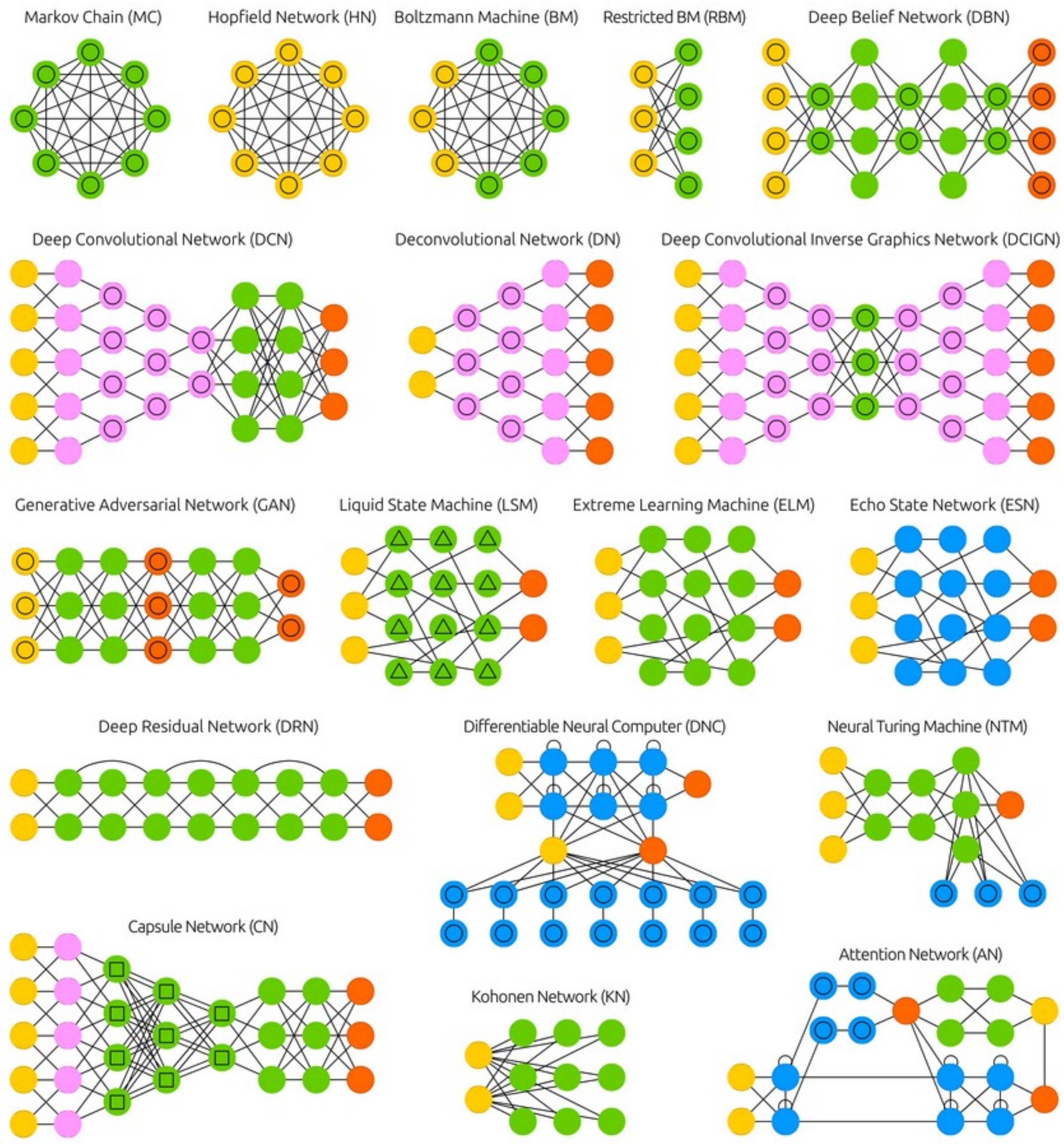
Machine learning
Lecture 9



- Mixed Density Network

Marcin Wolter

IFJ PAN

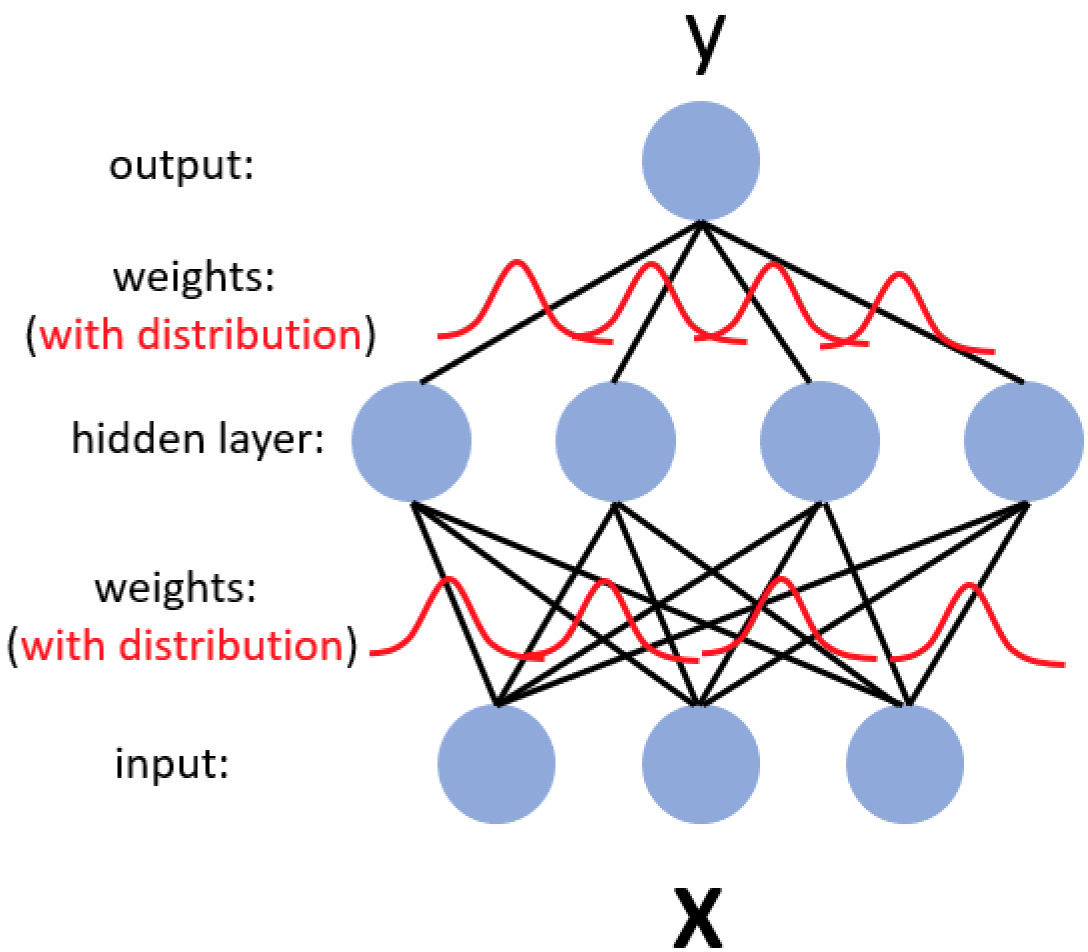
14 February 2020

A Deep Neural Network Applet

Applet showing the performance of Deep Convolutional NN:

http://cs.stanford.edu/people/karpathy/convnetjs/

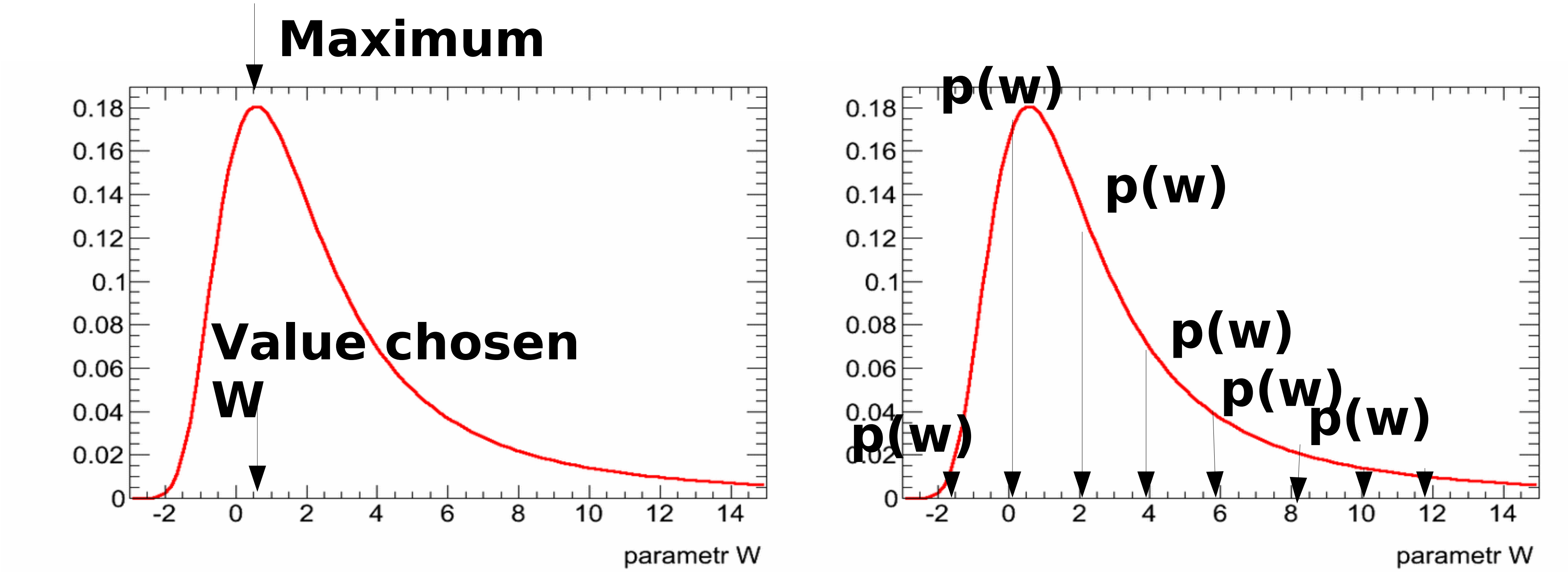
We have already learned about Bayesian Neural Networks



Bayesian neural networks differ from plain neural networks in that their weights are assigned a probability distribution instead of a single value or point estimate.

These probability distributions describe the uncertainty in weights and can be used to estimate uncertainty in predictions.

Machine and bayesian learning



Machine learning

We chose one function (or a value of a parameter describing the function).

Bayesian learning

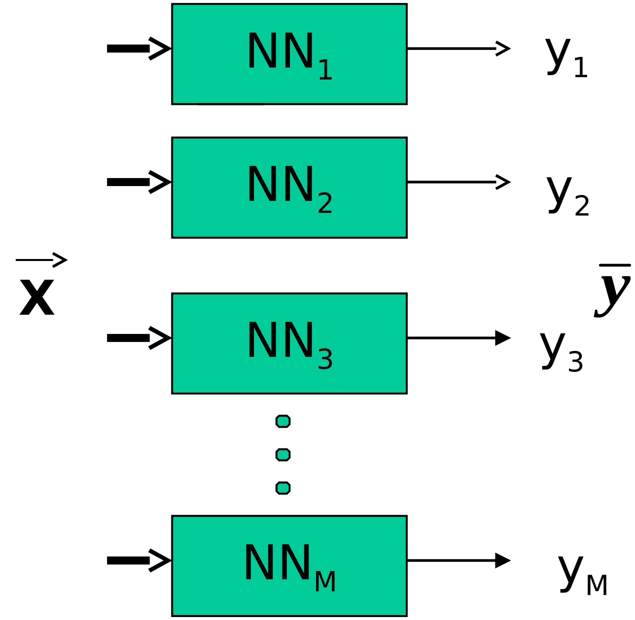
Each function (or a parameter value) is given some probability (weight).

Implementation of bayesian networks:

Instead of choosing a single set of weights describing the NN we should find the probability density for the entire space of weights.

Many neural networks.

Having many NN we can get the weighted mean or he most probable network and also the estimation error

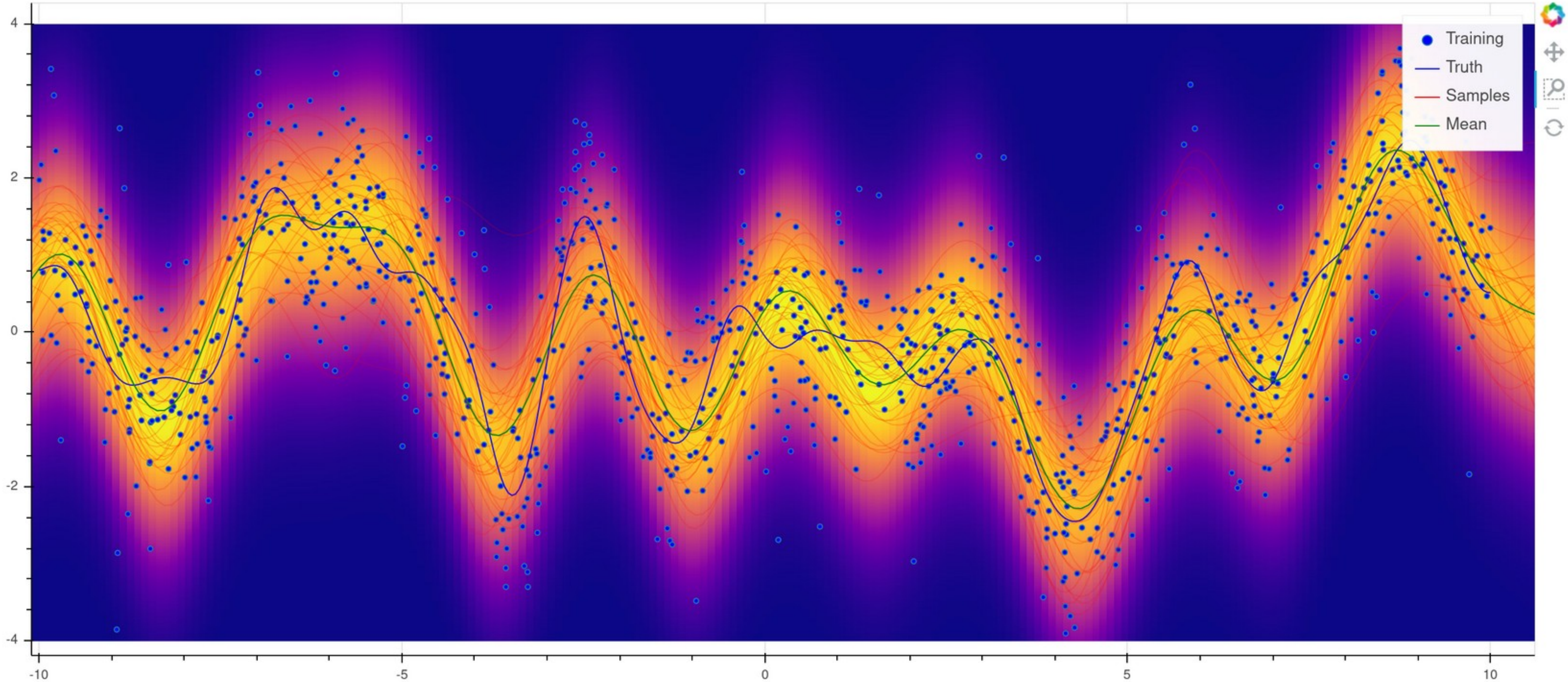


$$\bar{y} = \sum_i a_i y_i(x)$$

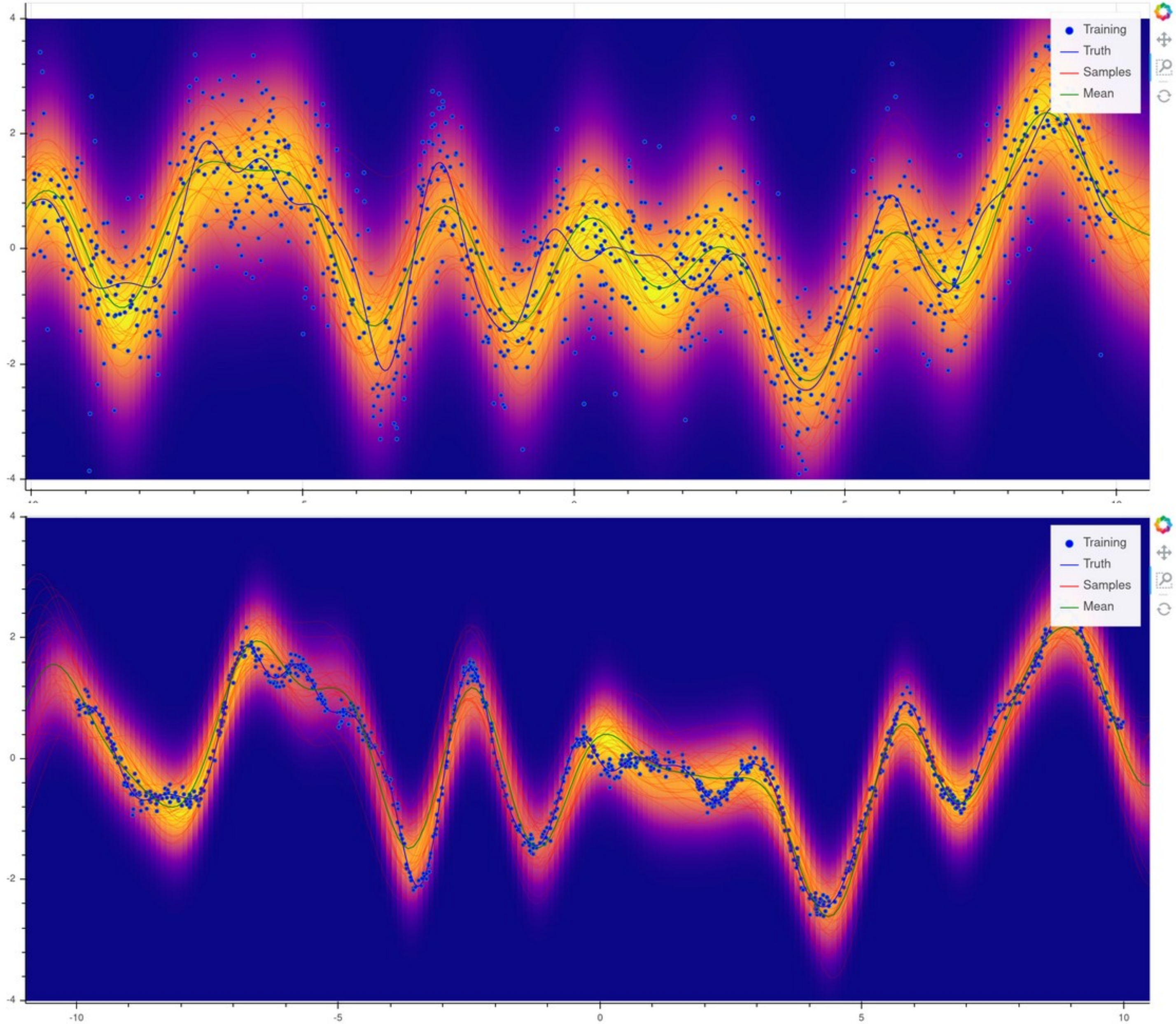
C.M. Bishop
"Neural Networks for Pattern Recognition",
Oxford 1995

Regression BNN example

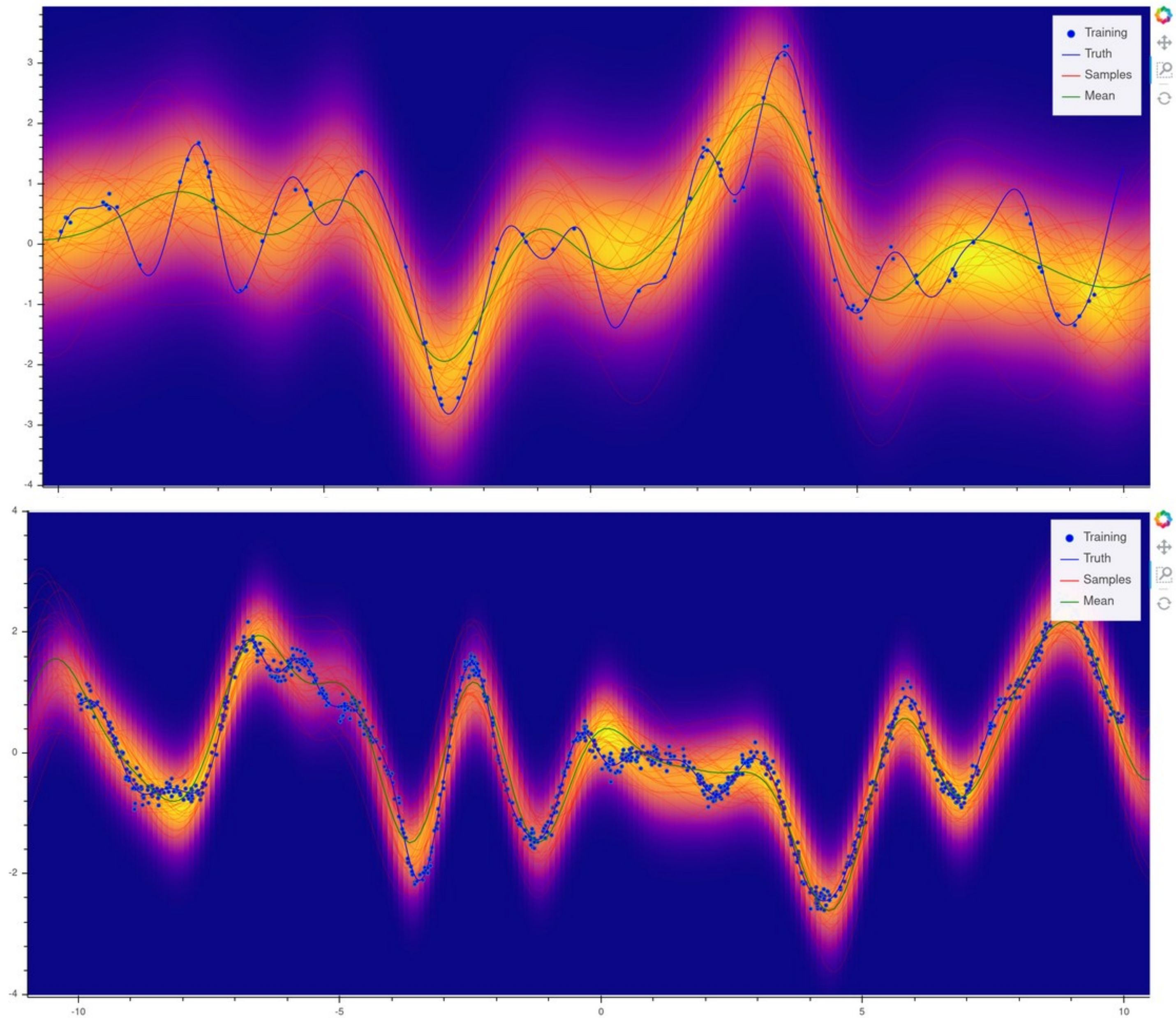
- Fit using Neural Network – example codes from https://github.com/gradientinstitute/aboleth/tree/master/demos
- Fit to the points drawn from the true function (blue) https://github.com/marcinwolter/MachineLearnin2019/blob/master/regression



Different noise

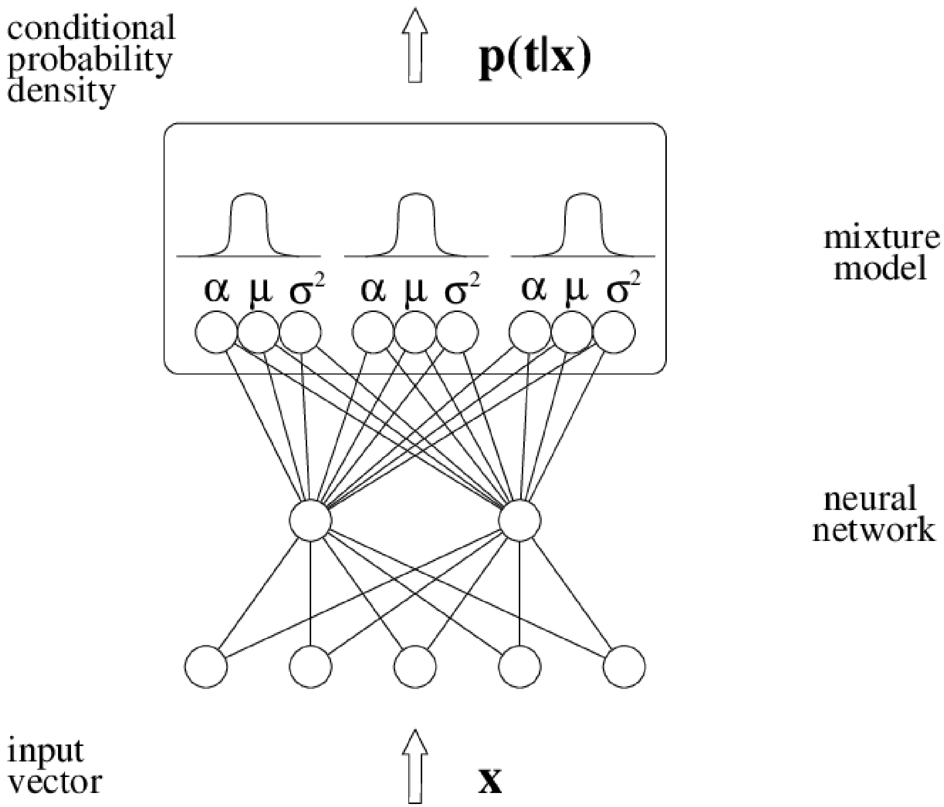


Different number of data points



Where is a problem?

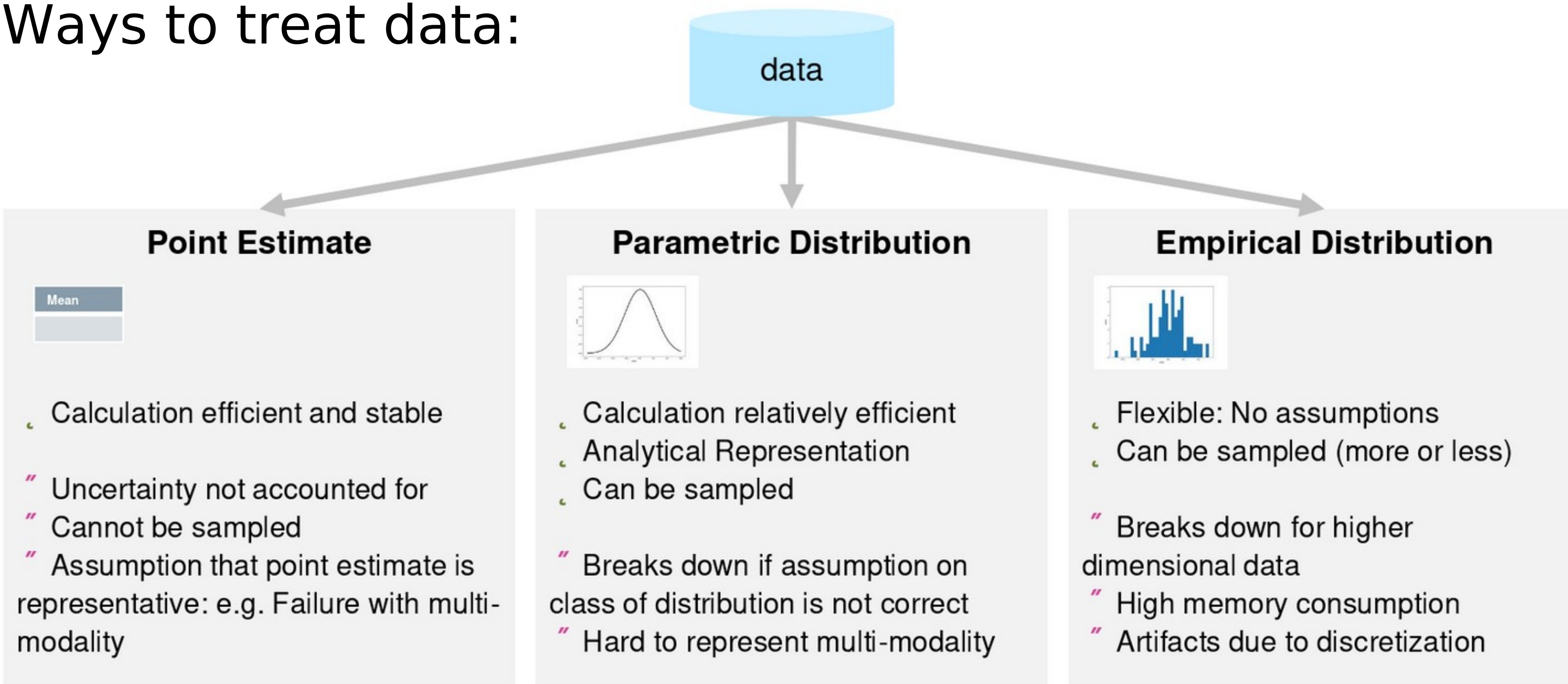
- We had to train many, many networks… It takes time.
- So, maybe there is a way to do it in a more efficient way?
- Idea:
  - Create a network, which returns the parametrized probability distribution



It is called a Mixed Density Network (MDN).
The output is parametrized by few Gaussian distributions.

Mixture Density Network (MDN)

Ways to treat data:



Classic Neural Network

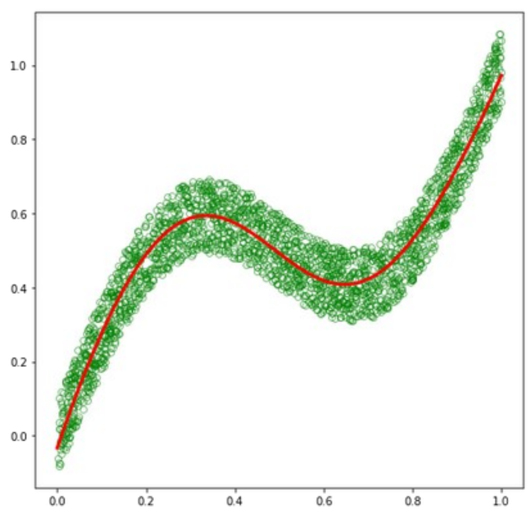
Mixture Density Neural Network

Bayesian Networks

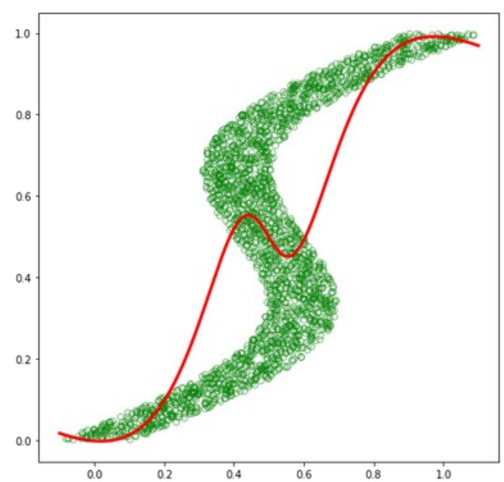
Mixture Density Network (MDN)

- MDN is an interesting model formalism working on supervised learning problems in which the target variable cannot be easily approximated by a single standard probability distribution.
- Conditional probability distribution *p(y|x)* is modeled as a mixture of distributions (few Gaussians), in which the individual distributions and the corresponding mixture coefficients are parametrized by functions of the inputs *x*.

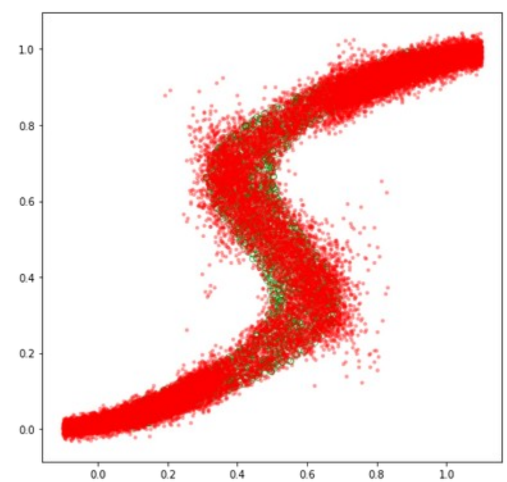
Regression:



Regular network - OK



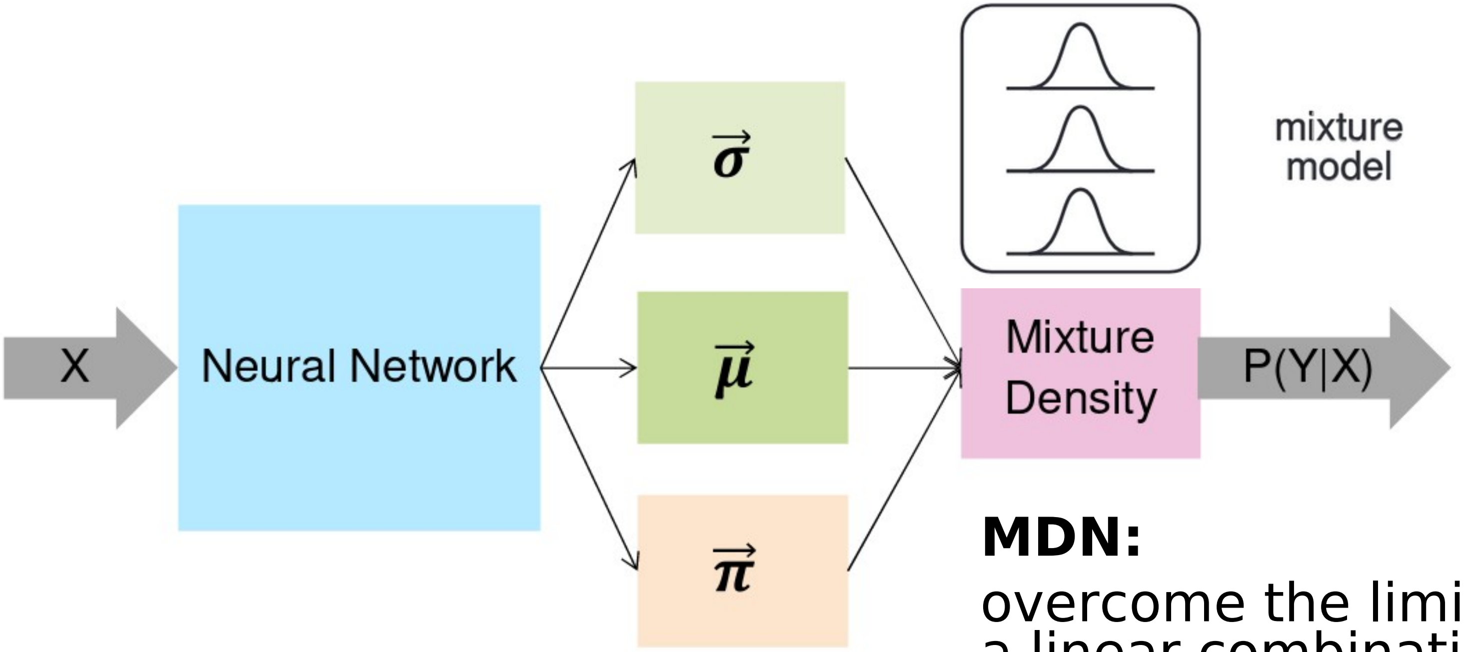
Regular network - problems



MDN sampling - OK

Nice presentation:
http://www.dbs.ifi.lmu.de/Lehre/DLAI/WS18-19/script/06_uncertain.pdf

Mixture Density Network (MDN)



$\vec{\sigma}$: standard deviations
$\vec{\mu}$: means
$\vec{\pi}$: cluster weights

Loss: $-\log P(\mathbf{Y}|\mathbf{X})$

MDN:
overcome the limits by using a linear combination of kernel function

MDN returns not only maximum probability value, but the probability distribution: it returns errors!

Reference: *Bishop, Christopher M. Mixture density networks. Technical Report NCRG/4288,Aston University, Birmingham, UK, 1994*
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.120.5685&rep=rep1&type=pdf

Running MDN example

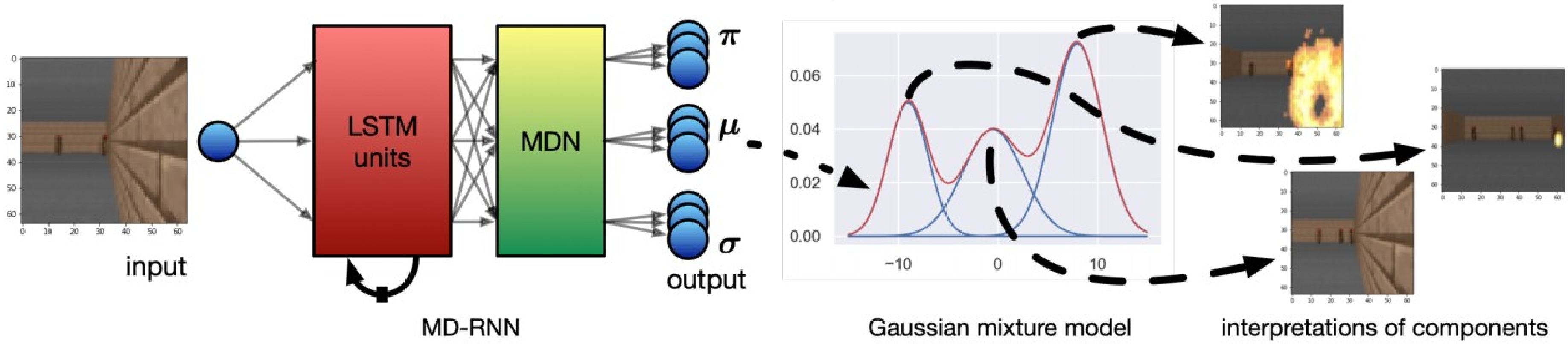
Example from
https://github.com/cpmpercussion/keras-mdn-layer

Google Colaboratory runable adapted example:
https://github.com/marcinwolter/MachineLearnin2019/blob/master/MDN_1D_sine_prediction.ipynb

An application example

How do Mixture Density RNNs Predict the Future? Kai O. Ellefsen, Charles P. Martin,Jim Torresen
https://arxiv.org/pdf/1901.07859.pdf

Problem: predict the future in the computer game.



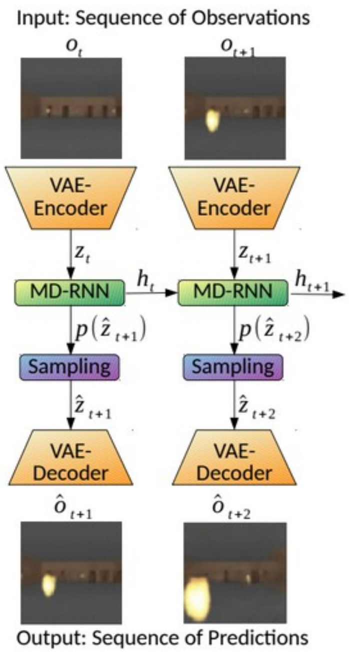
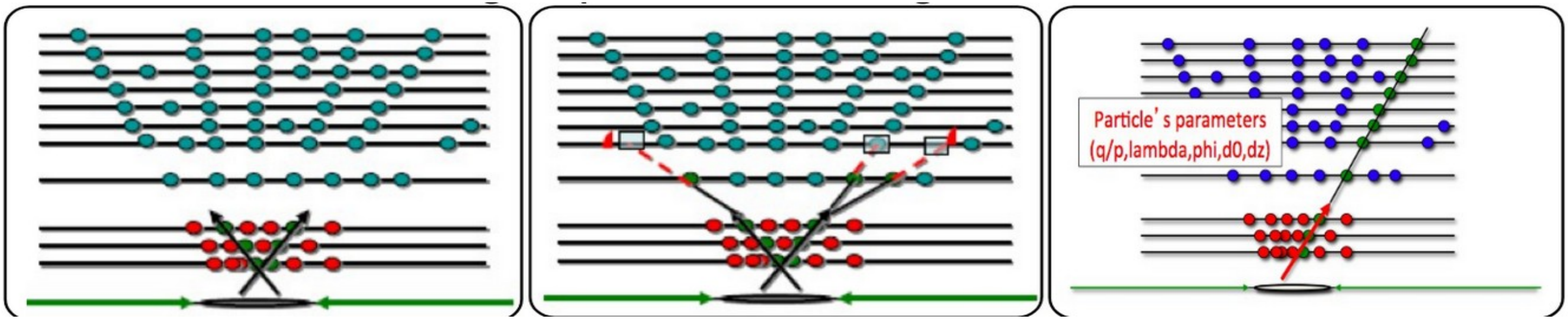


Figure 2. World model predicting future frames by combining a variational autoencoder and an MD-RNN. We follow the architecture suggested in (Ha & Schmidhuber, 2018).

Model gives a sequence of predictions based on previous frames. Gives also errors.

More scientific - Pattern Recognition in High Energy Physics

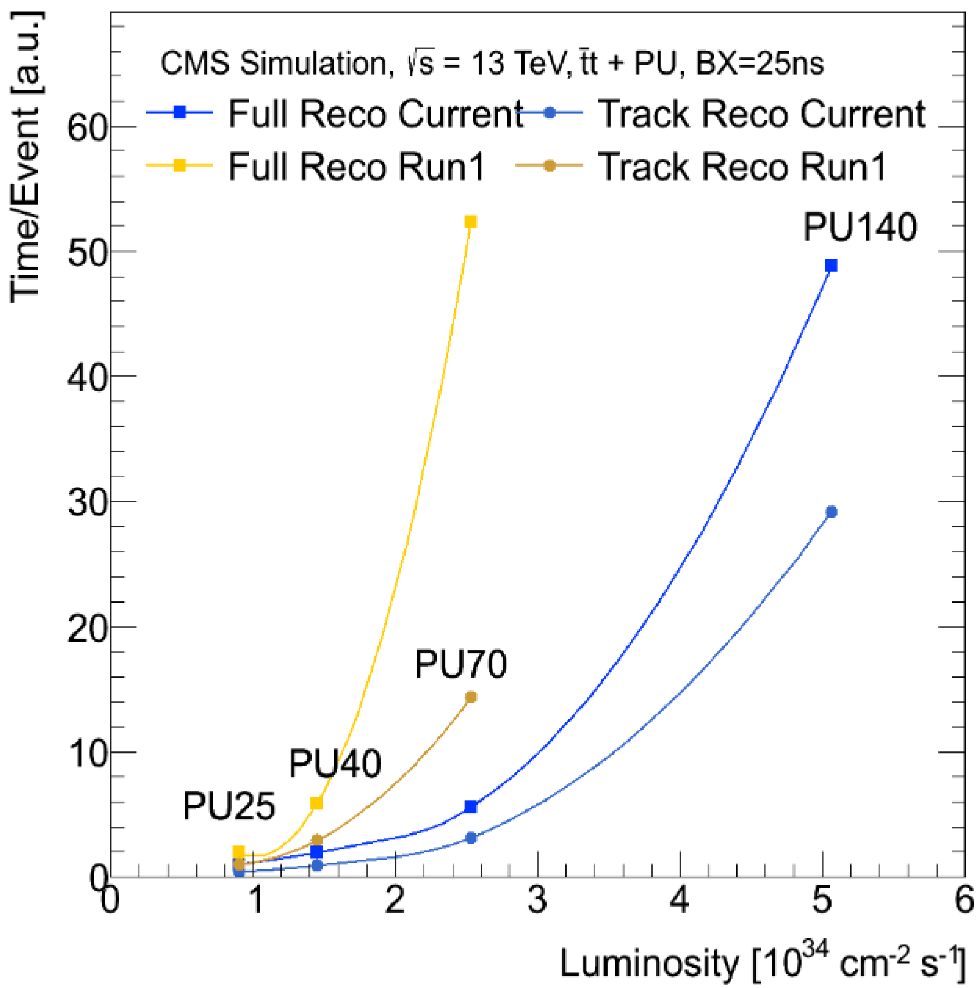


Seeding **Track Building** **Track Fitting**

- **Track seeding** – finding the seeds (initial sets of hits) from which the track starts
- **Track building** ≡ pattern recognition HEP jargon
  - Creating a 2- and 3-dimensional lines and assigning to them all the hits within a certain window
  - Fitted frequently with “robust fit”
- **Track fitting** – final fitting of the track parameters (usually a Kalman filter used for tracking)

Usually this method works fine, is robust and efficient!

So, where is the problem?



CMS experiment simulation
J.-R. Vlimant, Machine Learning for Charged Particle Tracking, MIT, 2018

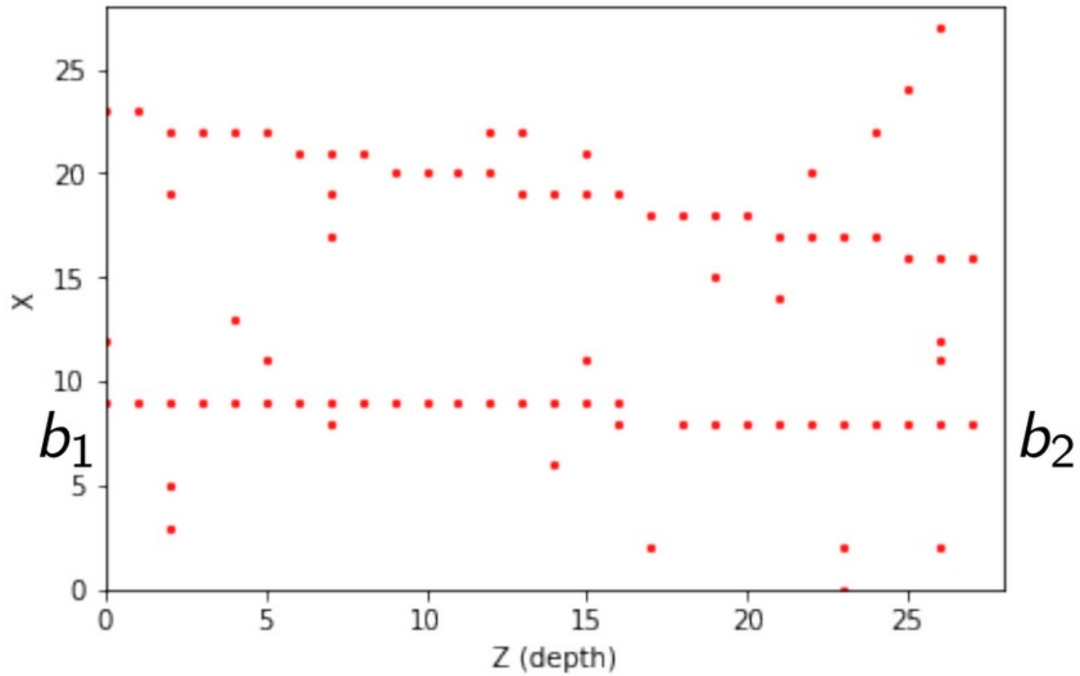
- The time needed to process one event grows quickly (worse than quadratic) with luminosity (number of collisions).
- Huge part of CPU consumption is the track finding.

Deep Neural Network (DNN)?

- Fast, parallel, in principle does pattern recognition “at once”, without looping over hits.
- Also experiments with lower occupancy might profit from DNN’s – higher precision and efficiency.
- There is a HEPTrkx group working on tracking for HEP experiments: *https://heptrkx.github.io/*

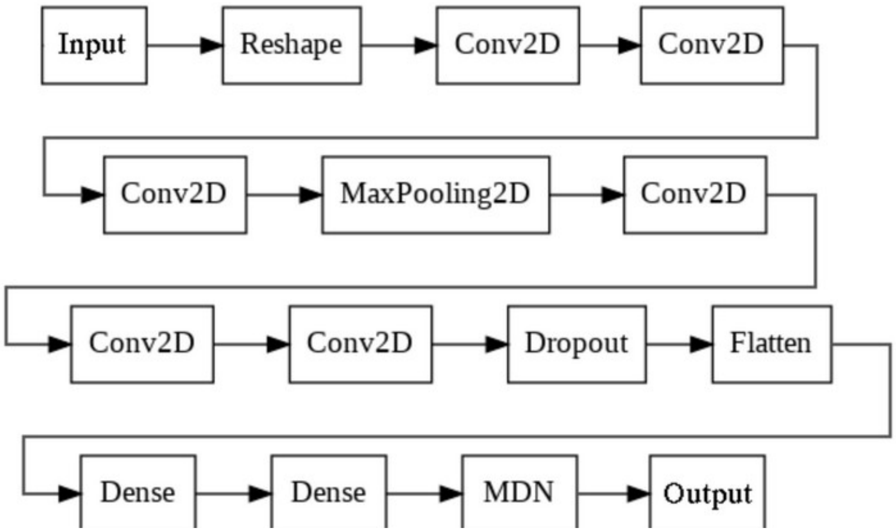
Tracking in 2D toy model

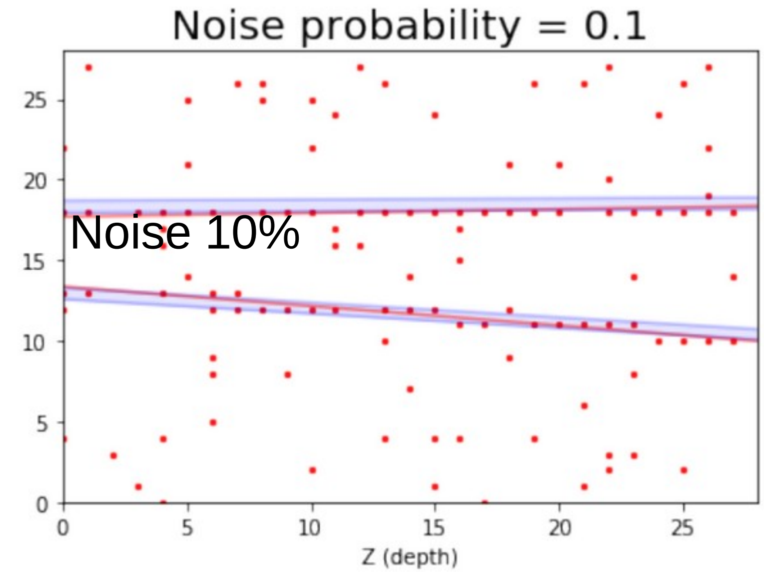
- **CNN returns track parameters (regression)**
- What about many tracks?
  - Solution: add **Mixture Density Network(MDN)** layer to process many tracks.
  - Straight tracks – described by two parameters. Each parameter has associated MDN Gaussian
  - If a number of tracks lower than expected some MDN outputs have very low amplitude.
  - **Important** - we are getting errors of track parameters

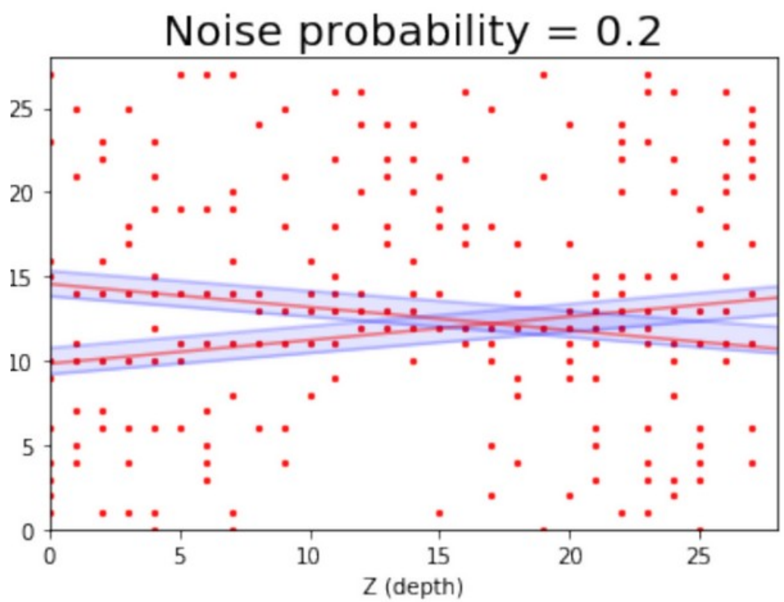


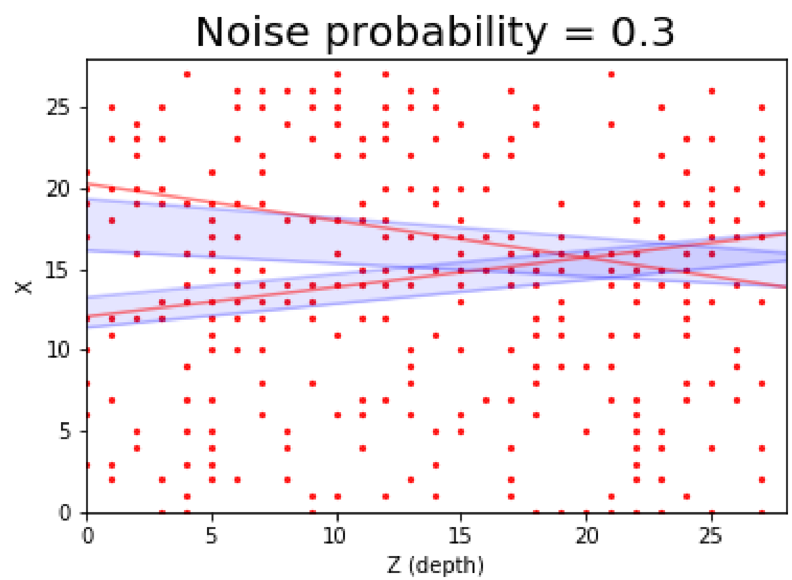
Track described by to interception parameters b_1 and b_2.

Mixed Density Network



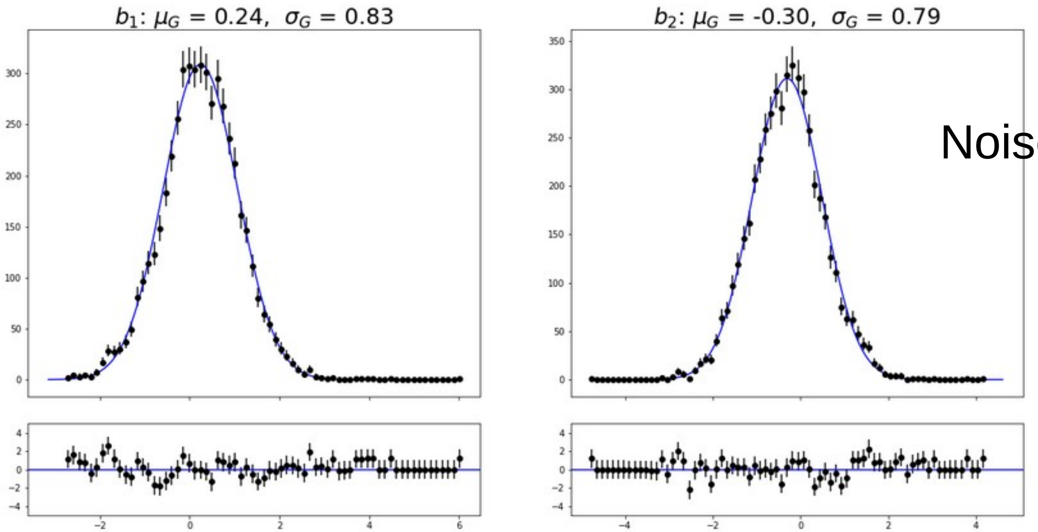




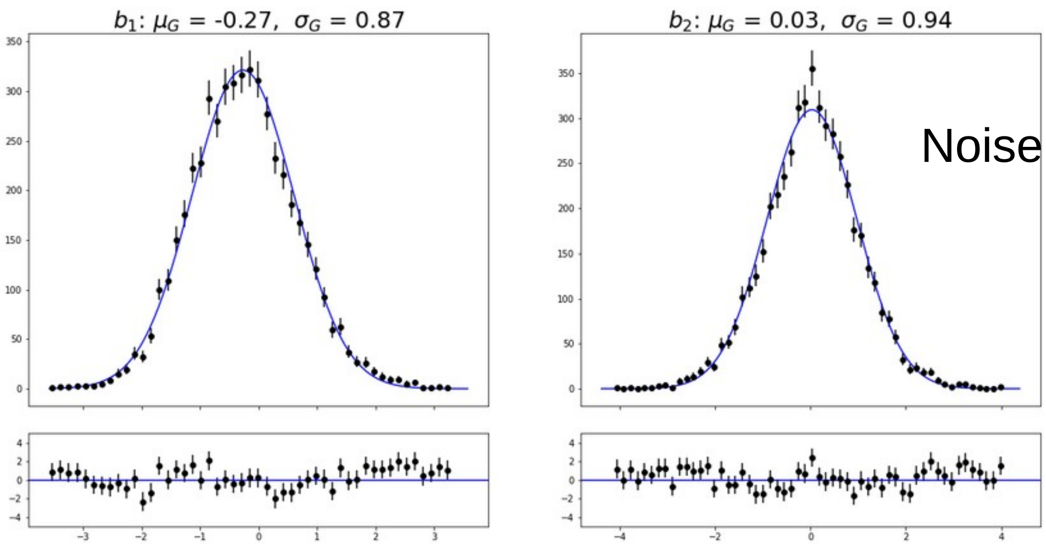


Designed by **Karol Białas** and **Mateusz Słysz** summer students at IFJ:
https://github.com/marcinwolter/MachineLearnin2019/blob/master/dnn_tracking_2D_mdn_multimod.ipynb

MDN Tracking – error estimation



Noise 10%



Noise 30%

Pull plots have a width not far from 1.

If a random variable x is generated repeatedly with a Gaussian than the pull distribution:

$$g = \frac{x - \mu}{\sigma}$$

will be distributed as a standard Gaussian with mean zero and unit width.

Mixed Density Networks

- Quite an old idea
- But a very nice one: we get as an output the probability distribution!

Choi, Sungjoon, et al. "Uncertainty-aware learning from demonstration using mixture density networks with sampling-free variance modeling." 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018.
https://arxiv.org/pdf/1709.02249.pdf

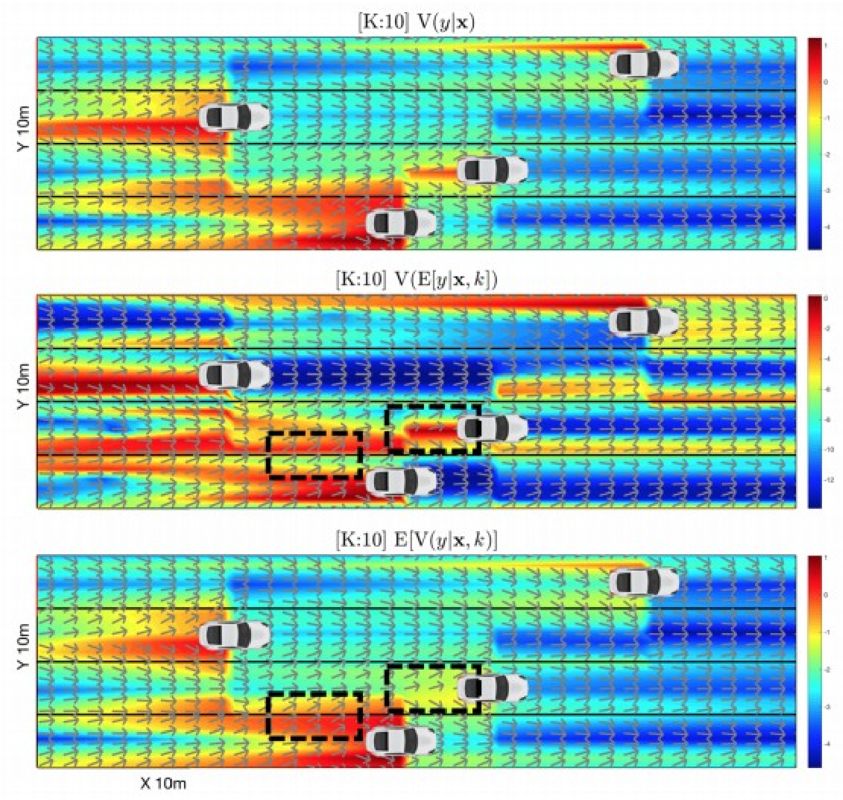
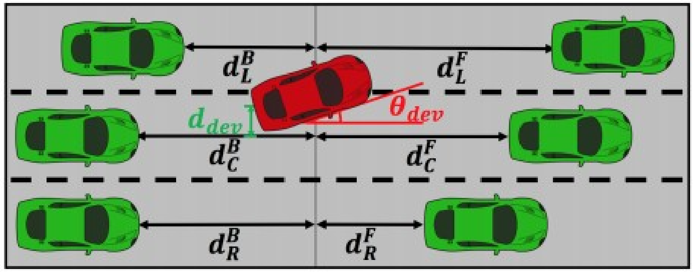


Fig. 5: Different uncertainty measures on tracks.

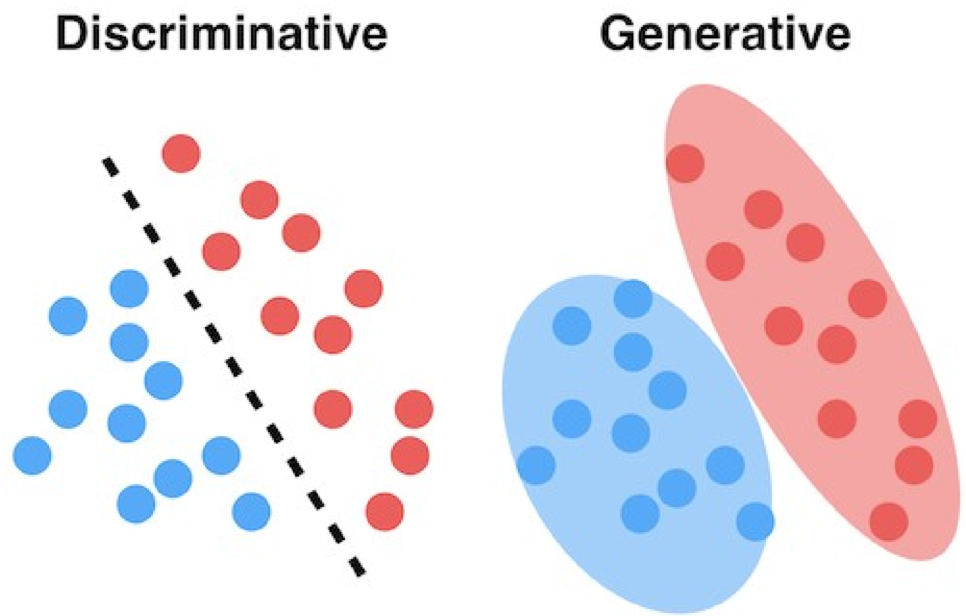


Generative Adversarial Nets (GANs)

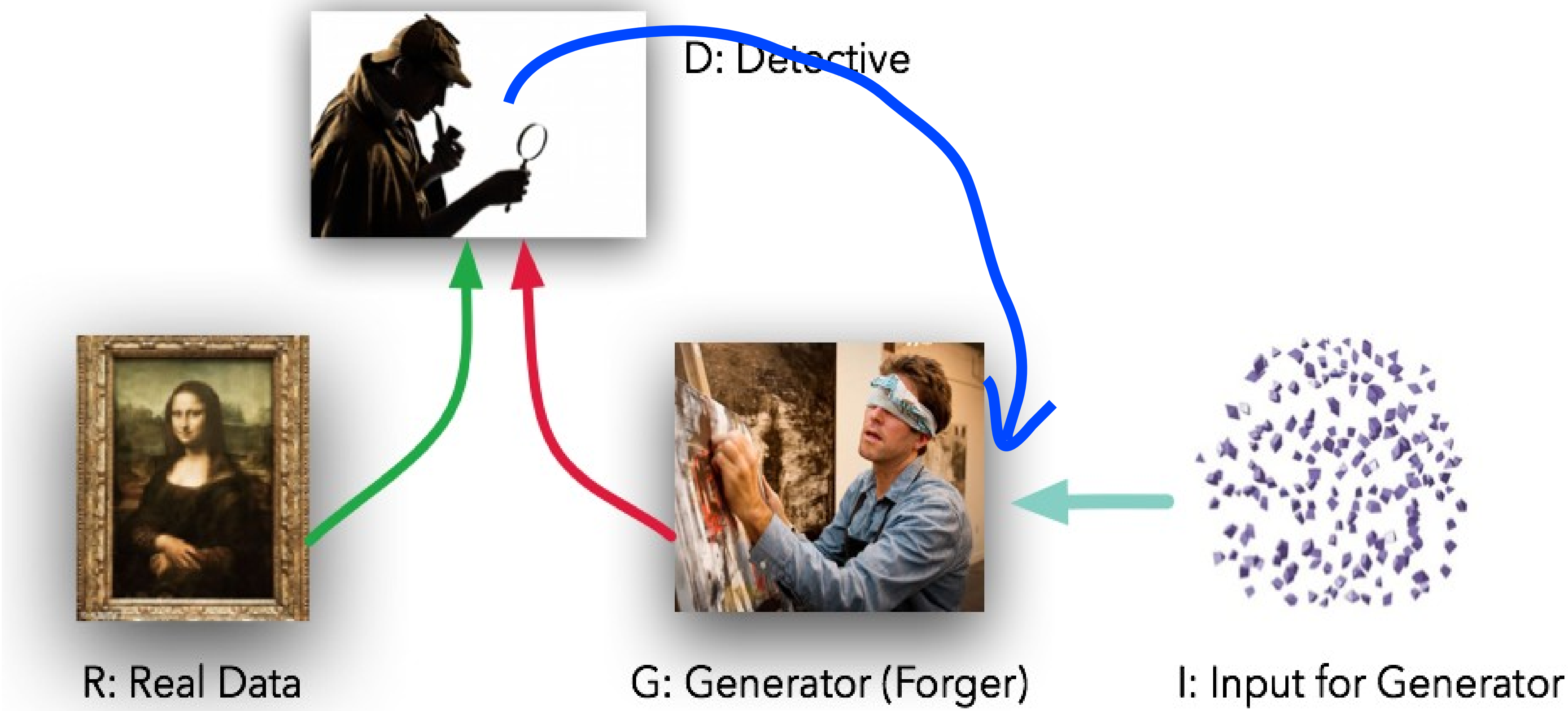
- GANs were introduced Ian Goodfellow and others in 2014 . Yann LeCun called adversarial training “the most interesting idea in the last 10 years in ML.” https://arxiv.org/abs/1406.2661
- GANs’ can learn to mimic any distribution of data. They can be taught to create worlds similar to our own in any domain: images, music, speech, prose. They are robot artists!

How do GANs work?

- **Discriminative algorithms** - classify input data; given the features, they predict a label or category to which that data belongs (*signal* or *background*)
- **Generative algorithms** – do the opposite, assuming the event is *signal*, how likely are these features?
- Another way to distinguish discriminative from generative like this:
  - **Discriminative models** learn the boundary between classes
  - **Generative models** model the distribution of individual classes



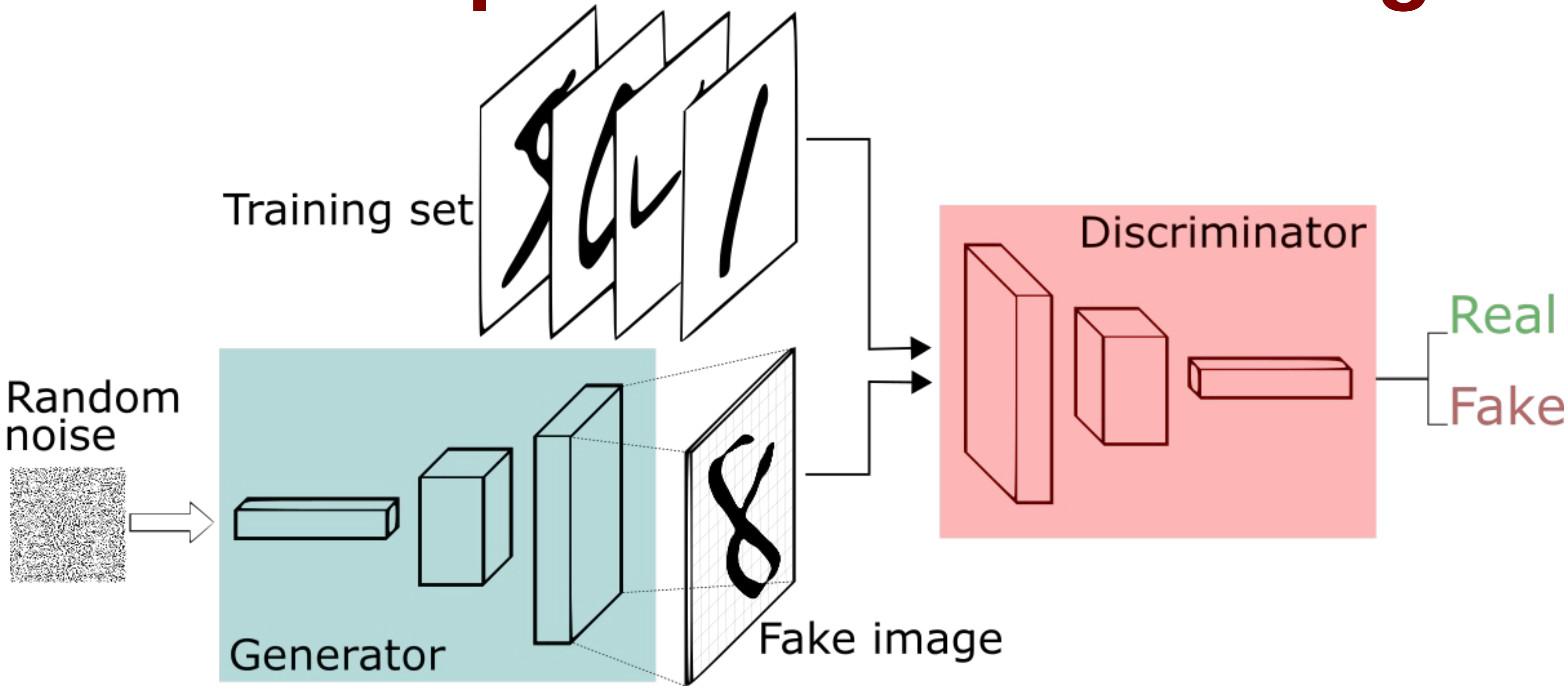
Blind forger and detective



The forger has never seen Mona Lisa, but gets the judgments of detective and tries to fool him (i.e. paint something that looks like Mona Lisa).

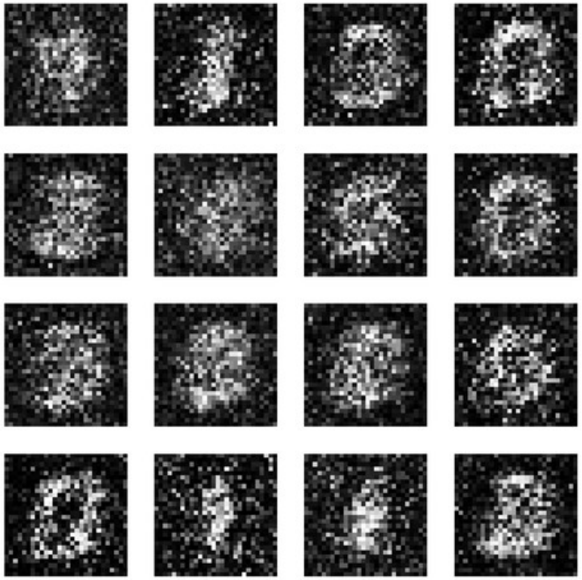
They both (forger and detective) have to train in parallel (important), since if detective is to clever the forger will never paint anything acceptable.

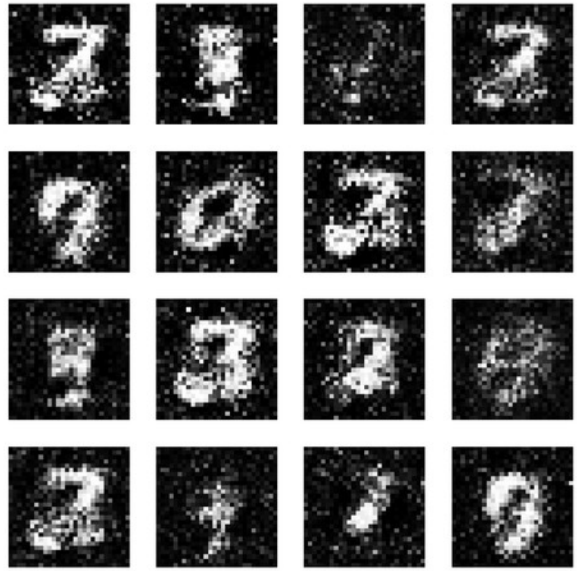
GANs example – hand-written digits



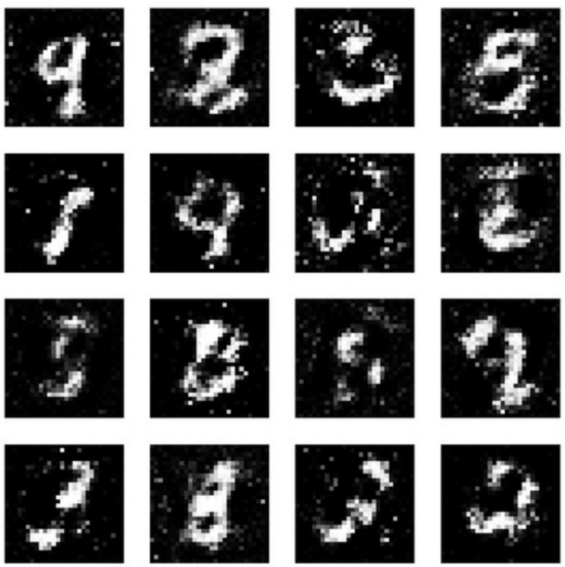
- **Training set** – MNIST: hand-written digits supplied by US post.
- **Discriminator** – convolutional neural network labeling images as real or fake.
- **Generator** - inverse convolutional network (while a standard convolutional classifier takes an image and downsamples it to produce a probability, the generator takes a vector of random noise and upsamples it to an image).

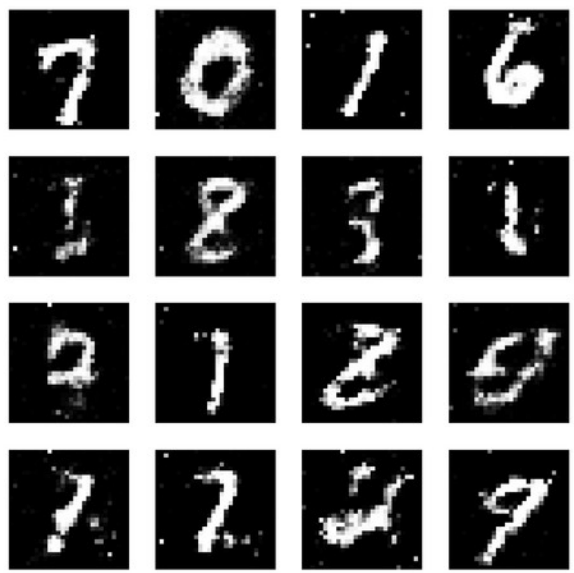
Implementation: Python code using Keras interface and TensorFlow backend.





400 cycles 800 cycles 1200 cycles





2400 cycles 8000 cycles 19900 cycles

Each cycle digits look more and more realistic.

Updated code from the net:

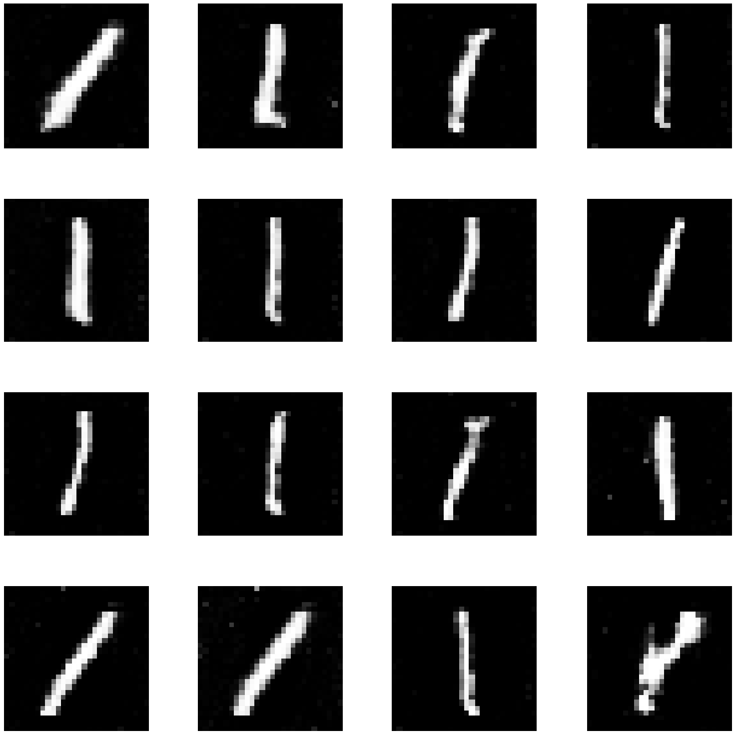
https://github.com/marcinwolter/MachineLearnin2019/blob/master/gan_generate_letters.ipynb

Training a GAN

- It is important to keep balance between generator and discriminator – none of them should be to smart!
- If in the previous example we modify discriminator to make it much better (ConvDNN, many layers) the GAN starts to generate just one digit.

  https://github.com/marcinwolter/MachineLearnin2019/blob/master/gan_generate_letters_failing.ipynb

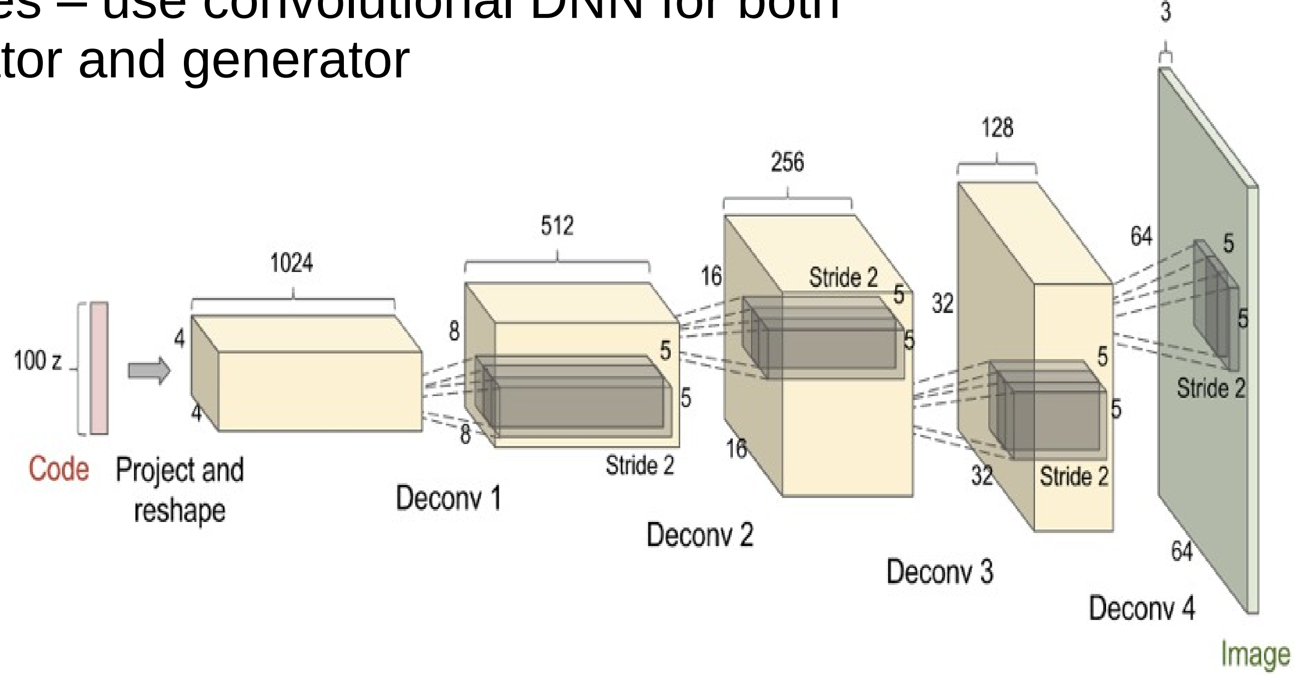
FAILING



Deconvolutional GANs (DCGAN)
(Radford et al., 2015)

Goodfellow (2017)

For pictures – use convolutional DNN for both discriminator and generator



Batch normalization important here, apparently

Deconvolutional GANs (DCGAN)
(Radford et al., 2015)

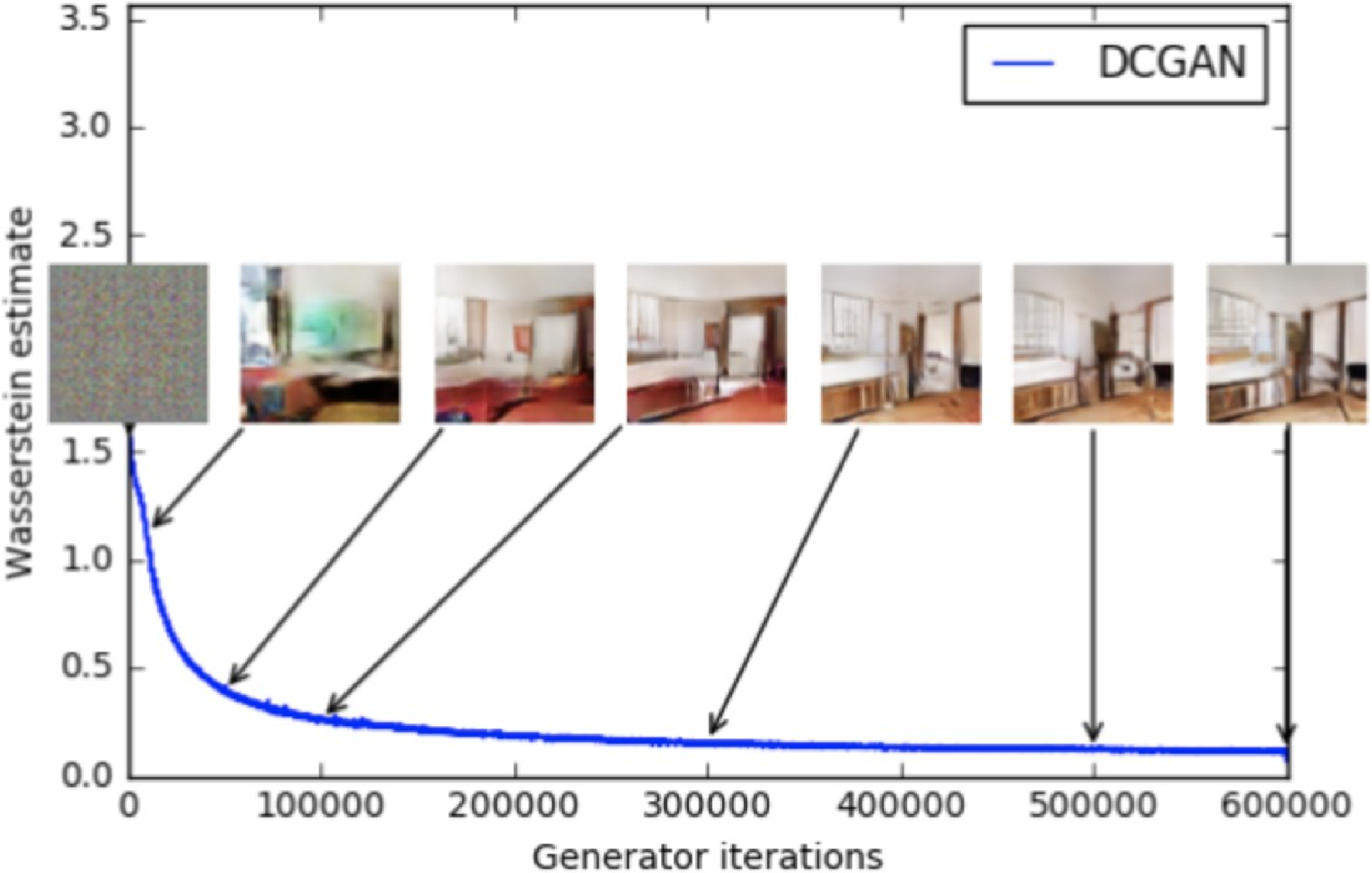
Goodfellow (2017)

Based on LSUN data set

- 10 scene categories
- 20 object categories

[ArXiv 1506.03365]()

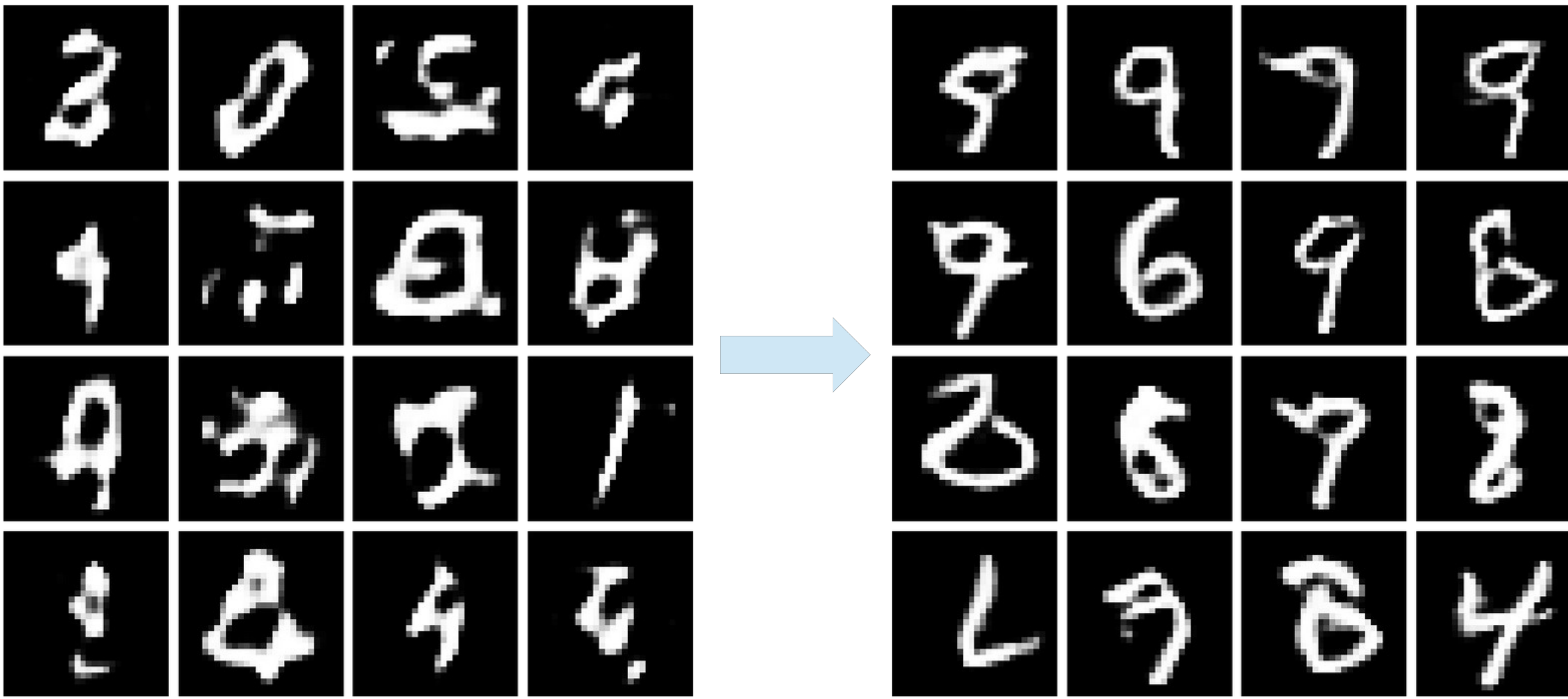
DCGAN training



An example of a DC-GAN training and improving its quality "Improving the GAN training process"

DCGAN example

Generating hand-written letters again:
https://github.com/marcinwolter/MachineLearnin2019/blob/master/dcgan_cnn_mnist.ipynb



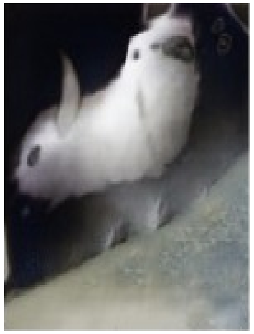
Cherry Picked Results

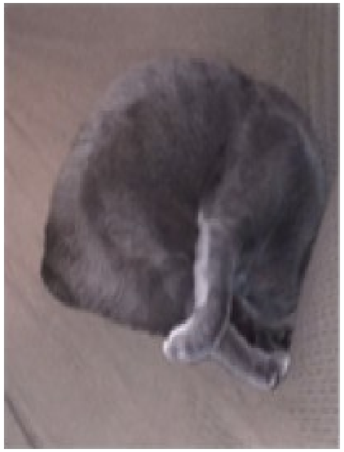
Goodfellow (2017)

Problems With Counting

Goodfellow (2017)

Problems With Perspective

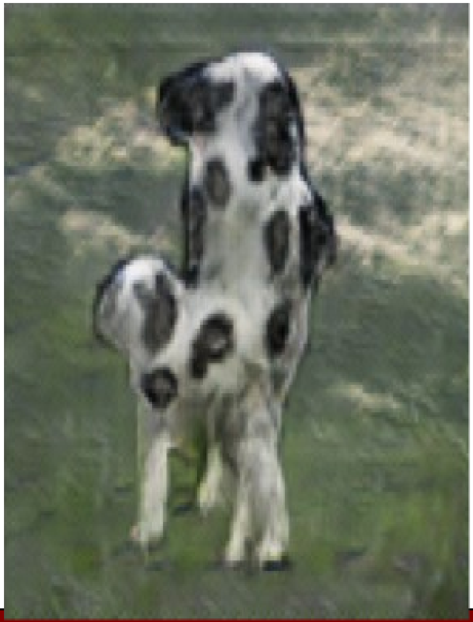



Goodfellow (2017)

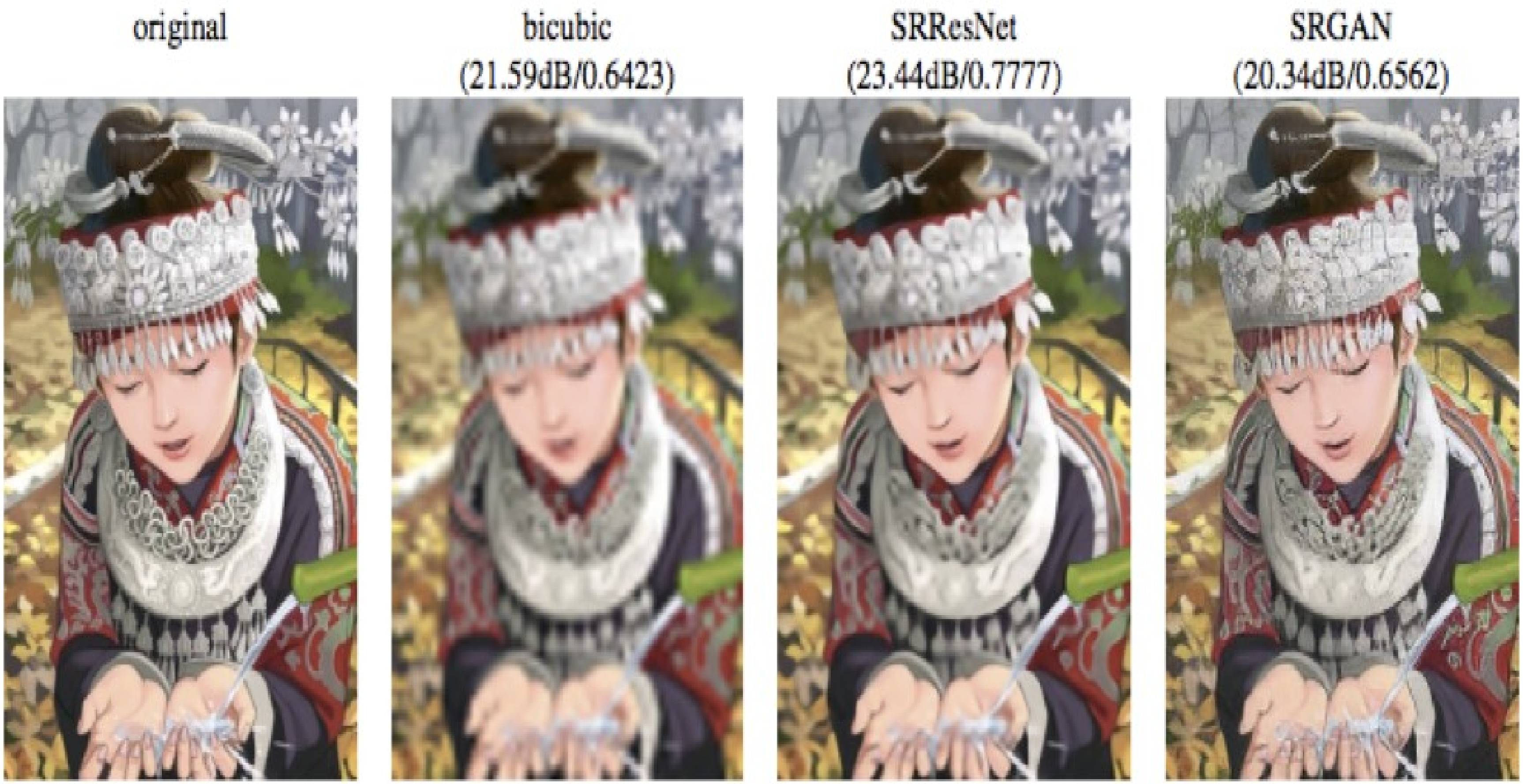
Problems With Global Structure



Goodfellow (2017)

Beyond Labels: Providing Images as Input to Generator: Image Super-Resolution (Ledig et al., 2016)

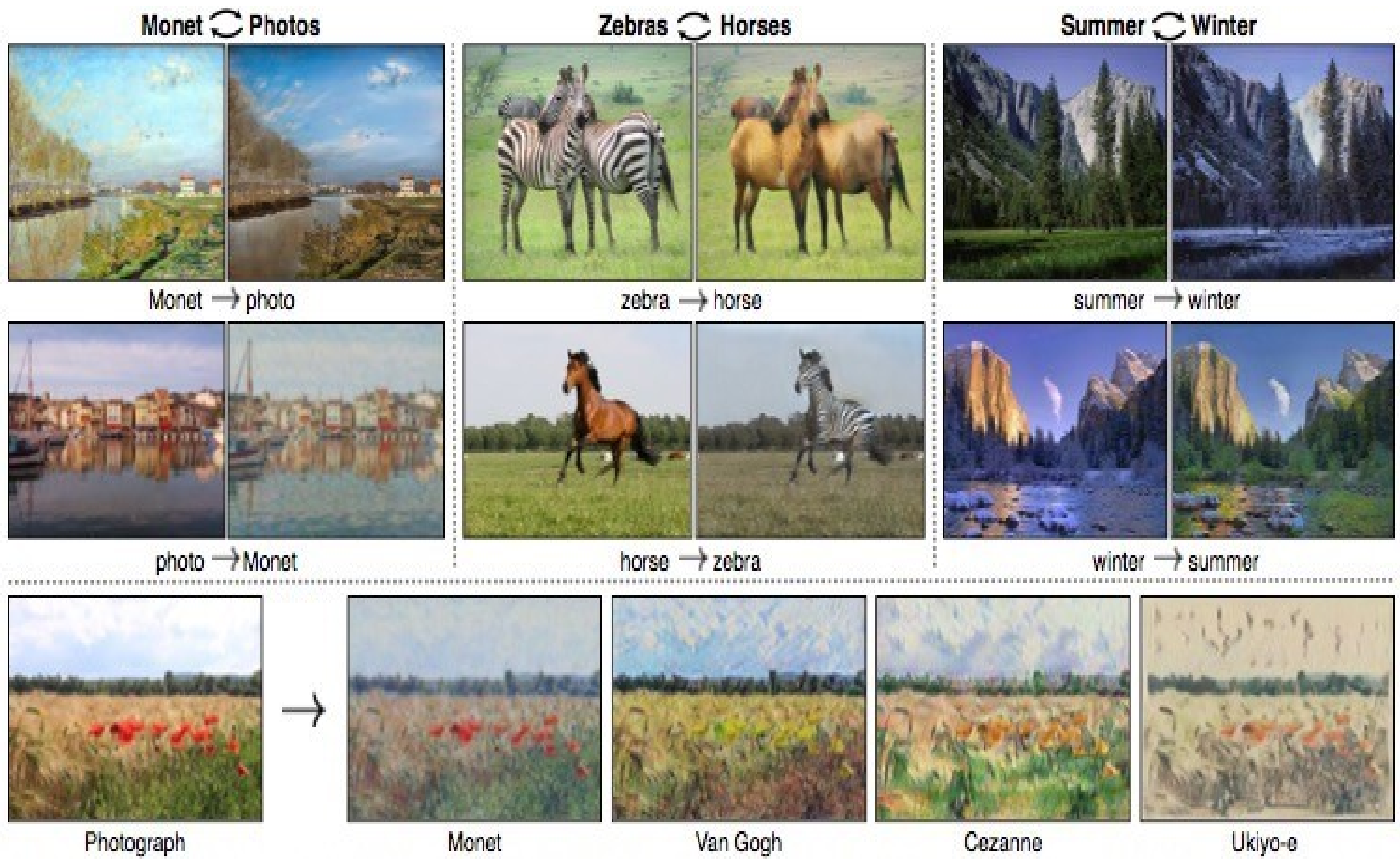


Cycle GANs

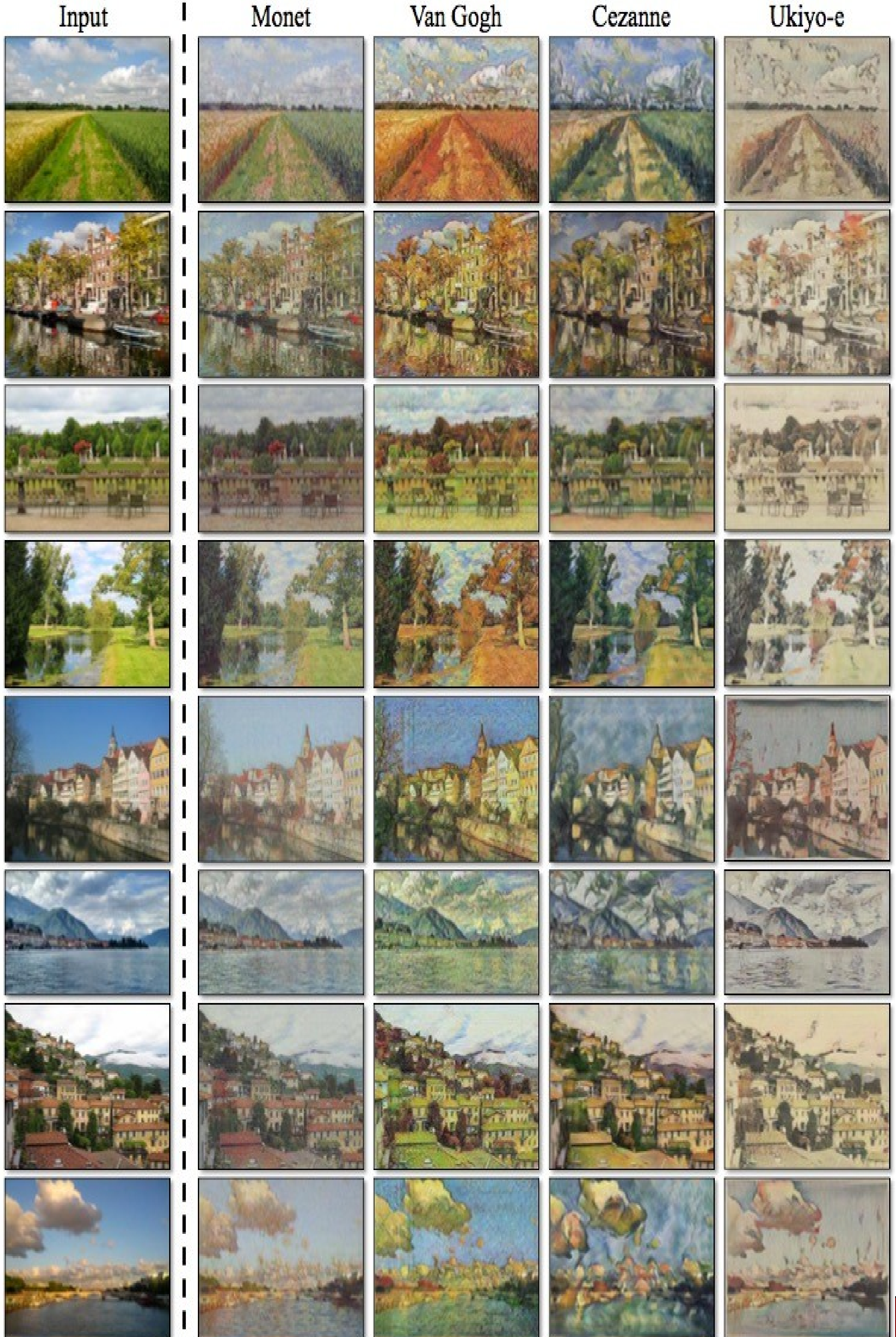
(Zhu et al., 2017; arXiv:1703:10593v2 [cs.CV])

Given two image collections

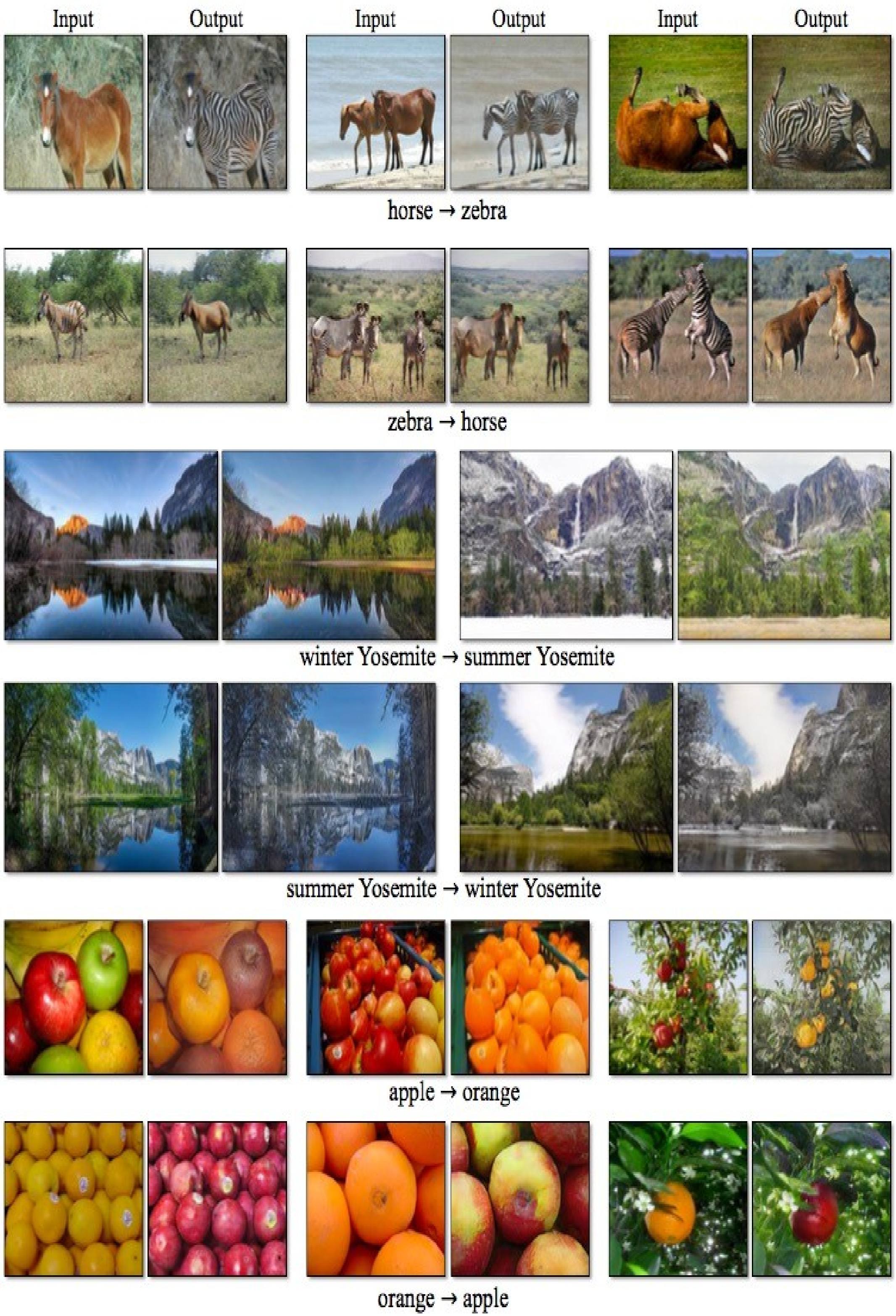
- **algorithm learns to translate an image from one collection to the other**
- **does not require corres-pondence between images**



Photos to paintings



Paintings to photos


Input
Output
Input
Output
Input
Output
horse → zebra
zebra → horse
winter Yosemite → summer Yosemite
summer Yosemite → winter Yosemite
apple → orange
orange → apple

Applications of GAN

- In science - peed up generation of simulated data – new simulated look similar to the once already simulated / collected data.
- Overview of GAN applications:

  https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/

Example of GAN Reconstructed Photographs of FacesTaken from Generative Face Completion, 2017.

Make money with GAN!!!

"Edmond de Belamy" is part of a fictitious family created by a "generative adversarial network," of which there's ten other paintings. "Edmond" is one of the most striking of the paintings, and will likely become an important part of art history going forward thanks to its huge selling price. The generator behind the painting created new portraits based on 15,000 from the last 600 years, taking existing art and crafting something wholly original and quite alien.

Conclusions

- Many new methods were developed recently.
- Machine Learning approach becomes to be used not only for classification, but also for other tasks.
- Each month new application appear!

Artificial intelligence is no match for natural stupidity.

- The development is driven by AI applications (image recognition, autonomous cars etc). But the physics community can profit!
- More and more advanced ML techniques have application in HEP. Try to find a new one!