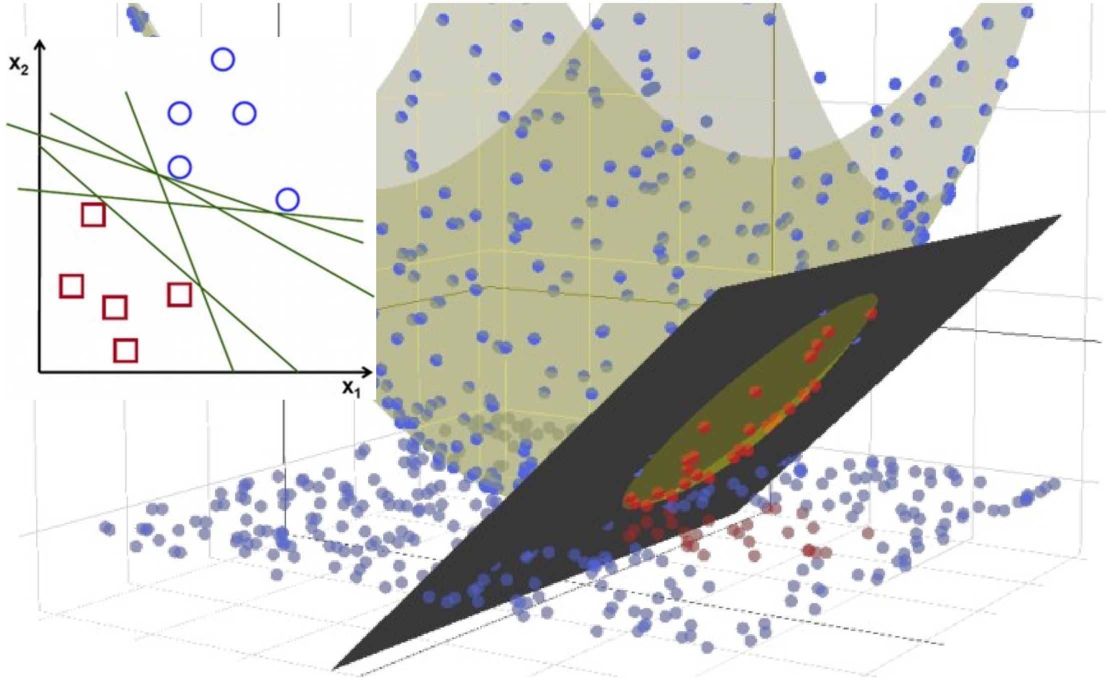
Machine learning
Lecture 6



Marcin Wolter

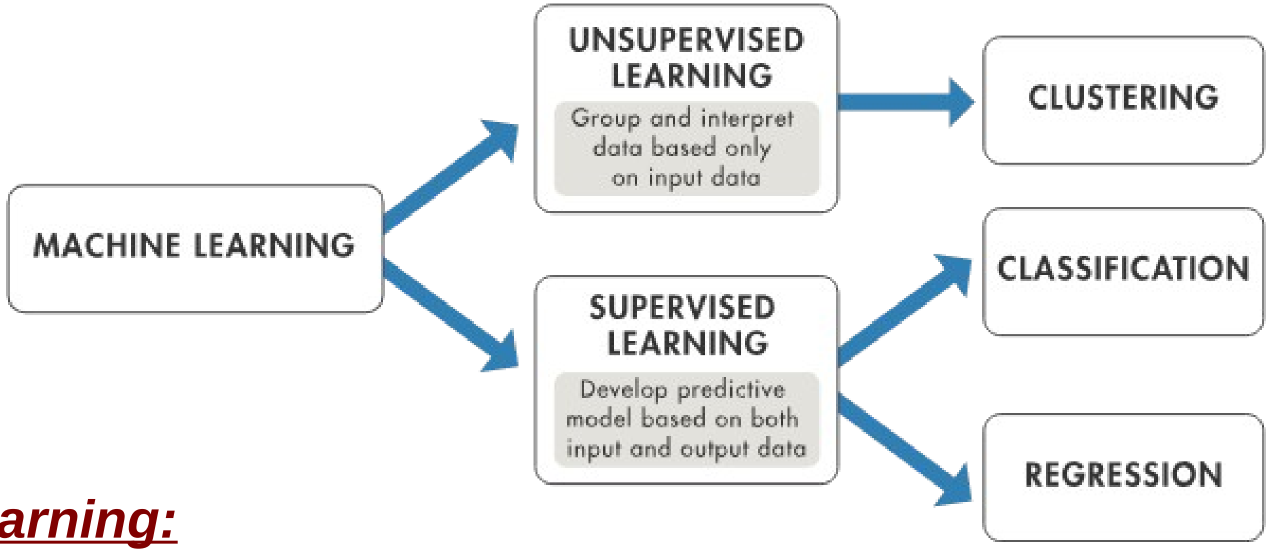
IFJ PAN

12 March 2019

- Practical exercise – pattern recognition using KERAS
- Generative networks

How could we get rid of Monte Carlo?

- The Monte Carlo simulation used for training Machine Learning methods always differs to some extend from data.
- So, the best would be to train algorithms on data…
- … => towards **unsupervised learning**?
- … or maybe only **weakly supervised**?

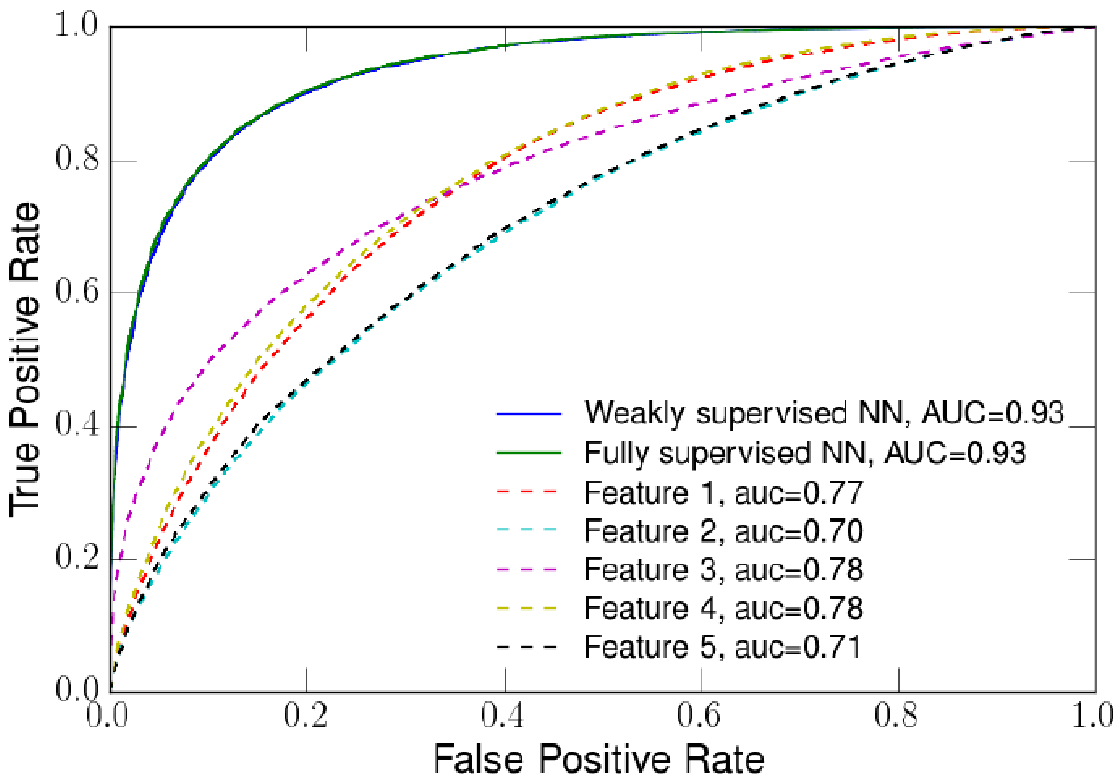


Unsupervised learning:

No training datasets are provided, the data is clustered into different classes based on similarity.

Weakly Supervised Classification

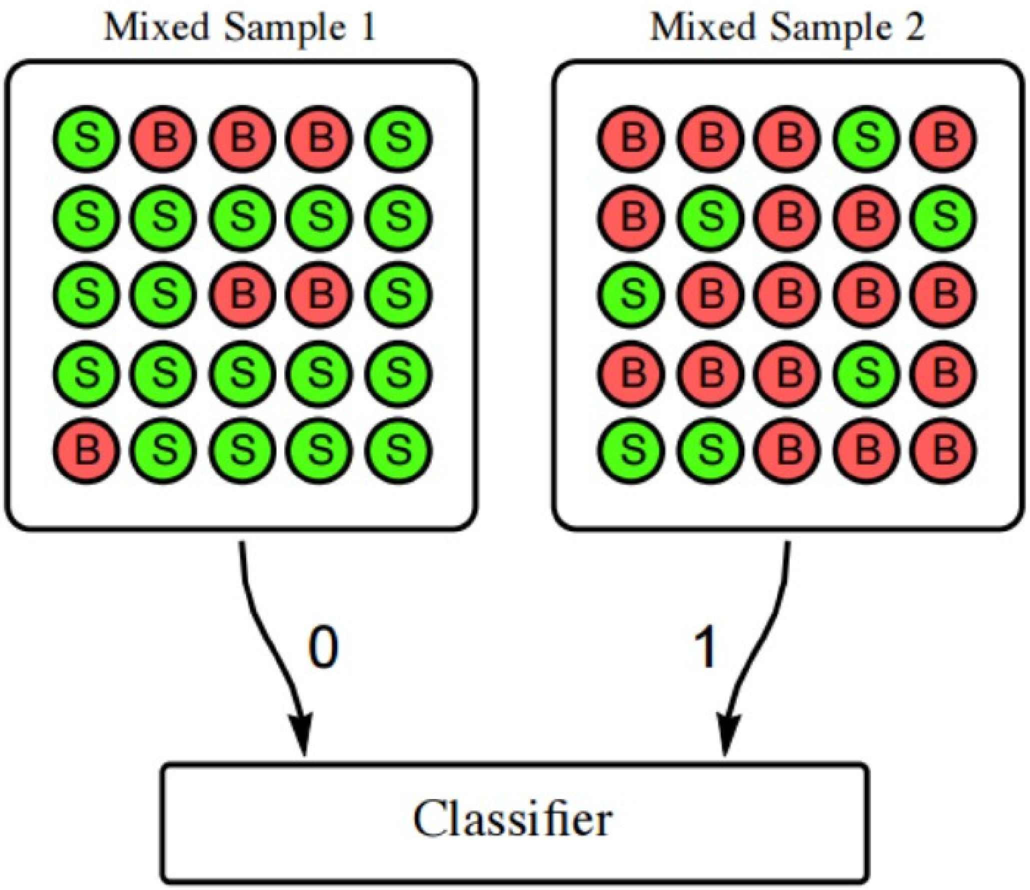
- A new approach in Machine Learning: **weakly supervised** classification in which **class proportions are the only input** into the machine learning algorithm.
- **Quark versus gluon tagging** - weakly supervised classification can match the performance of fully supervised algorithms.
- By design, the new algorithm is insensitive of MC mis-modelling – trained on data.
- **Problem:** we have to find in data what is the proportion of gluon and quark jets.
- Maybe template fits in one/some variable/s using again MC?



arXiv:1702.00414

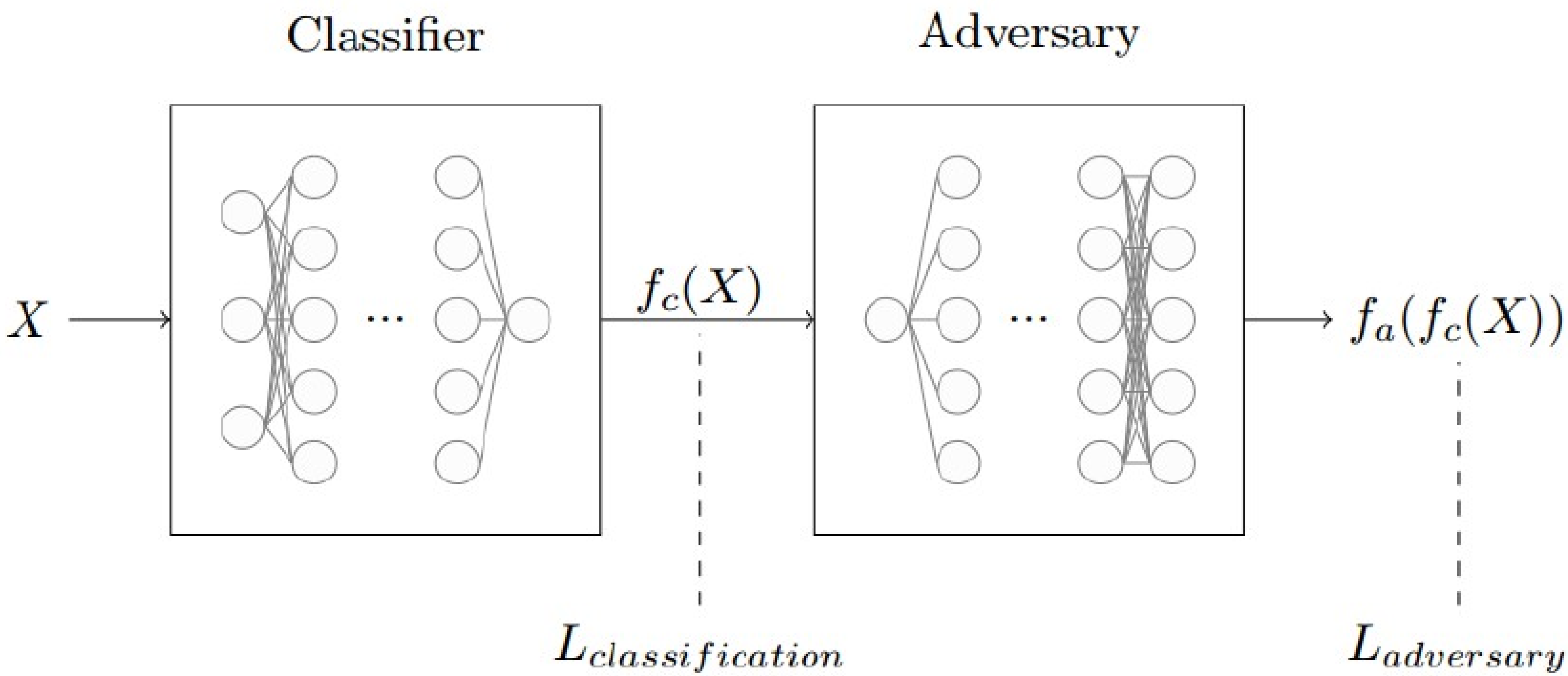
Learning from Data
Classification w/o Labeling

- A step even further is classification w/o labeling (CWoLa) https://arxiv.org/abs/1708.02949
- A classifier is trained to distinguish sample 1 from sample 2, which are mixtures of signal and background with different (and unknown) fractions.
- Such a classifier is optimal for distinguishing signal from background



Adversarial Neural Network

- **ANN** is combination of two regular NNs
- L = L(classifier)-λL(adversary), where λ is a hyper-parameter
- **The adversary function explicitly penalizes the classifier for using information from certain (poorly modeled) variables.**

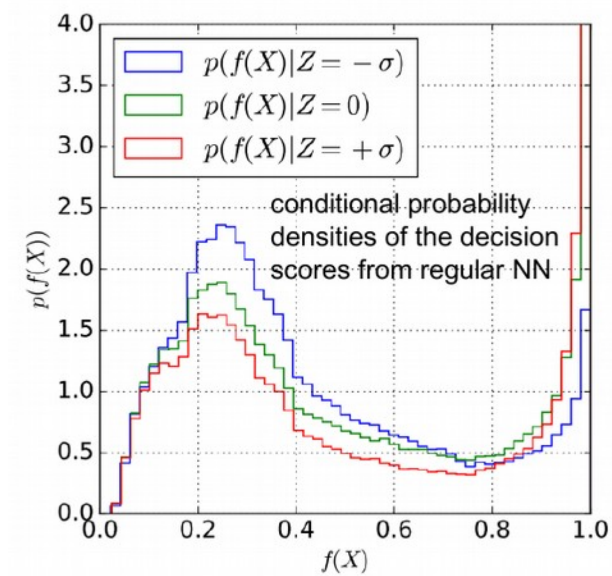


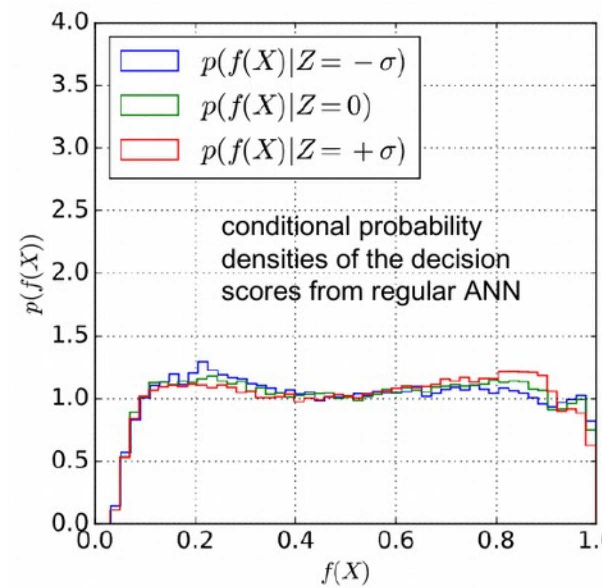
arXiv:1611.01046

Toy example with Adversarial NN

- Most classifiers we use are trained with nominal values of systematics nuisance parameters
- Some might be poorly modeled in MC.
- These classifiers have dependence on these nuisance parameters and are sub-optimal for real data (top plot – dependence of classifier on parameter Z).
- ANN can mitigate the classifier dependence on the nuisance parameter (bottom plot).

Toy example: distinguish two 2D Gaussians, the parameter Z is a shift between their centers.





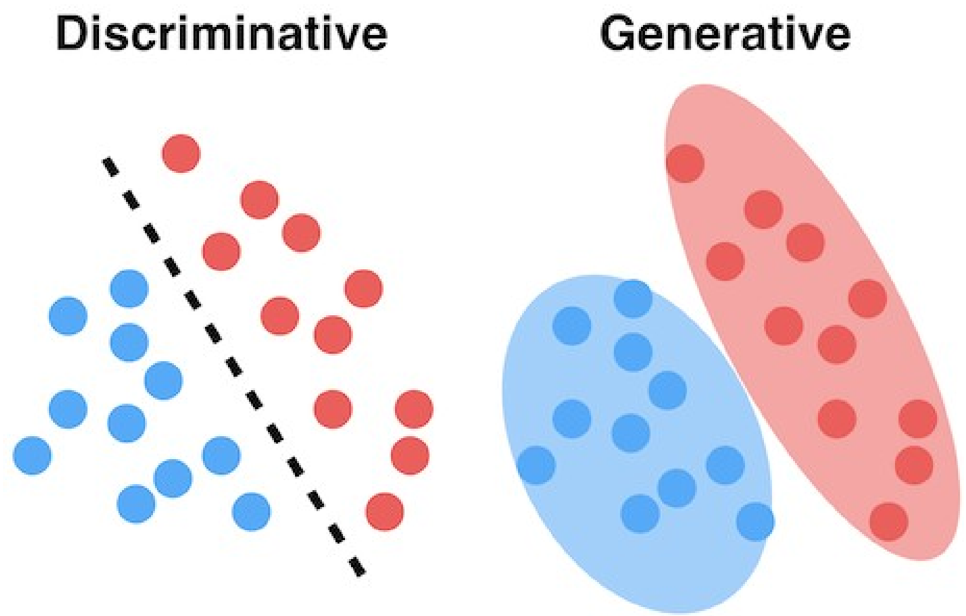
arXiv:1611.01046

Generative Adversarial Nets (GANs)

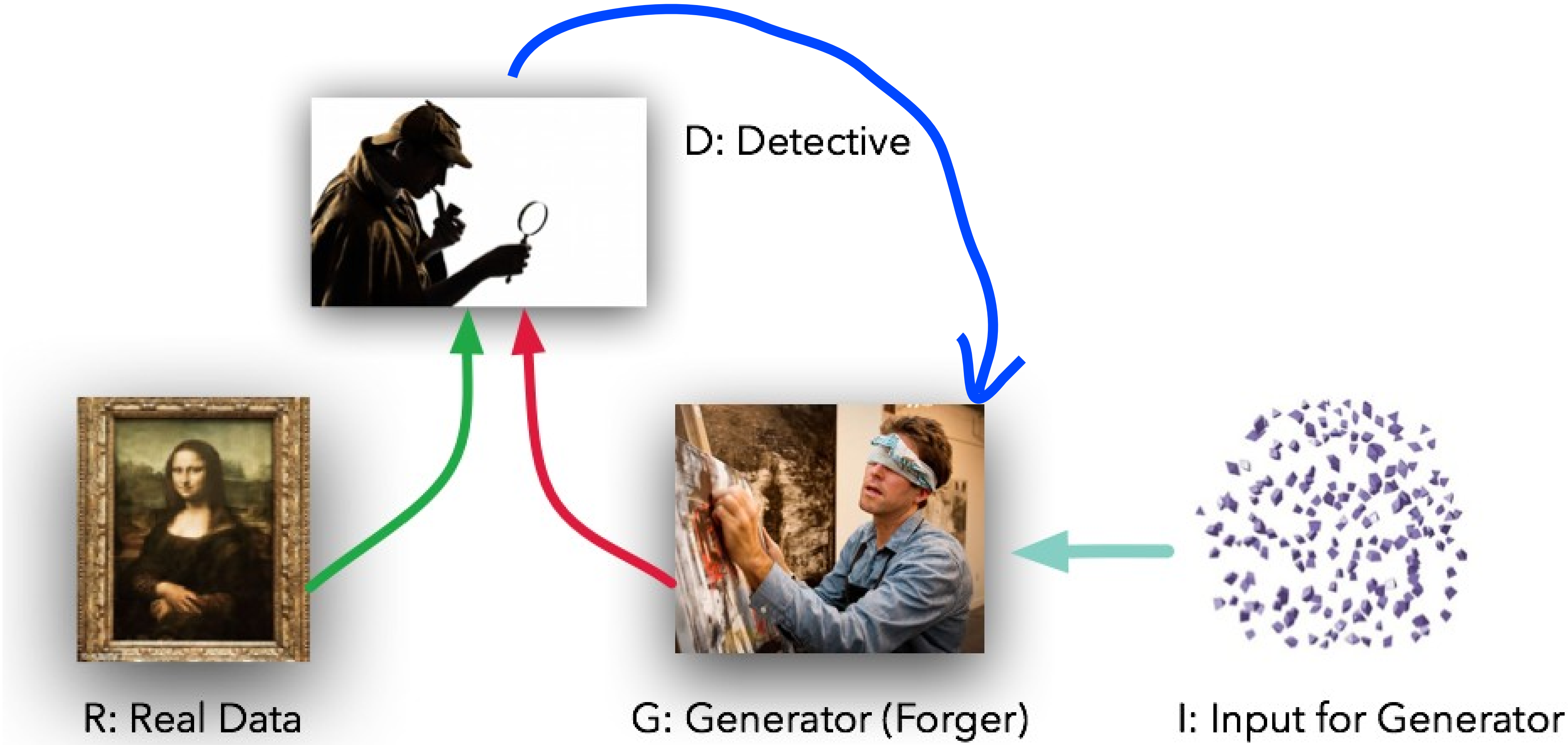
- GANs were introduced Ian Goodfellow and others in 2014 . Yann LeCun called adversarial training “the most interesting idea in the last 10 years in ML.” https://arxiv.org/abs/1406.2661
- GANs’ can learn to mimic any distribution of data. They can be taught to create worlds similar to our own in any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



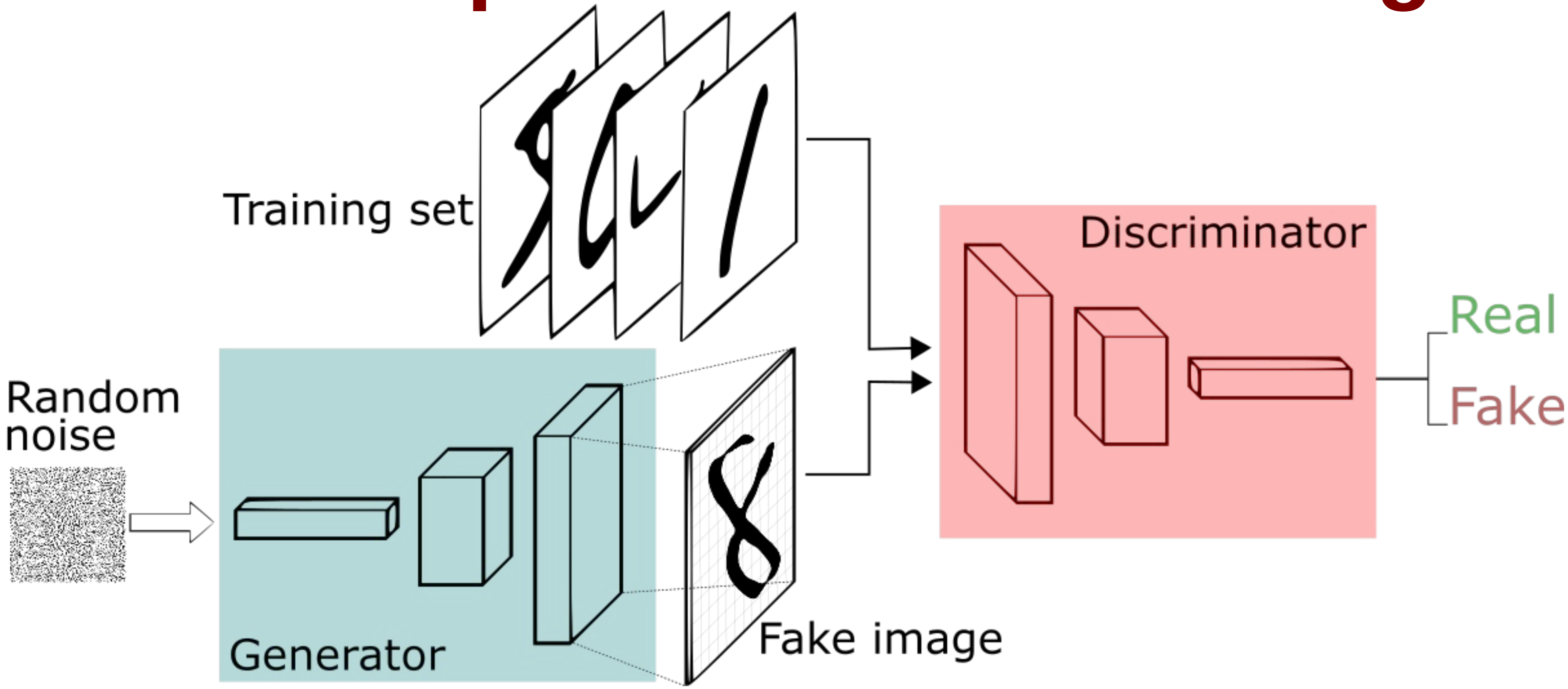
Blind forger and detective



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).

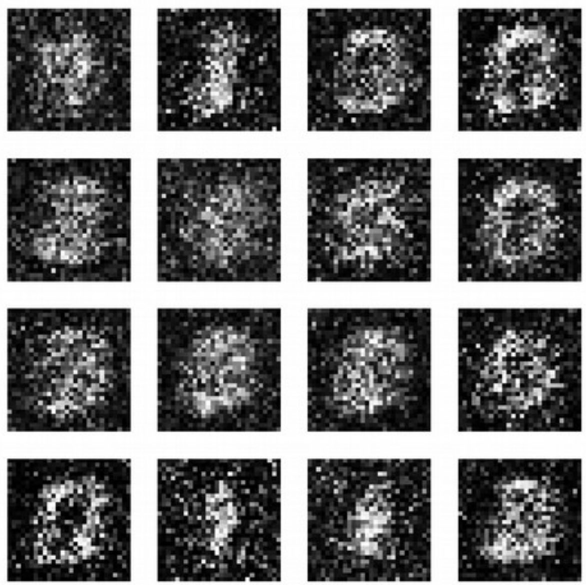
They both (forger and detective) have to train in parallel (important), since if detective is to clever the forger will never paint anything acceptable.

GANs example – hand-written digits

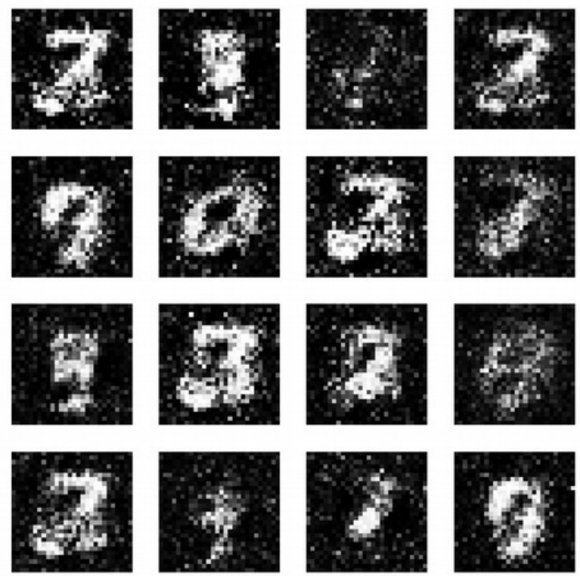


- **Training set** – MNIST: hand-written digits supplied by US post.
- **Discriminator** – convolutional neural network labeling images as real or fake.
- **Generator** - inverse convolutional network (while a standard convolutional classifier takes an image and downsamples it to produce a probability,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

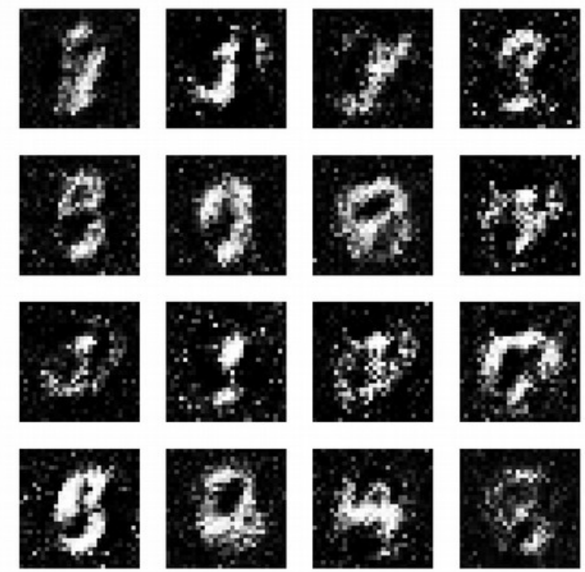
Implementation: Python code using Keras interface and TensorFlow backend.



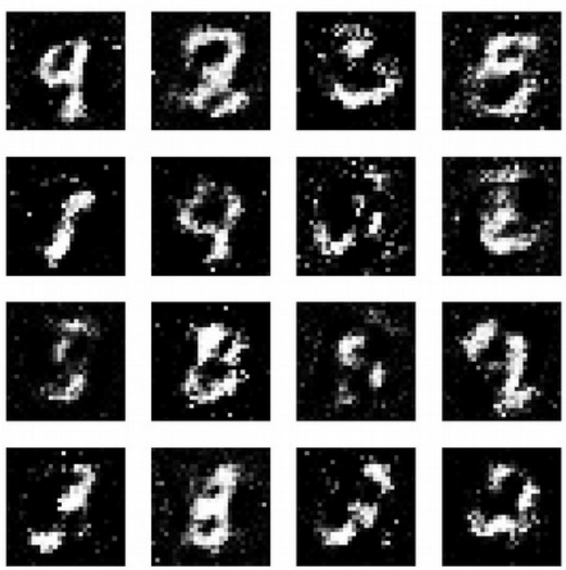
400 cycles



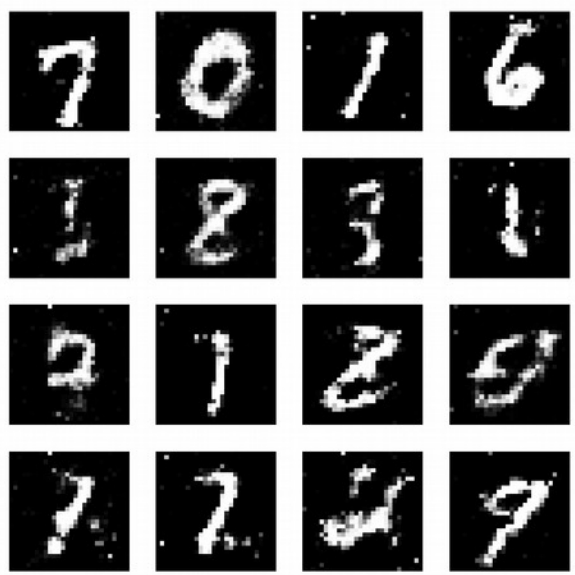
800 cycles



1200 cycles



2400 cycles



8000 cycles

19900 cycles

Each cycle digits look more and more realistic.

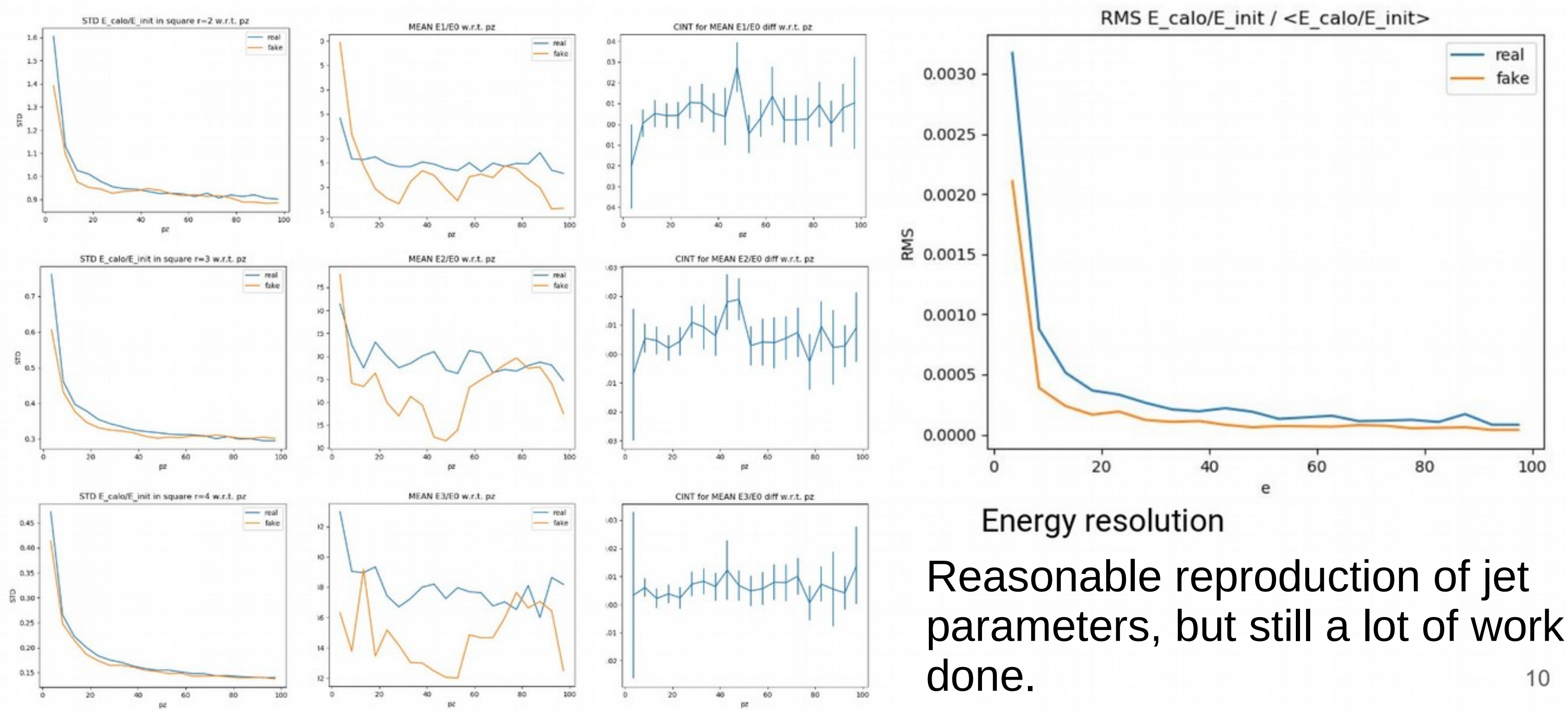
Example code: gan_mw.py on *https://indico.ifj.edu.pl/event/232/*

Could GANs be useful in physics?

Example - GANs can be used to speed up the **Monte Carlo** simulation

- **LHCb project** – speed up calorimeter simulation

Distributions inside calorimeter regions (bins represent different energy levels)



Energy resolution

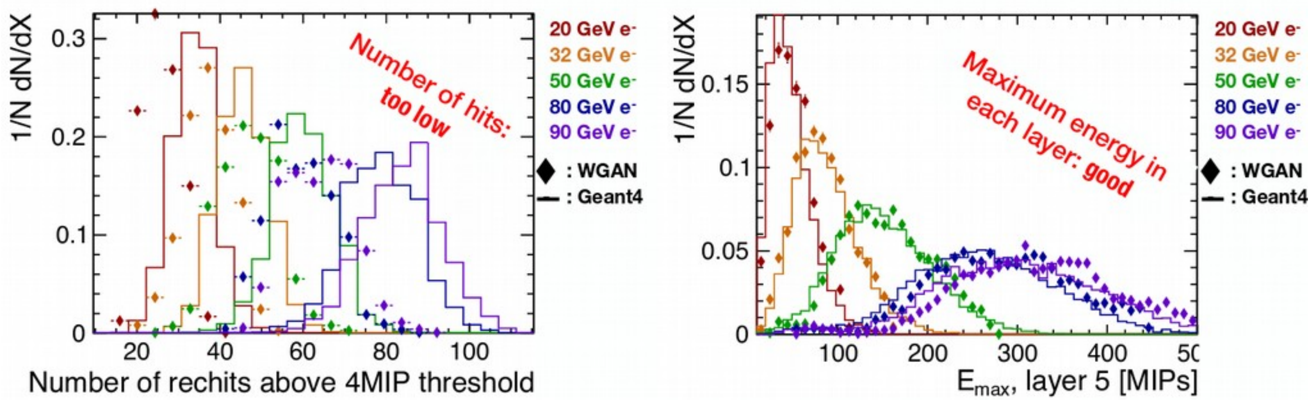
Reasonable reproduction of jet parameters, but still a lot of work to be done.

10

https://indico.cern.ch/event/668017/contributions/2947021/attachments/1629774/2597329/IML_18_Zakharov.pdf

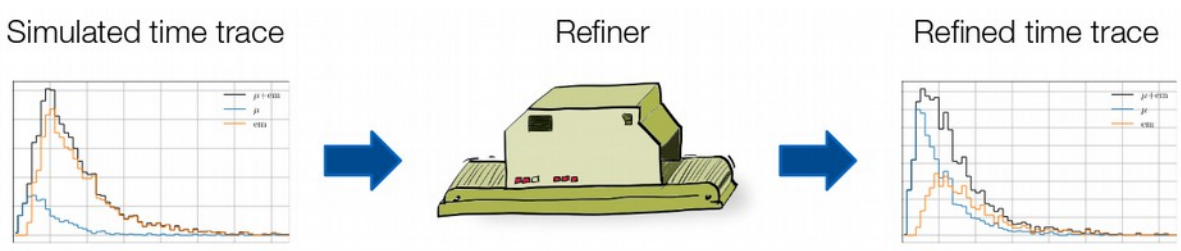
Wasserstein GANs in HEP
(modification of GANs)

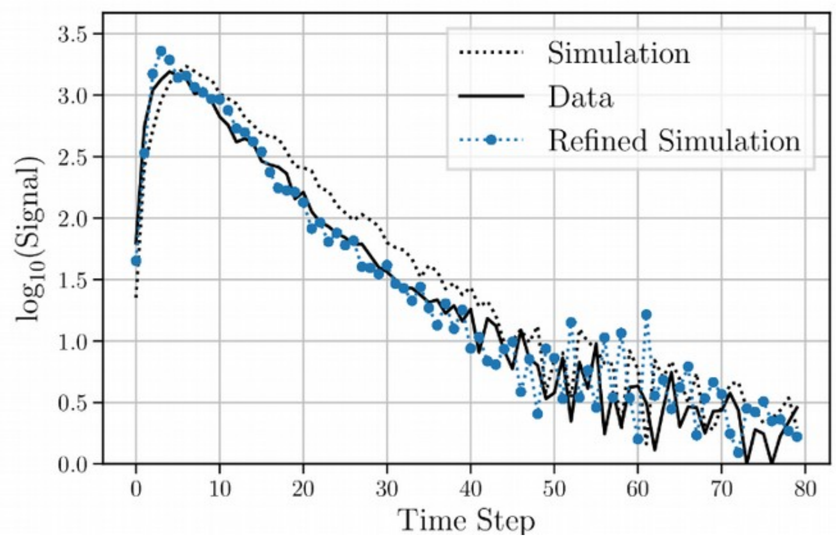
- Conditional Wasserstein GANs for fast simulation of electromagnetic showers in a CMS HGCAL prototype.
  - O(1000) faster simulation! Still work to be done…



- **Pierre Auger** experiment – refinement

  of the simulation to fit better to the data





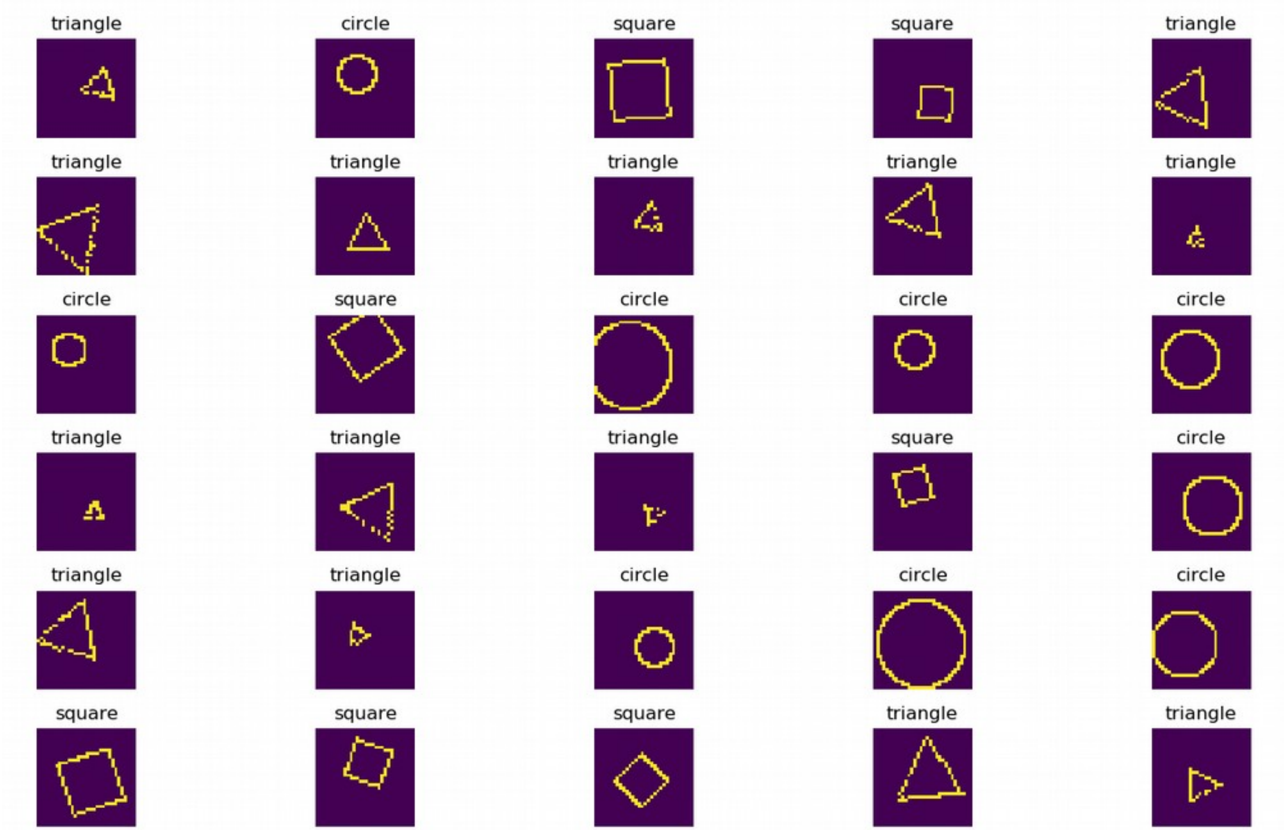
https://indico.cern.ch/event/668017/contributions/2947033/attachments/1629594/2597027/WGAN_JointTalk.pdf

Conclusions

- Many new methods were developed recently.
- Machine Learning approach becomes to be used not only for classification, but also for other tasks (Monte Carlo simulation, tracking etc).
- Advanced ML techniques have wider applications within HEP community.
- New approach to training – data driven training?

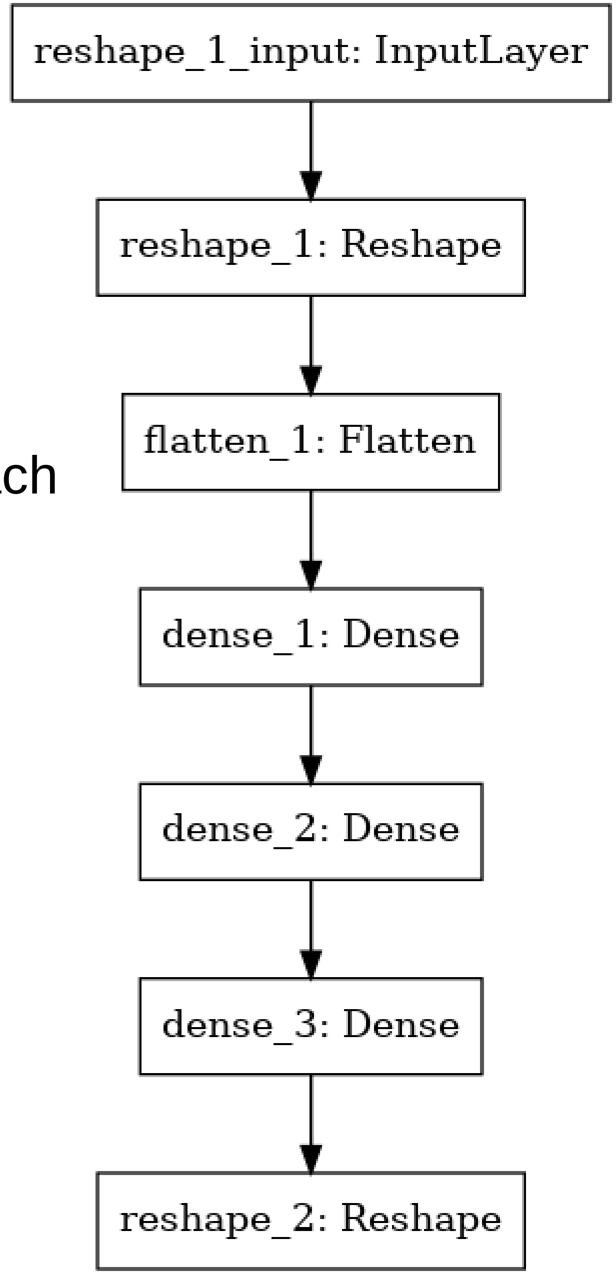
Qualification task

- **Task – recognize geometrical figures using Deep NN**
- Generation of images:
  - 10 000 images
  - Generate 32x32 pixel images of: circle, square or triangle
  - Different sizes and positions (including rotation)



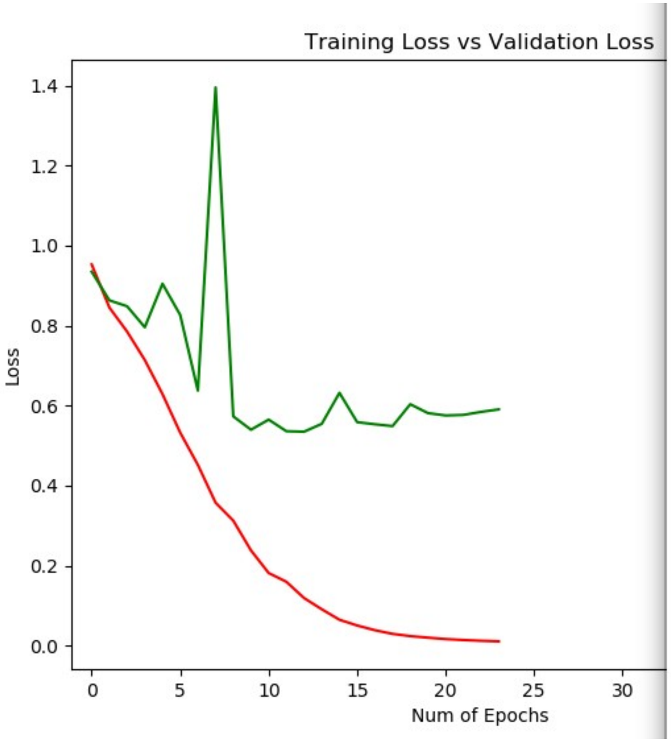
Simple fully connected network

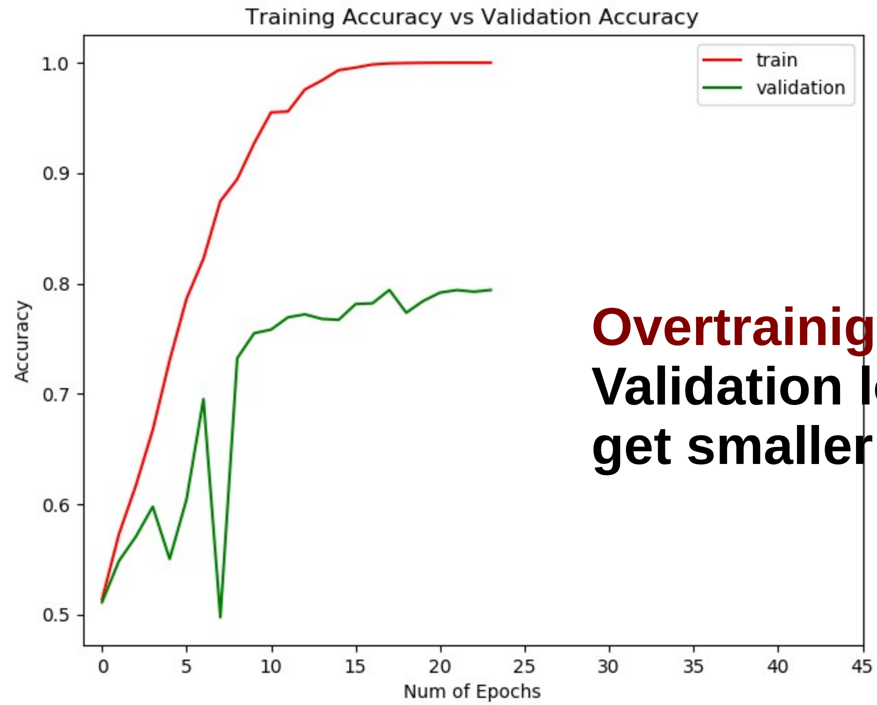
- Simple two layer fully connected network built
  - Two hidden layers, 512 nodes in each layer
  - One output layer – 3 nodes (one node for each shape)
  - Output: three numbers giving the „probability” of each figure shape.



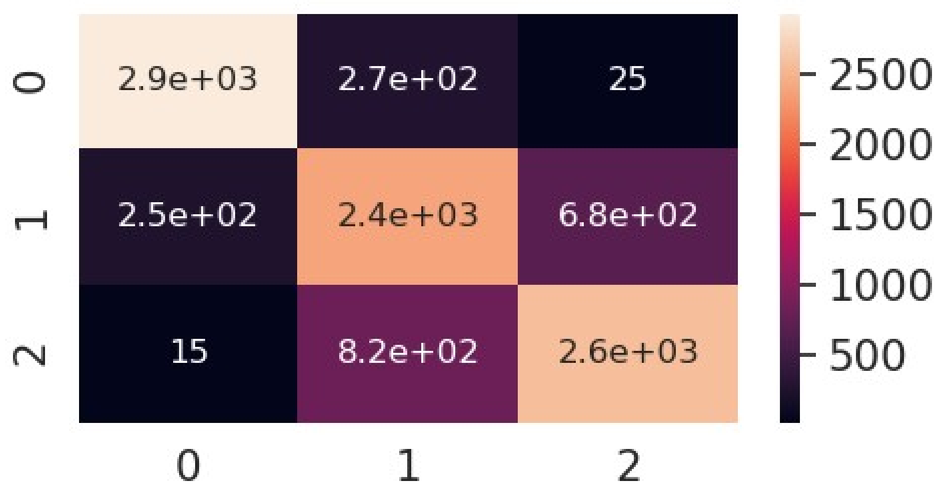

```
Total params: 788,995
Trainable params: 788,995
Non-trainable params: 0
_____________________________________________________________________________
             OPERATION           DATA DIMENSIONS   WEIGHTS(N)

                 Input   #####      1   32   32
               Reshape     |   -------------------         0
                         #####     32   32    1
               Flatten   ||||| -------------------         0
                         #####       1024
                 Dense   XXXXX -------------------    524800
                  relu   #####        512
                 Dense   XXXXX -------------------    262656
                  relu   #####        512
                 Dense   XXXXX -------------------      1539
               softmax   #####          3
               Reshape     |   -------------------         0
                         #####      1    3
```

Simple network performance





Overtrainig
Validation loss doesn’t get smaller

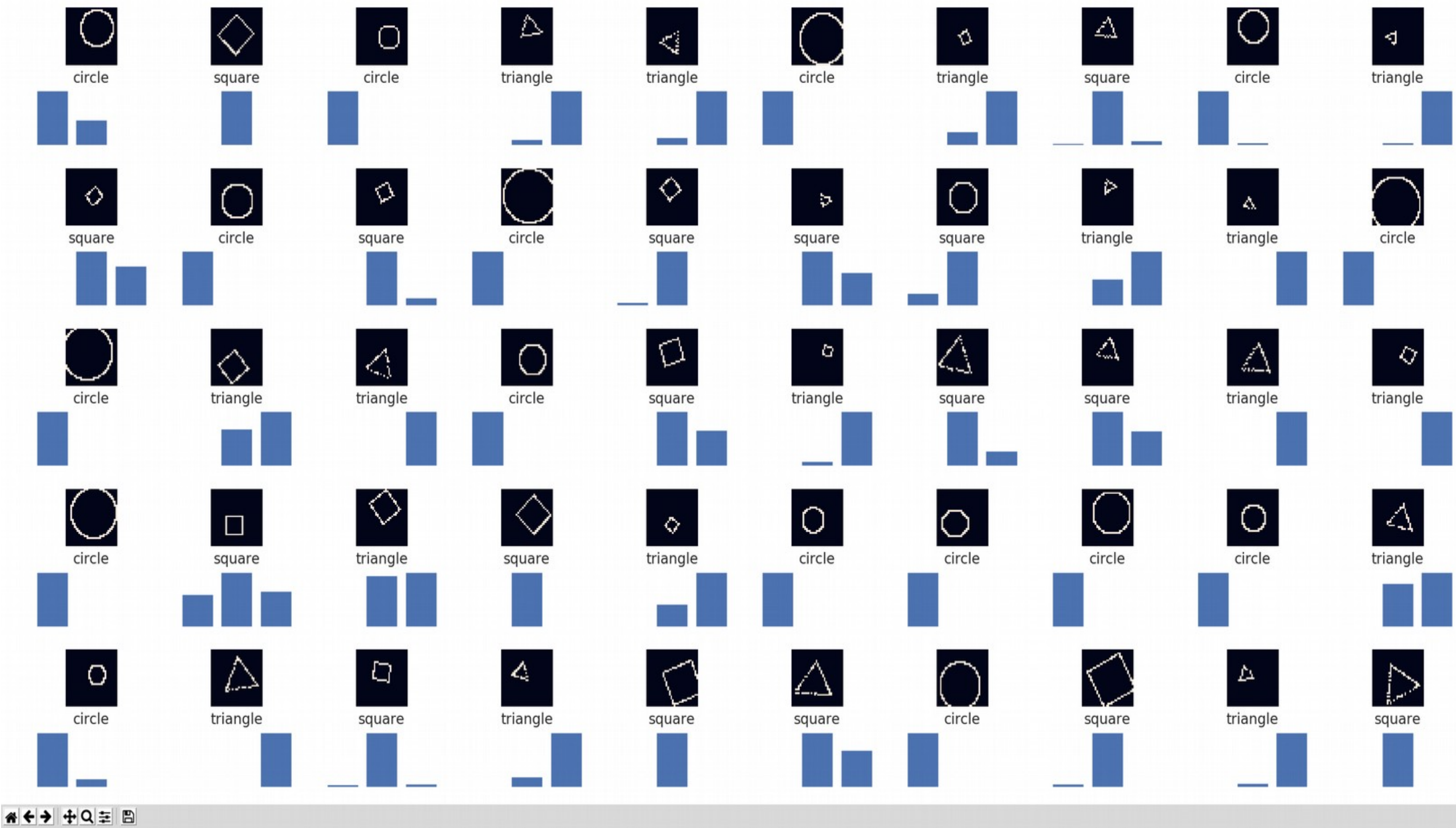


Confusion matrix – shows which classes are misclassified.

Accuracy: 79.51%

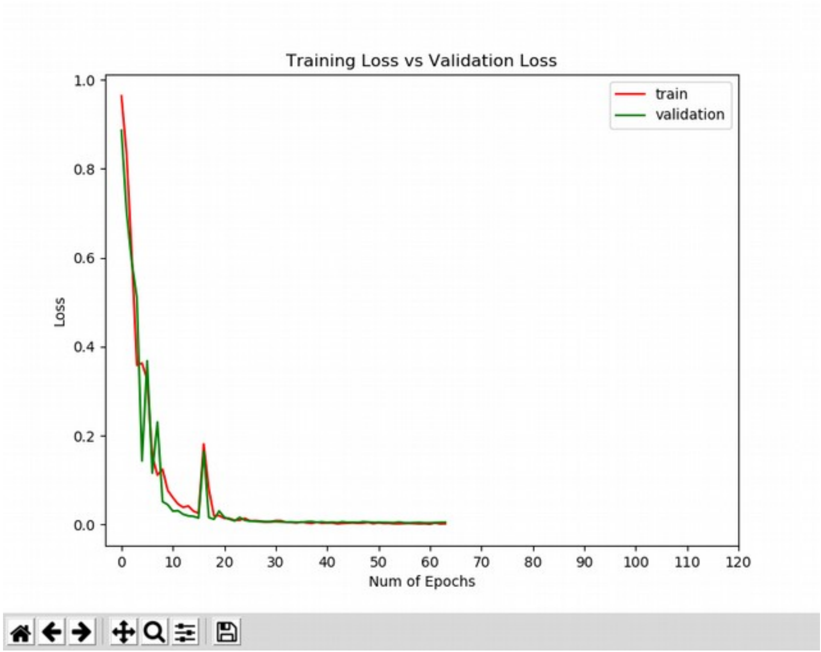
Code for KERAS network:
https://indico.ifj.edu.pl/event/232/
file: figure_cnn_simple.py

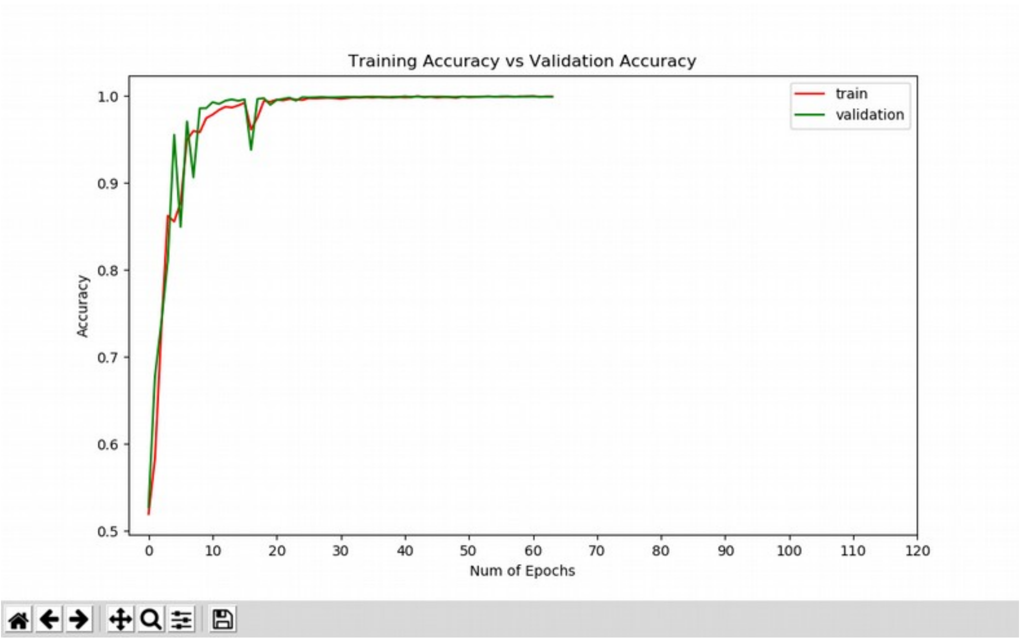
Simple network performance

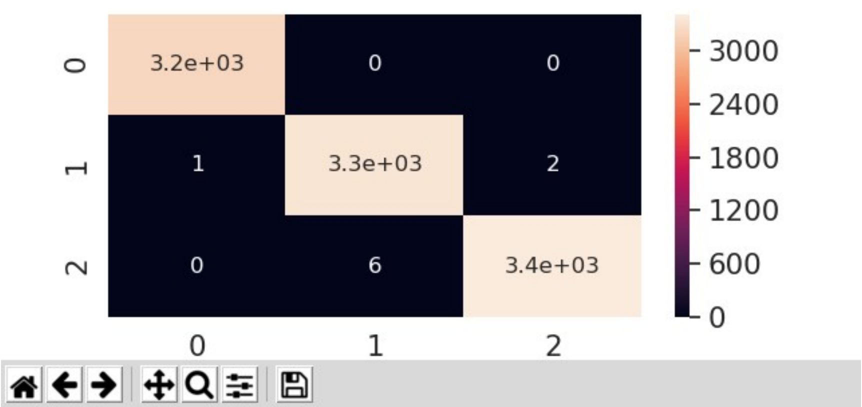


Convolutional network

- Bigger, convolutional deep neural network
- Uses Conv2D layers, MaxPooling2D and Dropout layers
- *Performance:*

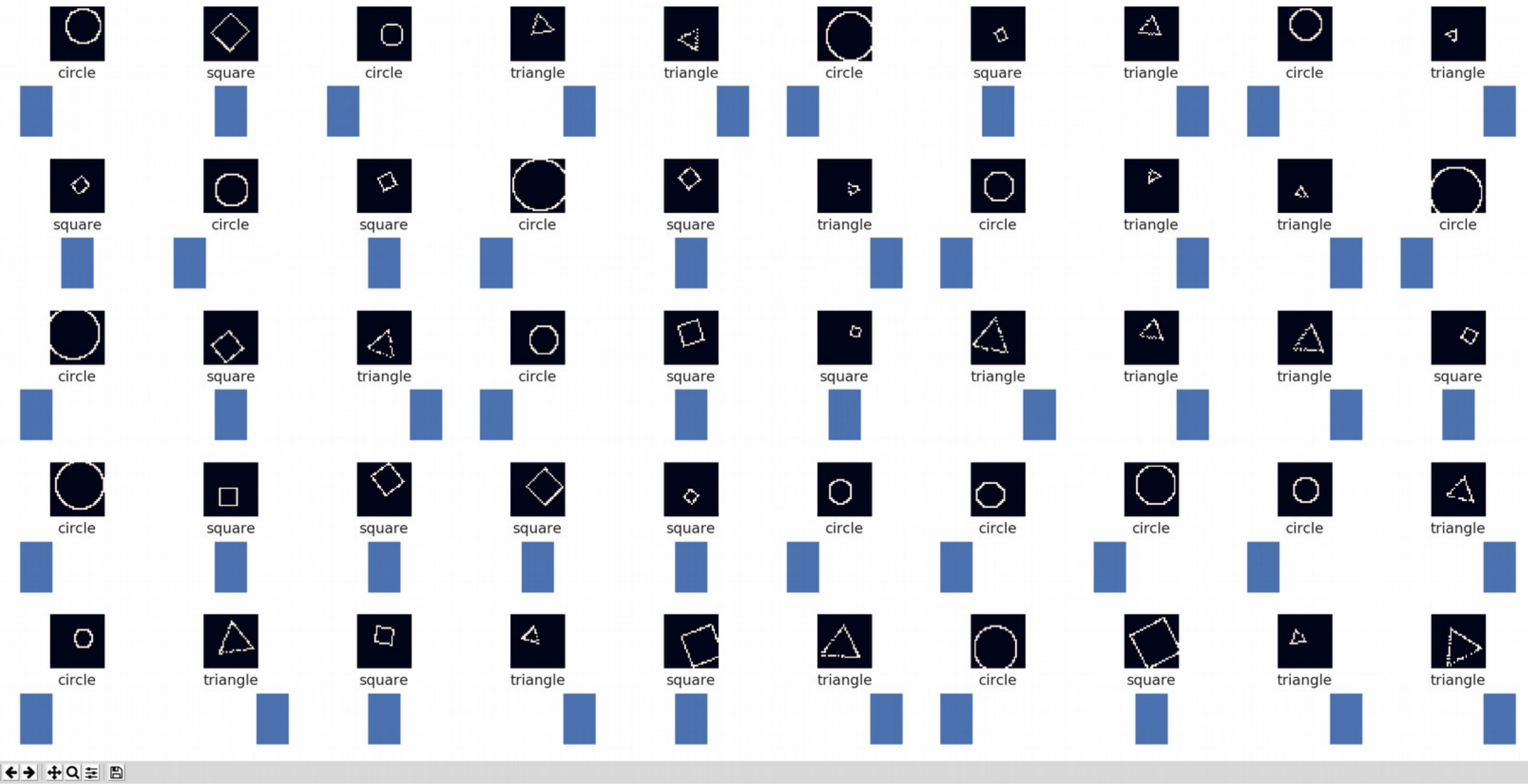






Accuracy: 99.91%

Convolutional network



Your task

- *Your task: **IMPROVE THE SIMPLE MODEL**. Try to match and outperform the network above!*
- ***Let's make a competition!***

Competition results: *write your results here:*

https://docs.google.com/document/d/1Y63SPeJRx95JtSpMqemniQxKc0-WqGQf4AgXWCveSyk/edit?usp=sharing

Good luck!