2nd International School on
Heterogeneous Computing Infrastructure- NEC'2017
Montenegro, Budva, Becici, 25-29 September, 2017

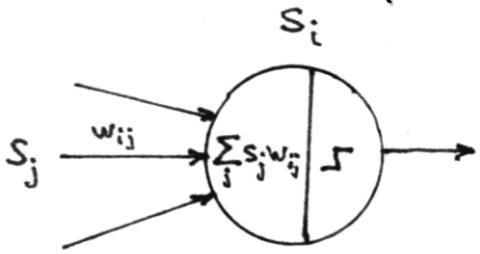
Введение в глубокие нейронные сети

Ососков Г. А., Гончаров П.В.,
Лаборатория Информационных Технологий,
Объединенного института ядерных исследований
email: ososkov@jinr.ru

История:

1.Ramón y Cajal & Camillo Golgi Нобелевская премия 1907 за теорию нейронных систем

2. MacKallok-Pitts (1943) математическая модель биологи`ческого нейрона



общий входной сигнал, поступающий
на i-й нейрон от всех остальных нейронов

$$h_i = \Sigma_j w_{ij} s_j$$

Синаптический вес

3.D.Hebb (1949) правило обучения нейронов $w_{ij}(t+1) - w_{ij}(t) = \Delta w_{ij} = \eta s_i(t) s_j(t)$

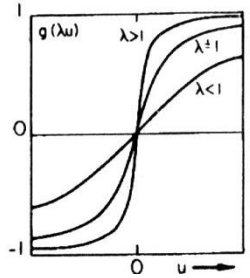
Что получилось

Архитектура искусственных нейронных сетей (ИНС). ИНС состоит из *нейронов*, – простых логических устройств, характеризуемых:

1. уровнем активации;
2. топологией связи друг с другом;
3. мерой взаимодействия с другими нейронами, называемой синаптической силой связи или *весом*.

Функция активации

$$g(u) = \frac{1}{1+e^{-\lambda u}}$$

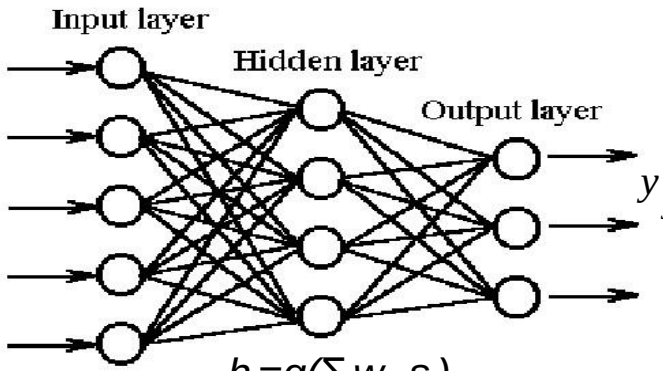


Веса этих связей различны и должны определяться в зависимости от решаемой задачи.

Вся система состоит из большого числа нейронов. Результат работы ИНС малочувствителен к характеристикам конкретного нейрона.

ИНС, применяемые в экспериментальной физике

1. Прямоточная ИНС или многослойный персептрон (МСП,RBF-сети).



$y_j = f(\sum_k w_{kj} h_k)$

$h_i = g(\Sigma_j w_{ij}\, s_j)$

Обучение с учителем.

Цель обучения – **определить веса** так, чтобы обученная сеть решала задачу распознавания или классификации.

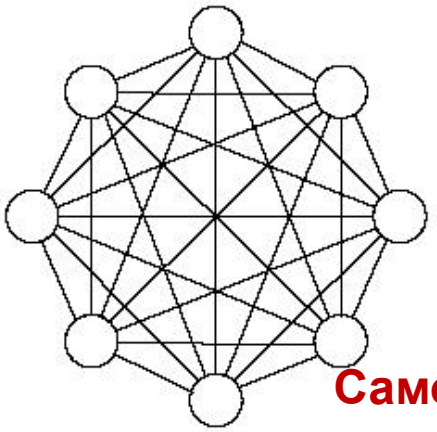
Этапы применения МСП:

1. **Создать обучающую выборку как набор пар (X_i,Z_i), где X_i – входное значение, Z_i – его целевое значение.**
2. **Обучить МСП методом обратного распространения ошибки**

$$E=\Sigma_m \Sigma_{ij} (y_i^{(m)} - z_i^{(m)})^2 \rightarrow min_{\{wij\}}$$

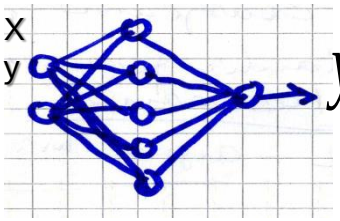
3. **Протестировать МСП на тестовой выборке**
4. **Обученная сеть реализуется как программа или нейрочип для очень быстрого применения**

2. **Полносвязная ИНС** (сеть Хопфилда)



Самообучение

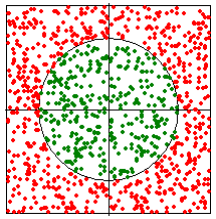
Тайны обучения. Что внутри черного ящика ИНС?



$h_i=g(\Sigma_j w_{ij}\, s_j)$

$$y_j = f\left(\sum_k w_{kj} h_k\right)$$

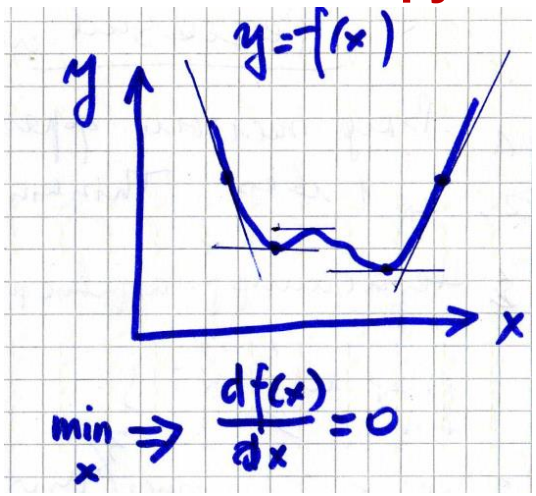
Всего 15 весов



Пример. Классификация с помощью МСП модель обучающей выборки: на вход сети подают по 3 числа (X,Y, Z) : координаты точки X,Y и признак Z (1 - внутри круга или 0 – вне его).

Чтобы обучить МСП методом обратного распространения ошибки, надо минимизировать по всем весам функцию ошибки сети: $E=\Sigma_m\Sigma_{ij}\,(y_i^{(m)} - z_i^{(m)})^2 \rightarrow min_{\{wij\}}$

Минимизация функции ошибки сети *E*



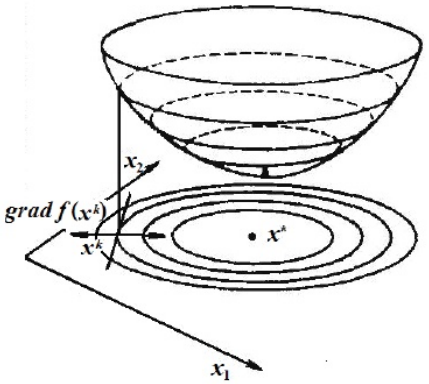
Т.о. надо решать систему из 15 уравнений $\frac{\partial E}{\partial w_{ij}} = 0$

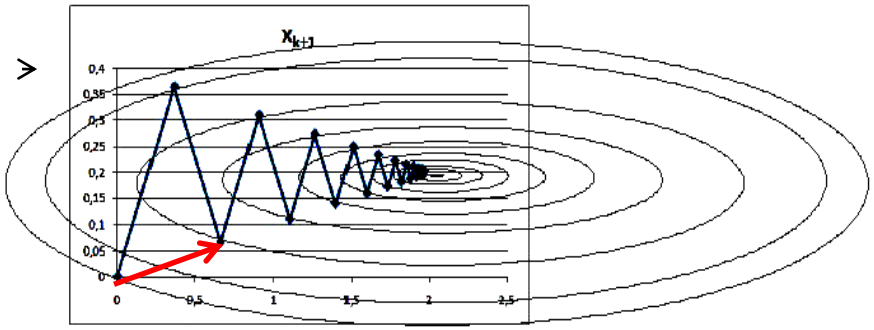
с 15 неизвестными значениями весов w_{ij} w_{jk}.

Обычный метод решения - антиградиентный спуск – хорошо работает для унимодальных функций при удачно выбранном начальном приближении, однако для функции E(w_{ij} ,w_{jk}) характерно овражное строение и наличие многих локальных минимумов.

Поэтому в программах, реализующих ИНС, используется вся «тяжелая артиллерия» вычислительных методов

Тайны обучения ИНС – вычислительные приемы



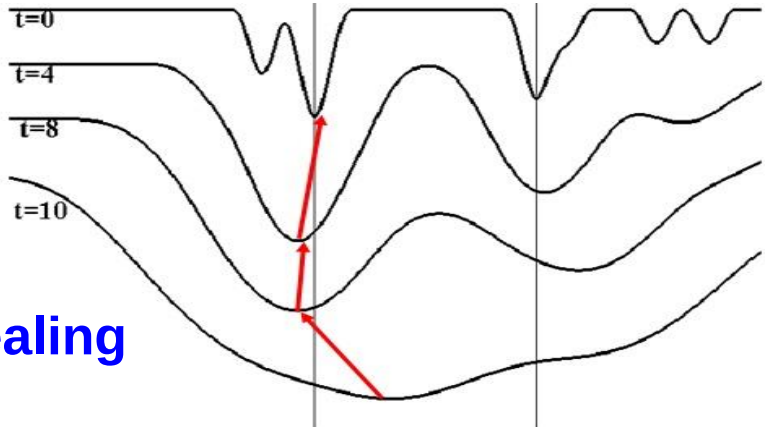


Выход – метод параллельных касательных «партан»

1. Наискорейший спуск для обычной и «овражной» поверхностей

2. Как выбраться из ложного локального минимума $\mathbf{E(w_{ij}, w_{jk})}$

$$g(u) = \frac{1}{1 + e^{-\lambda u}}$$ λ=1/t E=E(t,W)



Метод симуляции отжига – **simulated annealing**

3. Метод стохастического градиента (stochastic gradient descent - SGD)

Почему ИНС востребованы в ФВЭ

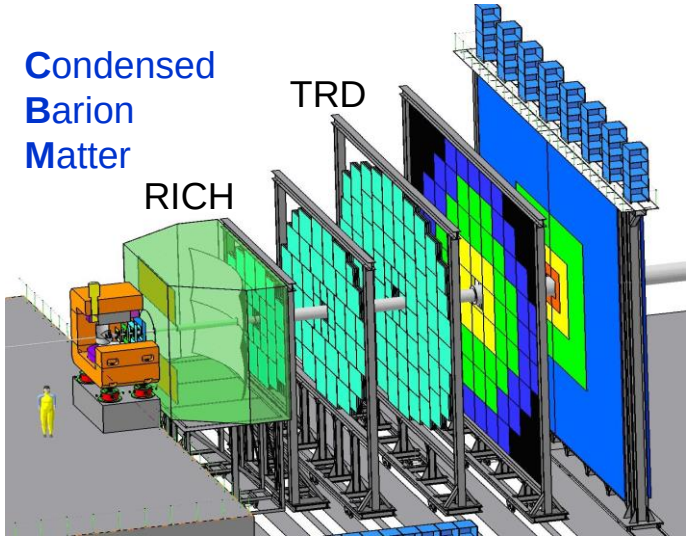
Artificial neural networks (ANN) with their ability for learning and self-learning are effective ML tools, so physicists accumulated a quite solid experience in various ANN applications in many HENP experiments for the recognition of charged particle tracks, Cherenkov rings, physical hypotheses testing, and image processing .

In particular, - MLP's are quite popular in physics, moreover namely physicists wrote in 80-ties one of the first NN programing packages – Jetnet. They were also among first neuro-chip users.

Main reasons were:

- The possibility to generate training samples of any arbitrary needed length by Monte Carlo on the basis of some new physical model implemented in GEANT simulation program
- neuro-chip appearance on the market at that time which make feasible implementing a trained NN, as a hardware for the very fast triggering and other NN application.
- the handy MLP realization with the error back-propagation algorithm for its training in TMVA – the Toolkit for Multivariate Data Analysis with ROOT

Пример изображений из эксперимента CBM

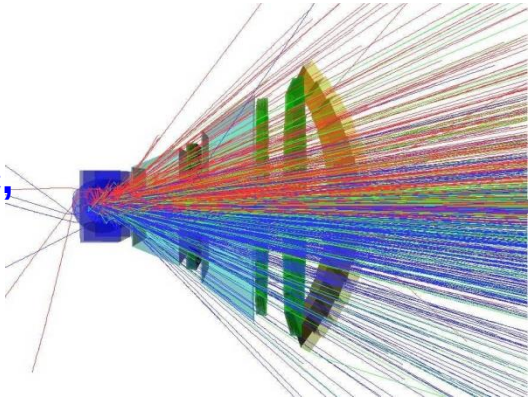


schematic view of the CBM setup

1. CBM experiment (Germany, GSI, to be running in 2018)

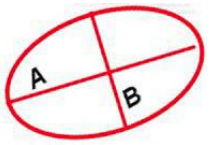
Data rate: 10^7 events per sec,
~1000 tracks per event
~100 numbers per track
Total: terabytes/sec !

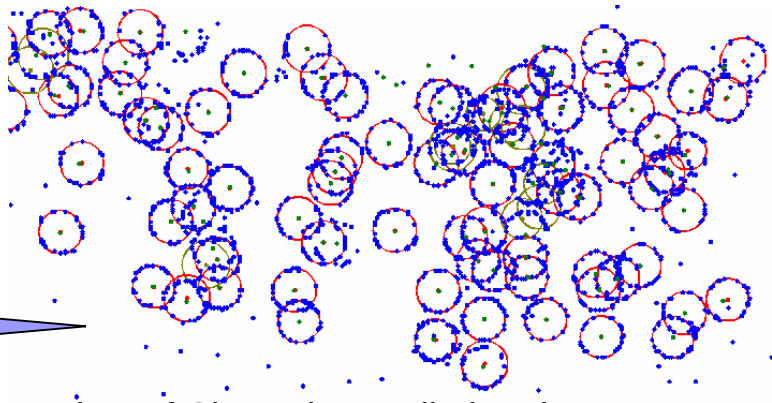
RICH - Cherenkov radiation detector



simulated event of the central Au+Au collision in the vertex detector

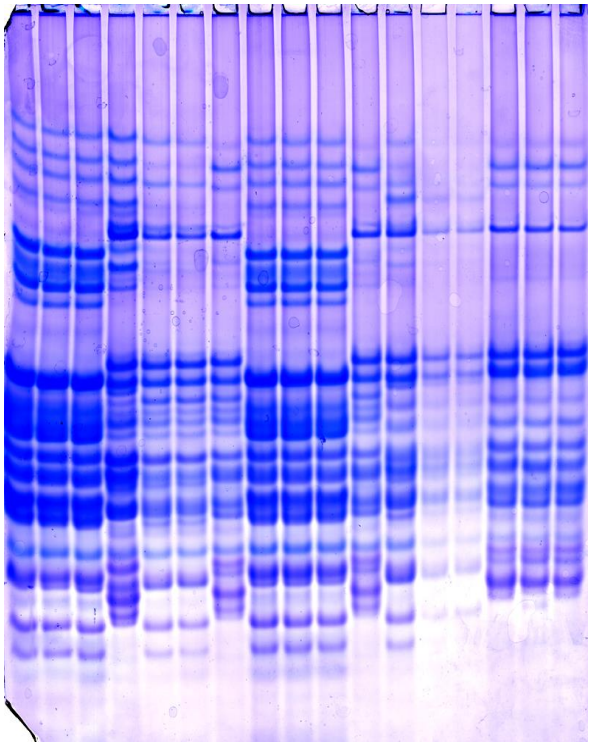
Problems are to recognize all of these tracks and RICH rings and evaluate their parameters despite of their overlapping, noise and optical shape distortions



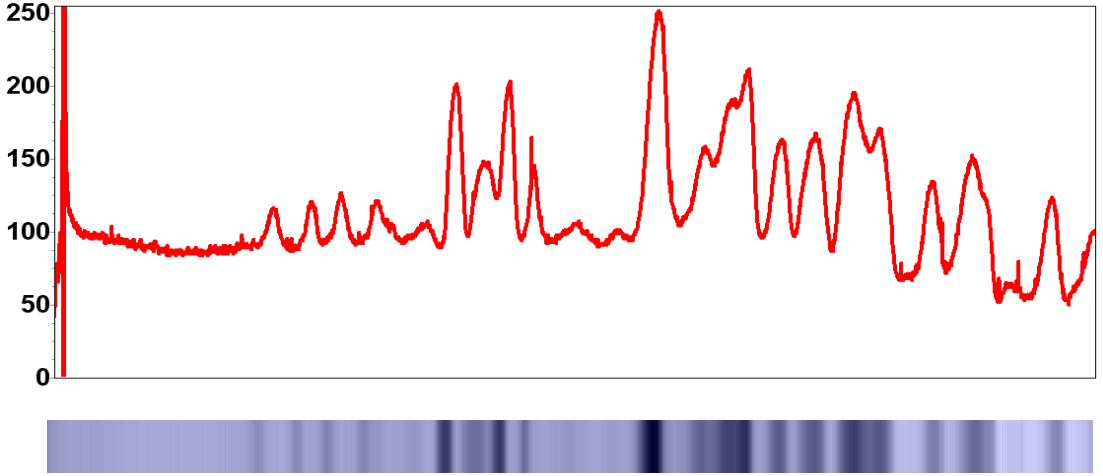


view of Cherenkov radiation rings registered by the CBM RICH detector

2. Пример изображения из генетики протеинов



The genetics of gliadin (alcohol soluble protein) have been studied in detail by a special gel electrophoresis. Each electrophoretogram strip after its digitalization became a densitogram spectrum consisting of about 4000 pixels.



Electrophoregram example with 17 wheat cultivar

It can be considered as a simple genetic formula, which allows for a qualified expert to classify any spectrum to its corresponding protein.

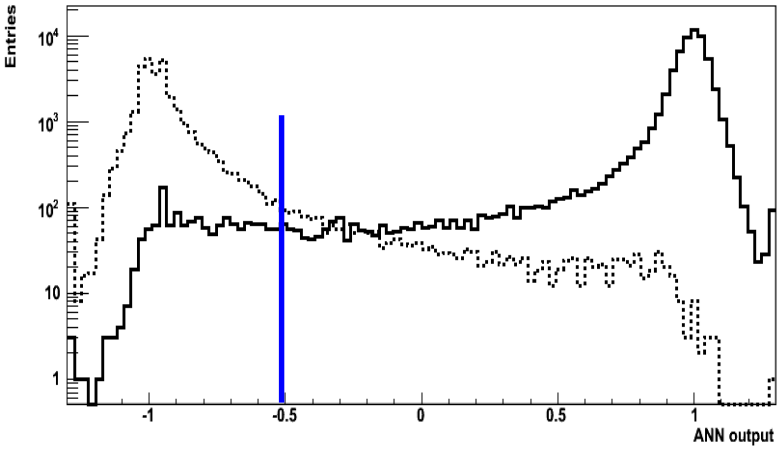
Such classifying procedure is of great importance in radiobiology and, especially, in agriculture
Therefore it is to be automatized.

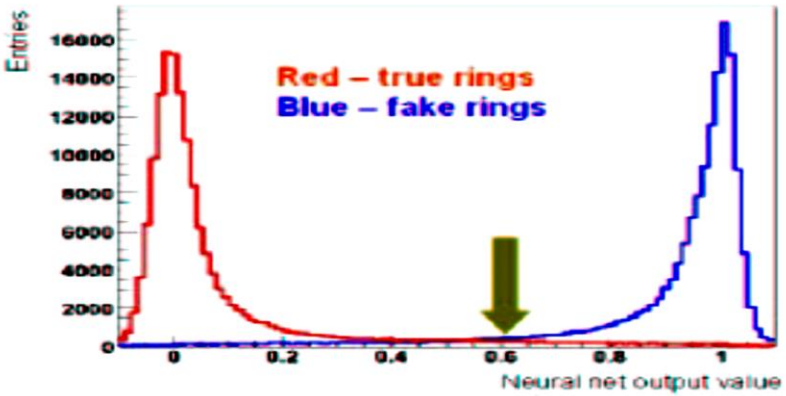
Применения МСП.

1. Классификация колец черенковского излучения

The study has been made to select the most informative ring features needed to (1) **distinguish between good and fake rings** and to (2) **identify electrons.**
Ten of them have been chosen to be input to ANNs, they are:

1. Number of points in the found ring
2. Its distance to the nearest track
3. The biggest angle between two neighbouring points
4. Number of points in the narrow corridor surrounding ring
5. Radial ring position on the photodetector plane
6. χ^2 of ellipse fitting
7. Both ellipse half-axes (A and B)
8. angle φ of the ellipse inclination to abscissa
9. track azimuth
10. track momentum





Two samples with 3000 e (+1)and 3000 π (-1) have been simulated to train NN. Electron recognition efficiency was fixed on 90% Probabilities of the 1-st kind error 0.018 and the 2-d kind errors 0.0004 correspondingly were obtained

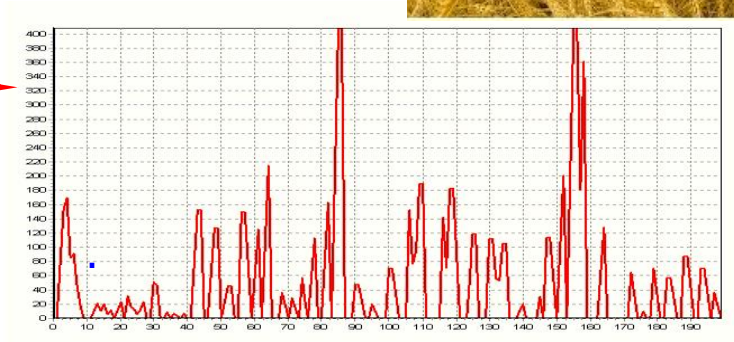
Применения МСП.

2. Genetics of proteins. EF-densitogram classifying by MLP

Important real case, - durum wheat classification

The real size of the training sample is 3225 EF-densitogramms for 50 sorts preliminarily classified by experts for each of wheat sorts

Curse of dimensionality problem



Input: 4000 pixels

Output: 50 sorts to be classified

ANN dimension D=4000*256+256*50>10^6,

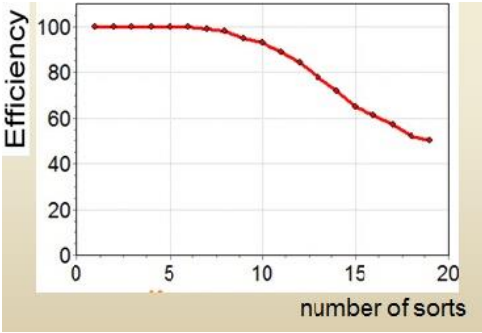
i.e millions of weights or equations to solve by the error back propagation method!

A cardinal reduction of input data preserving essential information is needed

Feature extraction approaches already used

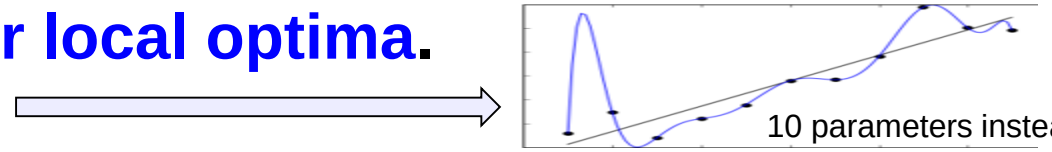
1.Spectrum coarsening from 4000 points into 200 zones
2. Fourier and wavelet analysis
3. Principal component analysis
4. Peak ranking by amplitudes

Input reduction in more than the order of magnitude, but **too low classification efficiency**



Недостатки метода обратного распространения ошибок

- The learning time does not scale well
  - It is very slow in networks with multiple hidden layers.
- It can get stuck in poor local optima.
- It tends to overfitting



- The problem known as the "Curse of dimensionality" hampering MLP applications for image recognition despite of any attempts to reduce input data preserving essential information.
- It requires labeled training data (what is the privilege of HENP).
  - Almost all data is unlabeled.

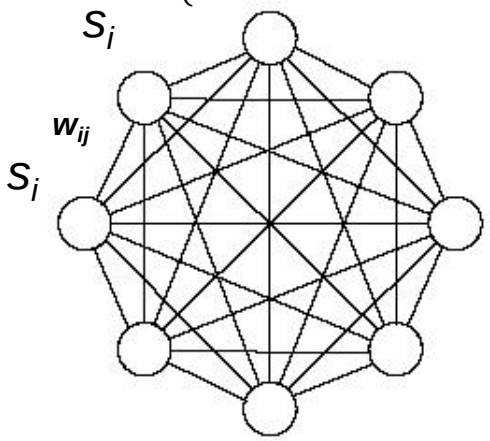
So let us see on different type of NN – reccurrent NN

The exhausted analysis of BP shortcomings and effective ways to overcome them one can find in Yann LeCun's paper "Efficient BackProp" : http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

Полносвязные сети Хопфилда

The recurrent NN considered as a dynamic system of binary neurons. All them are connected together with weights w_{ij}.

$$S_i = \begin{cases} 1, active \\ 0, non\ active \end{cases}$$



Hopfield's theorem: the energy function

$$E(s) = -\tfrac{1}{2} \Sigma_{ij} w_{ij} s_i s_j$$

of a recurrent NN with the symmetrical weight matrix $w_{ij} = w_{ji}$, $w_{ii} = 0$ has local minima corresponding to NN stability points.

However the usual way of $E(s)$ minimizing by updating the equation system, which defines the ANN dynamics: $s_i = \frac{1}{2}\left(1 + \mathrm{sign}\left(-\frac{\partial E}{\partial s_i}\right)\right)$ would bring us to one of many local minima.

Since our goal is to find the global minimum of ***E*** we have to **apply the mean-field-theory** (MFT). According to it, all neurons are thermalized by inventing temperature T, as λ=1/T and s_i are substituted by their thermal averages $v_i = \langle s_i \rangle_T$, which are now continuous in the interval [0,1]. Then ANN MFT dynamics is determined by the updating equation

$v_i = 1/2(1+\tanh(-\partial E/\partial v_i\ 1/T) = 1/2(1+\tanh(H_i/T)),$

where $H_i = \langle \Sigma_j w_{ij} s_j \rangle_T$ – is the local mean field of a neuron. Values of v_i are now defined **the activity level** of i^{th} neuron. Neurons with $v_i > v_{min}$ determine the most essential ANN-connections

Применение полносвязных ИНС для распознавания треков

Track recognition by Denby- Peterson (1988) segment model

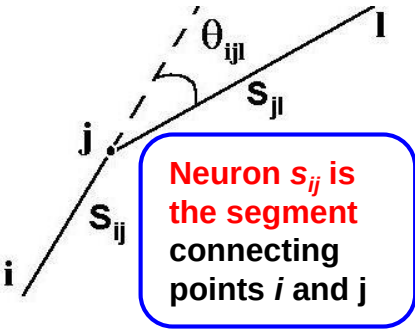
The idea: support adjacent segments with small angles between them. It is done by the special energy function:

$E = E_{cost} + E_{constraint}$, where

$$E_{cost} = -\frac{1}{2}\sum_{ijkl}\delta_{jk}\frac{\cos^m\theta_{ijl}}{r_{ij}r_{jl}}v_{ij}v_{kl}$$

W_{ijkl}

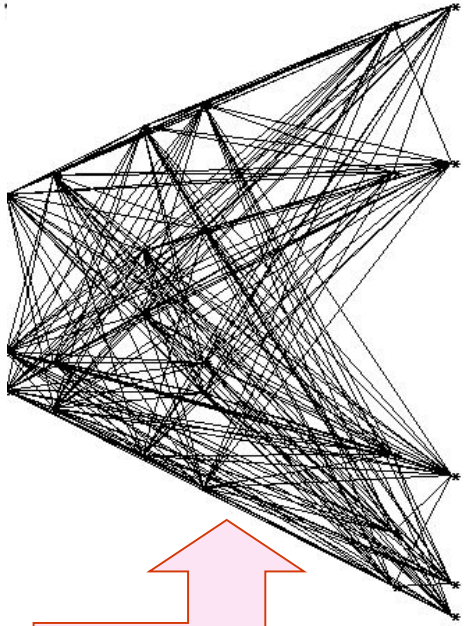
An easy example of the EXCHARM experiment with 6 multiwire proportional chambers registering hits from passing particle tracks and noise



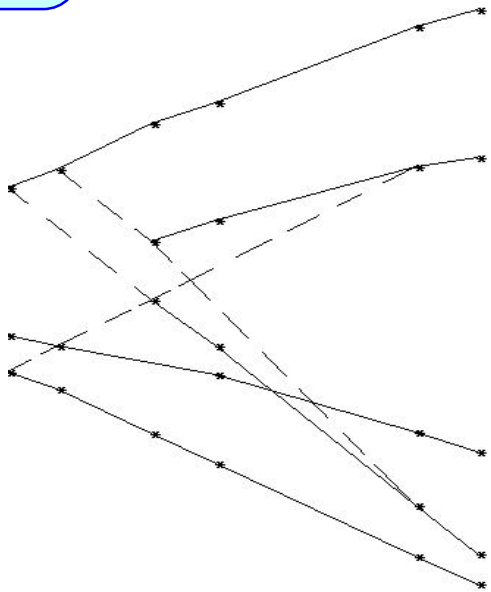
Neuron s_{ij} is the segment connecting points *i* and j

$E_{constraint}$ punishes segment bifurcations and balances between the number of active neurons and the number of experimental points.

Note: adding even a single noise point would generate ~80 extra hampering neurons



Zero iteration: 244 neurons.



After 30 iteration:
26 neurons with v_{ij} >0.5

Thus we have NN with unsupervised learning that trained itself during its evolution

Нейронные сети глубокого обучения

Big Data era.
Technological achievements: HPCs, GPU, clouds.
Biological observations: The mammal brain is organized in a deep architecture, animal visual cortex perceive 3D obejcts.

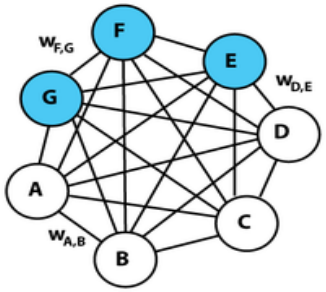
Problems to be solved are now getting more and more complicated, so the ANN architecture **needs to become deeper by adding more hidden layers** for better parametrization and more adequate problem modelling. However in most cases deepening, i.e. the fast grow of inter-neutron links, faces the problem known as the “Curse of dimensionality”, which leads to overfitting or to sticking the minimized BP error function in poor local optima.
Hopfield NN could not also be effective enough in cases of noisy and dense events.

Challenge of Deep Learning approach in Neural Networks, inevitability to use parallelism and virtuality.

Полносвязные сети как машина Больцмана

Geoffrey Hinton proposed in 2006 a new approach to combine in one neural net both types of NN, - with supervised learning by back-propagation and recurrent NN (RNN) with unsupervised learning.

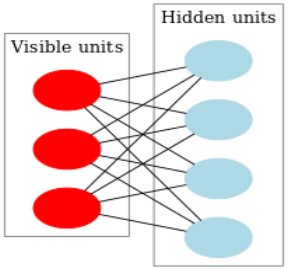
He considered Hopfield RNN as **stochastic NN** which state probabilities obey **Boltzman distribution** $p_i = \frac{e^{-\varepsilon_i/kT}}{\sum_{j=1}^{M} e^{-\varepsilon_j/kT}}$, as the function of that state's energy and the system temperature, and named it the **Boltzman machine (BM)** (hence **Boltzman distribution**).



Graphical BM representation

Then Hinton divided neurons into groups of **visible and hidden** ones where visible units V are those which receive information from the 'environment', i.e. our training set is a set of binary vectors over the set V. It allows to **find probabilities of hidden neuron states in the way of BM thermal equilibrium**. However, due to serious practical problems with BM training, Hinton proposed a new efficient way in an architecture called the "**restricted Boltzmann machine**" (**RBM**) which does not allow intralayer connections between hidden units.

After training one RBM, the activities of its hidden units can be treated as data for training a higher-level RBM. This method of stacking RBMs makes it possible to train many layers of hidden units efficiently and it is one of the most common deep learning strategies. As each new layer is added the overall generative model gets better.



Graphical representation of a restricted Boltzmann machine.

"Deep" refers to the number of NN layers. Whereas most current learning algorithms, as MLP, correspond to shallow architectures (1, 2 or 3 layers). The mammal brain is organized in a deep architecture.

Глубокая нейросеть доверия

Thus Hinton introduced **Deep Belief Networks** (**DBN**) which is composed of multiple hidden layers, with connections between the layers but not between units within each layer.

The word "belief" here does not mean "faith", but "confidence" what means we deal with stochastic NN and probability

These layers are **restricted Boltzmann machine** (**RBM**) that can learn themselves a **probability distribution** over its set of inputs. The evolution of RBF layers is realized by Markov chain Monte Carlo (MCMC) method, although its convergence needed to obtain mean values of wanted parameters is always **too long to be practical**.

Hinton showed, if the learning approximately minimizes a different function called "**contrastive divergence**" (CD), then Markov chain can run only for just one step, but this trick works surprisingly well.

During the last decade DBNs have been applied with success not only in classification, but also in regression, dimensionality reduction, modeling motions, information retrieval, object segmentation, natural language processing and many others.

Изучение глубоких нейросетей в ОИЯИ

In the Laboratory of Information Technologies of JINR the program complex for the image recognizing problem was developed. It realizes **four neural networks**:

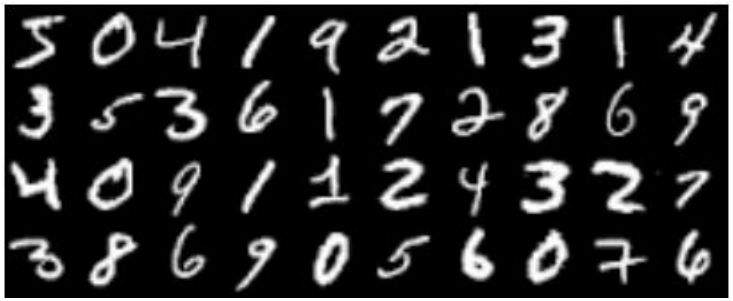
1. **Autoassociative NN** realizing the nonlinear principal component analysis (NLPCA) intended for the **significant compression of input data**;

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

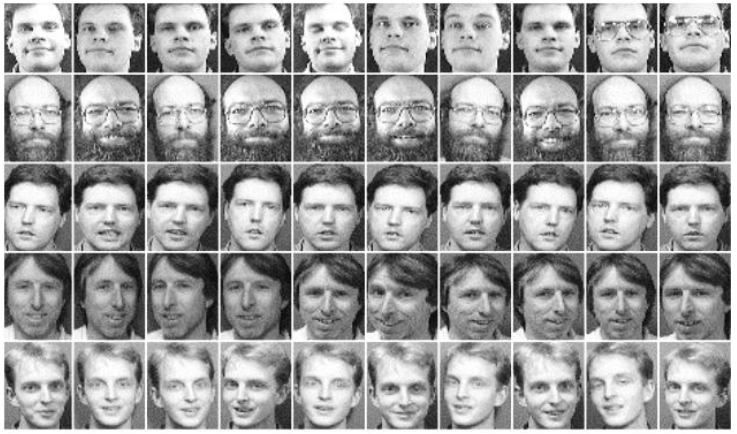
2. Perceptron with one hidden layer;
3. Deep Belief Network;
4. Deep Neural Network (DNN) with initialization of normalized weights.

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

- MNIST handwritten digit database (70,000 handwritten digits images);
- FERET face database (40 people to 10 photos for each).



MNIST handwritten database sample

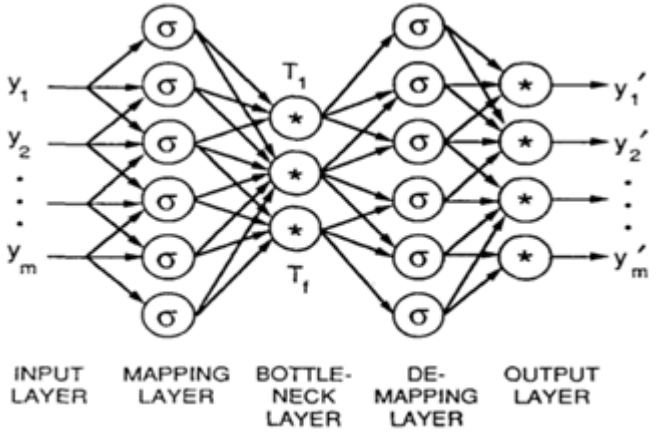


FERET face database sample

1-й этап: сжатие данных автоэнкодером

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 as an autoassociative neural network with **three hidden layers including the bottle neck** layer in the middle.

In our study we considerably modified Kramer's scheme in order to improve its efficiency and speed up training time by solving convergence problems



Kramer's AE determins f nonlinear factors,
σ indicates sigmoidal nodes,
*** indicates sigmoidal or linear nodes**

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

- it gives us the same set of compressed data as an input vector for three neural networks to be compared further as classifiers;
- AE realized as an autoassociative neural network needs to optimize its structure, tune parameters of activation function, choose a proper normalization of input data and obtain the fast convergence of the training process;
- AE provides highly effective data compression, which is getting especially important at the Big Data era.

Улучшение эффективности автоэнкодера

Important to note: the state-of-the-art level of our AE models.

Our improvement of Kramer's scheme is a result of the special study and consists in

(1) replacing sigmoidal activations by Leaky Rectified Linear (LReLU) activation with the carefully chosen parameter *a* different for each of data set;

(2) a specific input normalization (MinMax was chosen);

(3) applying tied weights;

(4) stochastic gradient descent (SGD) with Adam optimization.

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

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

To determine the **number of neurons in hidden layers** of autoencoder we calculate the dependence of the AE loss function for different numbers of neurons in the bottle-neck and in both mapping-demapping layers taking into account the training time as well.

Eventually, dimensions of 5 autoencoder layers were chosen as (**m,256,64,256,m**), where m is the length of the input vector.

After nonlinear compression input images to 64 principal features in bottle-neck layer AE recover them in output layer, as it is shown for MNHIST data with LReLU activation

original above, recovered below

Оптимизация глубоких классификаторов

We use **SGD Adam minimization** of the back-prop loss in our work with shallow, deep and deep belief neural networks. The numbers of hidden layers and optimal combination of hidden neurons per layer in our deep classifiers are chosen by the **grid search cross-validation** procedure taking into account NN efficiency dependencies on these numbers and the training time.
All it gives **two hidden** layers for both DNN and DBN while the optimal numbers of the hidden neurons per 1st and 2d layers are **300** and **100** respectively.

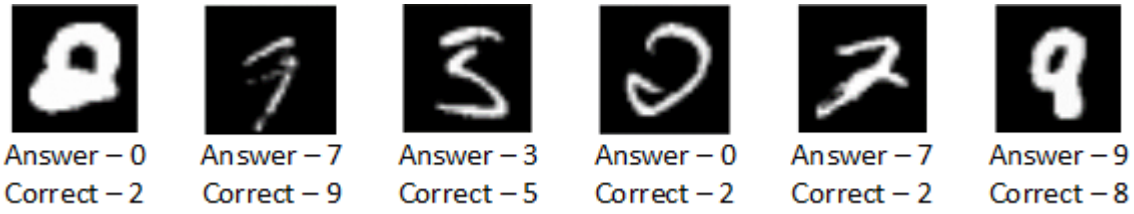
The images of both data sets being rescaled to the 64-dimensional eigenvectors were classified then by three our classifiers with the following results:

Результаты тестирования

	Perceptron	Deep Neural Network	Deep Belief Network
MNIST	0.7977	0.9285	0.9839
FERET	0.8750	1.0000	1.0000

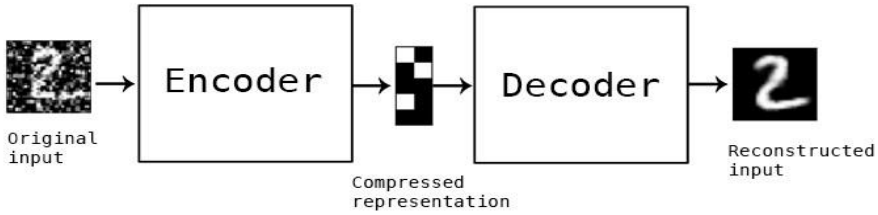
Deep learning techniques give 100% classification efficiency on the faces database and 92 –98 % efficiency on the handwritten digits dataset. Although it is not a problem for the human to distinguish various faces, the DBN can make better predictions, even in the case of digits, than the human can.

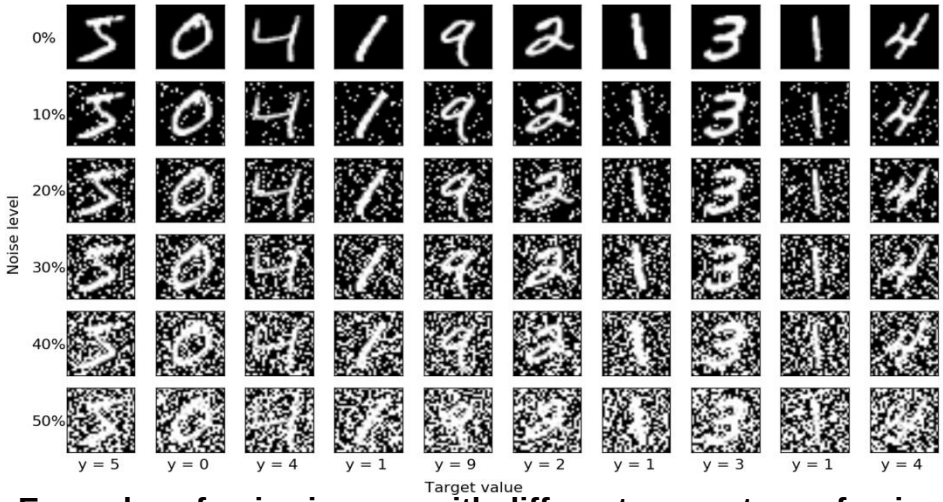
Examples of incorrect DBN predictions are shown to emphasize the special difficulties of classifying handwritten digits even for the human.



Применение автоэнкодеров для шумоподавления

A denoising procedure takes noisy data, as input, and outputs this data, where the noise is removed.





Examples of noisy images with different percentage of noise

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 range from 0% to 50% compressed previously by DAEN.

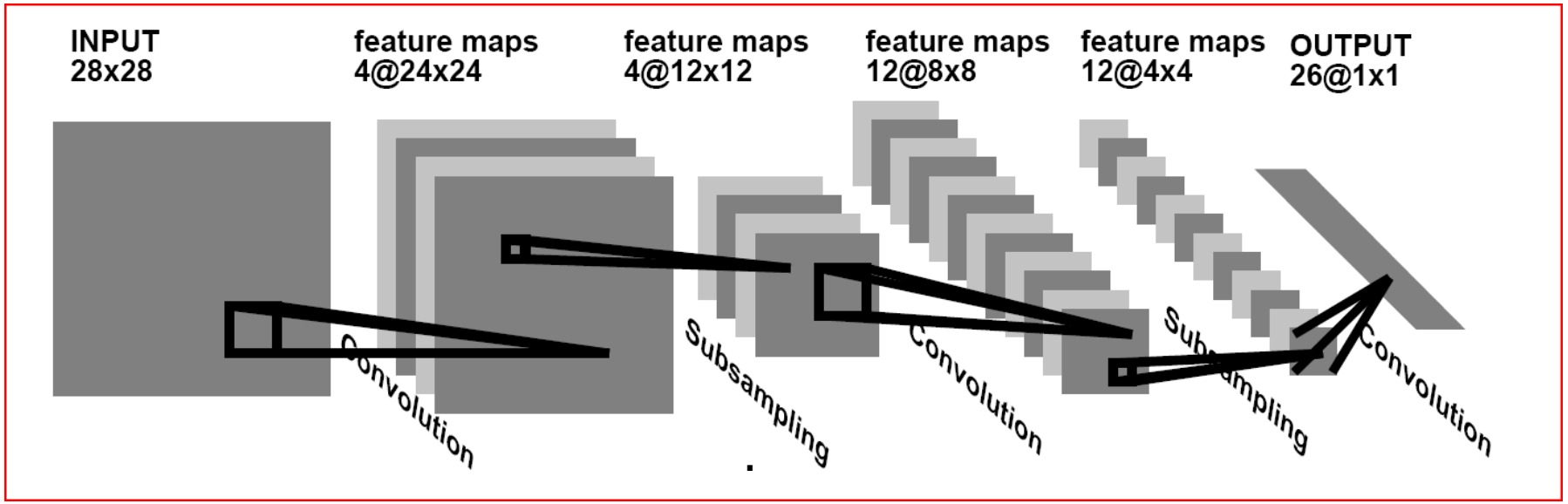
As results in table show, the efficiency keeps well even on the data with 30% of noise. Deep Belief Network performs better classification results again, but with noise increasing it yields to the DNN. Both deep classifier efficiencies are notably higher than for the perceptron with one hidden layer.

	FERET			MNIST		
Noise (%)	Perceptron	DNN	DBN	Perceptron	DNN	DBN
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

Efficiency results

Сверточные нейросети для распознавания изображений

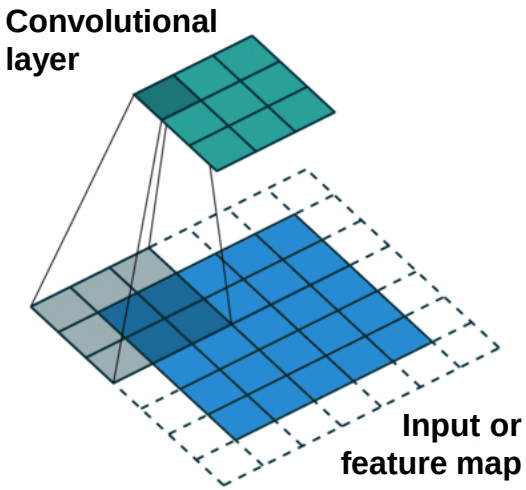
Recall to motivate: 1. DBN training time is too long
2. unsupervised learning algorithms for training RBMs by approximate maximum likelihood meet an intractable problem of measuring the likelihood that is to be optimized.
3. Direct applying regular neural nets to image recognition is useless because of two main factors:
(i) input 2D image as a scanned 1D vector means the loss of the image space topology;
(ii) full connectivity of NN, where each neuron is fully connected to all neurons in the previous layer, is too wasteful due to the curse of dimensionality, besides the huge number of parameters would quickly lead to overfitting.

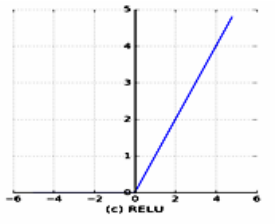


Instead, neurons of Convolutional Neural Networks (CNN) in a layer are only connected to a small region of the layer before it (**Le Cun & Bengio, 1995, see also http://cs231n.github.io/convolutional-networks/**)

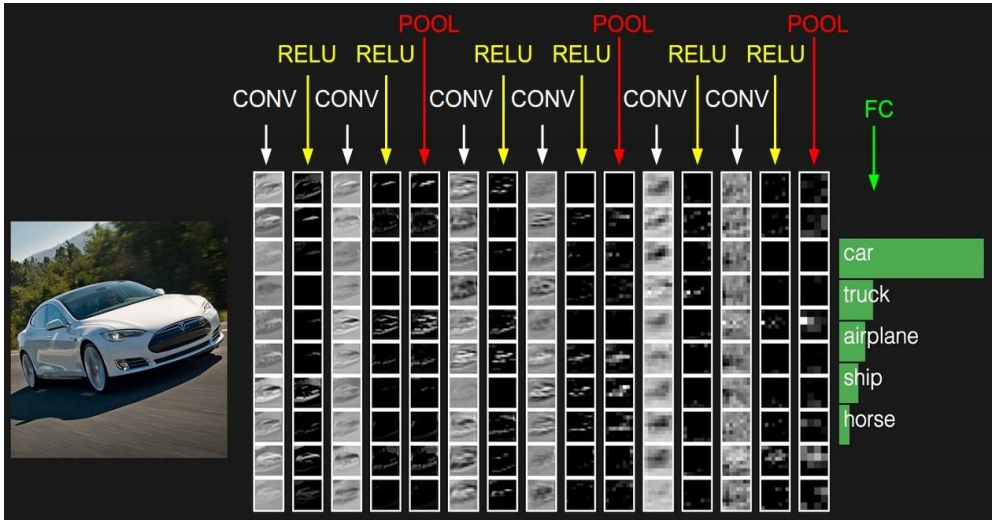
Основы архитектуры CNN

The CNN architecture is a sequence of layers, and every layer transforms one volume of activations to another through a **filter** which is a differentiable function.
There are three main types of layers to build CNN architectures: **Convolutional Layer**, **Pooling (*subsampling)* Layer**, and **Fully-Connected Layer** (just MLP with backprop)**.** There are also **RELU** (rectified linear unit) layers performing the max(0,x)

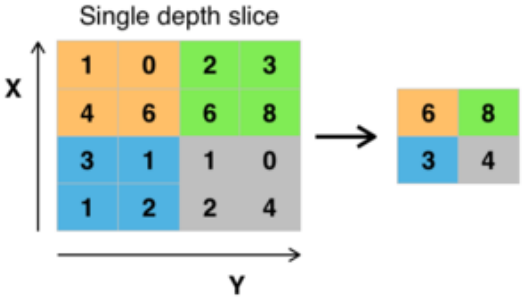




Activation: ReLU



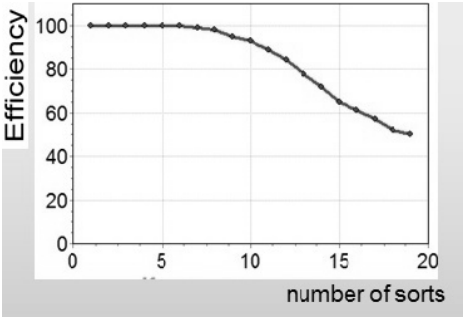
Example of classifying by CNN



Max **pooling** with a 2x2 filter

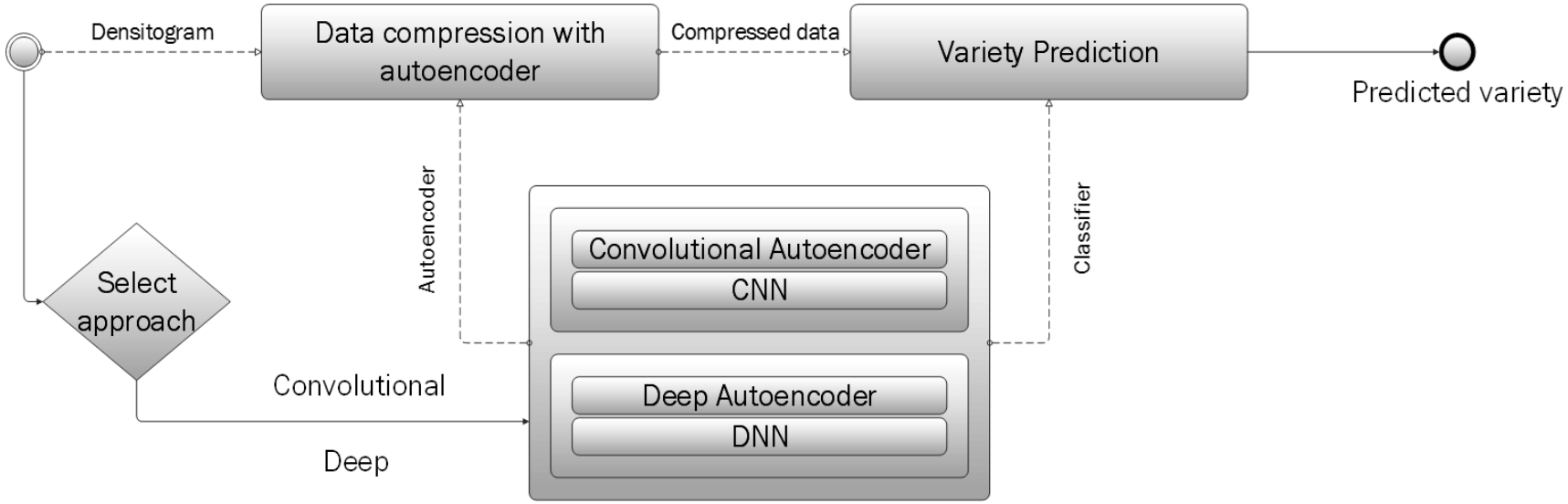
Each Layer accepts an input 3D volume (x,y,RGB color) and transforms it to an output 3D volume. To construct all filters of convolutional layers our CNN must be trained by a labeled sample with the back-prop method**.** See https://geektimes.ru/post/74326/ in Russian or tps://en.wikipedia.org/wiki/Convolutional_neural_network

CNN в задаче классификации белков твердой пшеницы



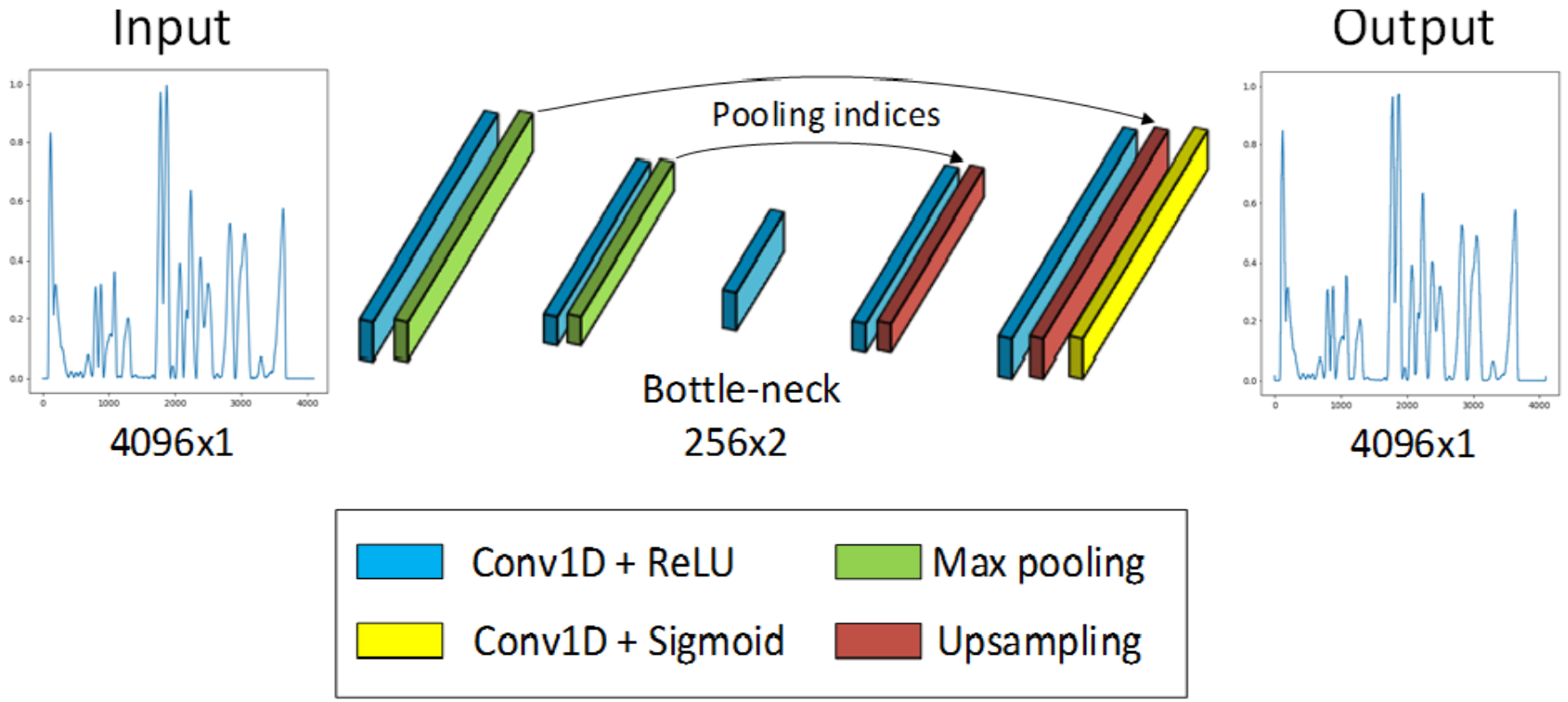
Recall,- the failure of any attempt to classify more than 10 sorts of wheat.

To solve the problem of classifying 50 wheat sorts presented as 3225 EF-densitogramms we propose to use two-step algorithm based on data-compression by an autoencoder on the first step followed then on the second step by applying either DNN or convolutional NN. To make a justified choice of AE we elaborate also a convolutional one. So in our study we have to develop two versions of autoencoders and two versions of classifiers in order to compare them for better efficiency with reasonable time consuming.



Scheme of comparative study of DNN and CNN classifications

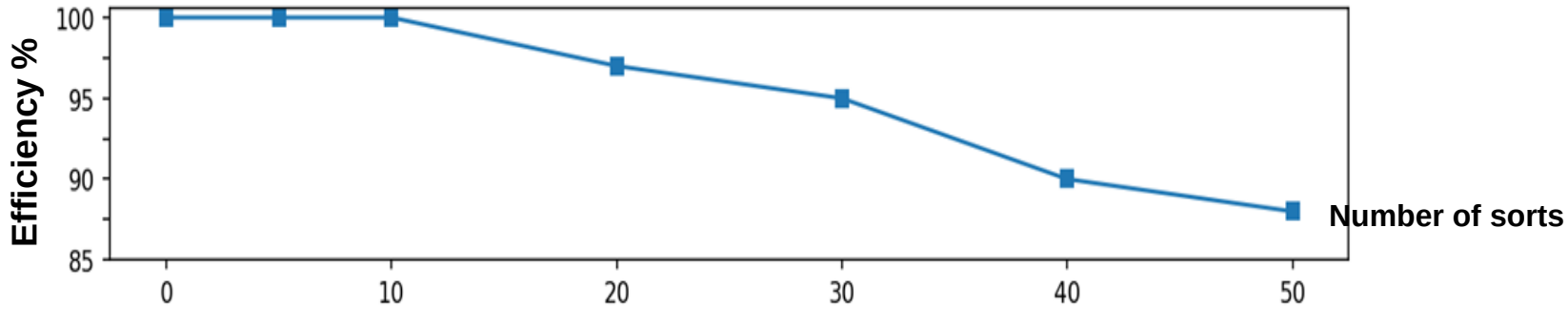
Сверточный автоэнкодер



We apply the improvements elaborated above for AE such as 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 AE and gives significant gain in memory to store AE as well as in training speed

Сравнительные результаты классификации протеинов

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



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

One can see advantages of convolutional network comparing to DNN. With the same efficiency CNN takes 22 times less memory, while its training time is faster.
Besides these results demonstrate the overwhelming superiority of the deep approach in the solution of protein classification problem.

Ускорение обучения ИНС путем применения карт GPU в облачной инфраструктуре

Classification model is developed on notebook Dell Vostro 5480 with hardware:

- **CPU*:* Intel Core i3-4500U @ 1.70 GHz (4 cores);**
- **RAM*:* 4096 MB;**
- **Graphic*:* Intel HD Graphics 4400.**

One training epoch for convolutional autoencoder takes 70 – 80 sec,

but operating in the cloud service facilities provided by HybriLIT using TensorFlow and Keras we have:

Nvidia K80 + Keras + TensorFlow = it takes **only 1 sec for 1 training epoch**

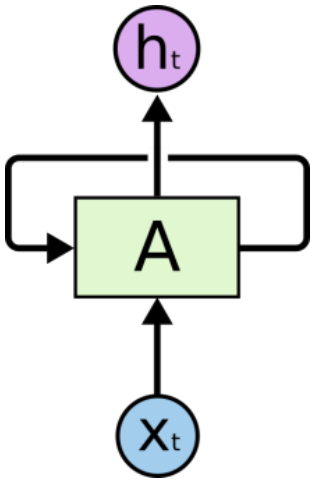
TensorFlow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. TensorFlow can do calculations in parallel by installing required drivers and define special flags. To install GPU version, user must add «gpu» postfix after the name of package.

Keras is the Python Deep Learning library, which is a high-level API that uses TensorFlow as a backend

Немного философии об изменяющемся мире

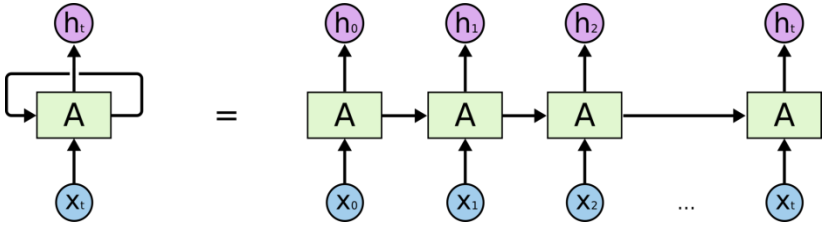
- В жизни мы имеем дело с объектами, изменяющимися во времени, и наш мозг, работая, всегда исходит из знания того, что уже было.
- Однако обычные нейросети с одним скрытым слоем, глубокие сети, включая DBN, и даже такие продвинутые сети, как сверточные, предназначены для работы со статическими объектами. Никакое обучение не поможет традиционной нейросети смоделировать будущее состояние объекта.
- Для описания динамического объекта нейронная сеть должна обладать некоей **памятью**, чтобы исходя не только из настоящего его состояния, но и из прошлого, нейросеть могла бы моделировать его последующее состояние.
- Эту проблему решает семейство новых глубоких нейросетей, называмых рекуррентными (Recurrent Neural Networks-RNN). Они содержат в себе **обратные связи**, позволяющие сохранять информацию.

Рекуррентные нейронные сети



На схеме фрагмент нейронной сети A принимает входное значение xt и возвращает значение ht. Внутри этой ячейки – обычная сеть с одним скрытым слоем.

Наличие обратной связи позволяет передавать информацию от одного шага сети к другому. RNN можно рассматривать, как несколько копий одной и той же сети, каждая из которых передает информацию последующей копии



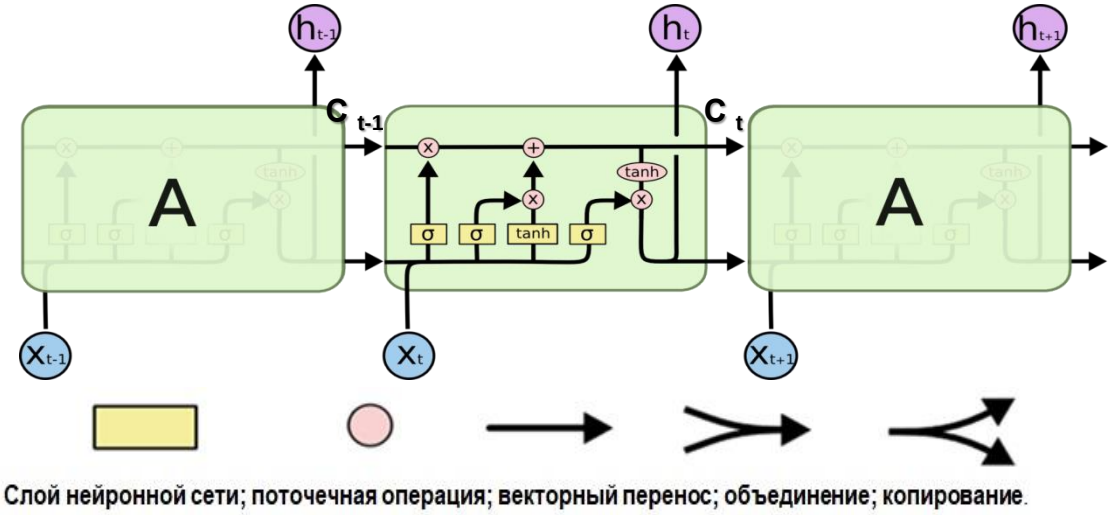
Так что RNN – самая естественная архитектура нейронных сетей для работы с последовательностями и списками. Однако, чтобы RNN могла **связывать** предыдущую информацию с текущей задачей, требуется ее усовершенствование до **LSTM сети** (Long short-term memory - долгая краткосрочная память) – особой разновидности архитектуры RNN, способной к обучению долговременным зависимостям.

Сети LSTM

Вместо одного слоя RNN сети LSTM содержат целых четыре взаимодействующих слоя, позволяющих удалять информацию из состояния ячейки с помощью **фильтров - gates**. Фильтры состоят из слоя сигмоидальной нейронной сети и операции поточечного умножения и позволяют пропускать информацию на основании задаваемых условий.

Основной компонент LSTM – это памяти сети (cell state) – линия, проходящая по верхней части схемы.

В LSTM три фильтра, позволяющих защищать и контролировать состояние ячейки.



1. слой фильтра забывания определяет какую часть информации выбросить

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

2. что сохранить - 2 слоя: слой входного фильтра + tanh-слои

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

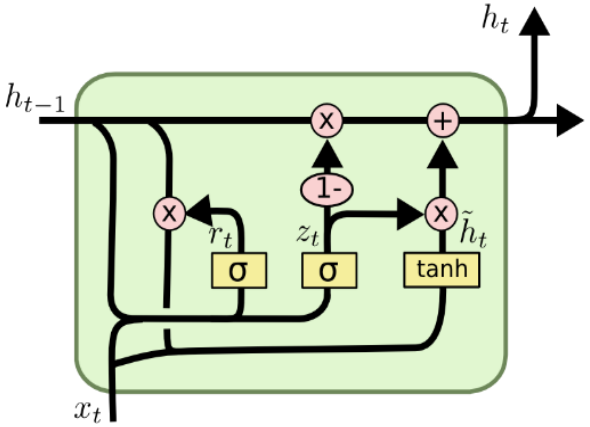
3. что получать на выходе

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Управляемые рекуррентные модули GRU (Gated Recurrent Unit)

Управляемый рекуррентный модуль GRU (Gated Recurrent Unit) является упрощенной версией LSTM. В нем фильтры «забывания» и входа объединяют в один фильтр «обновления» (update gate), а состояние ячейки объединяется со скрытым состоянием.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

С помощью именно такой GRU нейросети нам удалось решить задачу начального восстановления траекторий элементарных частиц в детекторе GEM эксперимента BM@N в ОИЯИ. Доклад об этом будет в пятницу 29 сентября.

Как создавать, обучать и тестировать нейросети

расскажет магистр Гомельского Технического университета

Павел Гончаров.

Задача по распознаванию рукописных цифр из базы данных MNIST будет решена с помощью трех нейронных сетей:

1. Обычный однослойный персептрон;
2. Сверточная нейронная сеть;
3. Рекуррентная нейронная сеть.

С помощью уже упомянутых библиотек KERAS и TensorFlow на ваших глазах будут созданы эти три сети с объяснением выбора их структуры и параметров. Обученные сети будут протестированы для получения цифр сравнительной эффективности.

Thanks for your attention!