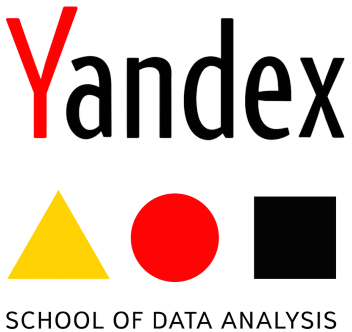
Machine Learning in Science and Industry

Day 4, lectures 7 & 8

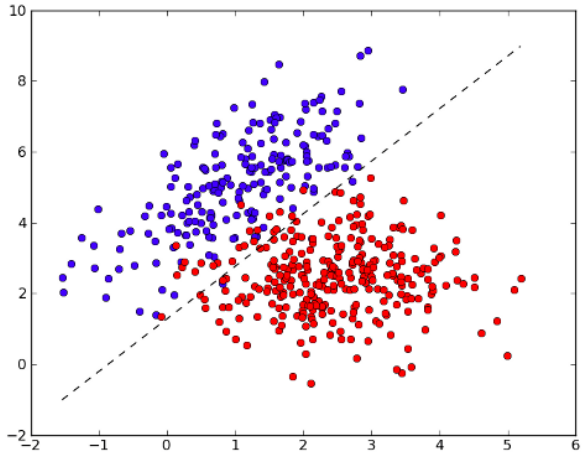
Alex Rogozhnikov, Tatiana Likhomanenko

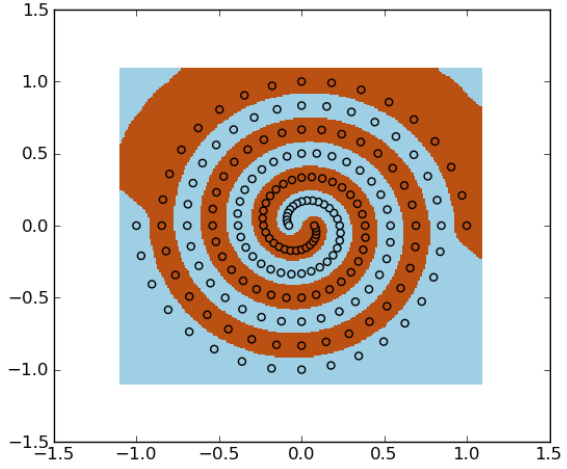
Heidelberg, GradDays 2017

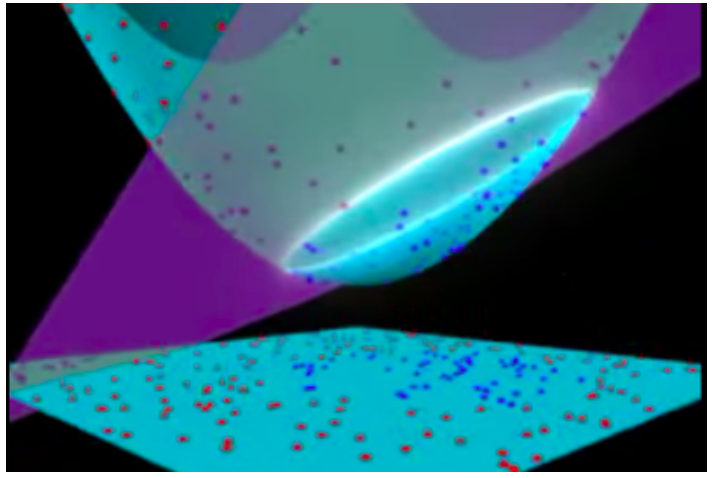


Recapitulation: linear models

- work with many features
  - bag-of-words for texts / images
- SVM + kernel trick or add new features to introduce interaction between features
- regularizations to make it stable







Recapitulation: factorization models

- recommender systems

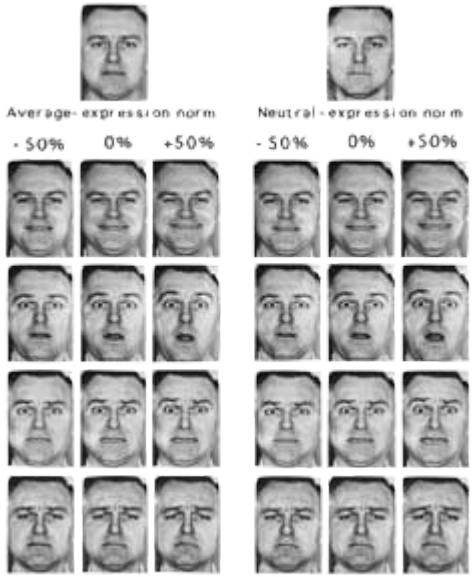
$$\hat{R}_{u,i} =< U_{u,}.V_{.,i} >$$

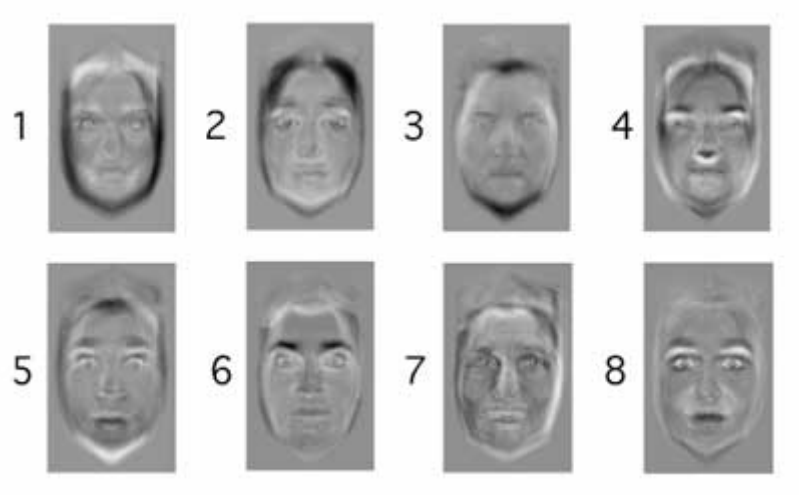
$$\sum_{\text{known } R_{u,i}} \left[\hat{R}_{u,i} - R_{u,i}\right]^2 \rightarrow \min$$

- learning representations
- learning through the prism of interaction between entities

Recapitulation: dimensionality reduction

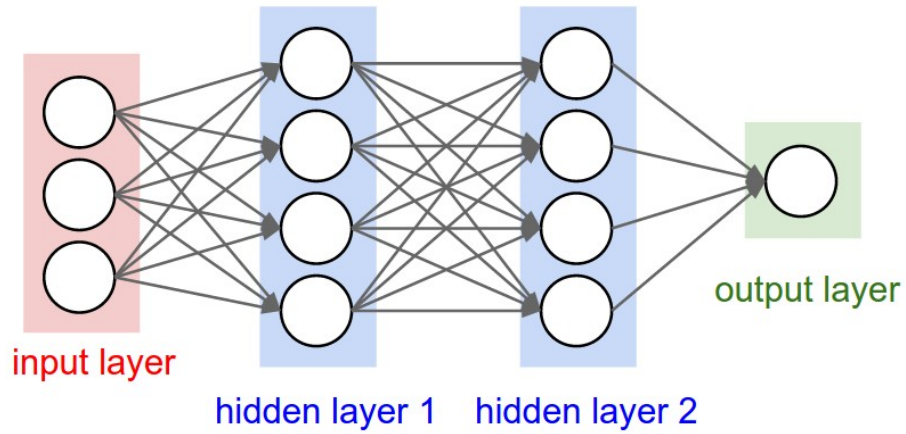
- compressing data of large dimensionality
- output is of small dimensionality and can be used efficiently by ML techniques
- expected to keep most of the information in an accessible form





Multilayer Perceptron

- feed-forward
- function which consists of sequentially applying
  - linear operations
  - nonlinear operations (activations)
- appropriate loss function + optimization
- in applications — one or two hidden layers



Tabular data approach

In rare decays analysis:

momenta, energies, IPs, particle identification, isolation variables, flight distances, angles ... $\rightarrow$ machine learning $\rightarrow$ signal or background

In a search engine

How many words from query in the document (url, title, body, links...), TF-IDFs, BM25, number of links to / from a page, geodistance, how frequenly a link was clicked before, estimates of fraud / search spam, ... $\rightarrow$ machine learning $\rightarrow$ real-values estimation of relevance

This approach works well enough in many areas.
Features can be either computed manually or also be estimated by ML (*stacking*).

When it didn't work (good enough)?

- feature engineering $\rightarrow$ ML is a widespread traditional pipeline
- which features help to identify a drum on the picture?
  - a car?
  - an animal?
- in the sound processing
  - which features are good to detect a particular phoneme?

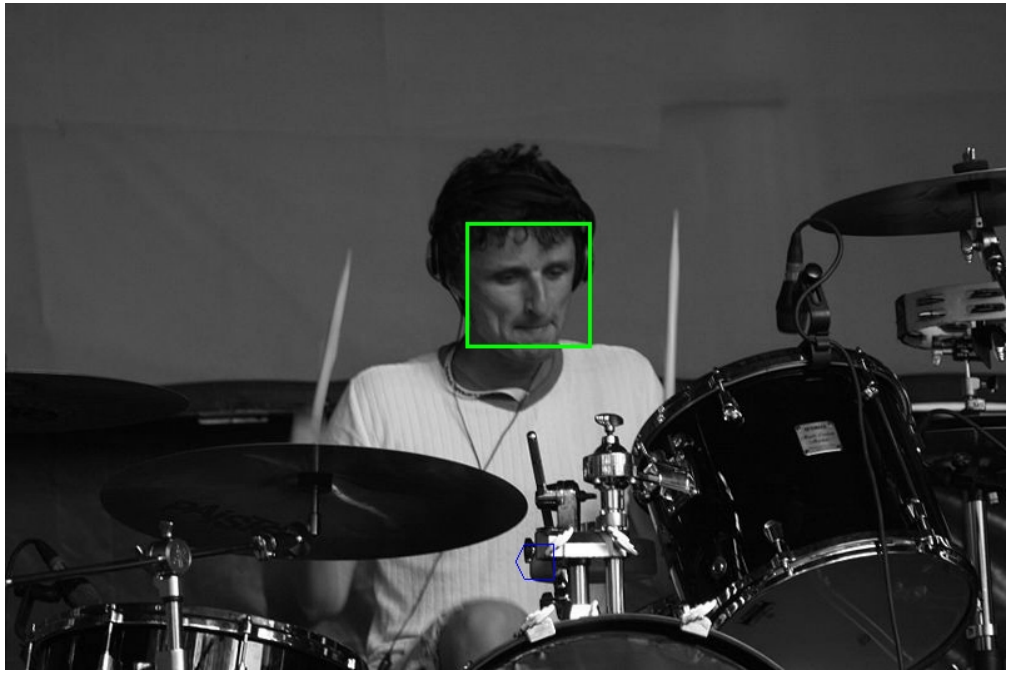
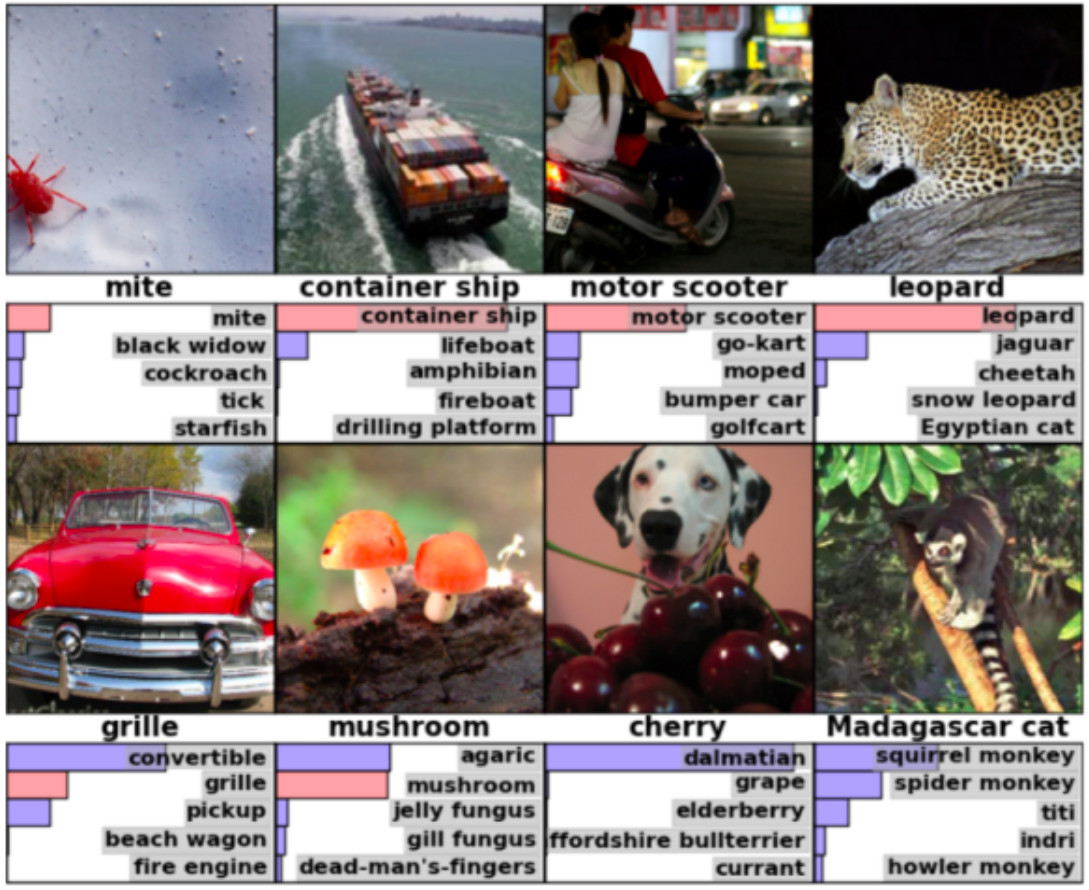


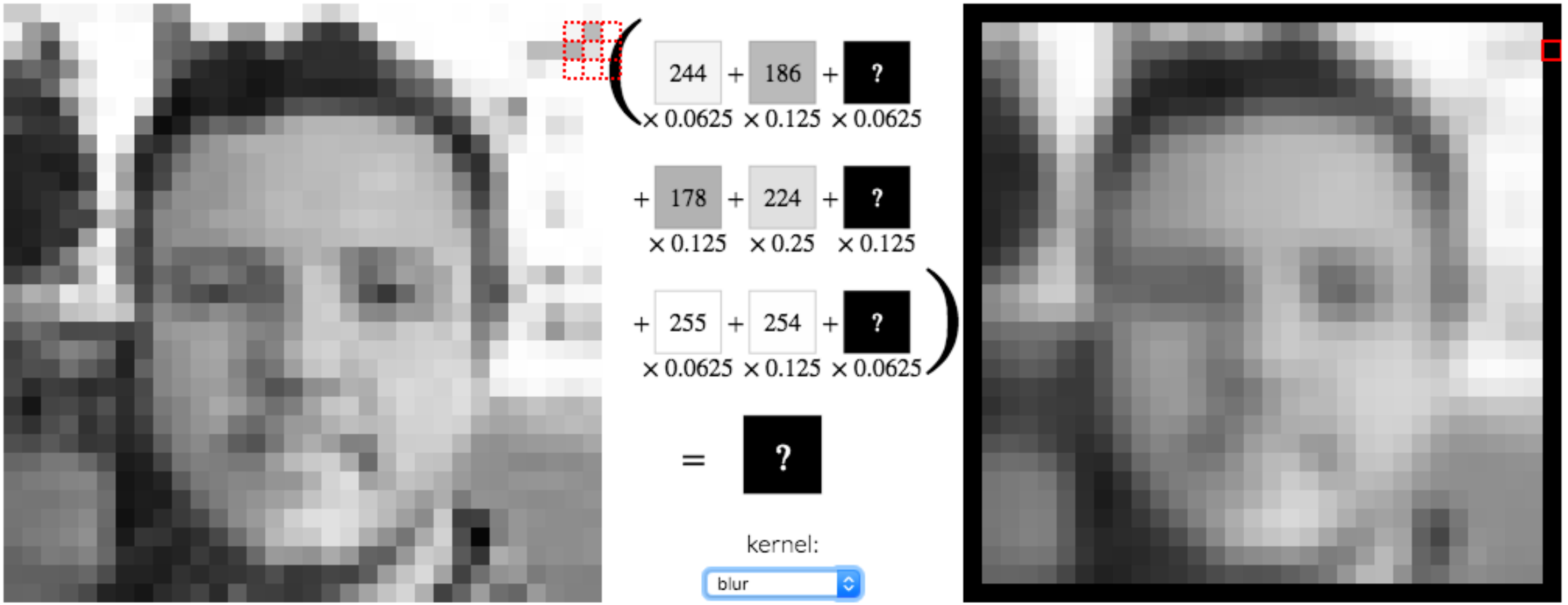
Image recognition today

- can classify objects on images with very high accuracy
- even if those are images of 500 x 500 and we have only 400 output neurons, that's $500 * 500 * 400 = 100'000'000$ parameters
  - and that's only linear model!

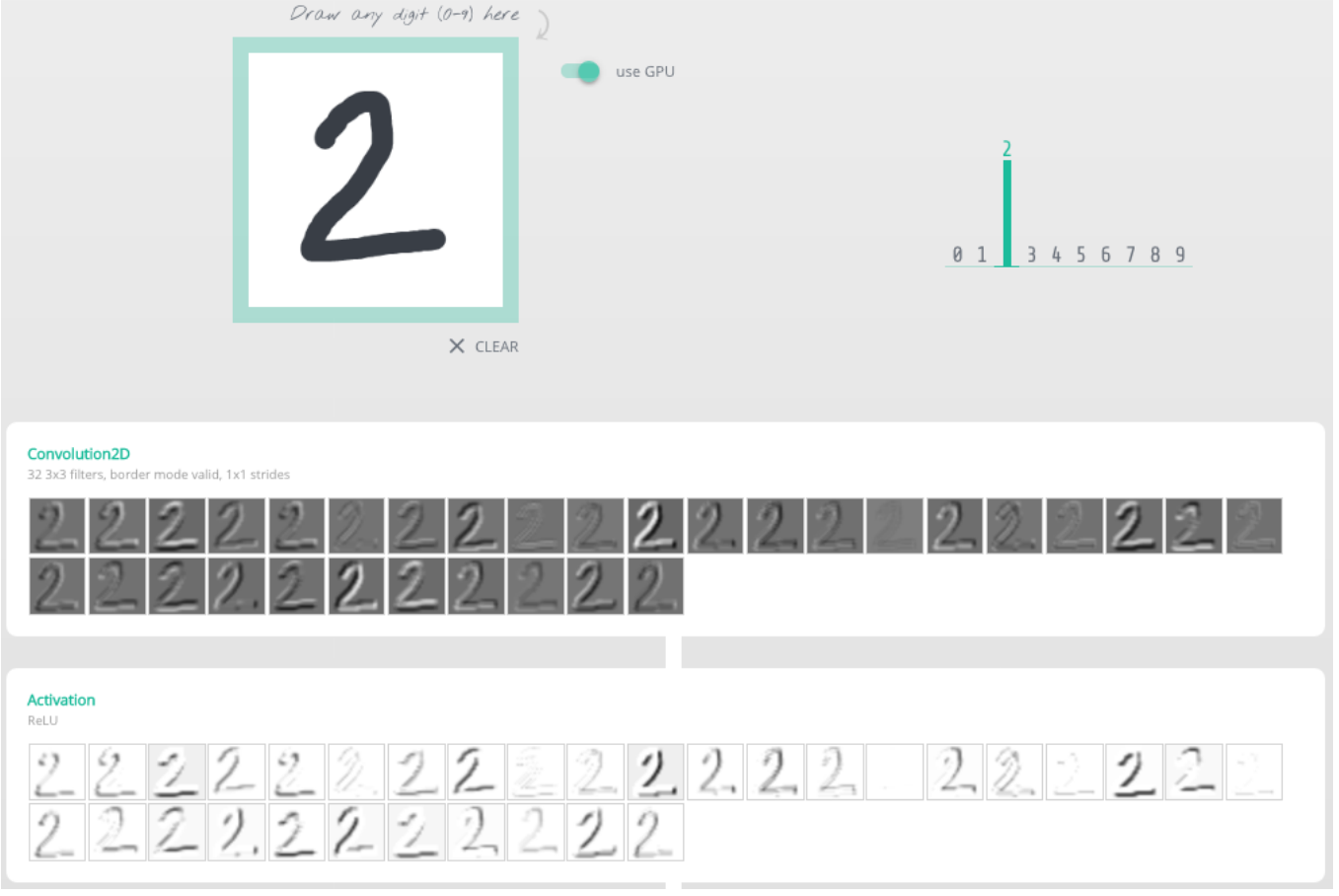
1. can we reduce the number of parameters?
2. is there something we're missing about image data when turning picture to a vector?



Convolutions

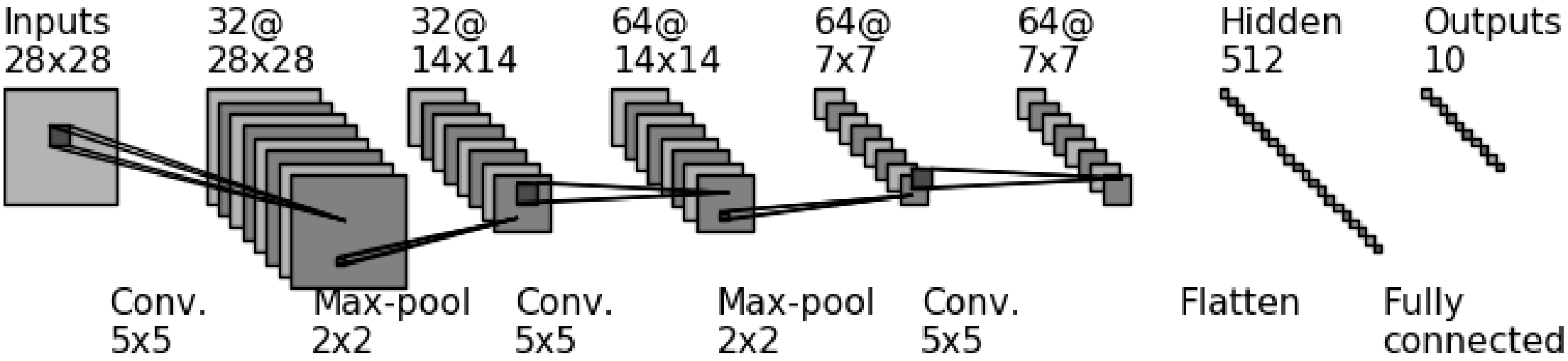


Convolutional neural network (also)



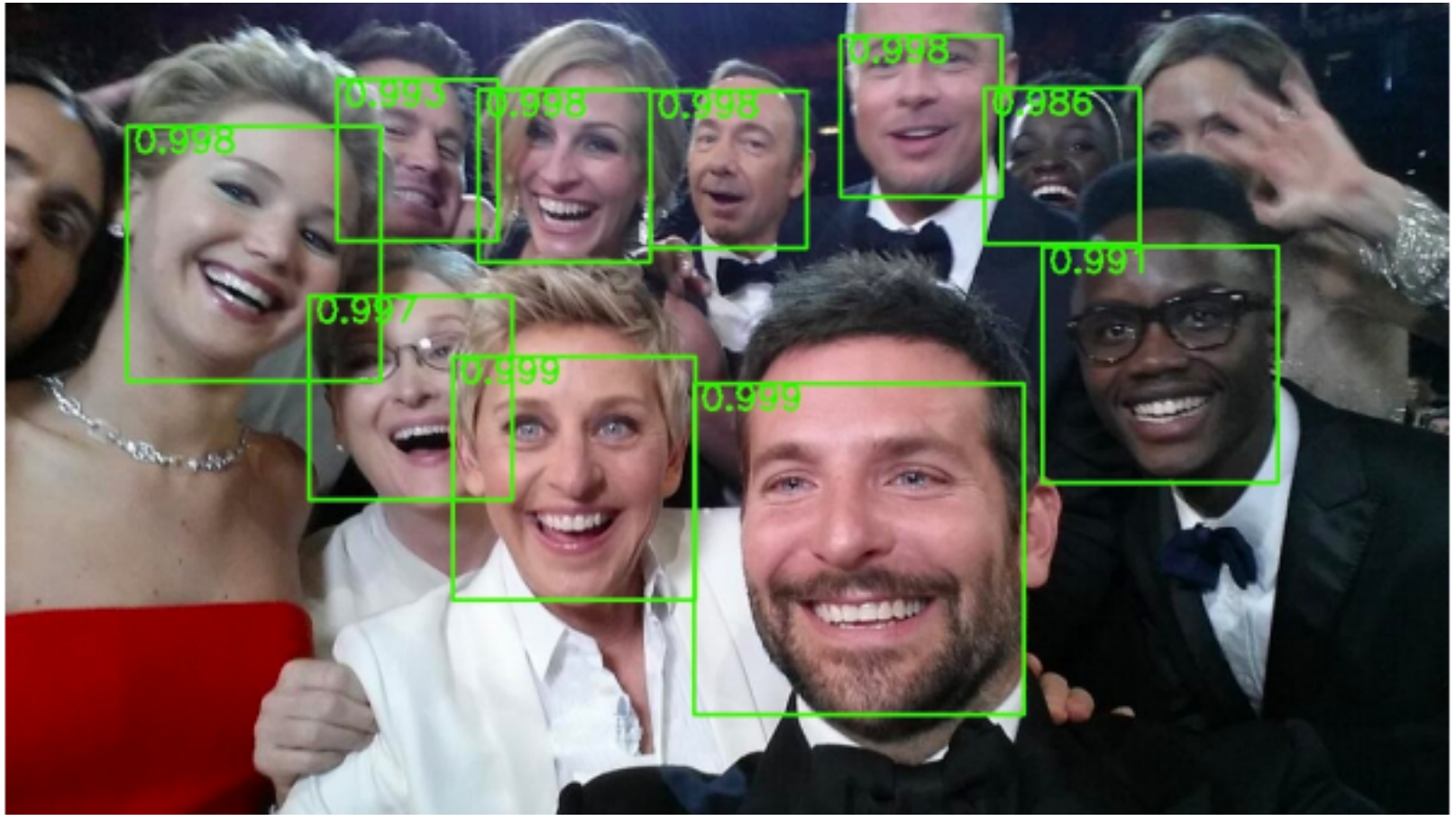
CNN architecture components:

- convolutions
- pooling (reducing the size)
- flattening
- simple MLP



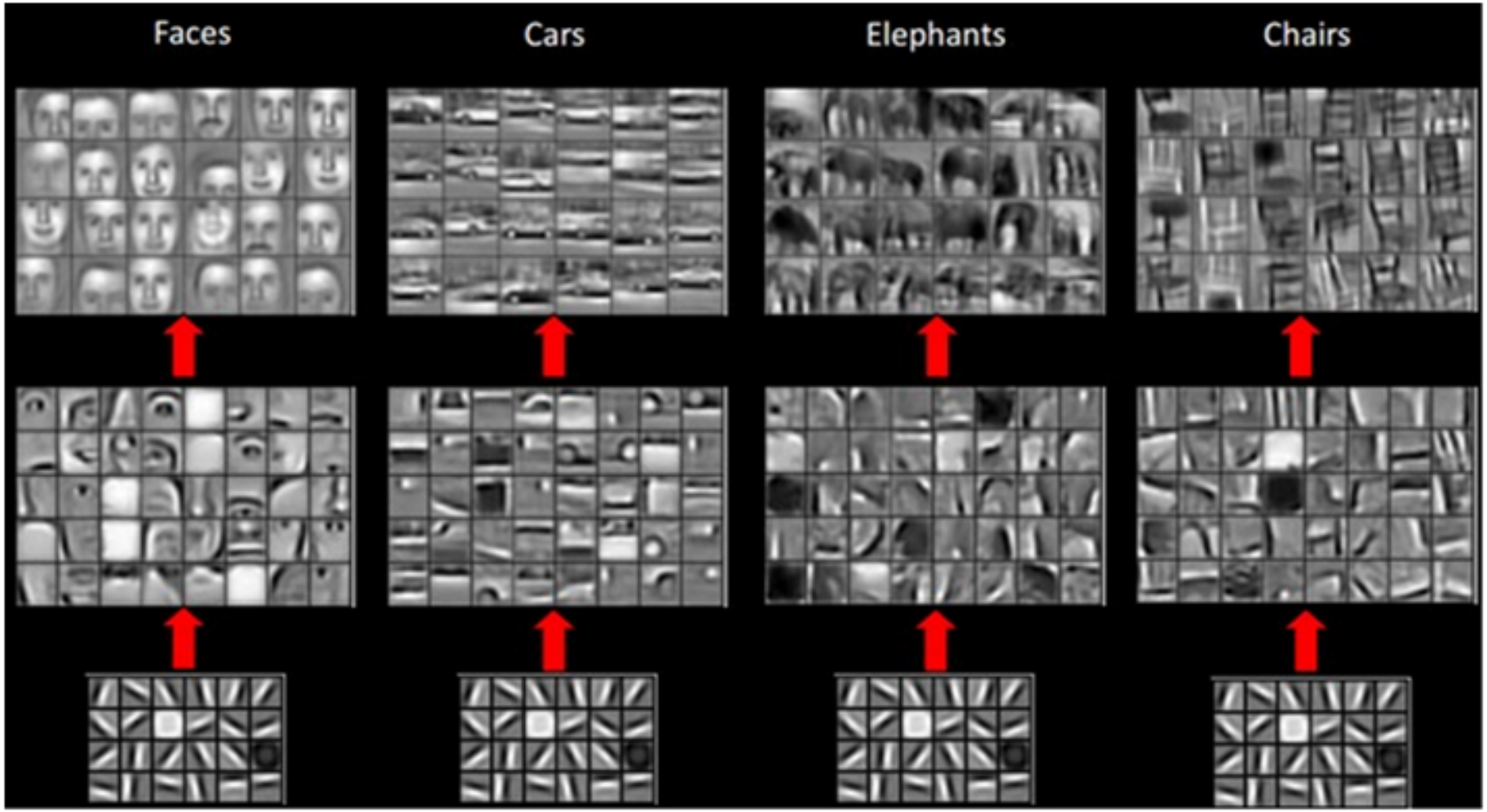
LeNet 1 based on CNNs — was developed [in 1993]

Some idea behind

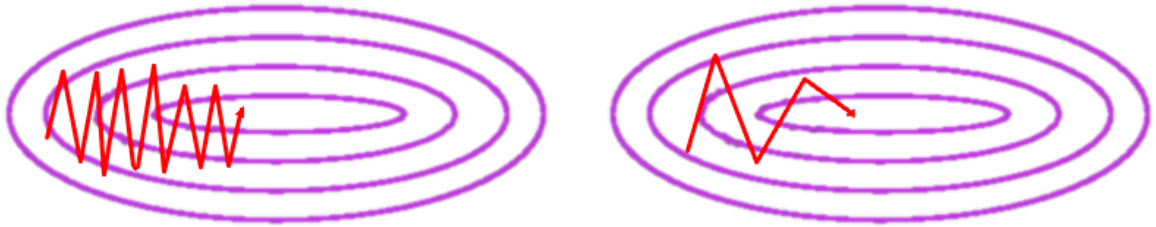


Detecting particular parts of a face is much easier.

Some idea behind



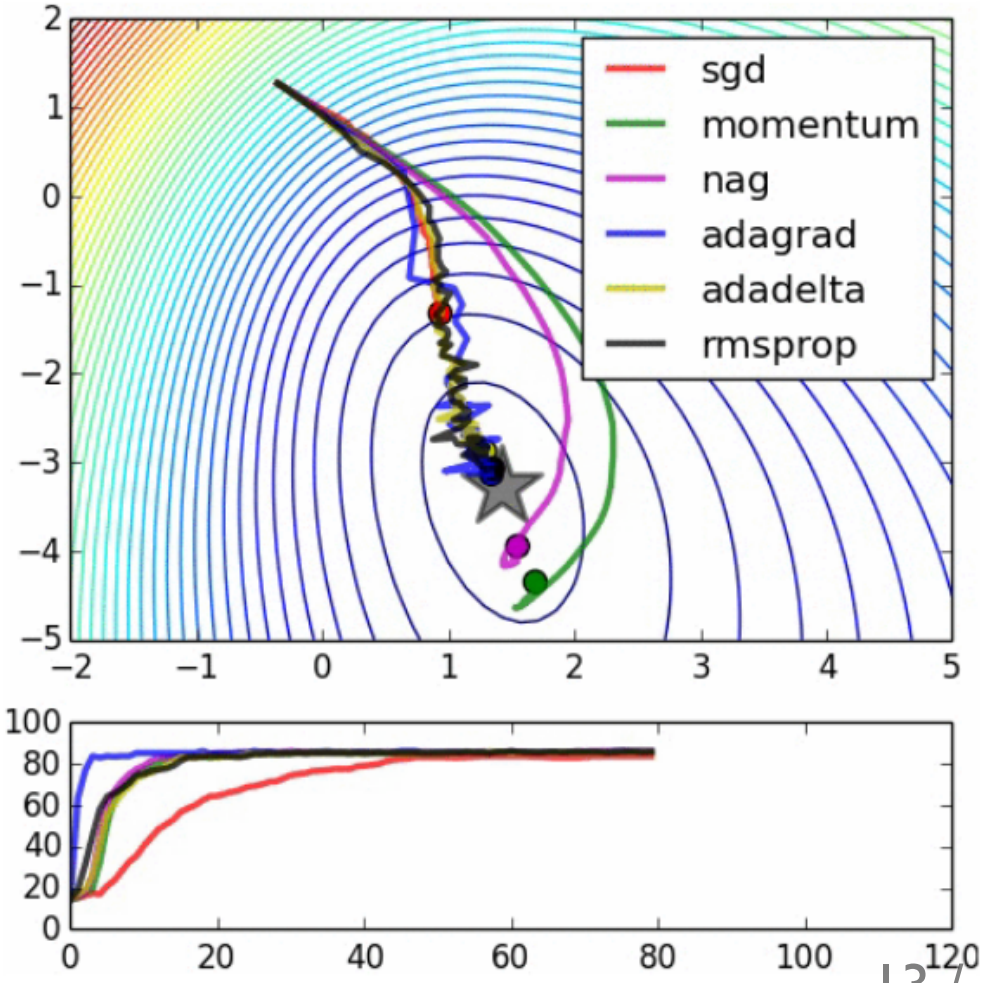
Problems with deep architectures



In deep architectures, the gradient varies between parameters drastically ($10^3 - 10^6$ times difference, this is not a limit).

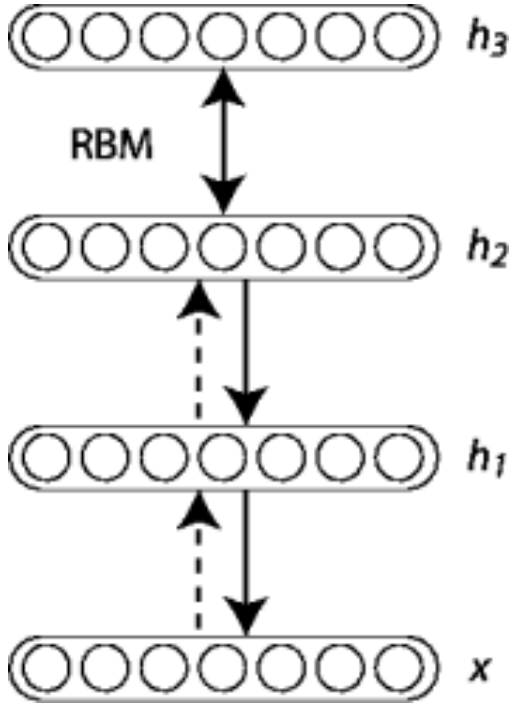
This problem usually referred to as 'gradient diminishing'.

Adaptive versions of SGD are used today.



Deep learning (as it was initially created)

Deep Belief Network (Hinton et al, 2006) was one of the first ways to train deep structures.



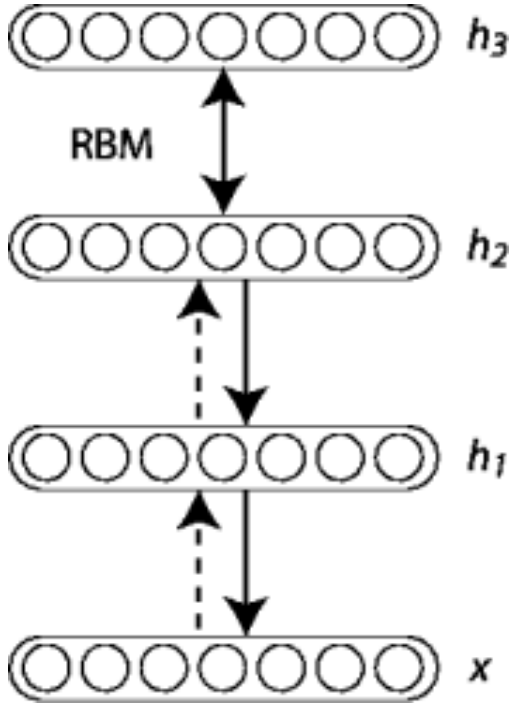
- trained in an unsupervised manner layer-by-layer
- next layer should "find useful dependencies" in previous layer output
- finally, a classification done using output of last layer
  (optionally, minor fine-tuning of the network is also done)

Why this is important?

- we can recognize objects and learn patterns without supervision
- gradient backpropagation in the brain??

Deep learning (as it was initially created)

Deep Belief Network (Hinton et al, 2006) was one of the first ways to train deep structures.



- trained in an unsupervised manner layer-by-layer
- next layer should "find useful dependencies" in previous layer output
- finally, a classification done using output of last layer
  (optionally, minor fine-tuning of the network is also done)

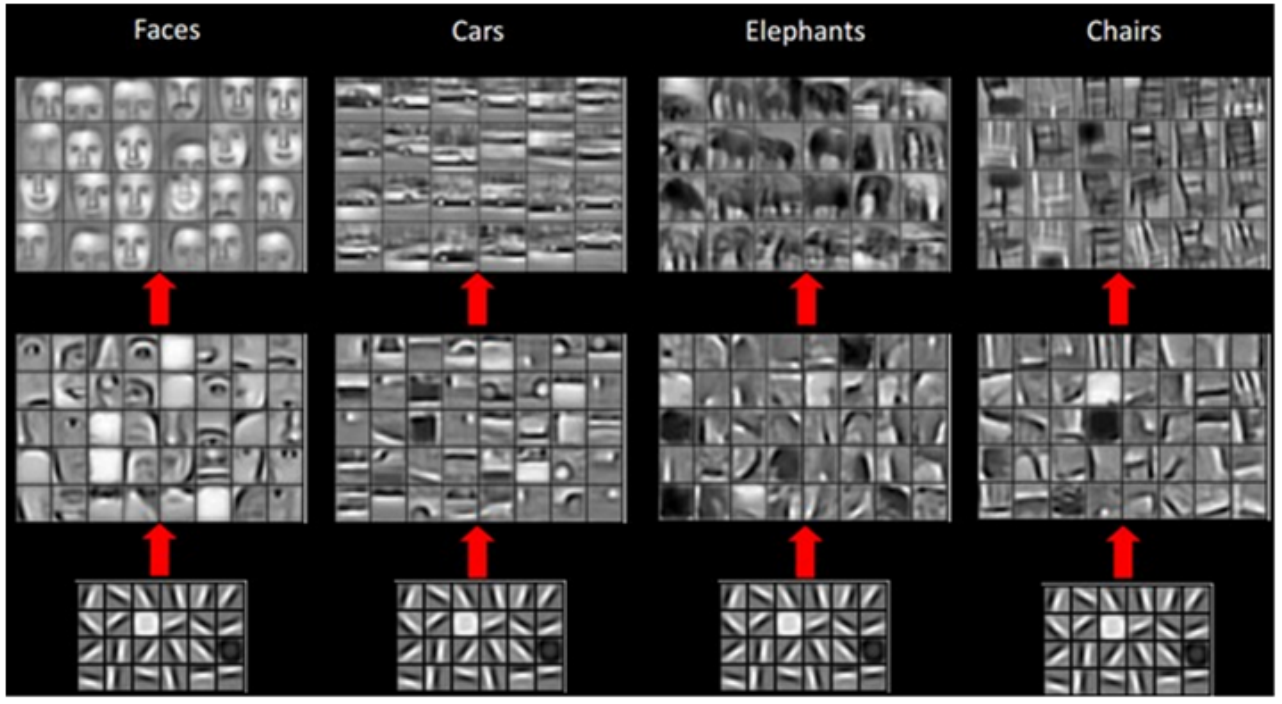
Why this is important?

- we can recognize objects and learn patterns without supervision
- gradient backpropagation in the brain??

Practical? No, today deep learning relies solely on stochastic gradient optimization

Deep learning today

- automatic inference of helpful intermediate representations ...
- with deep architecture
- mostly, trained altogether with gradient-based methods



Inference of high-level information

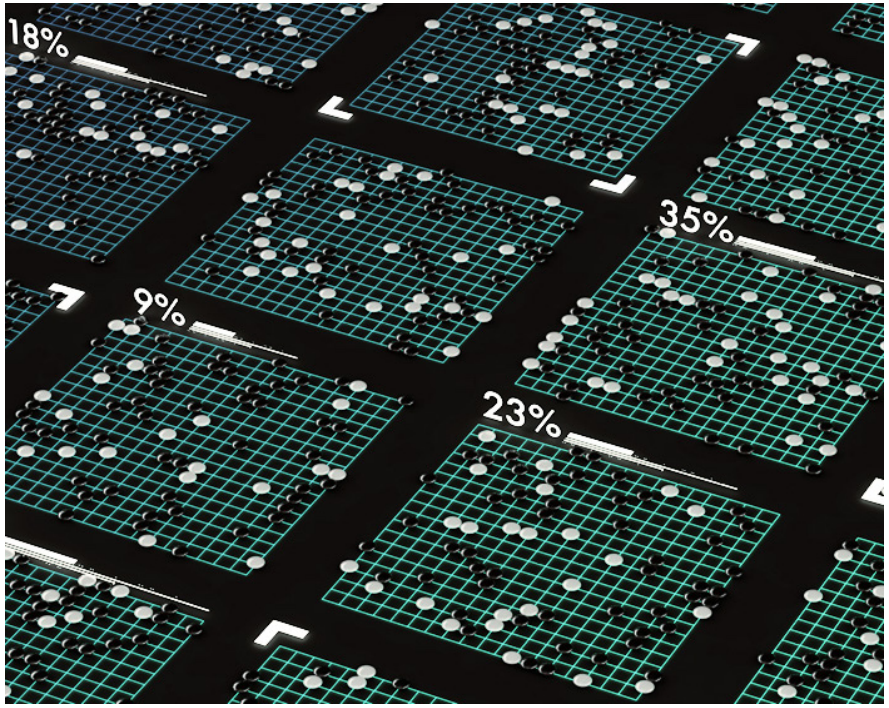
Ability of neural networks to extract high-level features can be demonstrated with artistic style transfer.

Image is smoothly modified to preserve the activations on some convolutional layer.

Sketch to picture (texturing)

AlphaGo

- too many combinations to check
- no reliable way to assess the quality of position

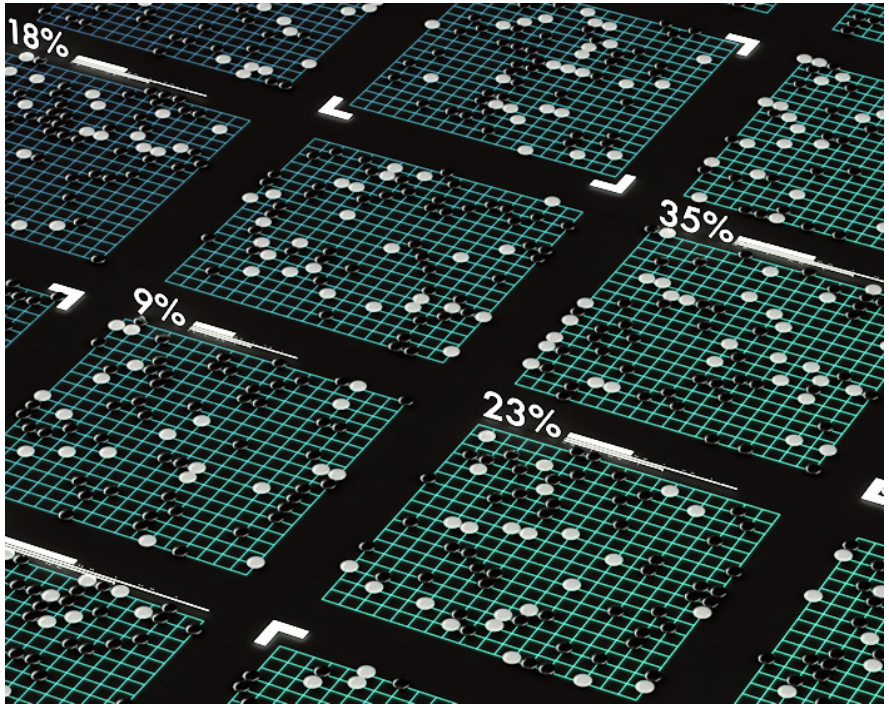


AlphaGo

- too many combinations to check
- no reliable way to assess the quality of position

AlphaGo has 2 CNNs:

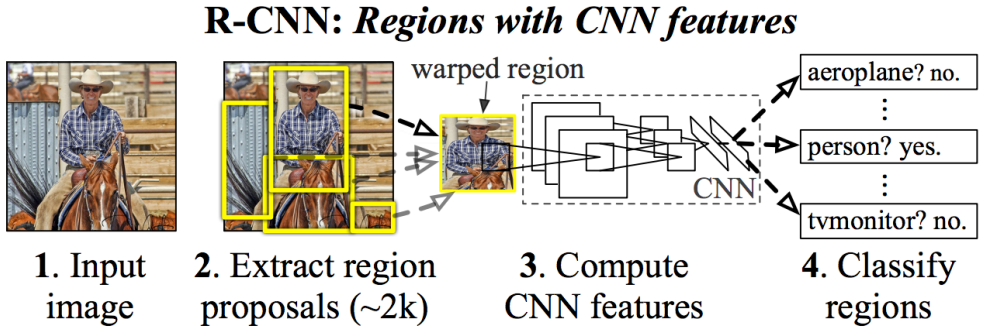
- value network predicts probability to win
- policy network predicts the next step of a player
- both are given the board (with additional features for each position on the board)

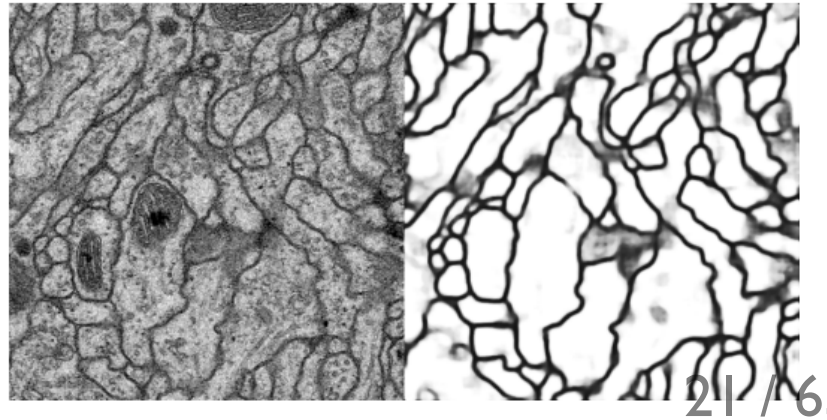


Applications of CNNs

Convolutional neural networks are top-performing on the image and image-like data

- image recognition and annotation
- object detection
- image segmentation
  - segmentation of neuronal membranes

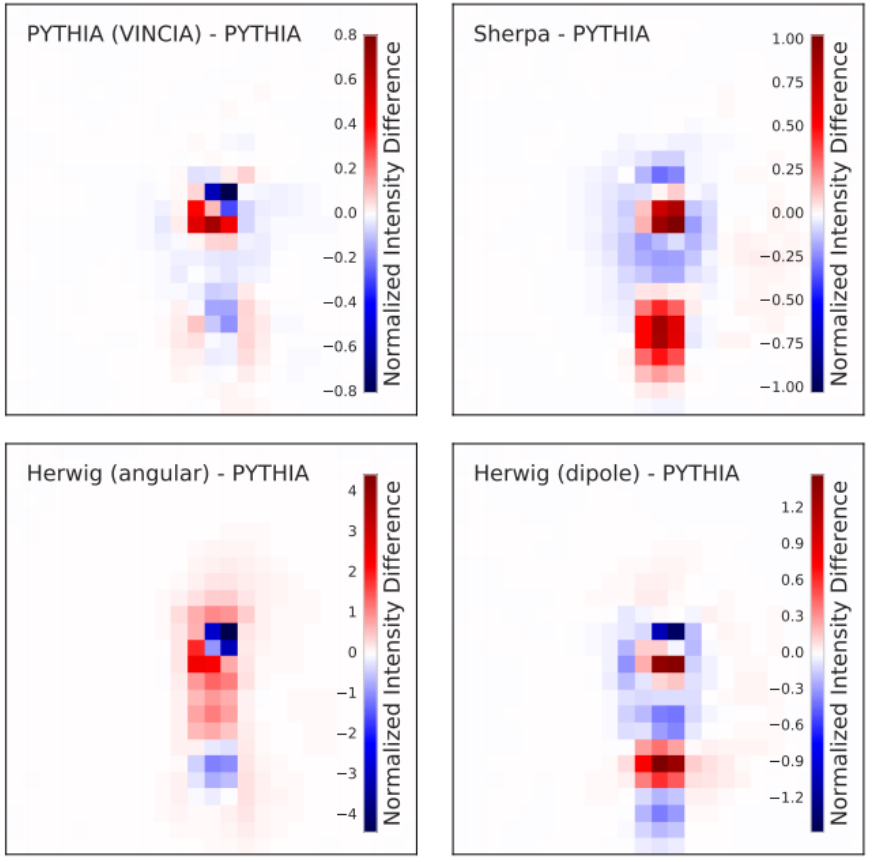




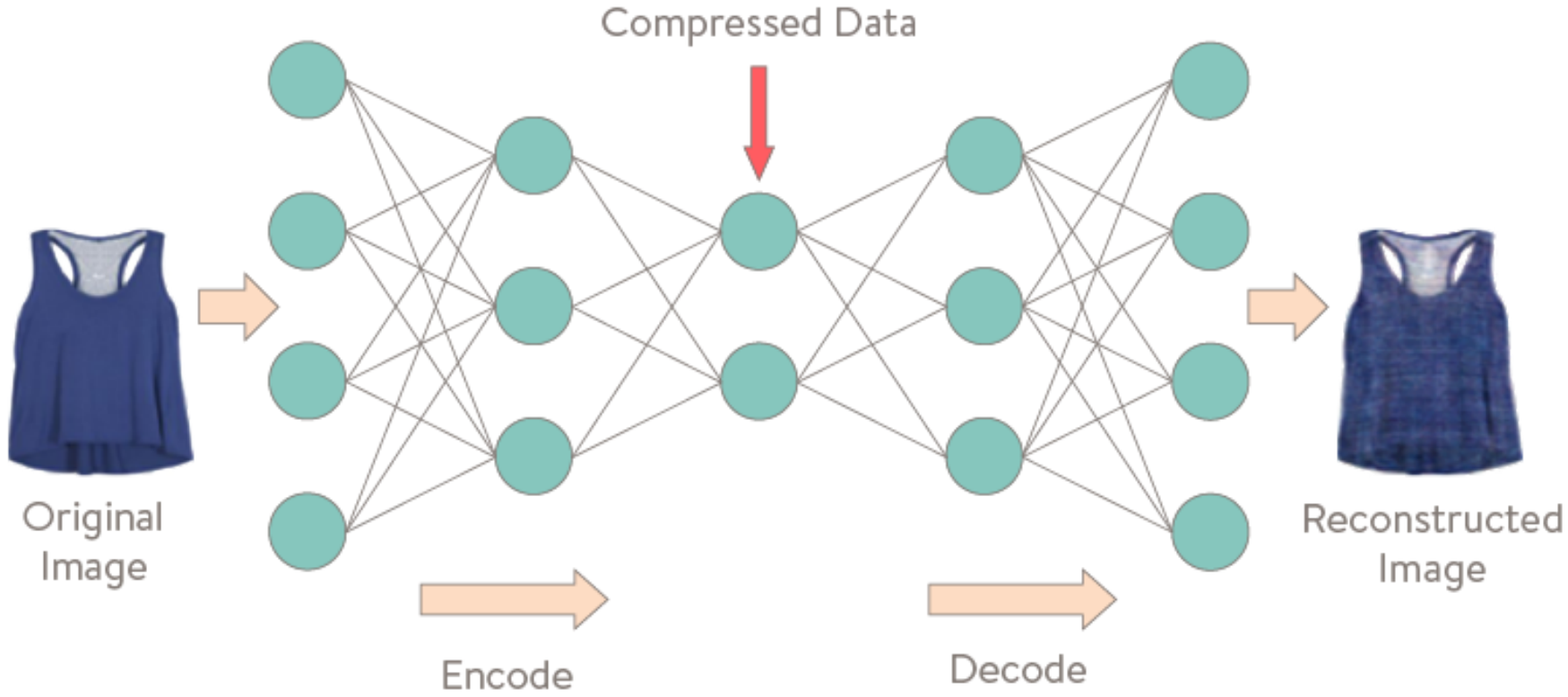
Pitfalls of applying machine learning

CNNs were applied to jet substructure analyses and shown promising results on simulated data, but a simple study shows model applied to different MC (Monte-Carlo) generators output has up to twice higher background efficiency

- thus, learnt dependencies are MC-specific and may turn out to be useless.
- there are approaches to domain adaptation, but their applicability is disputable

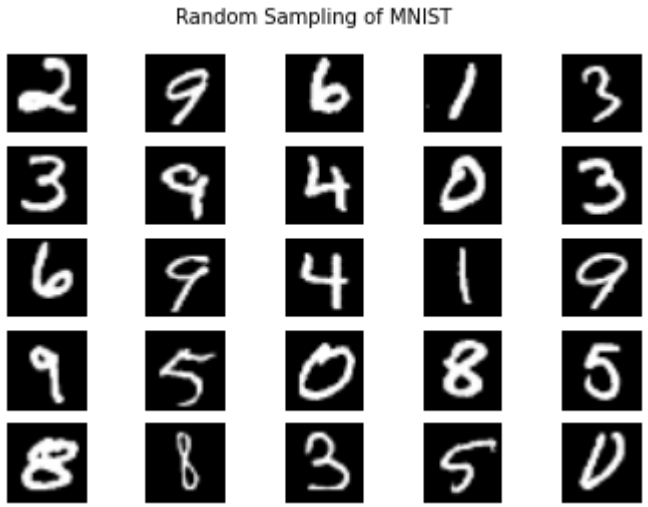


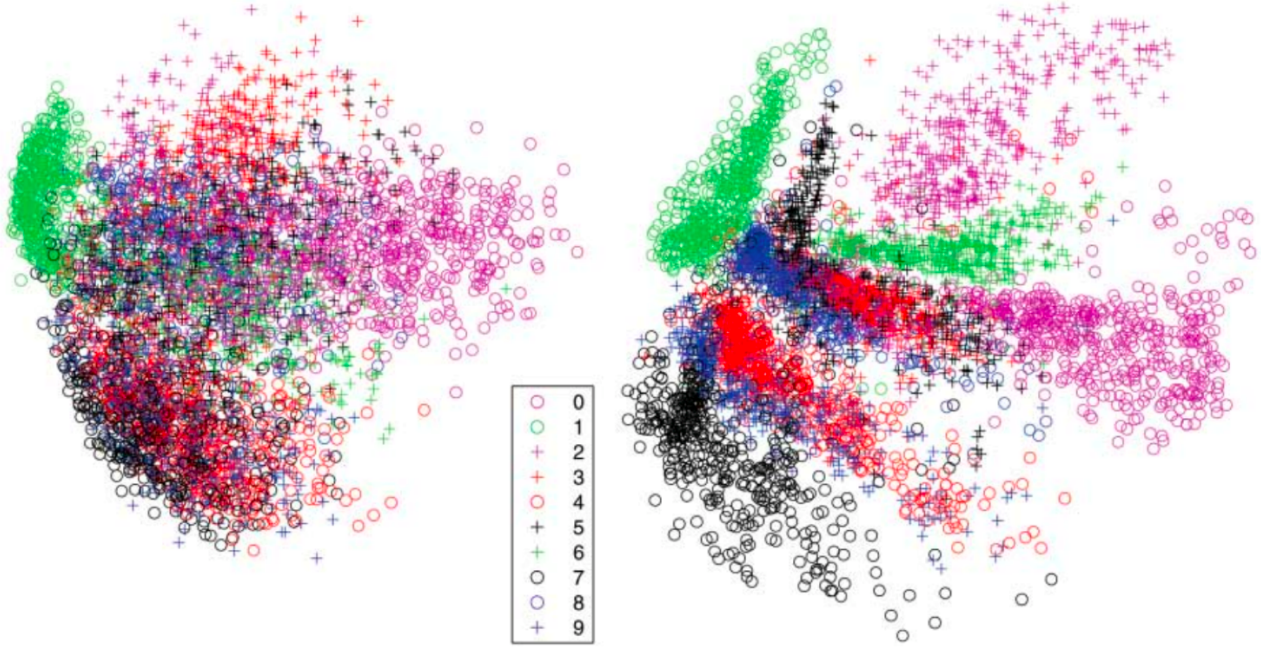
Autoencoding



- compressed data is treated as a result of an algorithm
- trained representations can be made stable against different noise

Autoencoding

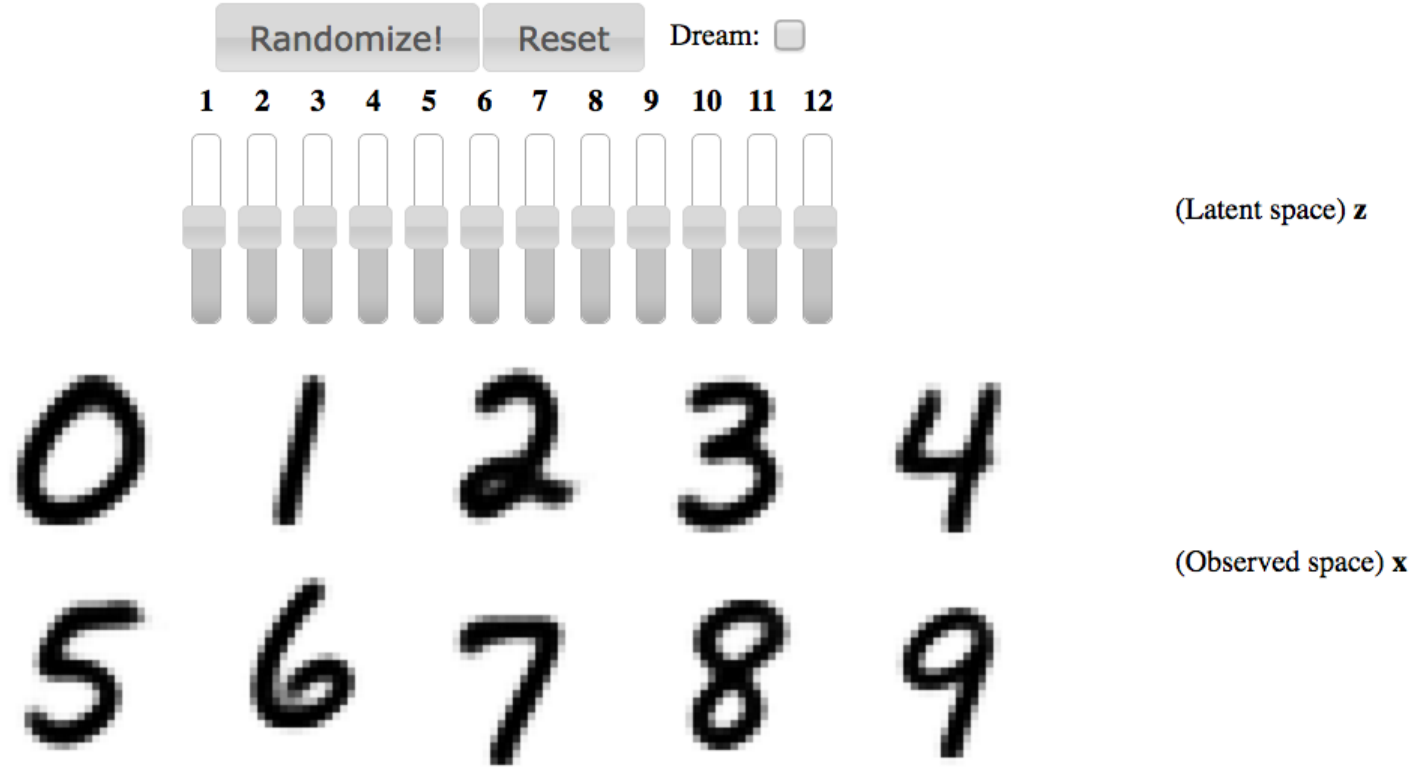




Comparison of PCA (left) and autoencoder with 2 neurons in the middle layer (right)

You can also play with it

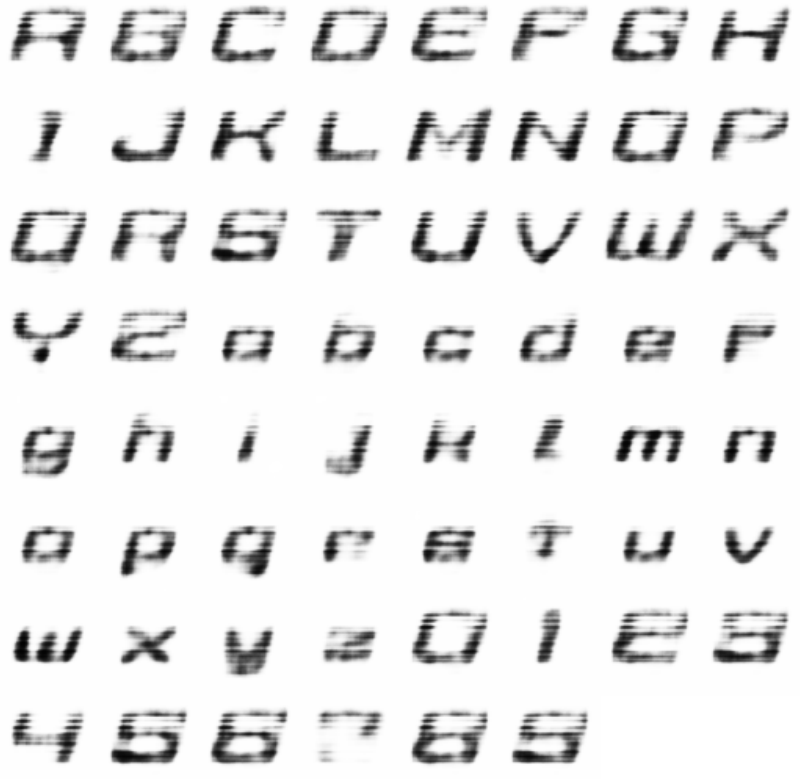
Variational autoencoder



(an example to play with the variables of latent space)

An example with fonts

autoencoder trained to reconstruct symbols from different fonts can generate new fonts when walking along the latent space



Gaze correction (demo)

- shift correction + light correction applied
- training data was collected specially for this task

n-minutes break

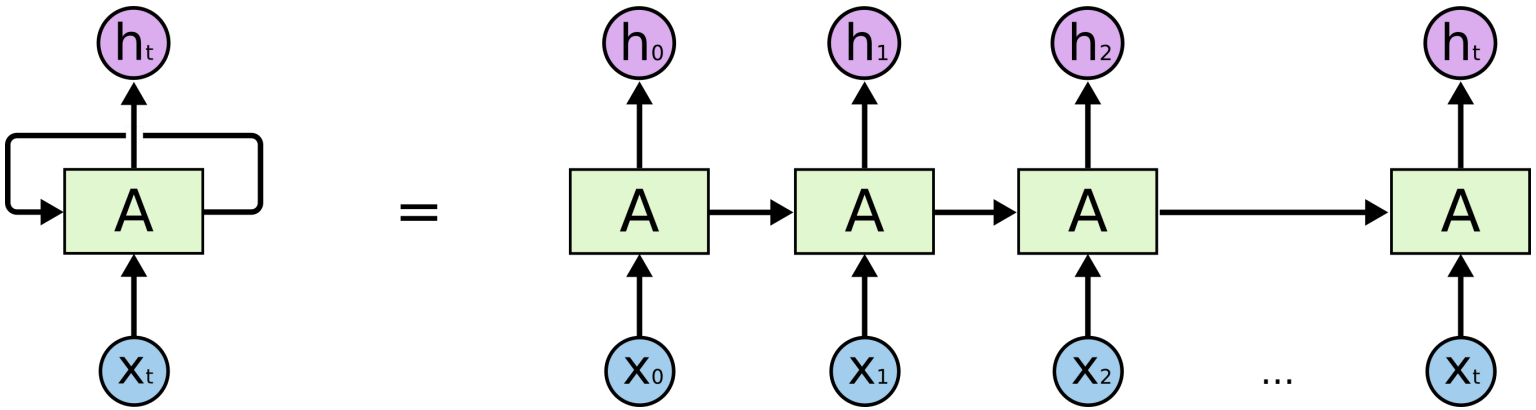
Recapitulation

- analysis of tabular data
- convolutional neural networks
    - and their applications to images and image-like data
- autoencoding

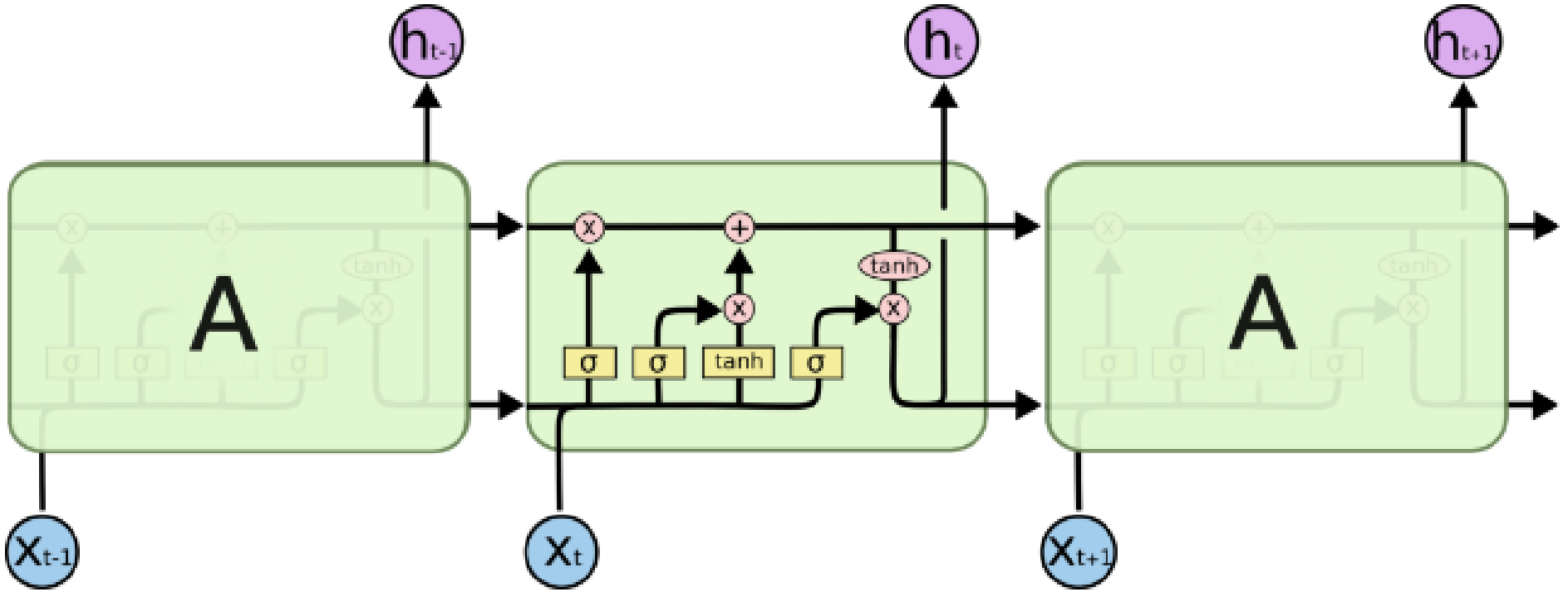
Sequences and Recurrent NN

- sound, text
- any information that is time dependent (daily revenues, stocks prices...)

Inspiration: brain contains information about past and is able to process the information given in sequences.

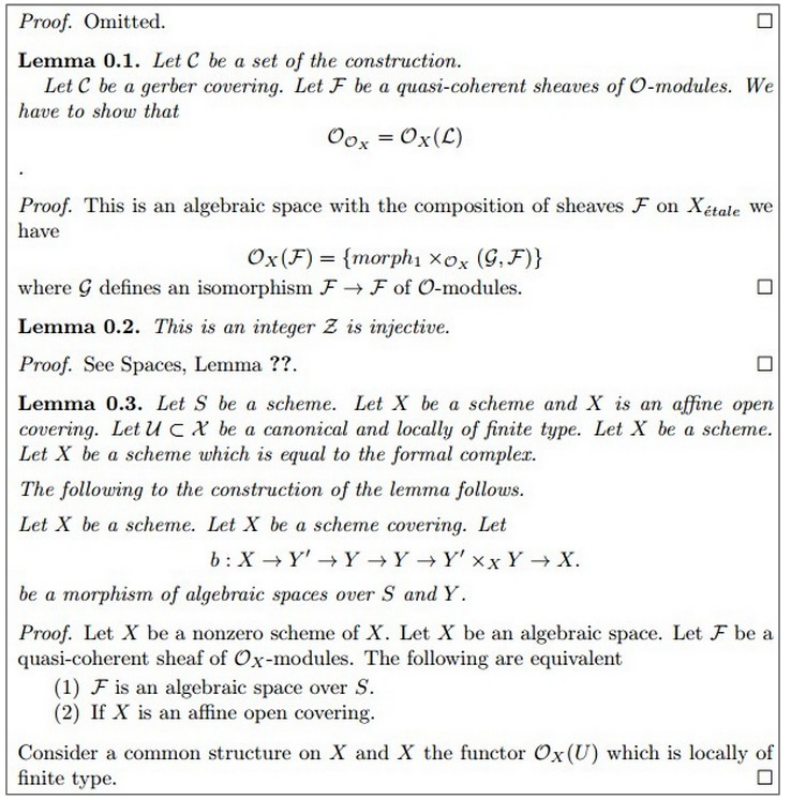


LSTM: a popular recurrent unit



Those are all tensor operations, see post for details

Character-level text generation



Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

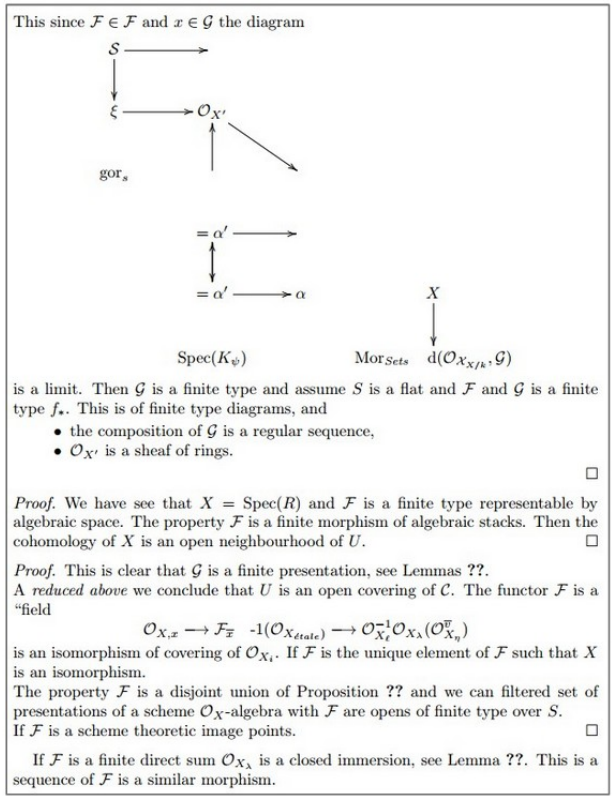
$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □



This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

$$\begin{array}{ccc} \mathcal{S} & \longrightarrow & \\ \downarrow & & \\ \xi & \longrightarrow & \mathcal{O}_{X'} \\ & \text{gor}_s & \uparrow \\ & & \\ & = \alpha' & \longrightarrow \\ & \updownarrow & \\ & = \alpha' & \longrightarrow \alpha \end{array} \qquad \begin{array}{c} X \\ \downarrow \\ \text{d}(\mathcal{O}_{\mathcal{X}_{X/k}}, \mathcal{G}) \end{array}$$

$$\text{Spec}(K_\psi) \qquad \text{Mor}_{Sets}$$

is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type $f_\star$. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \text{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_\iota}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{n}})$$

is an isomorphism of covering of $\mathcal{O}_{X_\iota}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

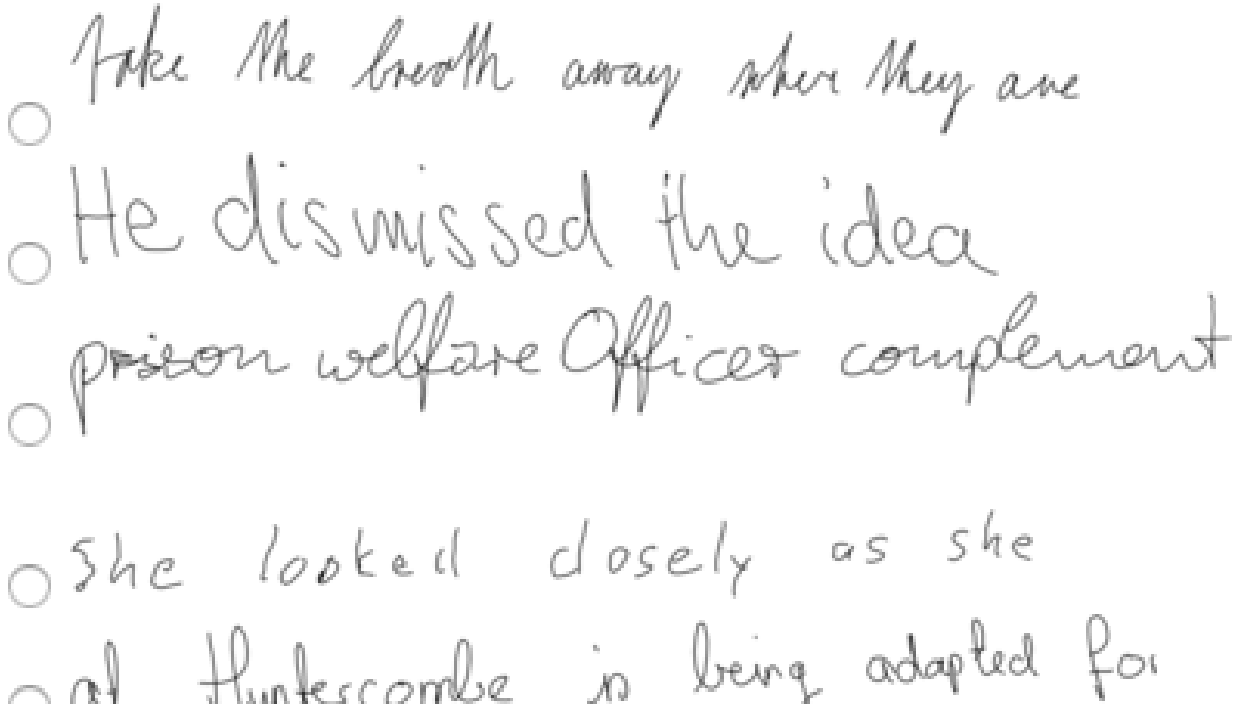
Given a text, network predicts next character. Then this network is used to generate a text character-by-character. More examples

Code generation

Network trained on the linux source:

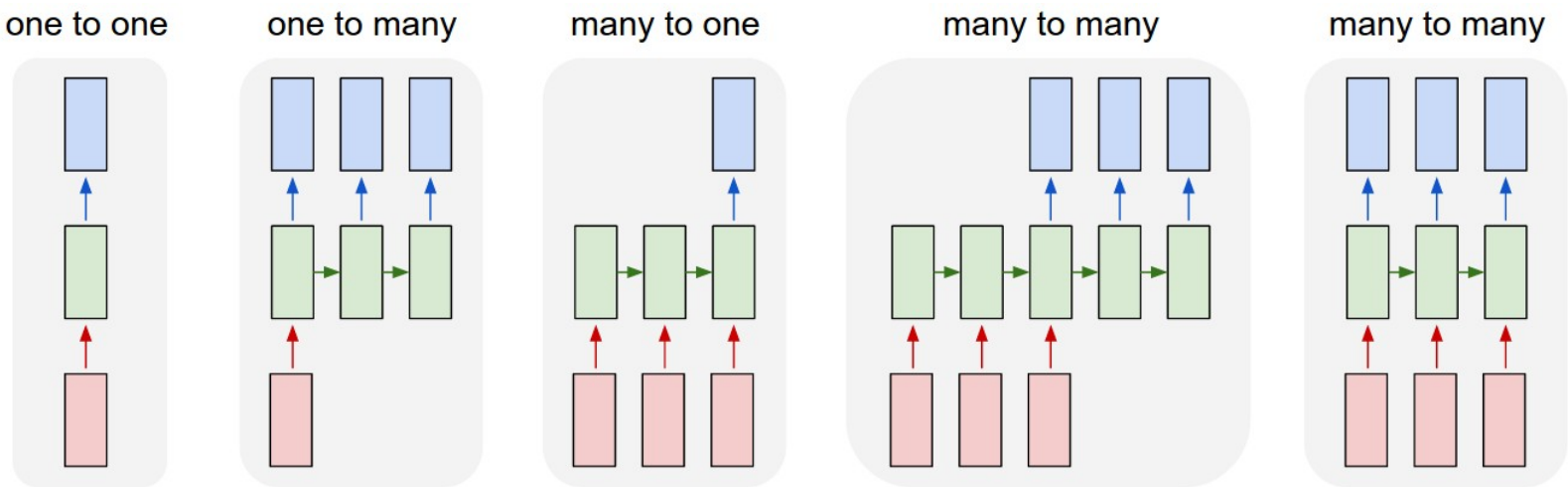
```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

Style in RNNs (demo)



Position of the pen in the next moment is predicted.

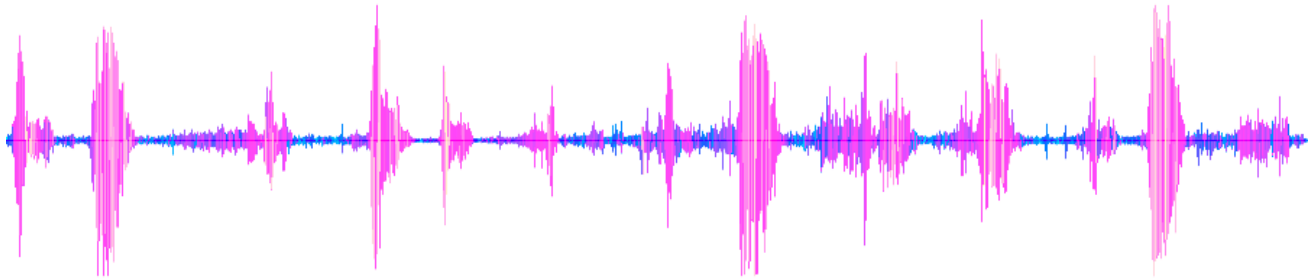
Different strategies for prediction



- one can predict a single observation from sequence
- or map sequence to sequence

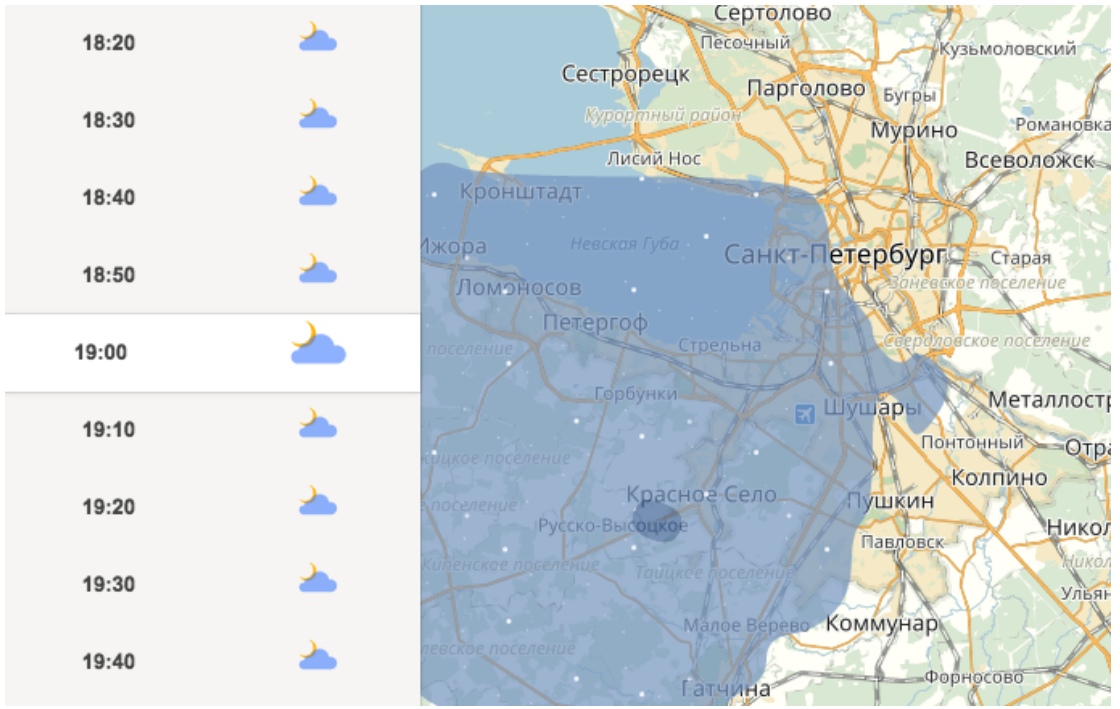
Applications of Recurrent NNs

- natural language processing
- translation
- speech recognition
  - sound record is a sequence!
- and speech synthesis
- demand forecasting in retail systems / stock prices forecasting



Mixing RNN and CNN

Sometimes you need both to analyze dependencies in space and time. For example, in weather forecasting a sequence of photos from satellites [can be used] to predict snow/rain



Why looking at sequence of images?

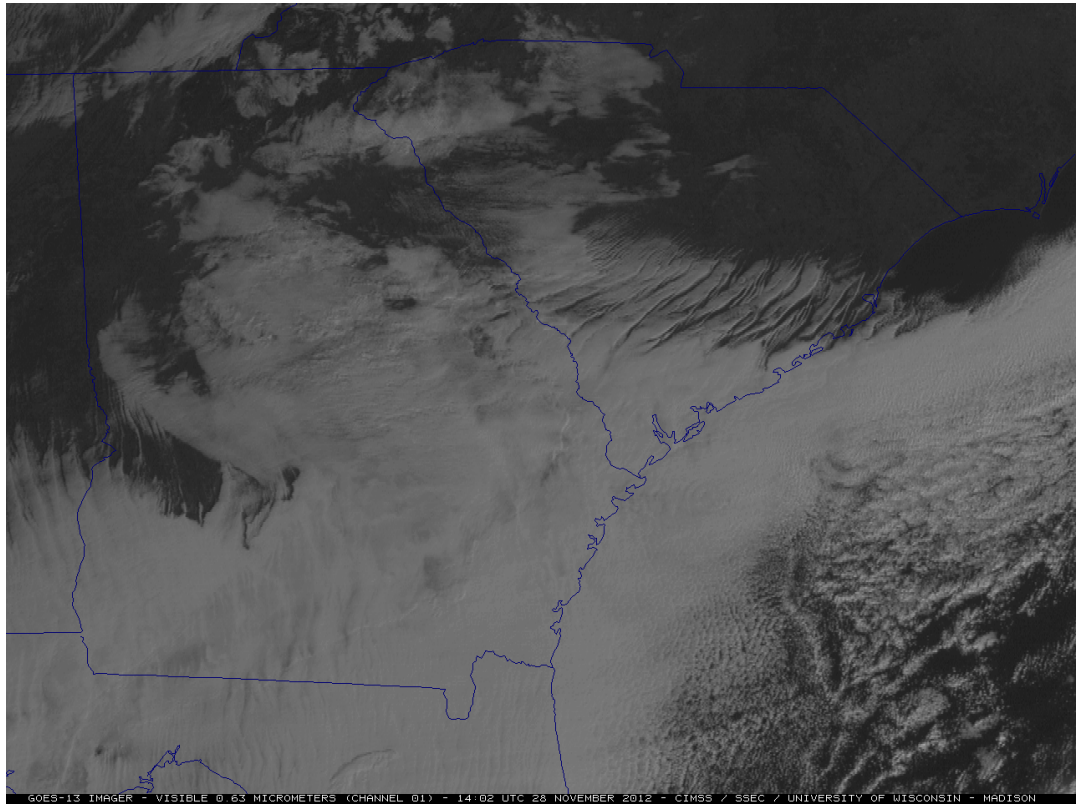


Image taken from [CIMSS Satellite Blog]

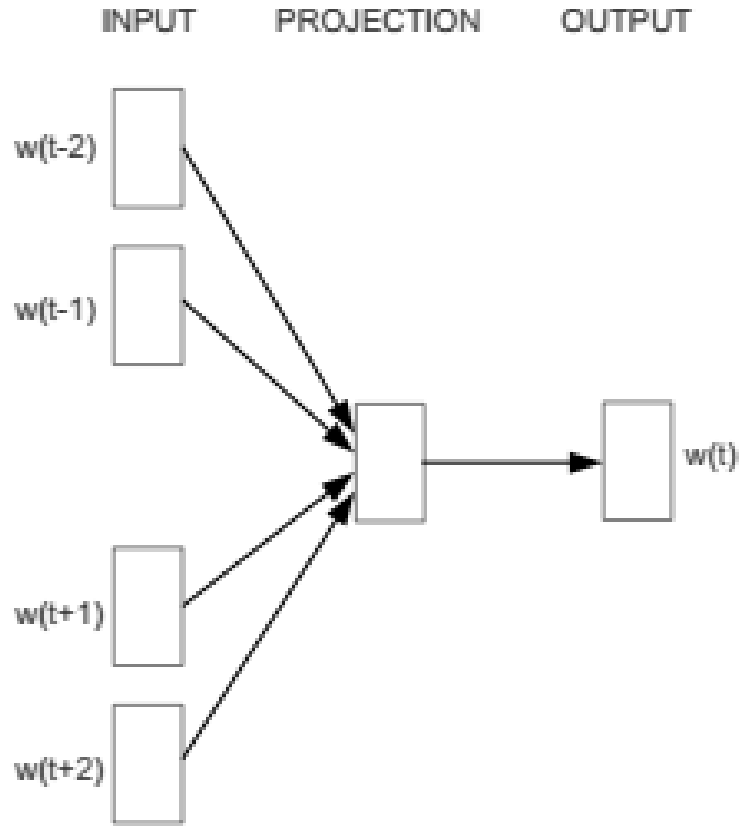
(Word) Embeddings

I would like a ? of coffee.

Skip-gram model takes a context around the word:

- 1 word to the left and one to the right:
  a ? of

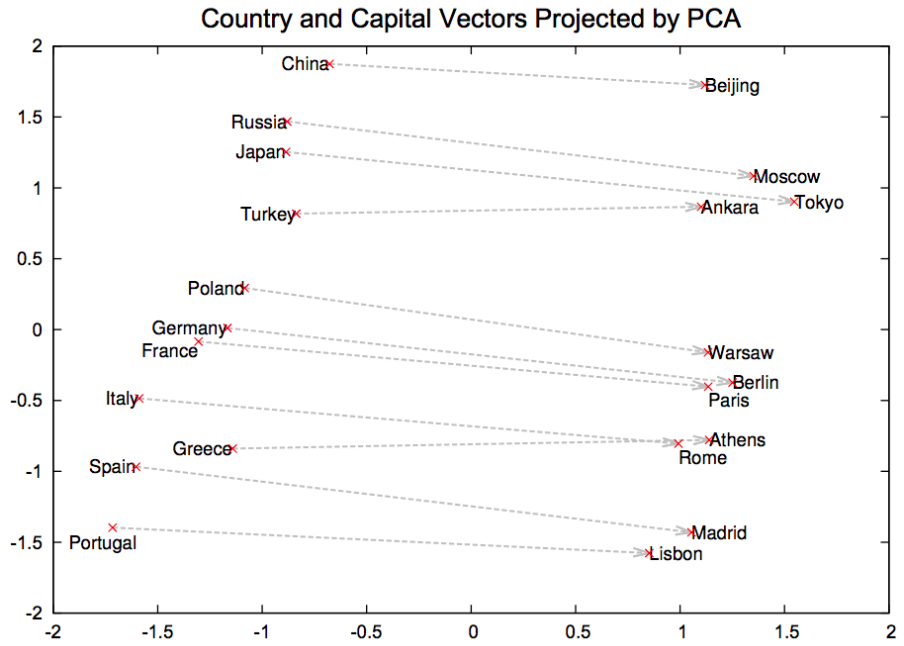
- 2 words to the left and right
  like a ? of coffee



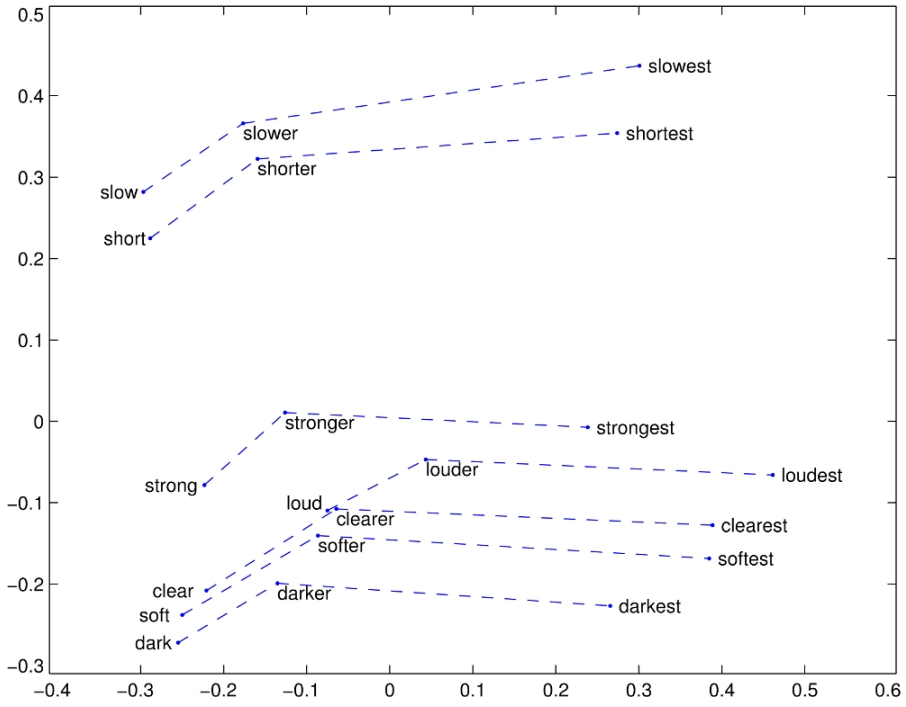
When neural networks are trained to predict which missing words are most probable, at the first stage each word is associated with (trainable) vector — representation.

If model can predict the word well, it already learnt something very useful about the words.

Embeddings



Capitals-countries example from Word2Vec



Comparative-superlitive example from GloVe

Similar (synonymic) words have very close representations.
Embeddings are frequently used to build representations e.g. of text to apply other ML techniques.

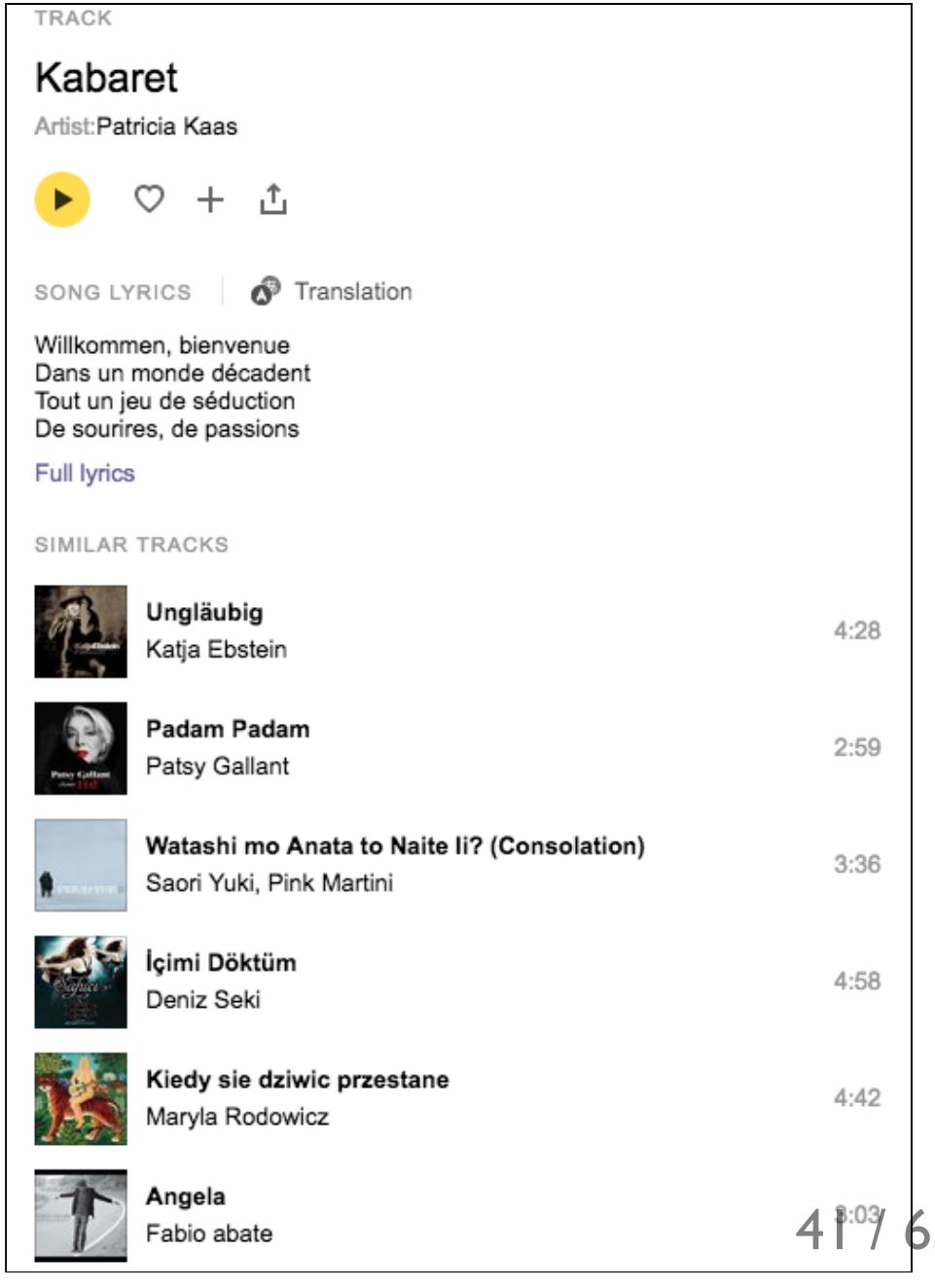
RNN + Embeddings

A problem of pure factorization is the *cold start*: until there is some feedback about a song, you can't infer representation, thus can't recommend it.

To find similar tracks using the track itself:

- RNN is trained to convert sound to embedding
- optimization goal: embeddings of known similar tracks should be closer

This makes it possible to recommend new tracks that have no feedback.



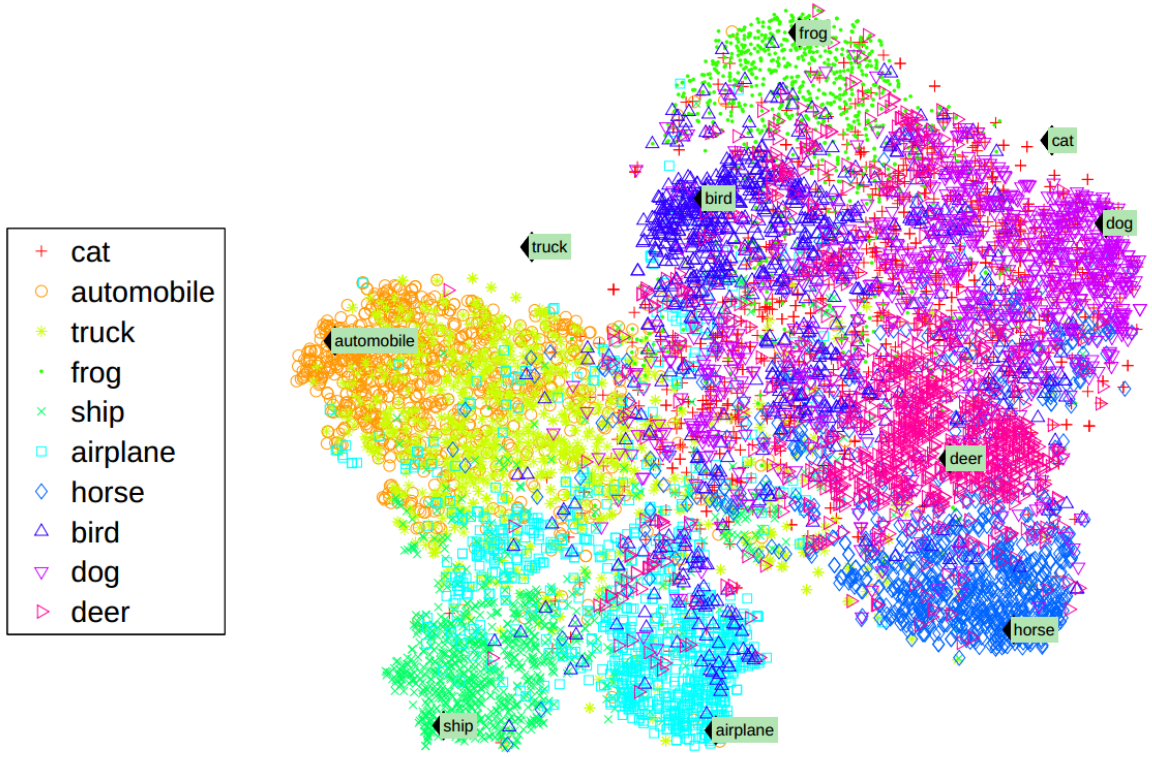
Joint Embeddings

Embeddings can be built for images, sounds, words, phrases.

We can use one space to fit them!

In this example embedding of images and words are shown.

'cat' and 'truck' categories were not provided in the training.



How can we generate new images? (source)

Generative adversarial networks

Discriminator and **G**enerator are working together

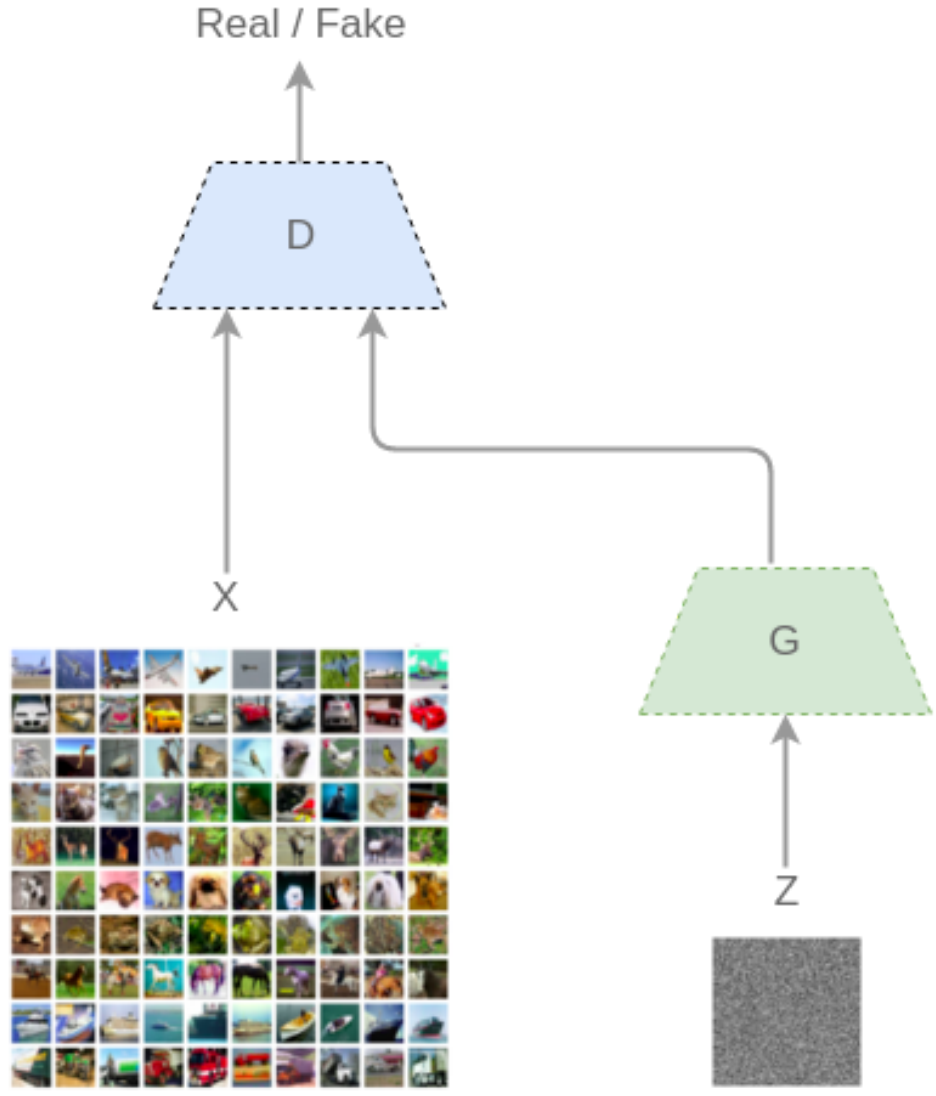
- Generator converts noise to image
- Discriminator distinguishes real images from generated

Log-loss is used, $D(x)$ is a probability that image is fake

$$\mathcal{L} = \mathbb{E}_{x-\text{real}} \log(1 - D(x)) + \mathbb{E}_{\text{noise}} \log D(G(\text{noise}))$$

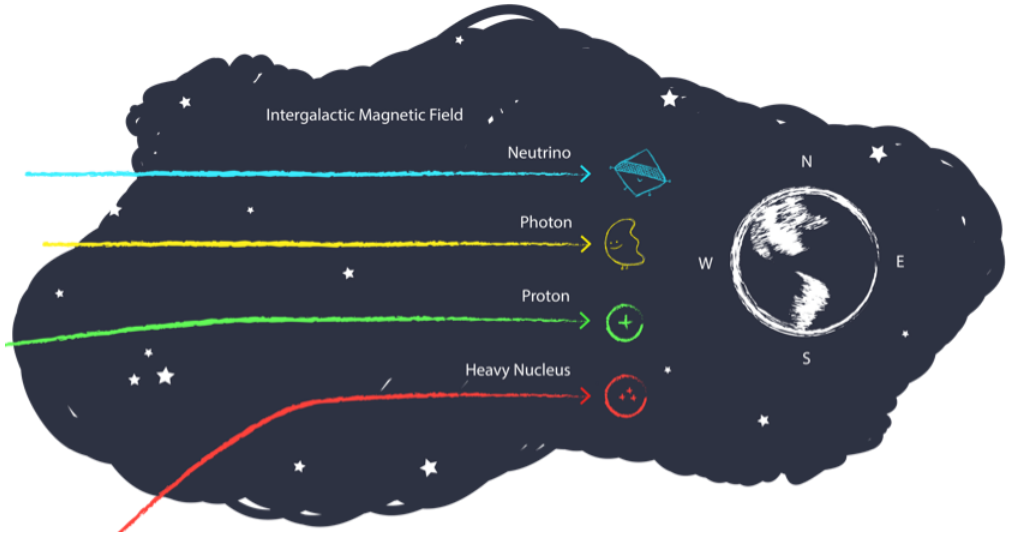
But the optimization is minimax:

$$\mathcal{L} \to \min_G \max_D$$



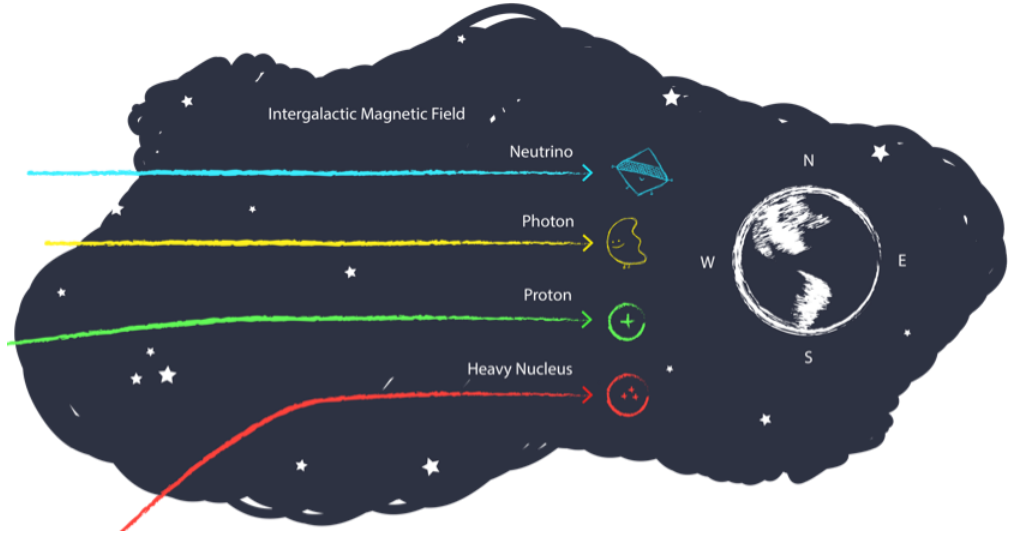
Detecting cosmic rays

- cosmic rays coming from space have huge energies
- we don't know their source
- those can be detected
- need to cover large areas with detectors



Detecting cosmic rays

- cosmic rays coming from space have huge energies
- we don't know their source
- those can be detected
- need to cover large areas with detectors



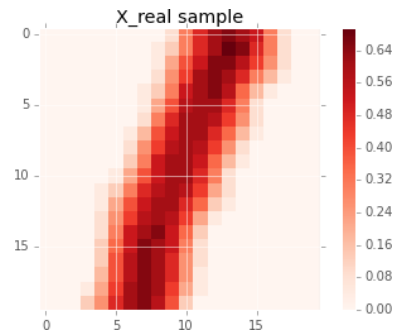
CRAYFIS idea

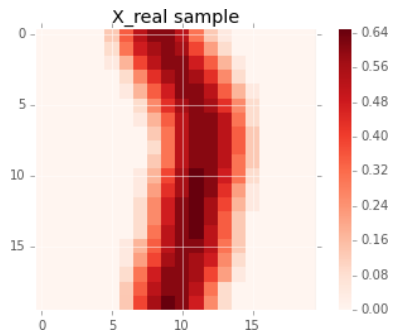
- smartphone camera is a detector
- a problem is to simulate well camera response for the particles

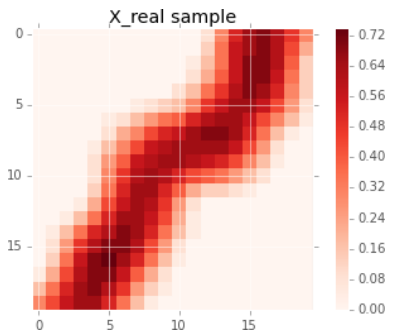
Materials were taken from slides (in russian) by Maxim Borisyak

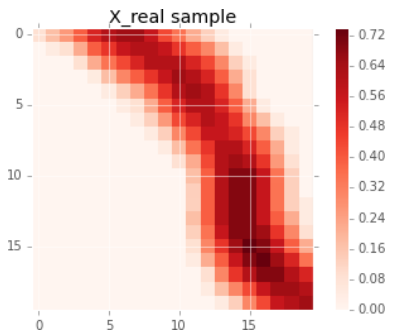
Transforming adversarial networks

We can 'calibrate' the simulation with adversarial networks. That's real data images:

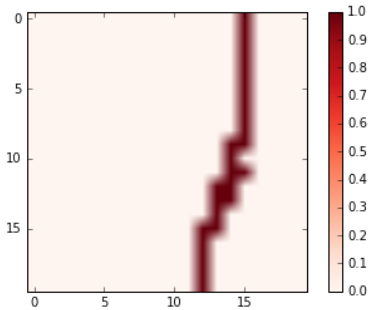




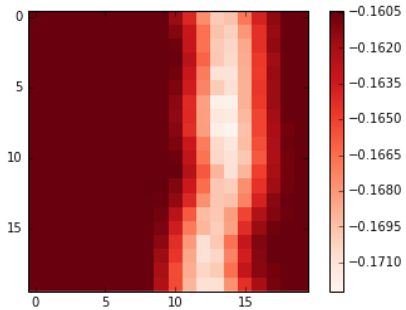




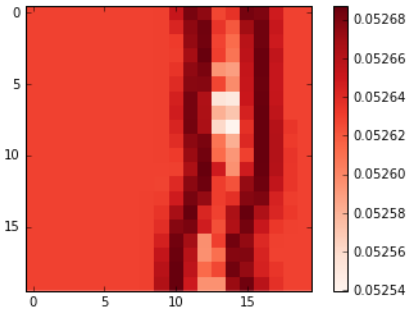
Input is not a noise, but output of a simulator



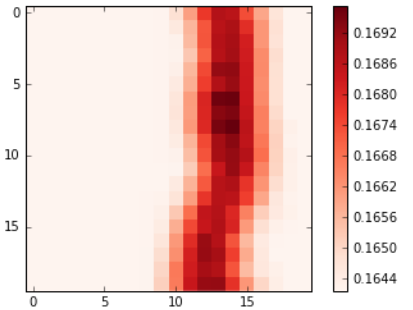
simulator output



1 epoch



2 epochs



3 epochs

Neural networks summary

- NNs are very popular today and are a subject of intensive research
- differentiable operations with tensors provide a very good basis for defining useful predicting models
- Structure of a problem can be effectively used in the structure of the model
- DL requires much data and many resources that became available recently
  - and also numerous quite simple tricks to train it better
- NNs are awesome, but not guaranteed to work well on problems without specific structure

Finding optimal hyperparameters

Motivation

- some algorithms have many hyperparameters (regularizations, depth, learning rate, min_samples_leaf, ...)
- not all the parameters are guessed
- checking all combinations takes too much time

How to optimize this process? Maybe automated tuning?

Finding optimal hyperparameters

- randomly picking parameters is a partial solution
  - definitely much better than checking all combinations
- given a target optimal value we can optimize it

Finding optimal hyperparameters

- randomly picking parameters is a partial solution
  - definitely much better than checking all combinations
- given a target optimal value we can optimize it

Problems

- no gradient with respect to parameters
- noisy results
- function reconstruction is a problem

Finding optimal hyperparameters

- randomly picking parameters is a partial solution
  - definitely much better than checking all combinations
- given a target optimal value we can optimize it

Problems

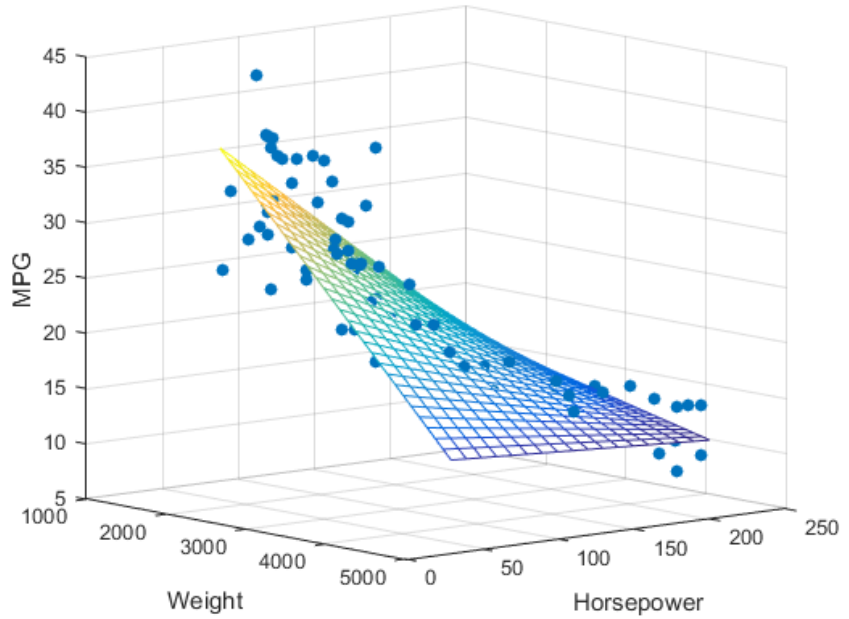
- no gradient with respect to parameters
- noisy results
- function reconstruction is a problem

Practical note: Before running grid optimization make sure your metric is stable (i.e. by train/testing on different subsets).

Overfitting (=getting too optimistic estimate of quality on a holdout) by using many attempts is a real issue, resolved by one more holdout.

Optimal grid search

- stochastic optimization (Metropolis-Hastings, annealing)
  - requires too many evaluations, using only last checked combination
- regression techniques, reusing all known information
  (ML to optimize ML!)



Optimal grid search using regression

General algorithm (point of grid = set of parameters):

1. evaluations at random points
2. build regression model based on known results
3. select the point with best expected quality according to trained model
4. evaluate quality at this points
5. Go to 2 until complete happiness

Question: can we use linear regression in this case?

Optimal grid search using regression

General algorithm (point of grid = set of parameters):

1. evaluations at random points
2. build regression model based on known results
3. select the point with best expected quality according to trained model
4. evaluate quality at this points
5. Go to 2 until complete happiness

Question: can we use linear regression in this case?

Exploration vs. exploitation trade-off: should we try explore poorly-covered regions or try to enhance currently seen to be optimal?

Gaussian processes for regression

Some definitions: $Y \sim GP(m, K)$, where m and K are functions of mean and covariance: $m(x)$, $K(x, \tilde{x})$

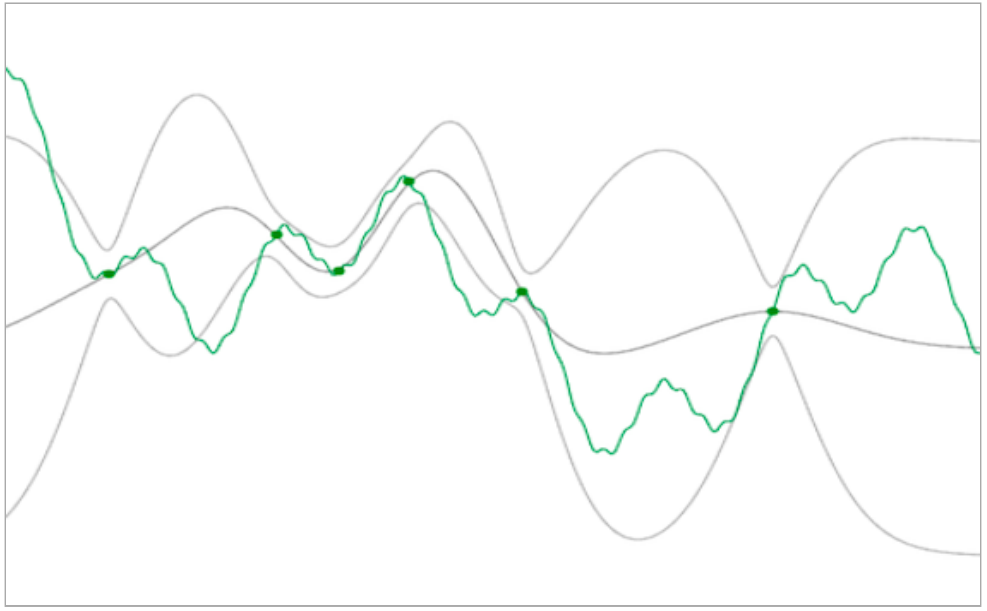
- $m(x) = \mathbb{E}Y(x)$ represents our prior expectation of quality (may be taken conszero)
- $K(x, \tilde{x}) = \mathbb{E}Y(x)Y(\tilde{x})$ represents influence of known results on the expectation of values in new points
- RBF kernel is used here too: $K(x, \tilde{x}) = \exp\left(-\frac{||x-\tilde{x}||^2}{h^2}\right)$
- Another popular choice: $K(x, \tilde{x}) = \exp\left(-\frac{||x-\tilde{x}||}{h}\right)$

We can model the posterior distribution of results in each point.

Gaussian processes and active learning demo

Active learning with gaussian processes

- Gaussian processes model posterior distribution at each point of the grid
- we know at which point we have already well-estimated quality
- and we are able to find regions which need exploration (those have large variance)



See also this demo.

Other applications of gaussian processes

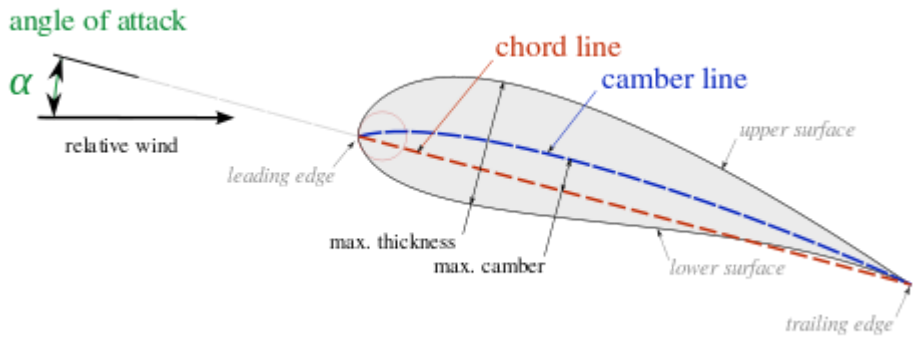
Gaussian processes are good at black-box optimization when computations are costly:

Optimizing parameters of Monte-Carlo simulations in HEP

- the goal is to decrease mismatch with real data

Optimizing airfoil of an airplane wing

- to avoid testing all configurations, gaussian processes are used
- gaussian processes provide a surrogate model which encounters two sources:
  - high-fidelity, wind tube tests
  - low-fidelity, simulation



Active learning

- optimization of hyperparameters is a major usecase
- can deal with optimization of noisy non-continuous functions

- machine learning should adapt dynamically to new situations
  - new trends / new tracks / new pages
- active learning is a way to include *automated experimenting* in machine learning

Instead of conclusion

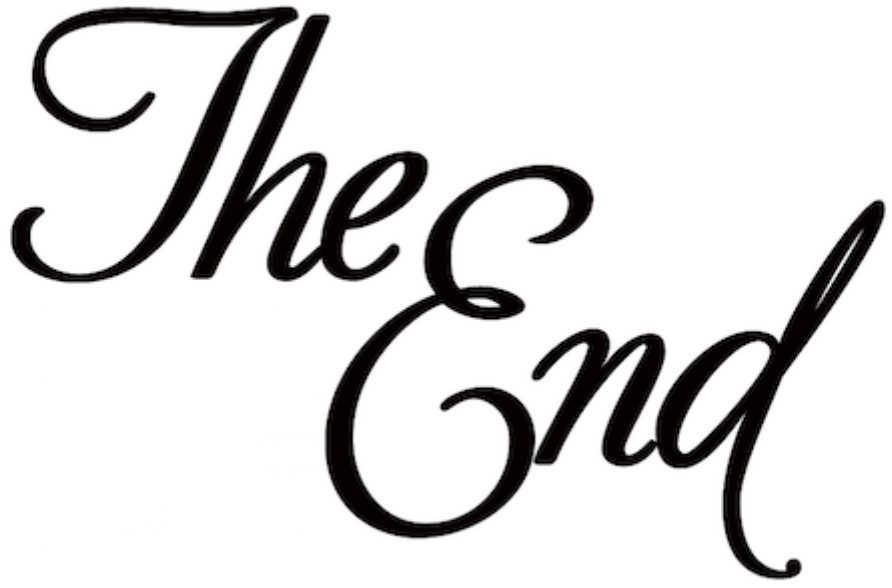
Summary of the lectures

- data arrives today in huge amounts
- there is much interest in obtaining useful information from it
  - both in science and in industry
- problems of interest become more and more complex

Summary of the lectures

- data arrives today in huge amounts
- there is much interest in obtaining useful information from it
  - both in science and in industry
- problems of interest become more and more complex

- machine learning techniques become more sophisticated and more powerful
- and influence our world



This mini-course was prepared by Yandex School of Data Analysis — a leading educational center in machine learning and data science in Russia.