Machine learning applications in subatomic physics

Artur Kalinowski
Faculty of Physics
University of Warsaw

Machine Learning

What is a Machine Learning (ML)?

Machine learning is a statistical analysis with complex and automatized methods.

• a main assumption is that a problem can be formulated as a quest for some probability distribution **p(x), x – a input data**

•machine learning development is mainly driven by so called "Data Mining" or "Big data": attempts to analyze large data sets available to "industry" in order to infer any possible knowledge

• image recognition is one of main applications driving ML development

• other driver is a NLP: Natural Language Processing

https://www.google.com/recaptcha

ImageNet Classification with Deep Convolutional Neural Networks (AlexNet 2012)

A neuron

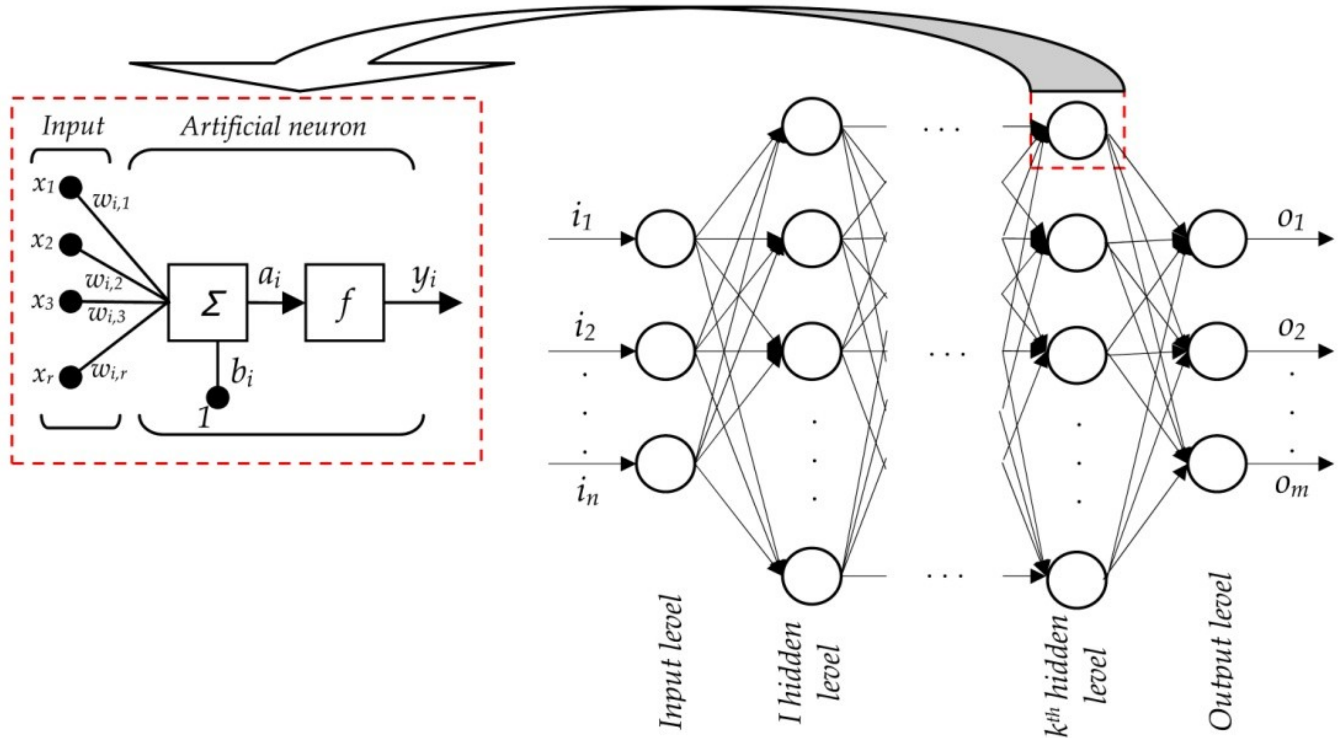
(Artificial) Neural Network (ANN):

- invented in 1957
- a system of connected units, neurons, performing averaging of input variables to obtain a number of output values
- averaging is performed at each neuron using a set of weights for its inputs, and "activation function"
- **training** – process of finding the parameters minimizing some loss function:
 f(output, expected value)

 often f(...) is a MSE: mean square error:

$$f\left(output\,, expected\ value\right) = \frac{1}{N}\sum\left(output - expected\right)^2$$

Artificial Intelligence Techniques for Modelling of Temperature in the Metal Cutting Process

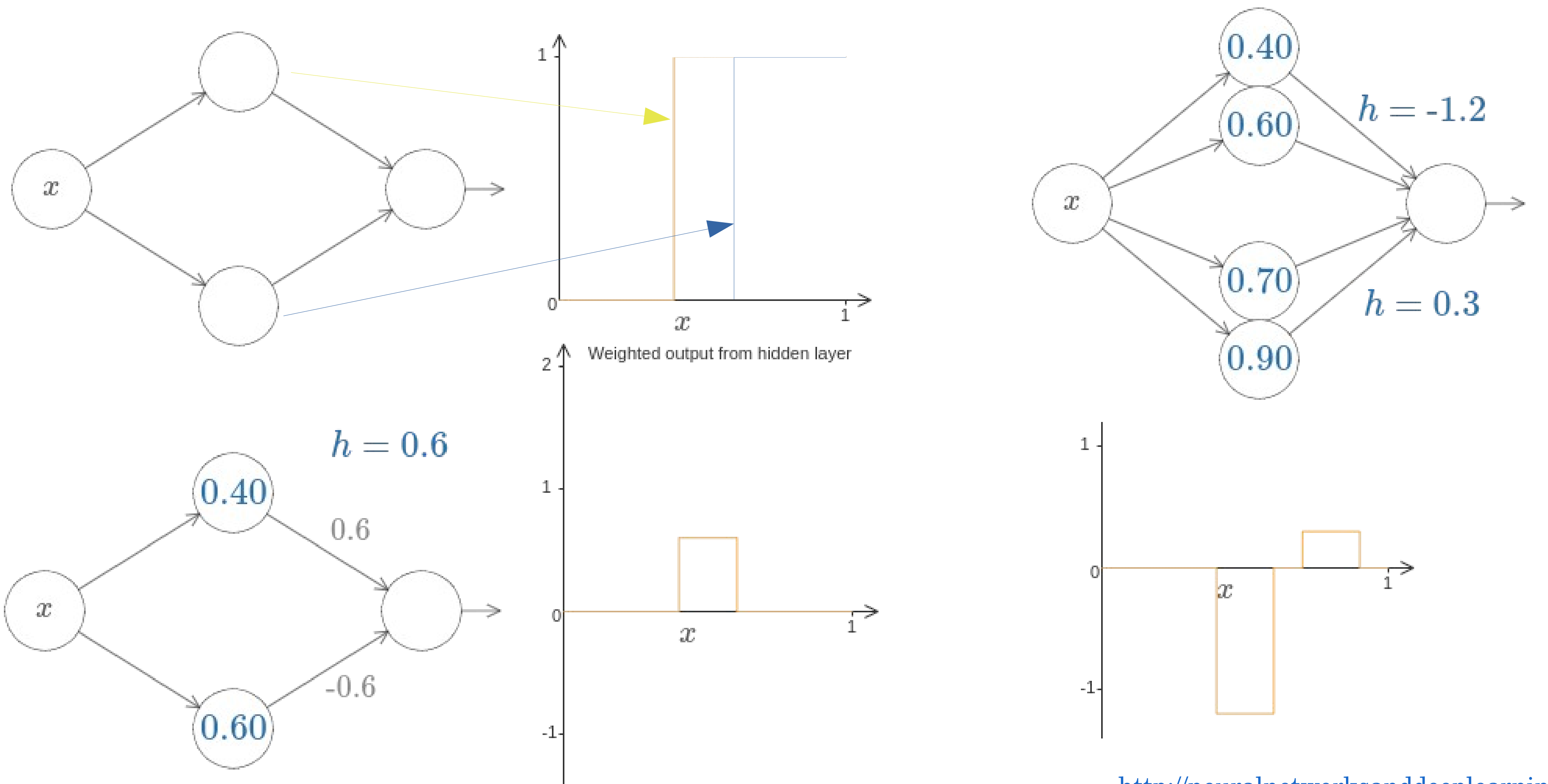


$$a_i = x_1 w_{i,1} + x_2 w_{i,2} + \ldots + x_r w_{i,r} + b_i$$

$$y_i = f\left(a_i\right) = f\left(\sum_{j=1}^{r} x_j w_{i,j} + b_i\right)$$

Neural Network approximator

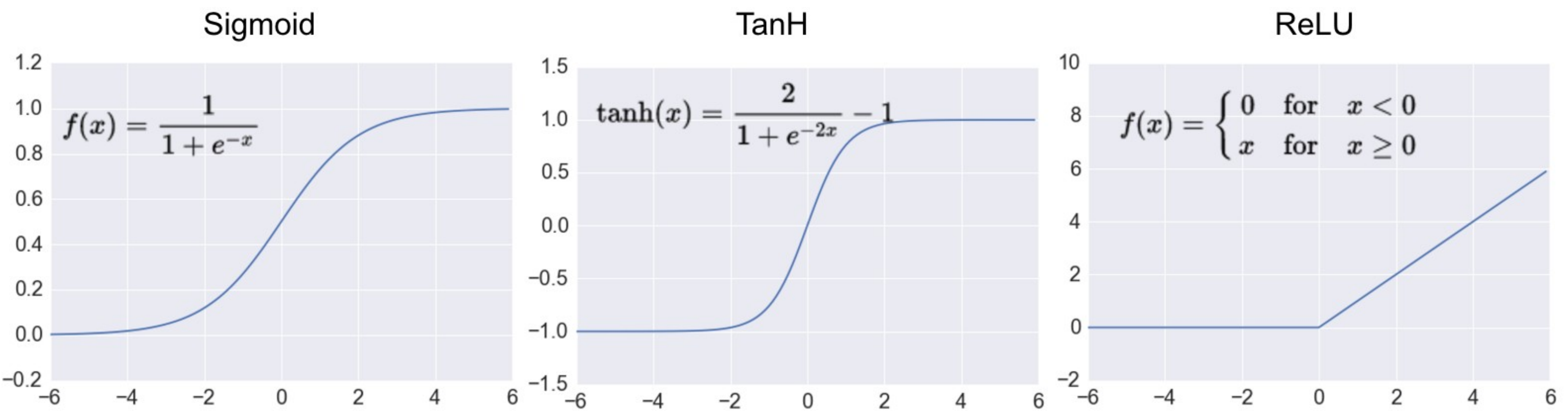
The universal approximation theorem: any smooth function can be approximated with a NN with a single hidden layer with finite number of neurons.



http://neuralnetworksanddeeplearning.com

Deep Learning advent

$$y_i = f(a_i) = f\left(\sum_{j=1}^{r} x_j w_{i,j} + b_i\right)$$



http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

Activation function:
- Rectified Linear Unit (ReLU): **nowadays a most common activation function.**

More computing power:
- Graphical Processing Units (GPUs) provide up to 100x faster training

More training data:
- Big memory, big CPU, big GPU allows use of BIG training datasets

A regression

K. Rolbiecki (IFT UW) et. al.

Reggression: instead for looking for a full **p(x), x – a input data, one seeks only a mean or median of p(x)**

The task: calculate NLO cross section for a MSSM process for any, out of 19, parameter value.

The current NLO codes (Prospino) take O(3’) to calculate $\sigma(pp \rightarrow \tilde{\chi}^{+}\tilde{\chi}^{-})$

The neural network was used to parametrise NLO cross sections from Prospino in pMSSM-19.

The data: 10^7 points in dim=19 parameter space of LO an 10^5 of NLO cross sections

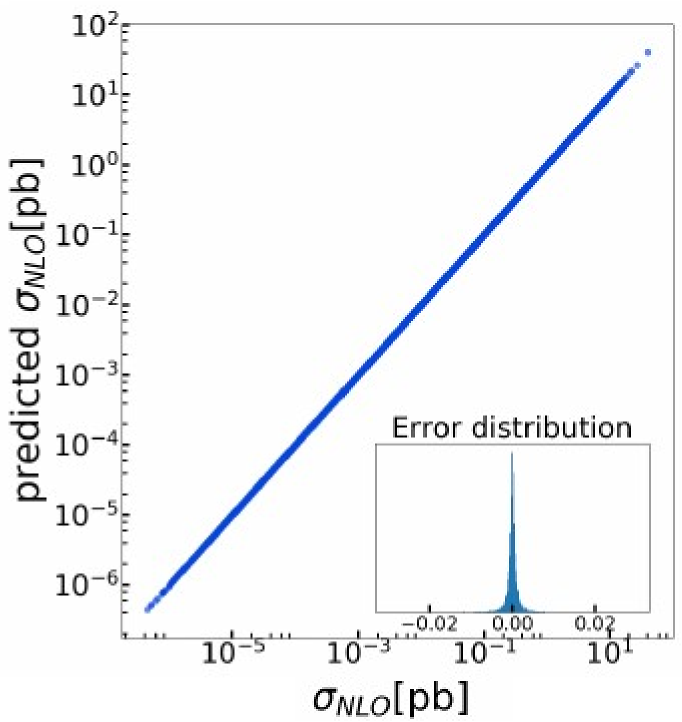
A regression model

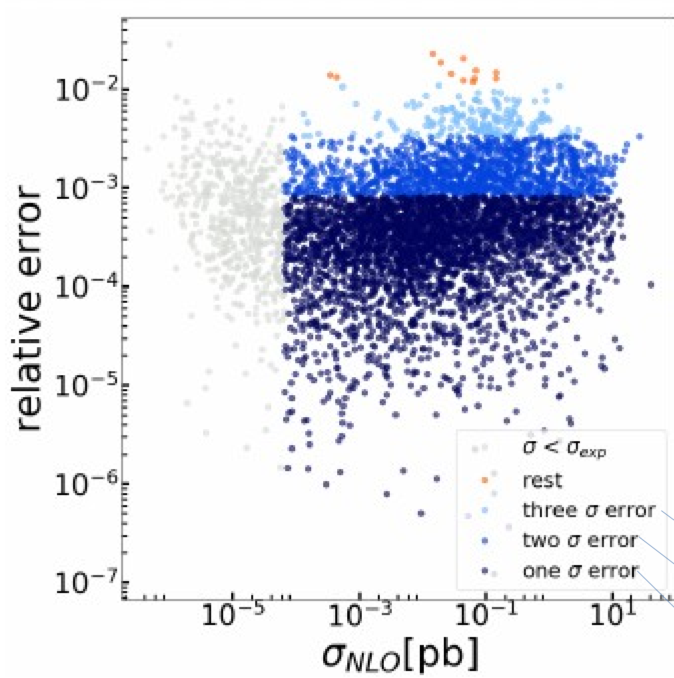
K. Rolbiecki (IFT UW) et. al.

The model: 8 hidden layers with 100 neurons each for LO parametrisation
8 hidden layers with 32 neurons each for NLO/LO k-factor parametrisation
Loss function: Mean Absolute Percentage Error:

$$\mathrm{MAPE} = \frac{1}{N}\sum_{i=0}^{N}\left|\frac{y_{\mathrm{true,i}} - y_{\mathrm{pred,i}}}{y_{\mathrm{true,i}}}\right|$$

arXiv:1810.08312





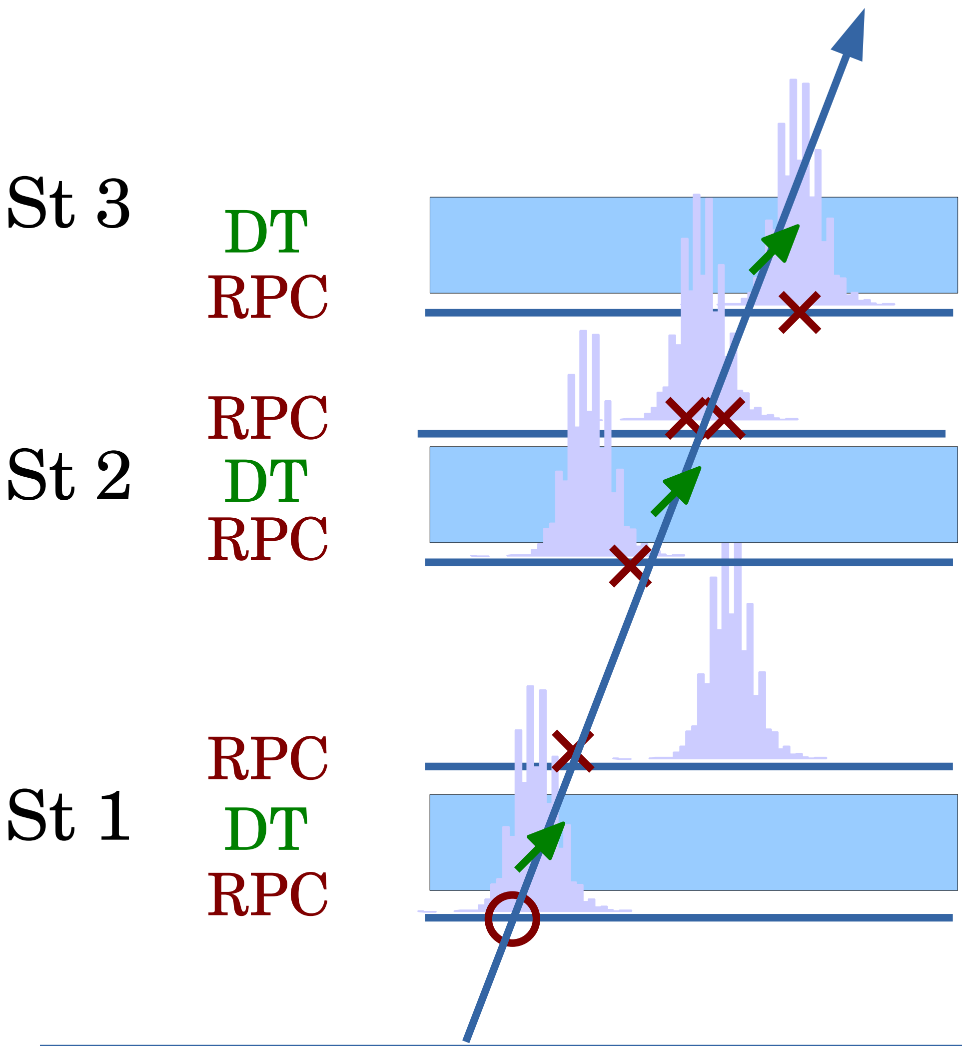
The result: cross section evaluated with precision of <2% for 95% of parameter space points.

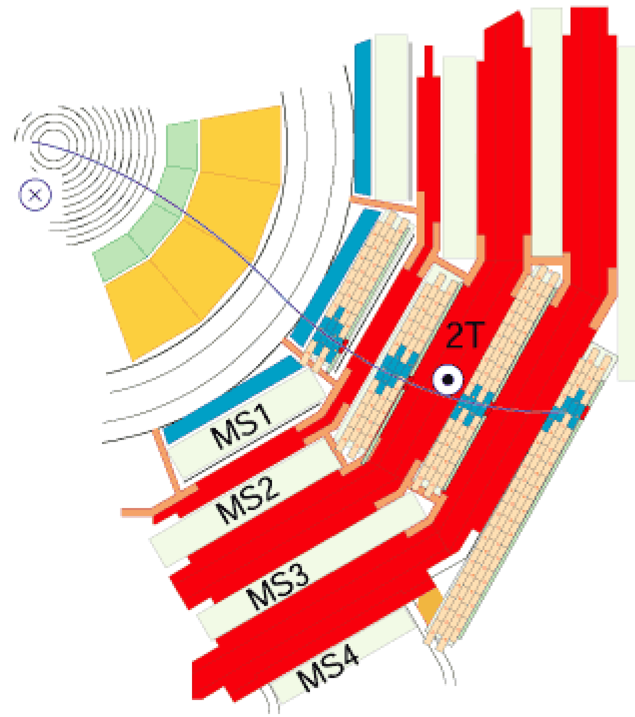
Computing time 5-6 orders of magnitude faster running on CPU

99.7% of points
95% of points
68% of points

CMS@Warsaw ML activities: OMTF

The task: use a NN model to reconstruct p_T at the CMS level 1 muon trigger



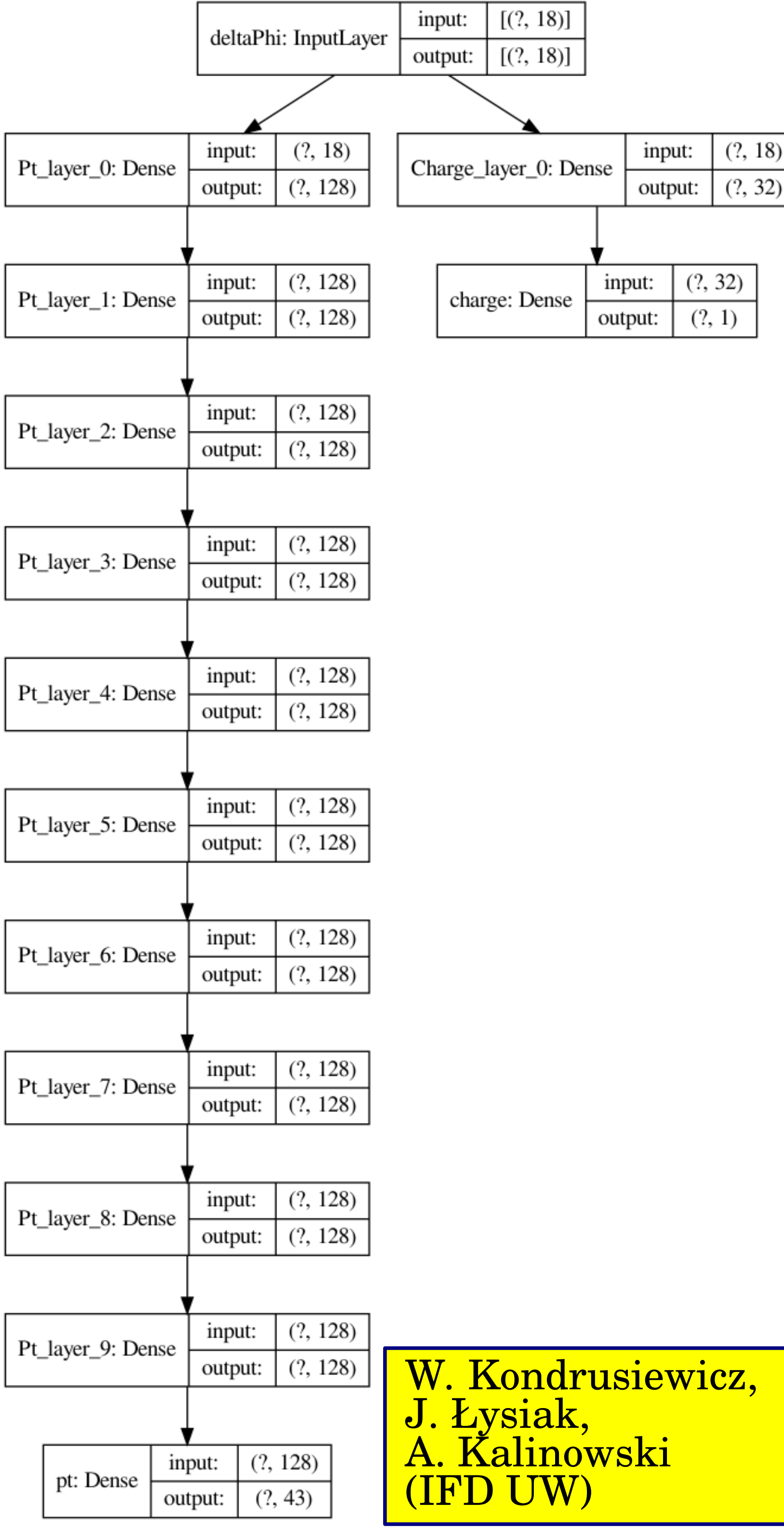


- current algorithm (**naive Bayes approximation**): given hit pattern, choose a p_T that maximizes the sum of hit probabilities in each layer. Neglects any interlayer correlations

OMTF NN model

The model:
- 10 fully connected layers, 128 neurons each
- output 43 neurons corresponding to 43 bins in p_T

The result:
- probability that a given candidate has p_T in given rage.



W. Kondrusiewicz,
J. Łysiak,
A. Kalinowski
(IFD UW)

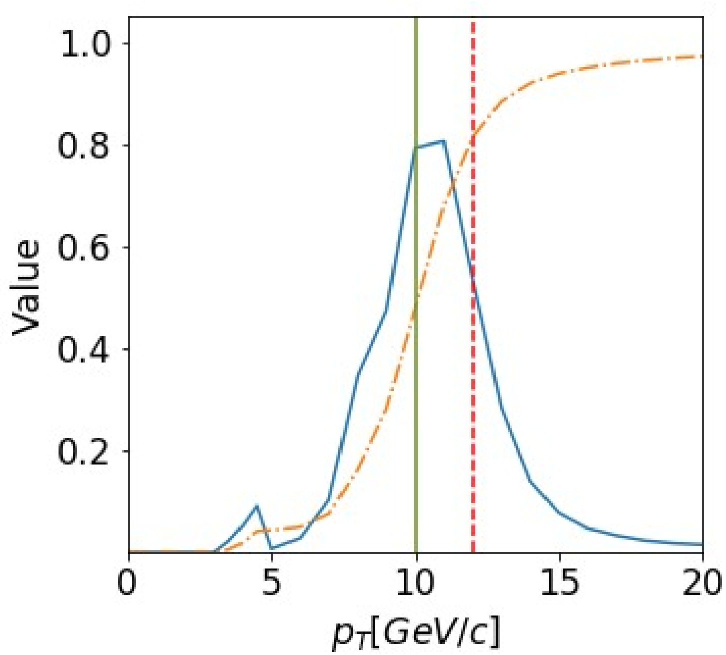
OMTF NN model

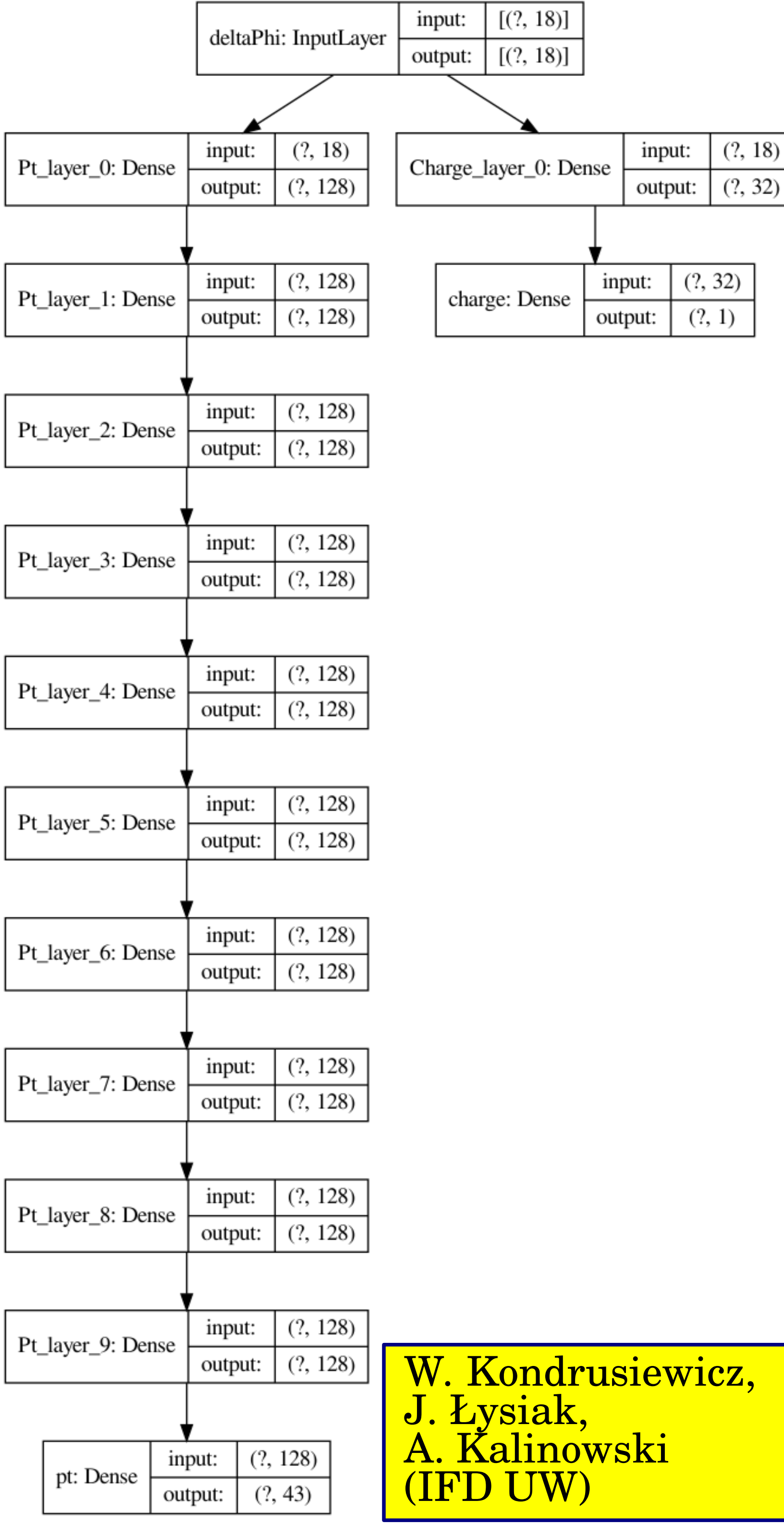
The model:

- 10 fully connected layers, 128 neurons each
- output 43 neurons corresponding to 43 bins in p_T

The result:

- probability that a given candidate has p_T in given rage.





W. Kondrusiewicz,
J. Łysiak,
A. Kalinowski
(IFD UW)

OMTF NN model

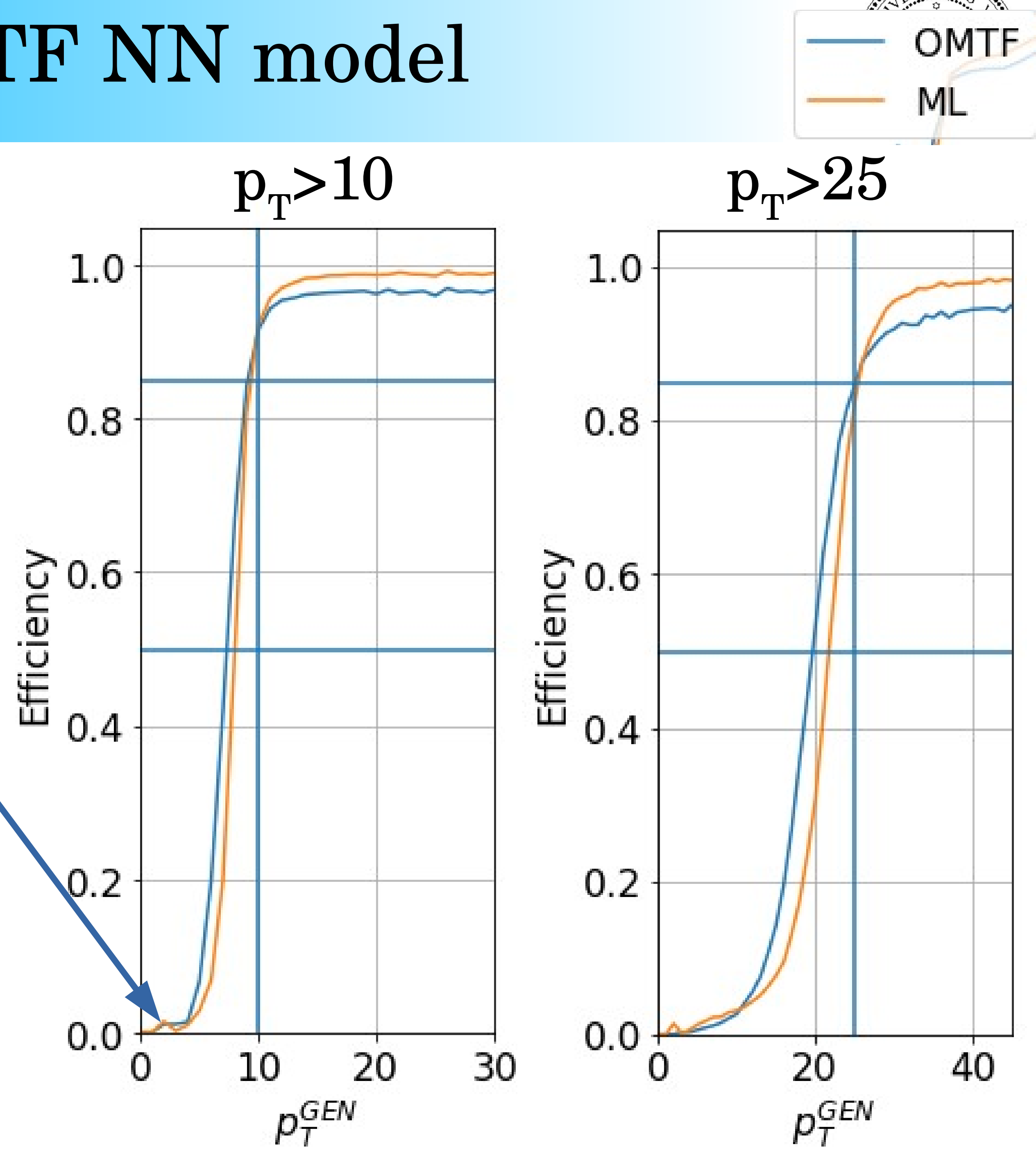
The trigger:
- does a candidate have p_T>X?

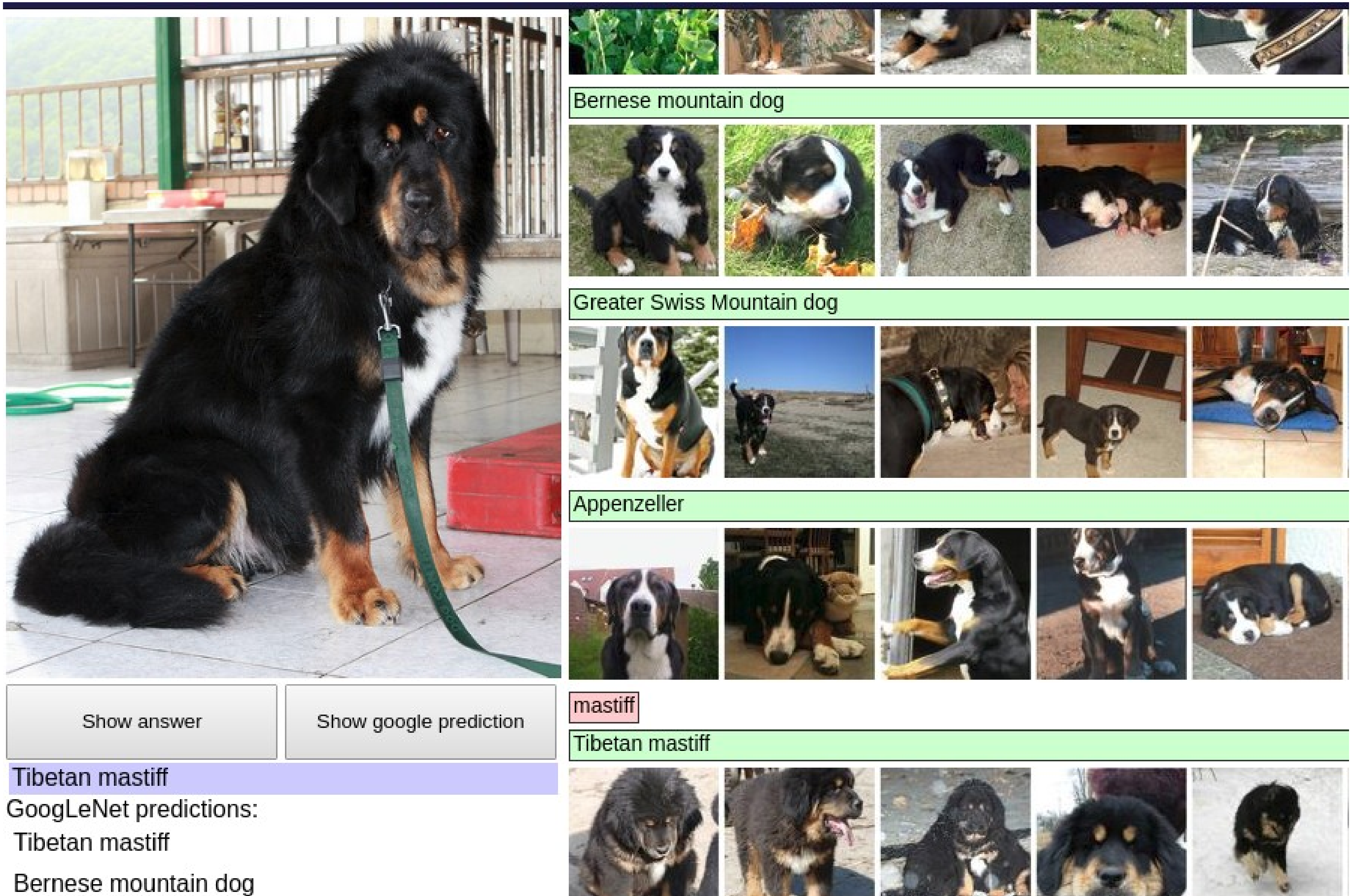
Human vs Machine:
- overall ML model works better

- still there are some specific cases, better treated by a model invented by a human

- in this case those rare specific cases are crucial for the model performance

- other issue is ML model implementation in trigger hardware (FPGA)





1k categories

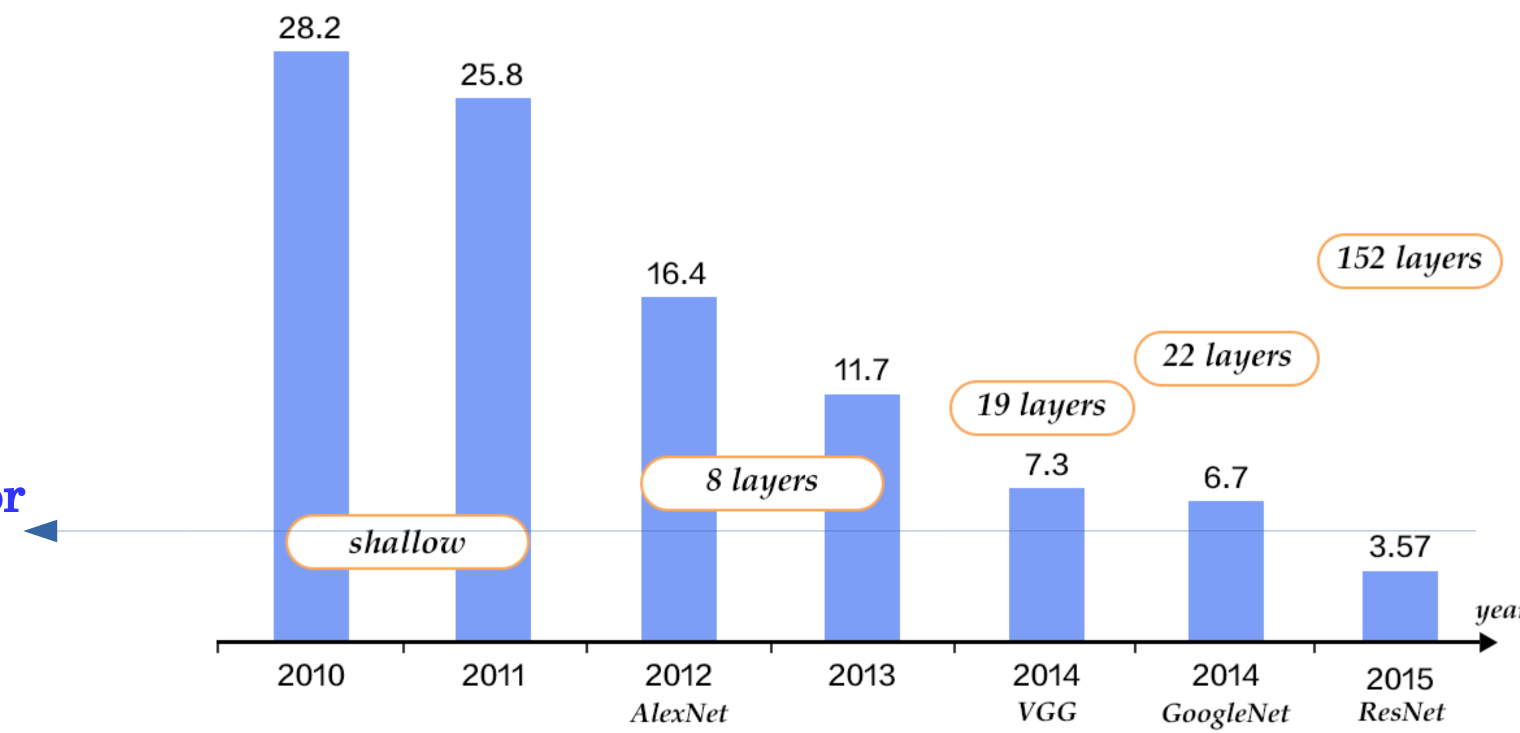
http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

Deep Learning

ImageNet is a data set for Large Scale Visual Recognition Challenge (ILSVRC) started in 2010

top-5 error rate – fraction of images where the correct label in not within 5 most probable (according to DNN)

Human top-5 error rate = 5%

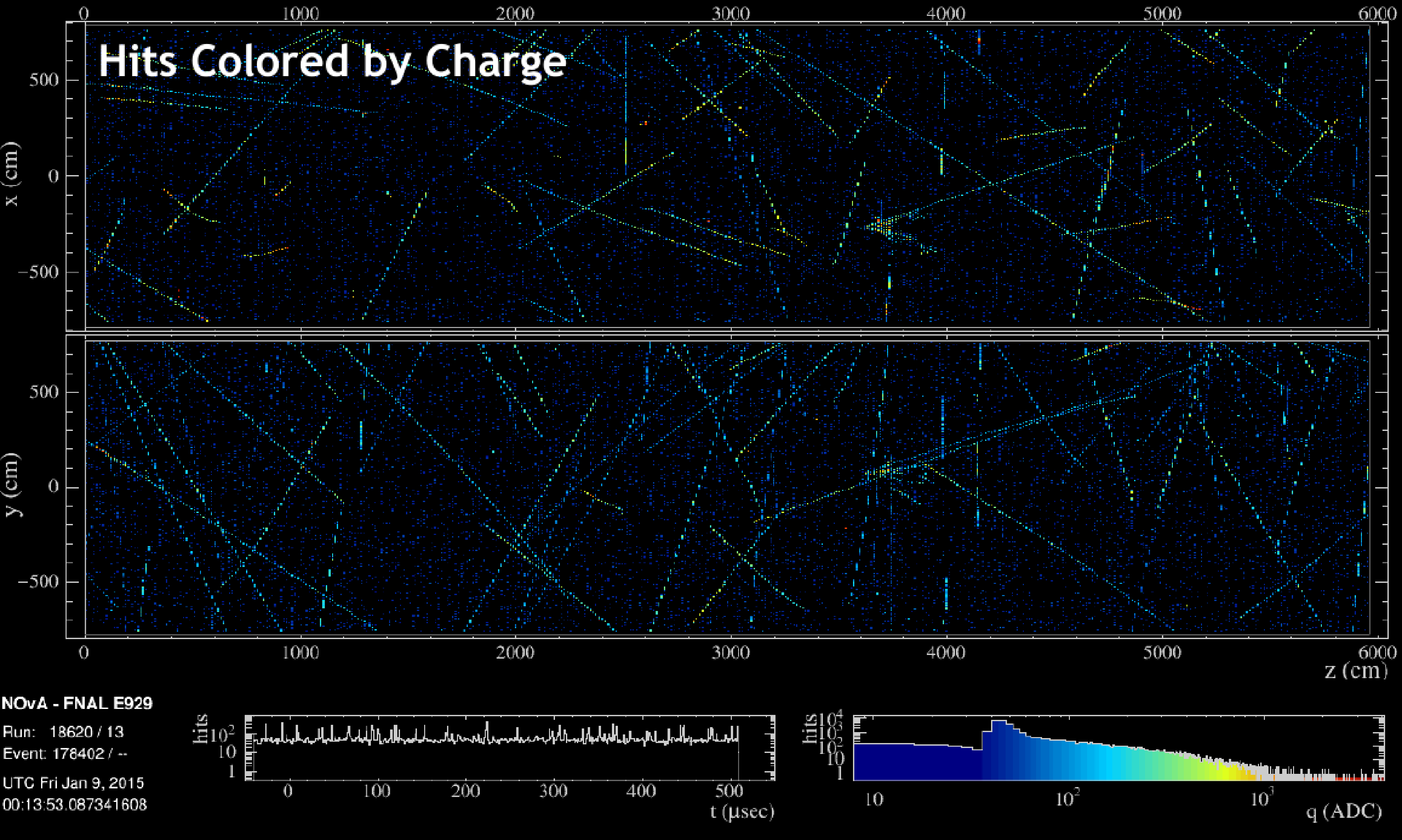


http://book.paddlepaddle.org/03.image_classification/

DNN in neutrino physics

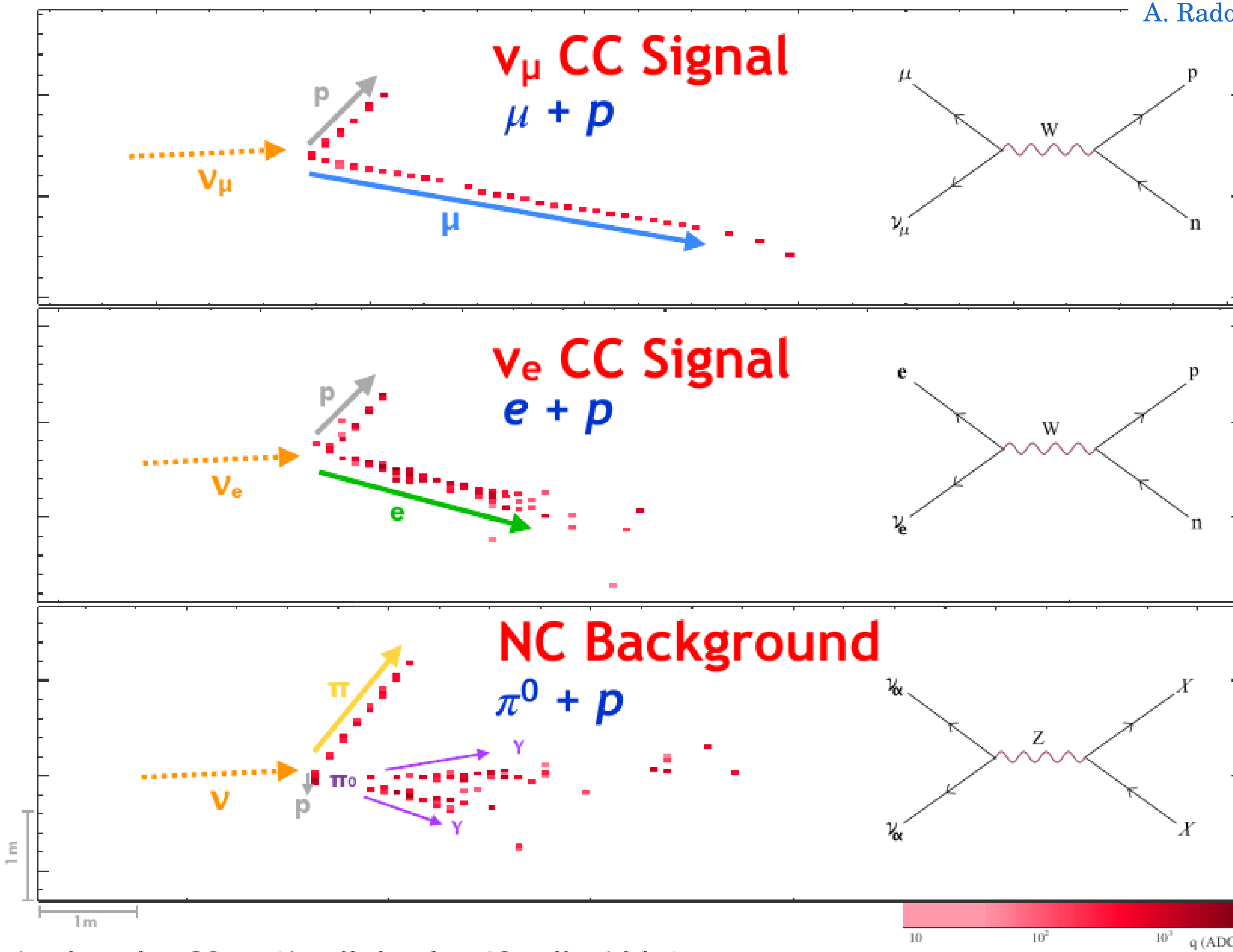
A. Radovic, DS@HEP 2017

550 μs exposure of the NOvA Far Detector



DNN in neutrino physics

A. Radovic, DS@HEP 2017

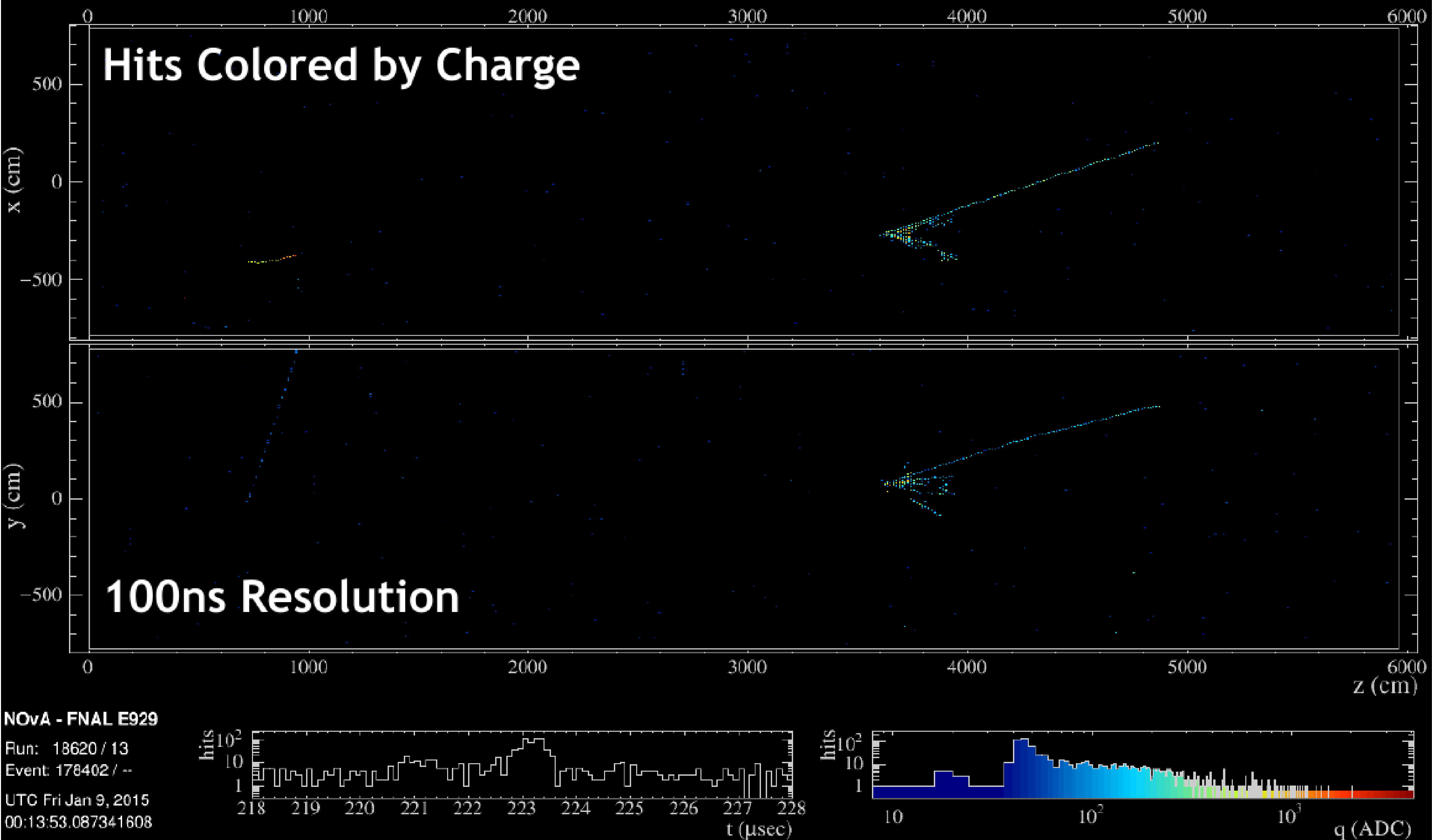


1 radiation length = 38cm (6 cell depths, 10 cell widths)

DNN in neutrino physics

Time-zoom on 10 μs interval during NuMI beam pulse

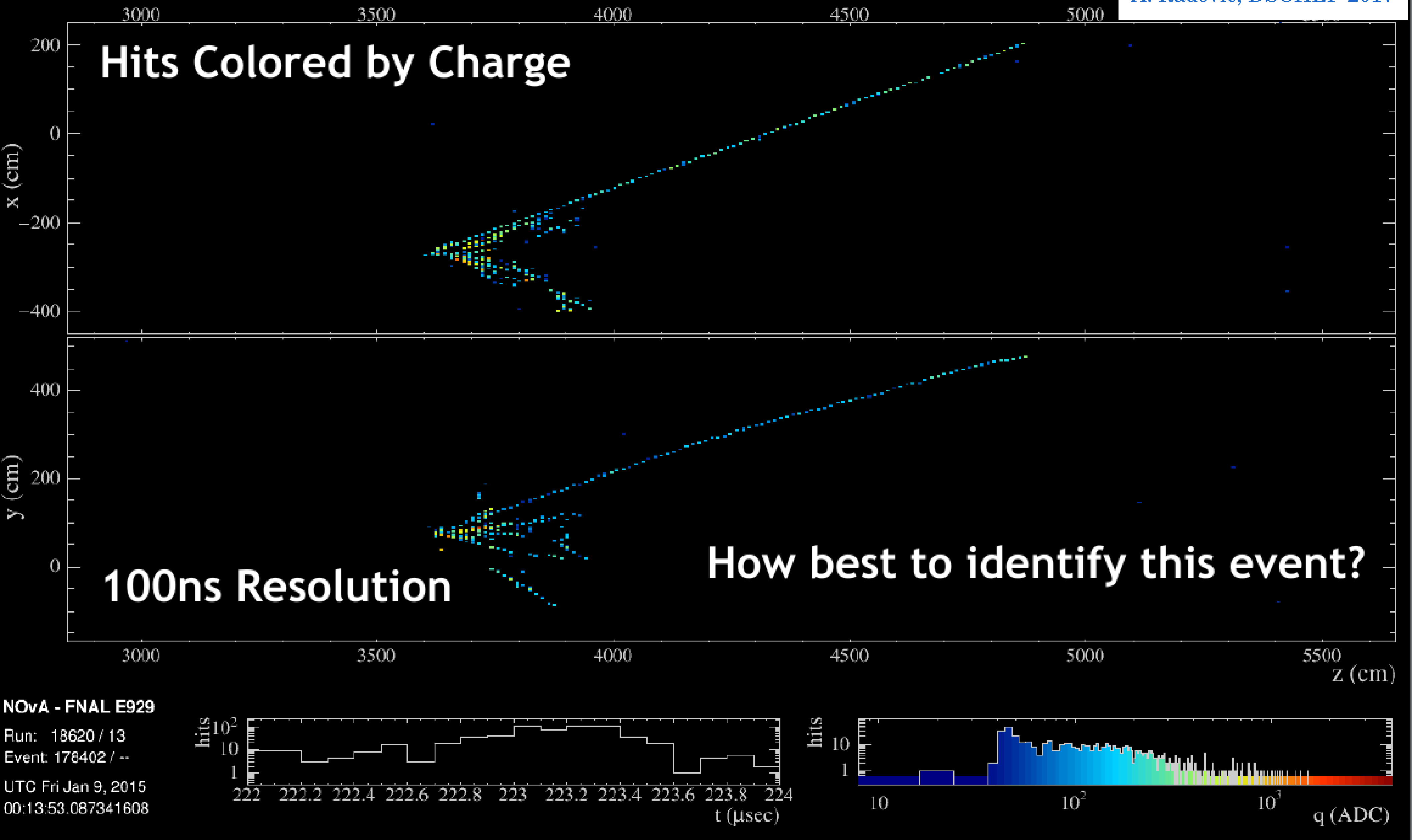
A. Radovic, DS@HEP 2017



DNN in neutrino physics

Close-up of neutrino interaction in the NOvA Far Detector

A. Radovic, DS@HEP 2017

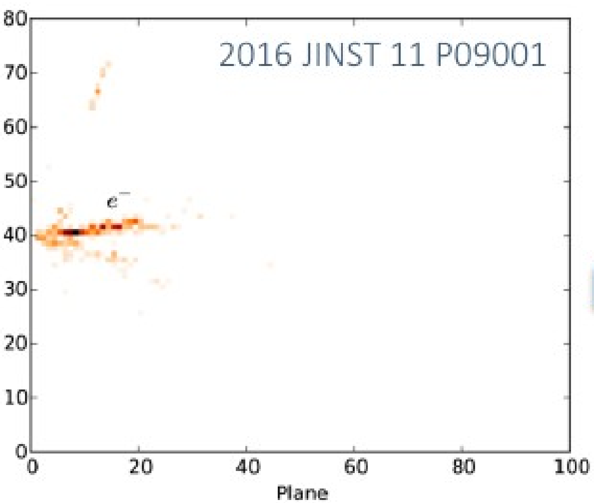


DNN in neutrino physics

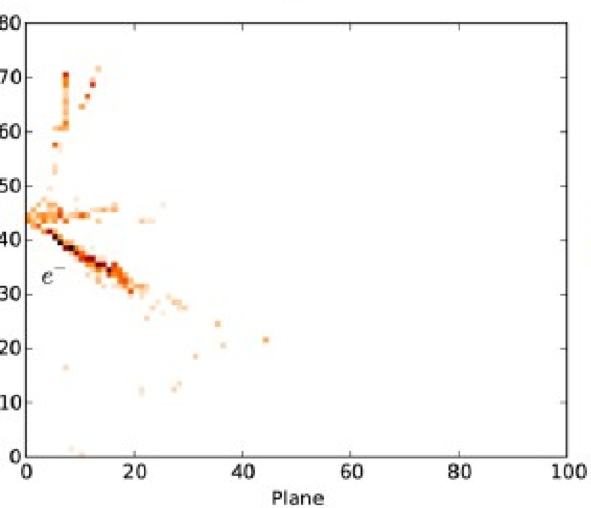
CNN applied to ν_e selection in NOvA

R. Sulej, CERN-EP/IT Data science seminar

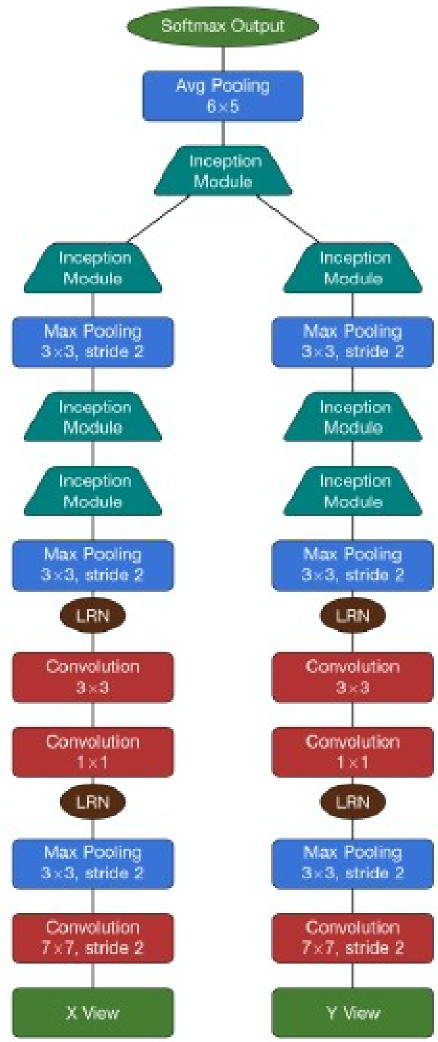
Input: image-like raw charge in 2D projections (NOvA),



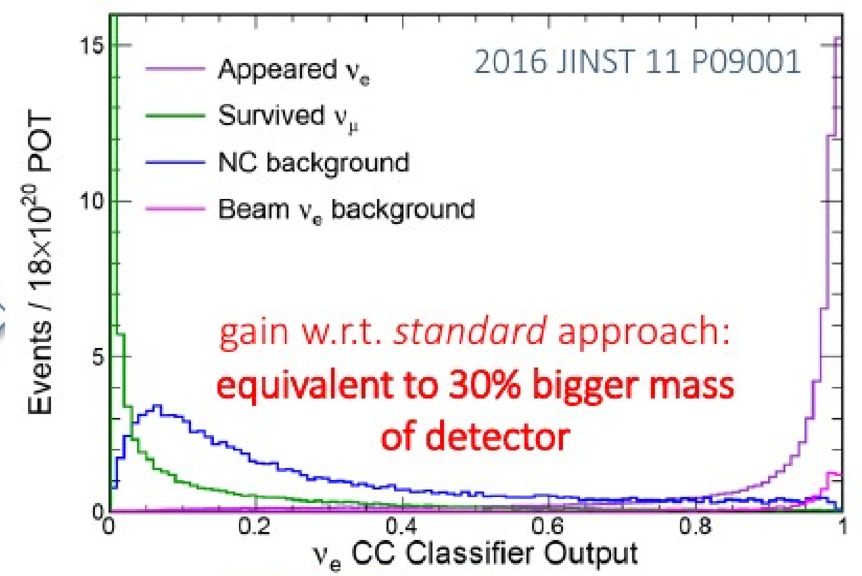
X-view



Y-view



reconfigured GoogLeNet, Caffe toolkit
2016 JINST 11 P09001



73% signal selection efficiency

[arxiv:1604.01444]

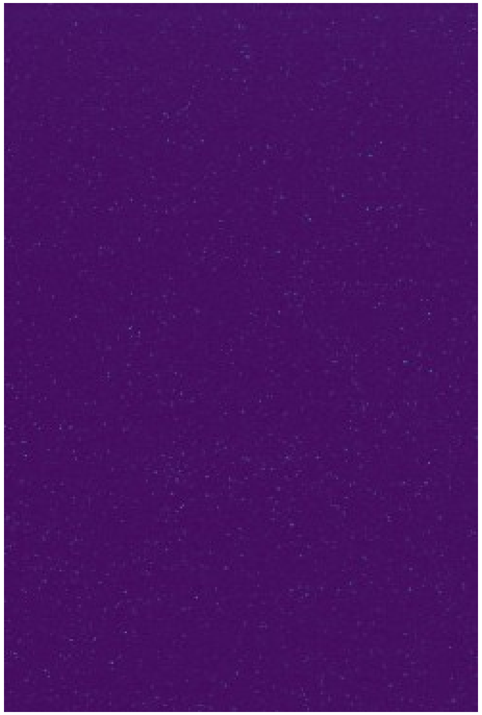
DNN in nuclear physics

N. Sokołowska
(IFD UW)

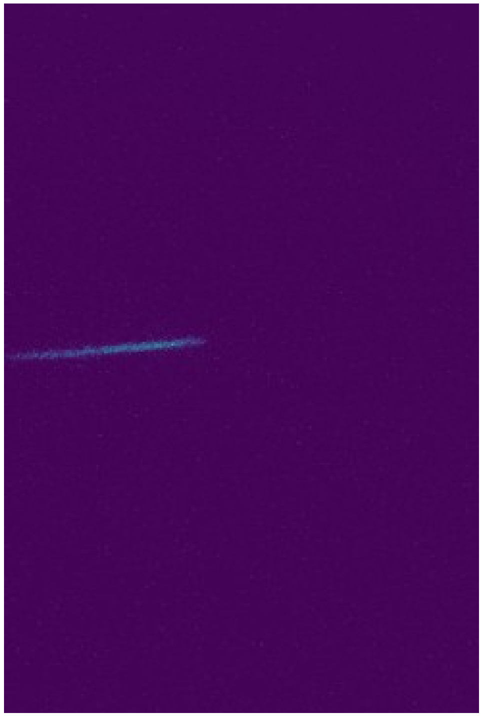
The data: $3 \cdot 10^6$ nuclear reaction photos from an TPC with optical readout (OTPC)

The task: assign one of five labels to a photo:

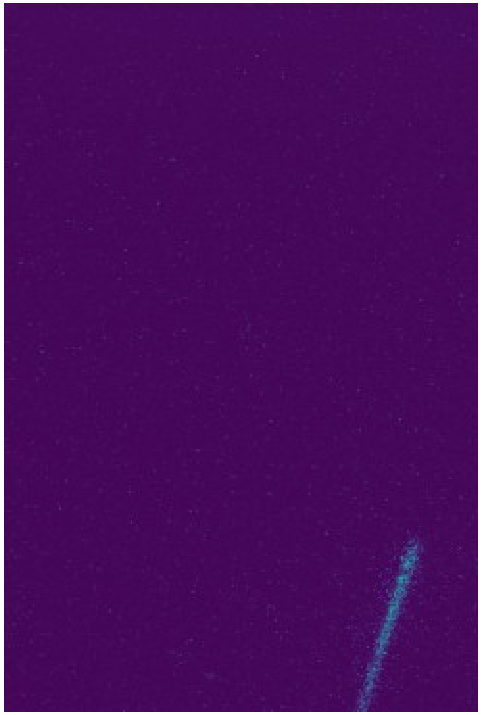
Empty (97%)



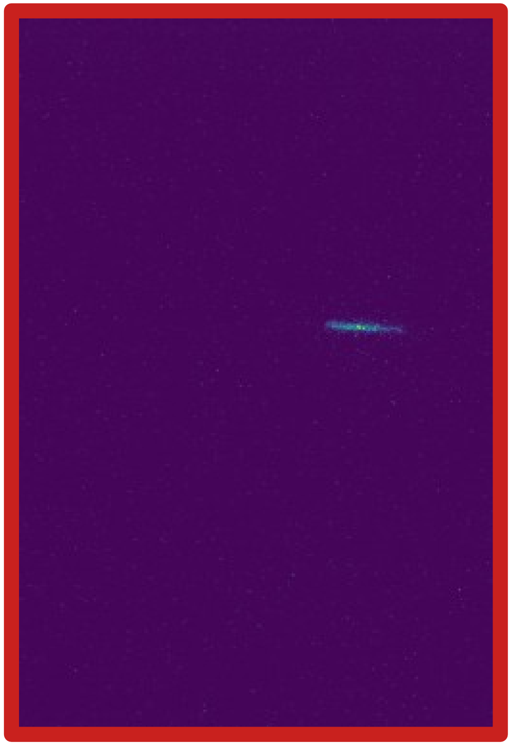
Calibration source (2%)



Physical backgrond (0.3%)



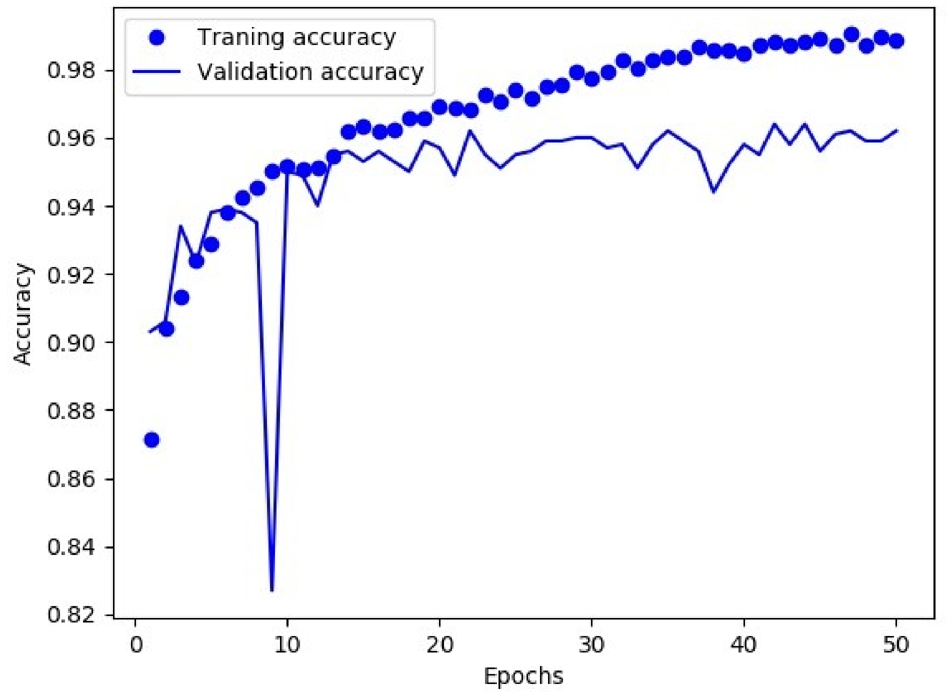
Signal (0.2%)



DNN in nuclear physics

N. Sokołowska
(IFD UW)

A preliminary result: 96% events with correct category assignment

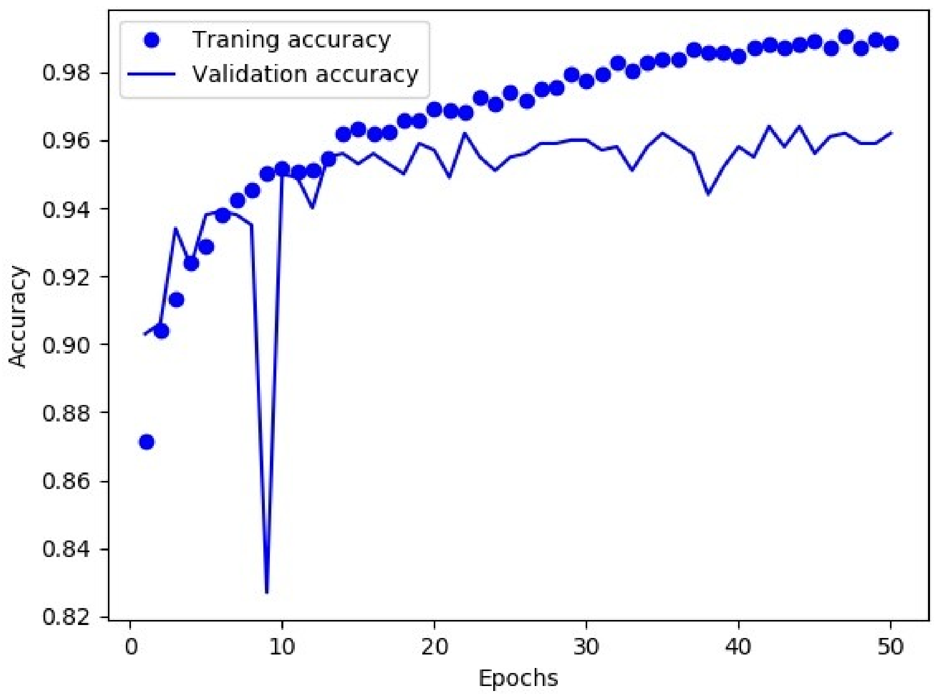


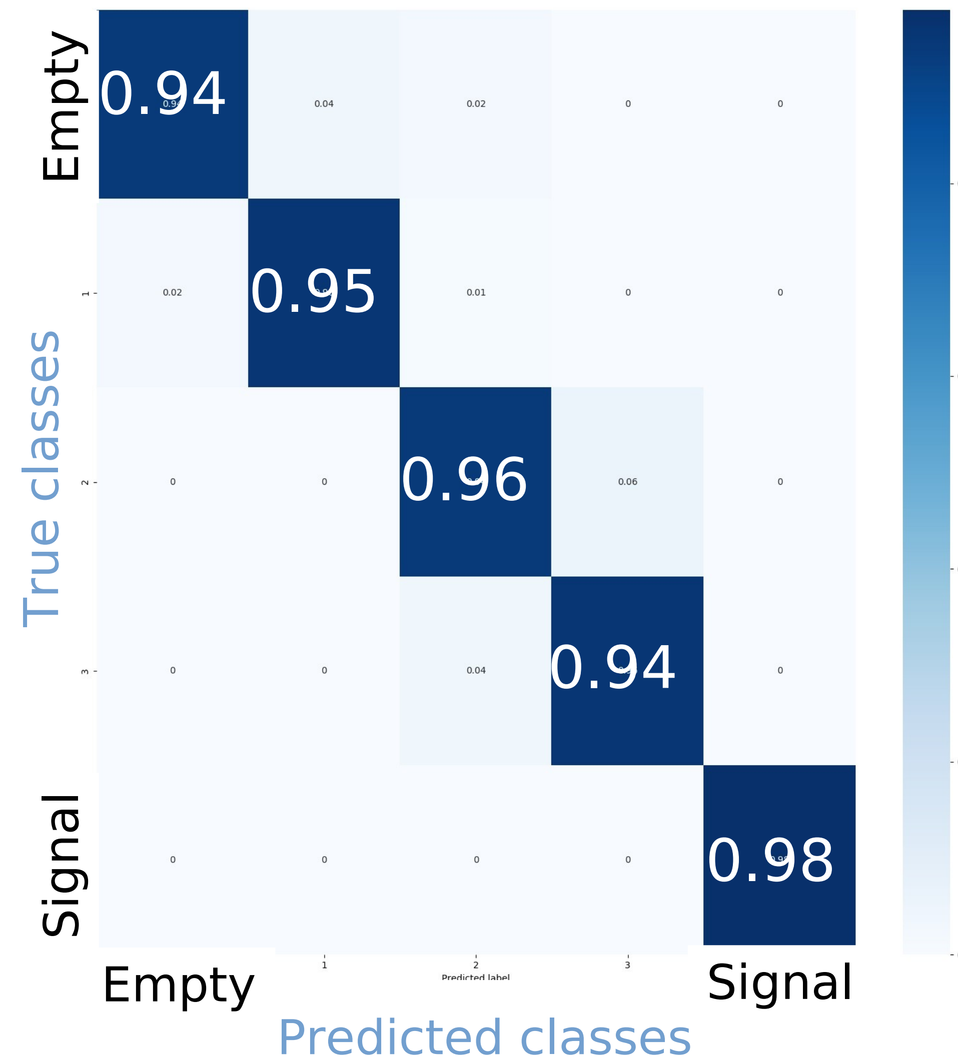
A small font note: 97% of events belong to the "empty" category.

DNN in nuclear physics

N. Sokołowska

A preliminary result: 96% events with correct category assignment





Confusion matrix – visualisation of true class predicted class correspondence

How to get started?

The software: many packages available on the market, all use Python. I use TensorFlow from Google. Many, large pretrained networks are available there:

Transfer learning with a pretrained ConvNet

Run in Google Colab | View source on GitHub | Download notebook

In this tutorial, you will learn how to classify images of cats and dogs by using transfer learning from a pre-trained network.

The hardware: one can start with just a bare web browser and use cloud resources from Google: the Google Colaboratory:

+ Code + Text | Copy to Drive | RAM Disk

Connected to "Python 3 Google Compute Engine backend"
RAM: 0.81 GB/12.72 GB Disk: 30.91 GB/107.77 GB

What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to learn more, or just get started below!

How to get started?

A small training: for not too big network, with ~1M parameters the GPUs do not give too much speedup wrt. a fast CPU. For an everyday work I just use my desktop:
Core i7 2700, 16 GB RAM, no GPU

```
Epoch 1/10
221/221 [==============================] - 20s 92ms/step - loss: 7.4700 - pt_loss: 7.1259 - charge_loss: 0.3441 - pt_accuracy: 0.1619 - charge_accuracy: 0.9217
s: 6.0529 - val_pt_loss: 5.8352 - val_charge_loss: 0.2178 - val_pt_accuracy: 0.2653 - val_charge_accuracy: 0.9383
Epoch 2/10
221/221 [==============================] - 19s 87ms/step - loss: 5.9526 - pt_loss: 5.7544 - charge_loss: 0.1982 - pt_accuracy: 0.2718 - charge_accuracy: 0.9385
s: 5.7389 - val_pt_loss: 5.5688 - val_charge_loss: 0.1701 - val_pt_accuracy: 0.2928 - val_charge_accuracy: 0.9410
```

A large training: for a serious training one can use the PLGrid infrastructure. Requires registration and application for a computing grant. **The service is free for all members of Polish scientific community.**

At the moment I use prometheus cluster (located at AGH) with NVIDIA K40 GPUs:

	Prometheus	2160	2	12	24	Intel Xeon E5-2680 v3	2,5	128	5,33	haswell_2500mhz	
		72	2	12	24	Intel Xeon E5-2680 v3	2,5	128	5,33	haswell_2500mhz,tesla_k40d	C0

```
Your active PL-Grid grants on THIS site:
+-----------+-----------+-----------+------------------+------------------+-------------------+-----------------+----------+
| GrantID   | Start Date |  End Date  | Total Walltime [h] | Used Walltime [h] | Total Storage [GB] | Used Storage [GB] |  Group  |
+-----------+-----------+-----------+------------------+------------------+-------------------+-----------------+----------+
| cmsml3 (*) | 2020-01-19 | 2020-12-30 |           10 000 |            2 557 |               100 |              37 | plggcmsml |
+-----------+-----------+-----------+------------------+------------------+-------------------+-----------------+----------+
```

Conclusions

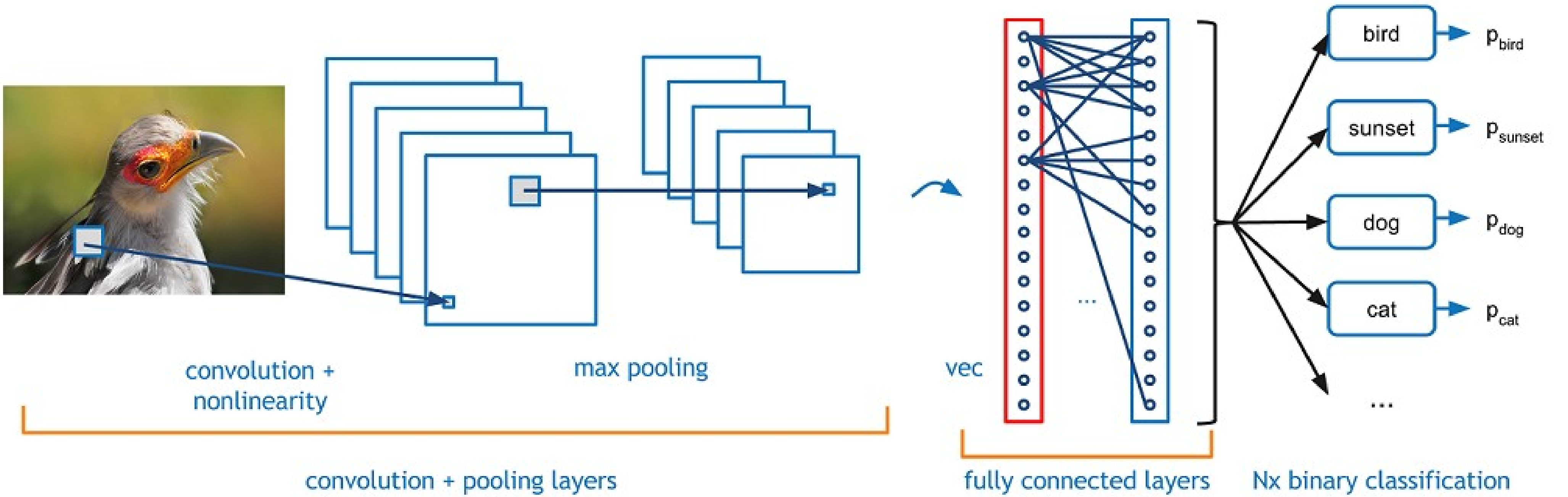
- Machine learning had made a huge development in last 5 years
- Ideas from industry are being extensively used within science
- **ML is the cutting edge of statistical data analysis (though not always as conscious as traditional approach)**

https://xkcd.com/1838/

Backup

A categorisation model

https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/



• a typical network (usually called a model) trained for image recognition consists of number of interleaved layers of convolution and pooling → **extraction of higher and higher level features**

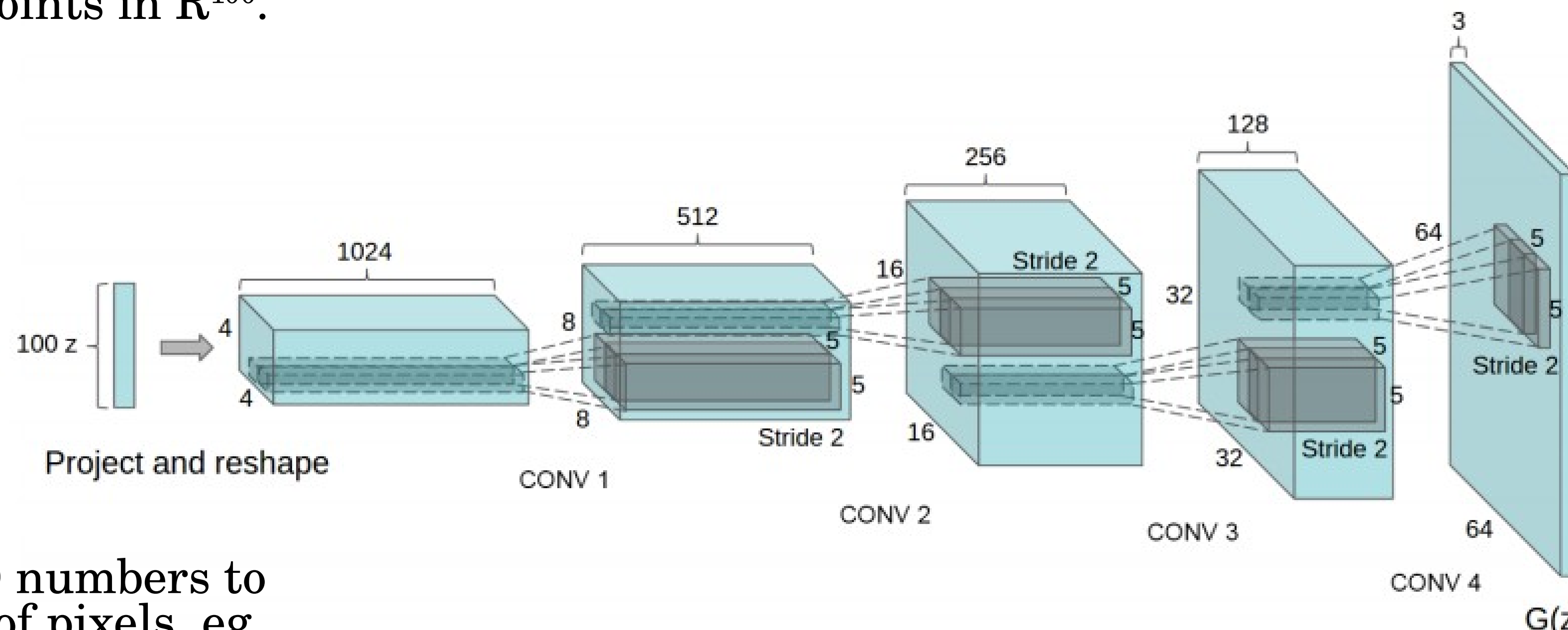
• final layers are responsible for decision making using the identified features

GAN: Generative Adversarial Networks

The task: code an RGB image as a point in R^{100}, then generate new images by drawing random points in R^{100}.

GAN: Generative Adversarial Networks

The task: code an RGB image as a point in R^{100}, then generate new images by drawing random points in R^{100}.



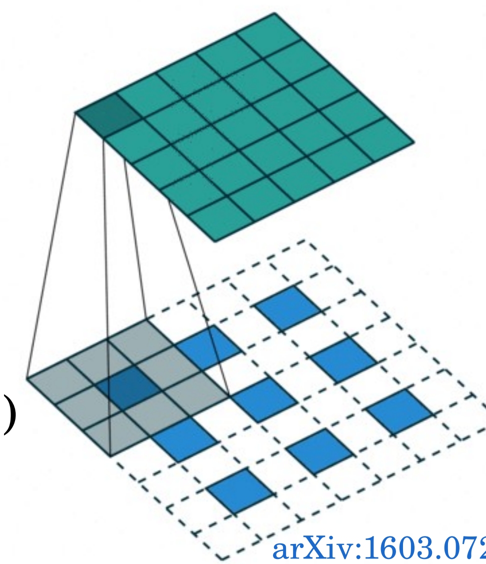
arXiv:1511.06434

Step 1: upscale 100 numbers to necessary number of pixels, eg. 64x64x3 = 12228 using a series of transposed convolutions. Each pixel has discrete values in 0-255 range.

output (6x6)

Transposed convolution:
resolution upscaling

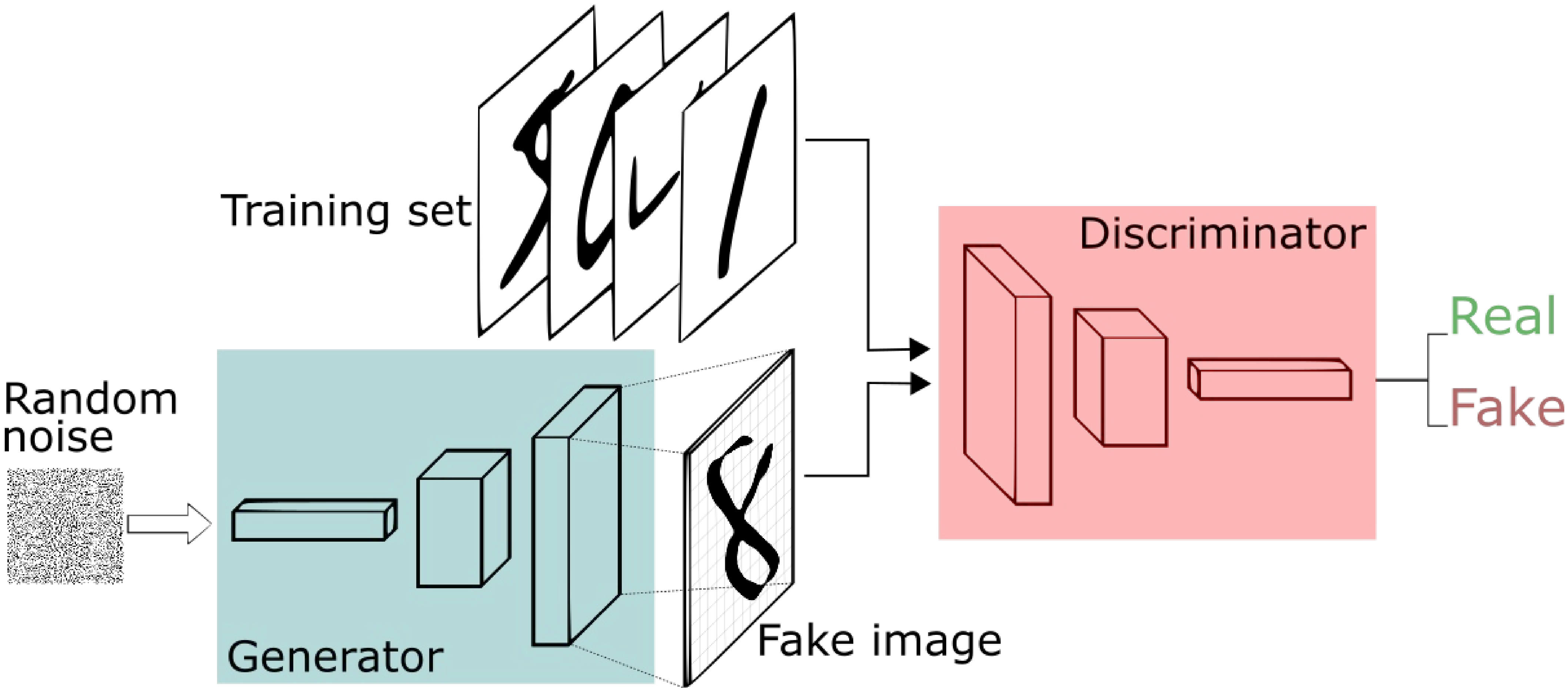
input (3x3)



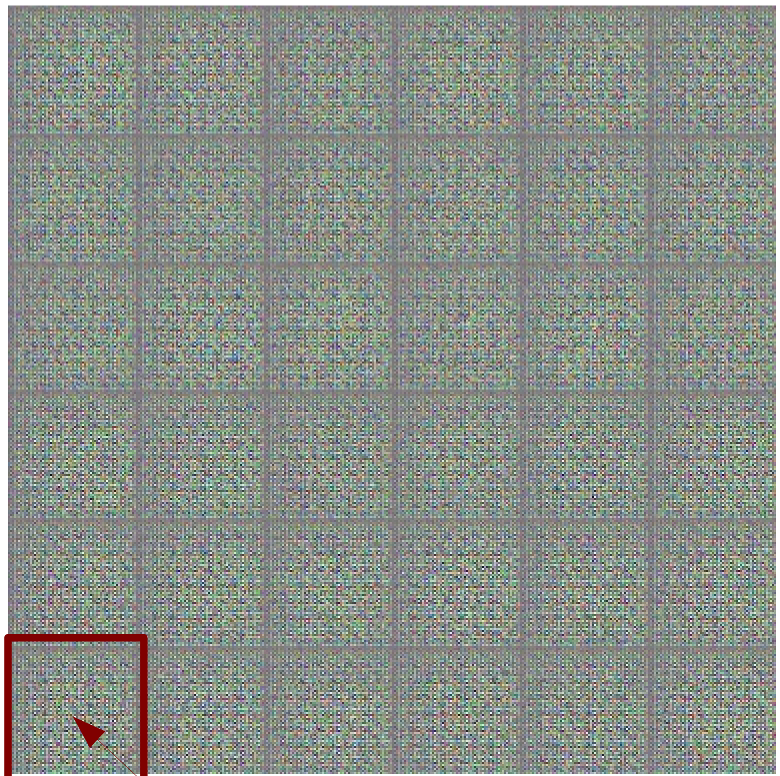
arXiv:1603.07285

GAN: Generative Adversarial Networks

Step 2: find mapping (= convolutions weights) from R^{100} to a subspace of R^{12228}.
Use two adversarial networks:
G – generator making an image from random noise
D – discriminator deciding if an image is real or generated



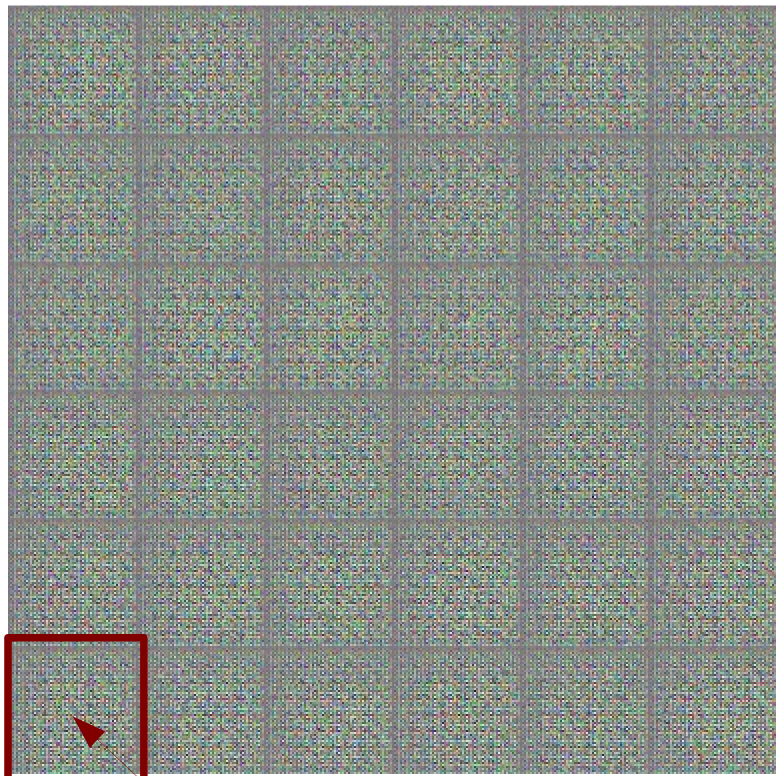
Starting point: random noise images generated by G



a single image

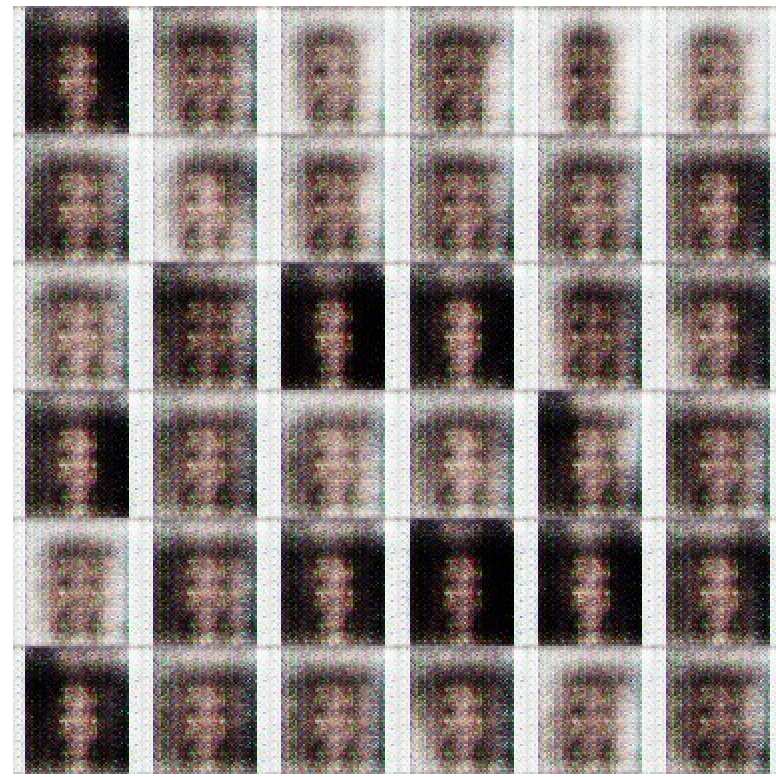
http://www.timzhangyuxuan.com/project_dcgan/

Starting point: random noise images generated by G



a single image

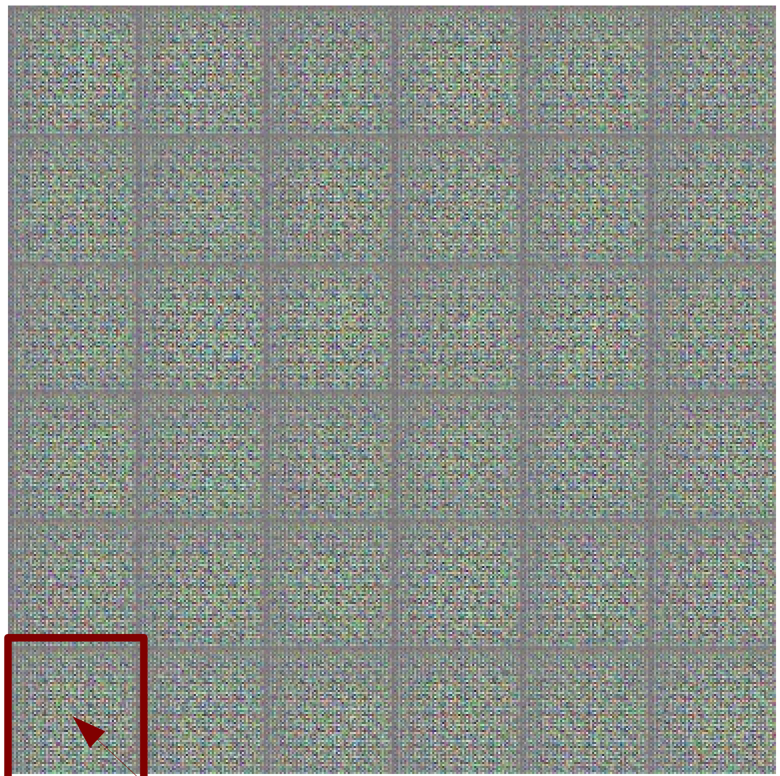
Epoch 150: 150 times transverse library of 200k real human face images.



http://www.timzhangyuxuan.com/project_dcgan/

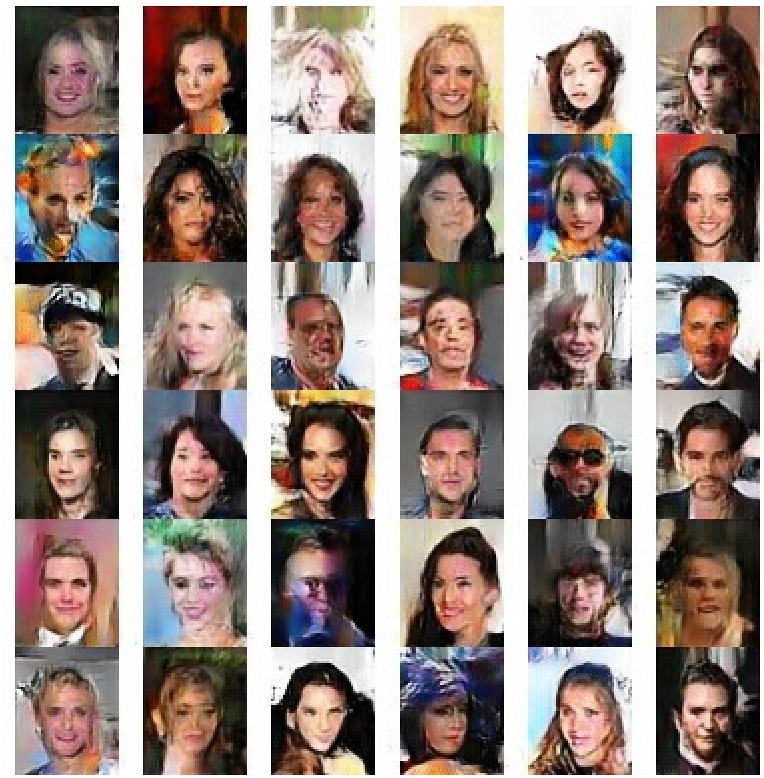
GAN: Generative Adversarial Networks

Starting point: random noise images generated by G



a single image

Epoch 16500: 16500 times transverse library of 200k real human face images.



http://www.timzhangyuxuan.com/project_dcgan/

GAN: Generative Adversarial Networks

Recent advance: progressive GAN – generate high resolution images by iterative resolution increase of generated image during the training process
Number of parameters: 23.1M in Generator and Discriminator networks respectively
Training time: 4 days on 8 Tesla V100 GPUs (single GPU cost: 50k PLN).

2015
64x64

2016
64x64

2017
128x128

2017
1024x1024

arXiv:1710.10196

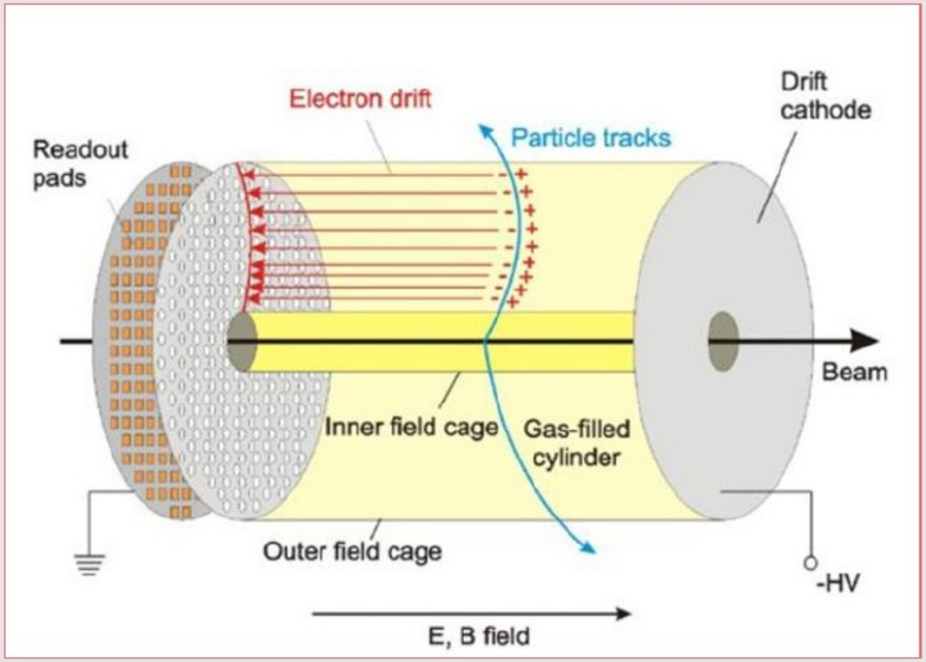
GAN in simulations

Example: simulation of particle passage through a detector: here ALICE TPC (work by group from the Warsaw University of Technology)

Particle clusters in TPC

ALICE

- Points in **3-dimensional space**, together with the energy loss, which were presumably generated by a particle crossing by.
- Input for particle tracks generation
- Up to **159 points per particle**
- Possible values **restricted** by the detector size ~ 5m x 5m x 5m
- **No clusters** in the inner field cage

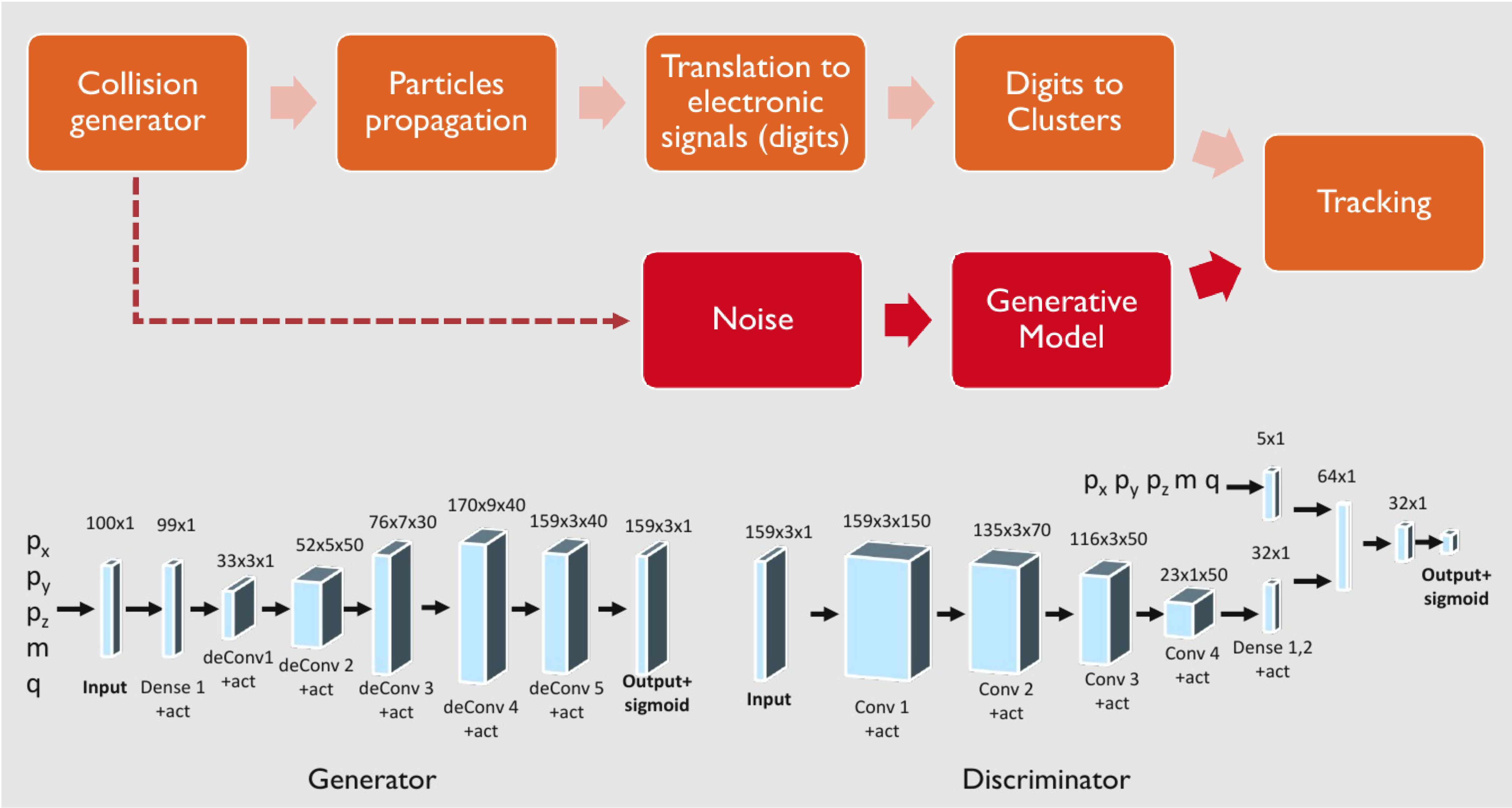


I.Konorov, Front-end electronics for Time Projection chamber

https://indico.cern.ch/event/587955/contributions/2937515/attachments/1683183/2707645/CHEP18.pdf

GAN in simulations

The idea: substitute time consuming full Geant 4 simulation by a GAN trained to generate "track images" = 100 + 4 dimensional paramatrisation of Geant4 output

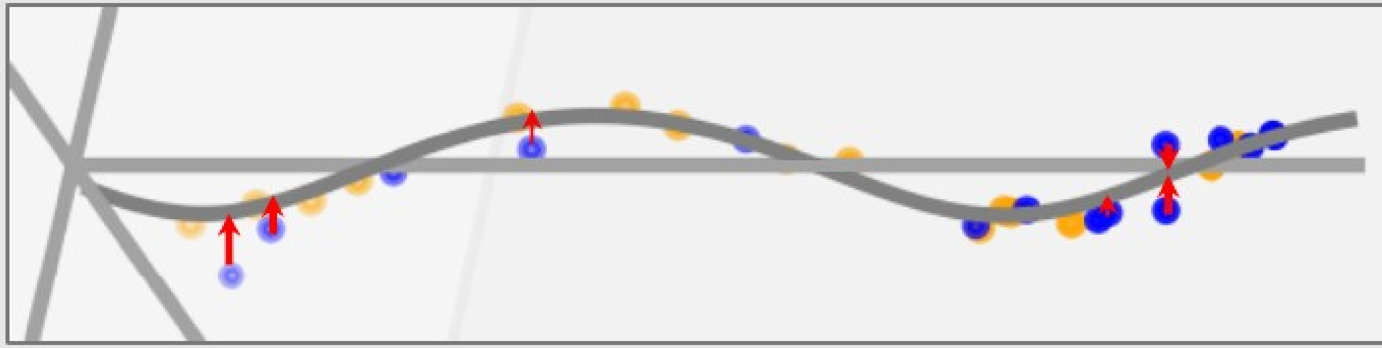


https://indico.cern.ch/event/587955/contributions/2937515/attachments/1683183/2707645/CHEP18.pdf

GAN in simulations

Quality criterion: mean square distance between generated hits and an ideal helix.

MSE visualisation:
Red - error
Grey- ideal helix
Orange – original clusters
Blue – generated clusters

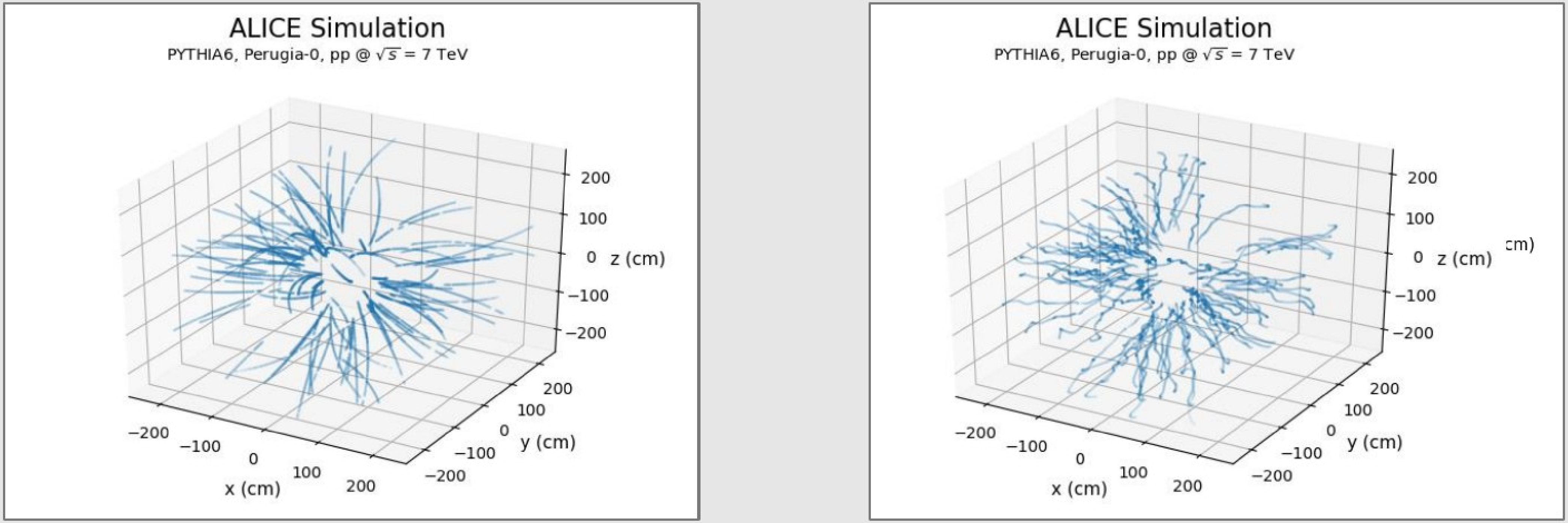


Speed increase: factor 25 for running GAN on CPU. Expected factor 250 for running on GPU

Method	Mean MSE (mm)	Median MSE (mm)	Speed-up
GEANT3	1.20	1.12	1
Random (estimated)	2500	2500	N/A
condLSTM GAN	2093.69	2070.32	100
condLSTM GAN+	221.78	190.17	
condDCGAN	795.08	738.71	**25**
condDCGAN+	**136.84**	**82.72**	

https://indico.cern.ch/event/587955/contributions/2937515/attachments/1683183/2707645/CHEP18.pdf

GAN in simulations



Original event

Generated event

https://indico.cern.ch/event/587955/contributions/2937515/attachments/1683183/2707645/CHEP18.pdf