feature of the deep autoencoders as their ability for denoising images after being trained on an image set corrupted in a special way. The denoising results of such autoencoder classification efficiency on FERET images are really striking: images corrupted with 50 percent of noise were classified with efficiency better than 90 percent.

Besides we extend our two stage deep learning model for the image classification to Convolutional Neural Networks. It was used to solve a quite actual problem of protein genetics on the example of the durum wheat classification with quite successful results.

In our study we speed up significantly our deep network calculations by using Tensorflow and Keras libraries and multicore computational systems and multiprocessor graphics cards or GPUs via the cloud service facilities provided at the JINR heterogeneous computing cluster HybriLIT.

References

[1] I. Kisel, V. Neskoromnyi, and G. Ososkov, Phys. Part. Nucl. **24**, 6, 657–676 (1993)
[2] G. E. Hinton and R. Salakhutdinov, Science **313**, 504–507 (2006)
[3] Y. Bengio, *Learning Deep Architectures for AI* (Now Publishers Inc, 2009) p. 144
[4] P. Lennie, Current biology **13**, (6), 493–497 (2003)
[5] Y. LeCun, L. Bottou, G. Orr, and K.-R. Müller, in *Neural Networks: Tricks of the Trade*, 9–50 (1998)
[6] M. Carreira-Perpinan, Aistats **10**, 33–40 (2005)
[7] *Web CNN description facility*, http://cs231n.github.io/convolutional-networks
[8] M. Kramer, AIChE Journal **37**, 2, 161–310 (1991)
[9] J. Schmidhuber, Neural Networks **61**, 85–117 (2015)
[10] *MNIST*, https://www.nist.gov/sites/default/files/documents/srd/nistsd19.pdf
[11] A. Phillips, H. Moon, P. Rauss, and S. Rizvi, IEEE Transactions on Pattern Analysis and Machine Intelligence **22**, 10 (2000)
[12] B. Xu, N. Wang, T. Chen, and M. Li, https://arxiv.org/abs/1505.00853
[13] D. Kingma and J. L. Ba, *Adam*, https://arxiv.org/abs/1412.6980
[14] X. Glorot and Y. Bengio, Aistats **9**, 249–256 (2010)
[15] M. Stone, Journal of the royal statistical society. Series B (Methodological), 111–147 (1974)
[16] A. Ruanet, A. Kudriavtsev, and S. Dadashev, Russian Journal of Genetics **37**, 10, 1207–1209 (2001)
[17] G.A. Ososkov and D.A. Baranov, Bulletin of PFUR Series Mathematics. Information Sciences. Physics. **3**, 142–148 (2010)
[18] *Web-portal of HybriLIT JINR computation facility*, http://hybrilit.jinr.ru
[19] *Speed up training with GPU-accelerated TensorFlow*, http://www.nvidia.com/object/gpu-accelerated-applications-tensorflow-benchmarks.html
[20] *Keras Python Deep Learning library*, https://keras.io/getting-started/sequential-model-guide/