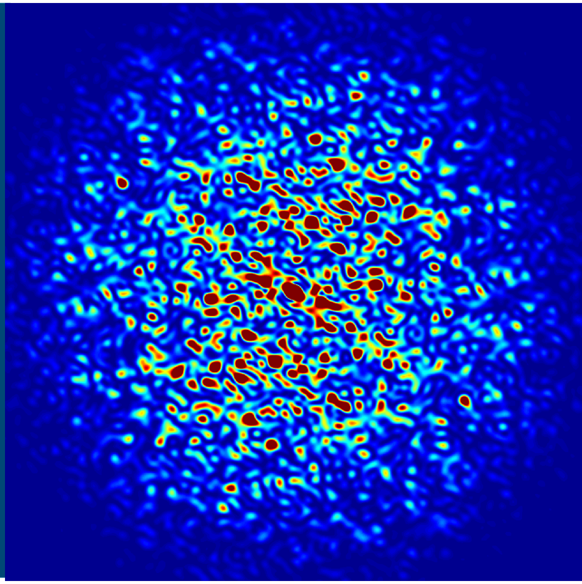
An introduction to artificial neural networks



Yipeng Sun
LLRF 2019 workshop, Chicago
Oct 3, 2019

Thanks to: Google Colab computing; Coursera.org for online courses.
Work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-06CH11357.

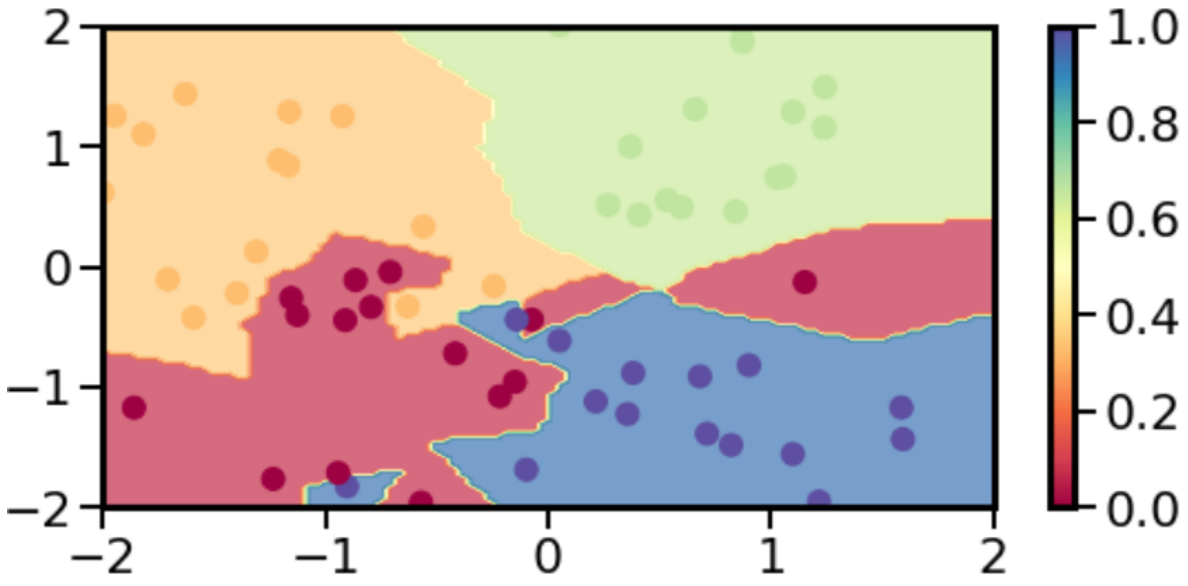
Outline

- Overview on artificial intelligence and Deep learning
- Artificial neural networks code developments
- Applications of this code:
  - Supervised and unsupervised learning
  - Classifier
  - Regression
- Conclusion

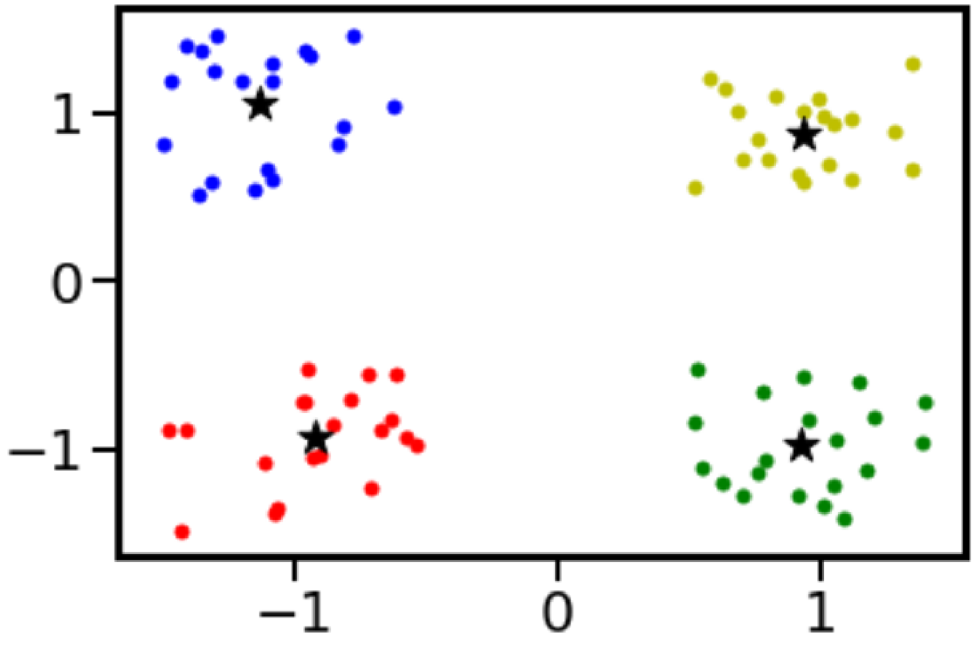
Machine learning examples

NO explicit instructions; NO need math/physics model; needs data

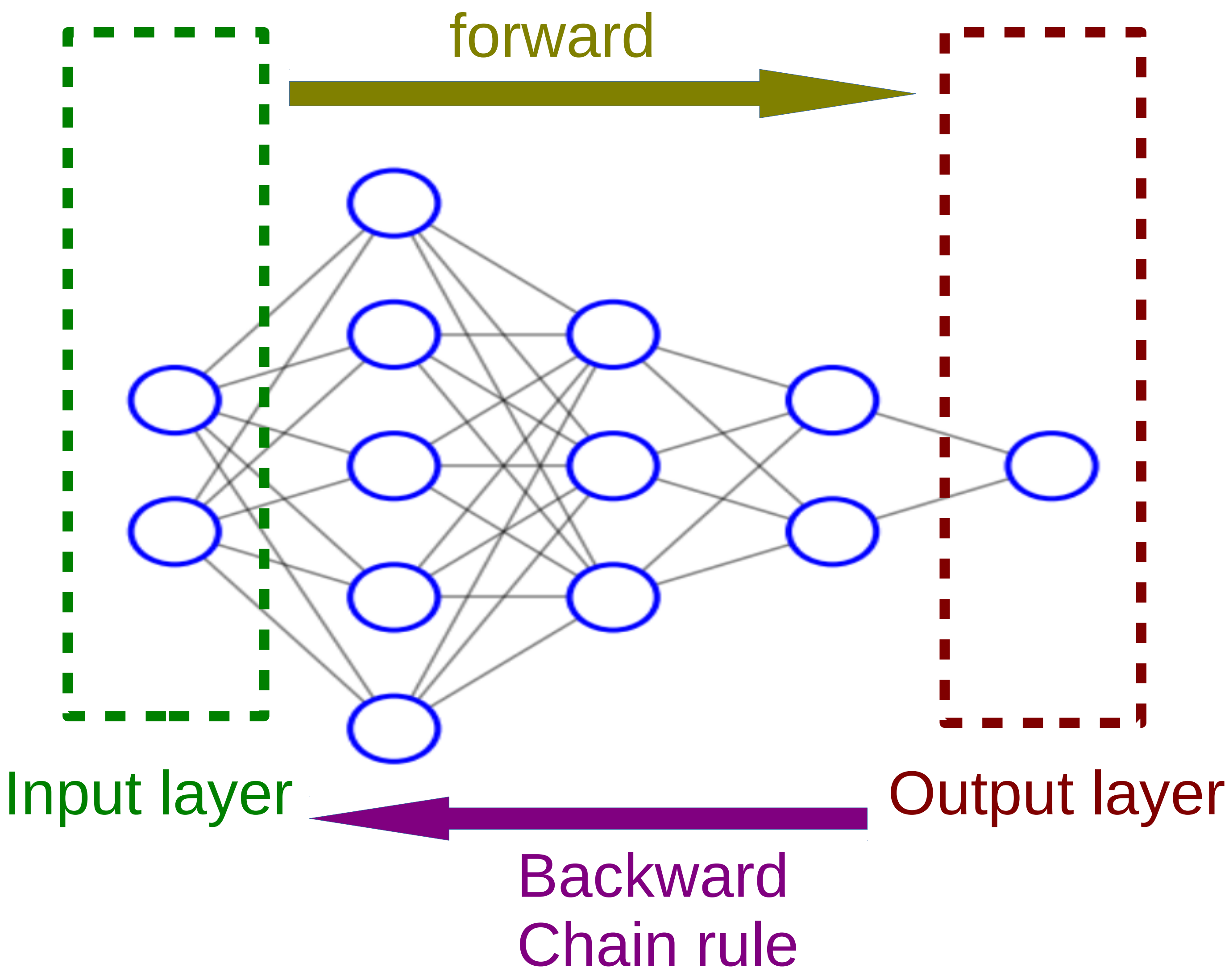
kNN: k nearest neighbors
k=1

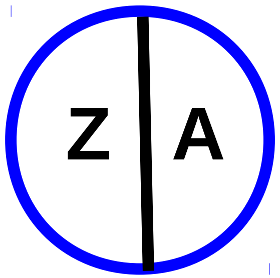


k-means clustering (unsupervised)
k=4



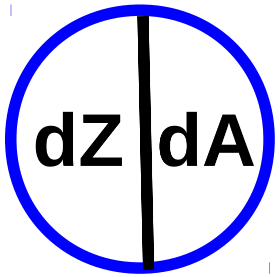
Neural network: forward and backward propagation





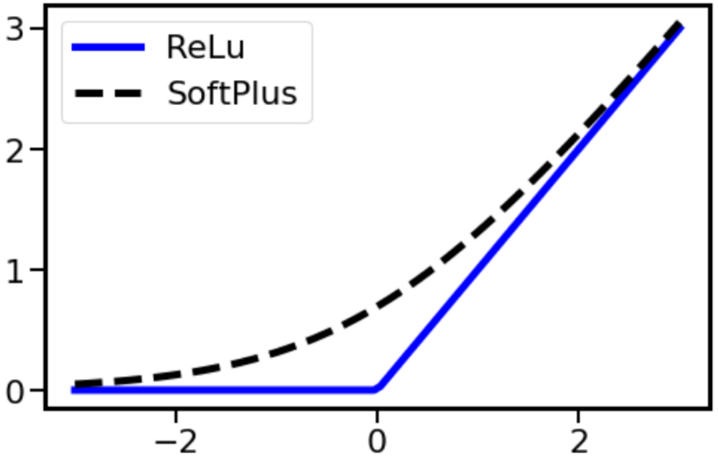
Z(n) = W*A(n-1) + b
A(n) = F(Z(n))

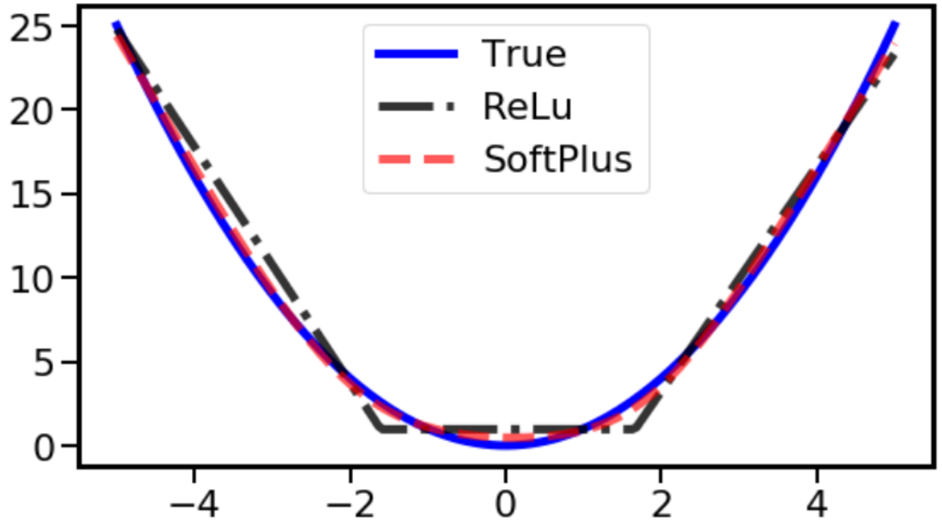
Cost function
(MSE, MAE, CE)



dA(n) = W*dZ(n+1)
dZ(n) = dA(n)*F'(Z(n))
dW, db from dZ(n) & A(n-1)

Neural network as arbitrary function approximator



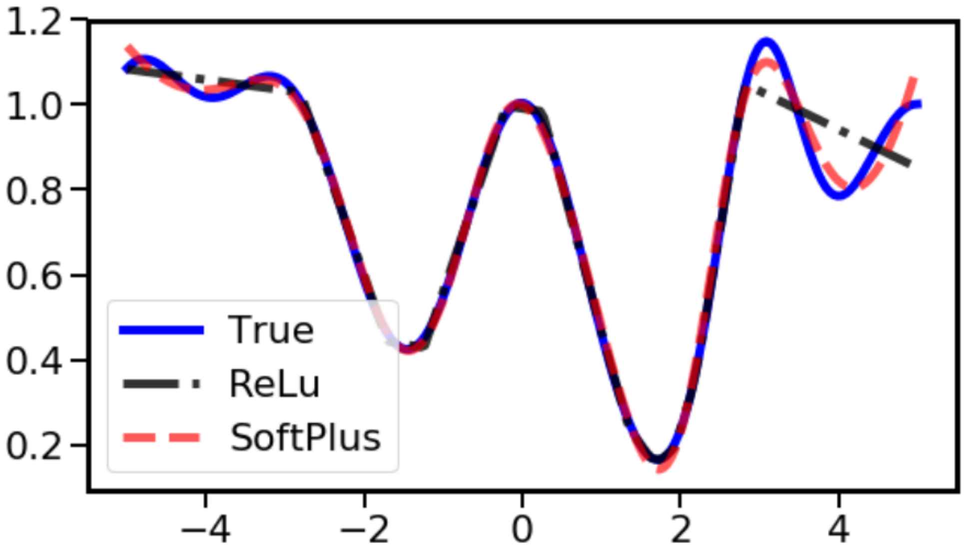


2 ReLu: x^2 (top); 50 ReLu: Radial-basis functions (bottom)

Activations

Rectified linear unit
ReLu: A = max(0,X)
SoftPlus: A = log(1+exp(X))

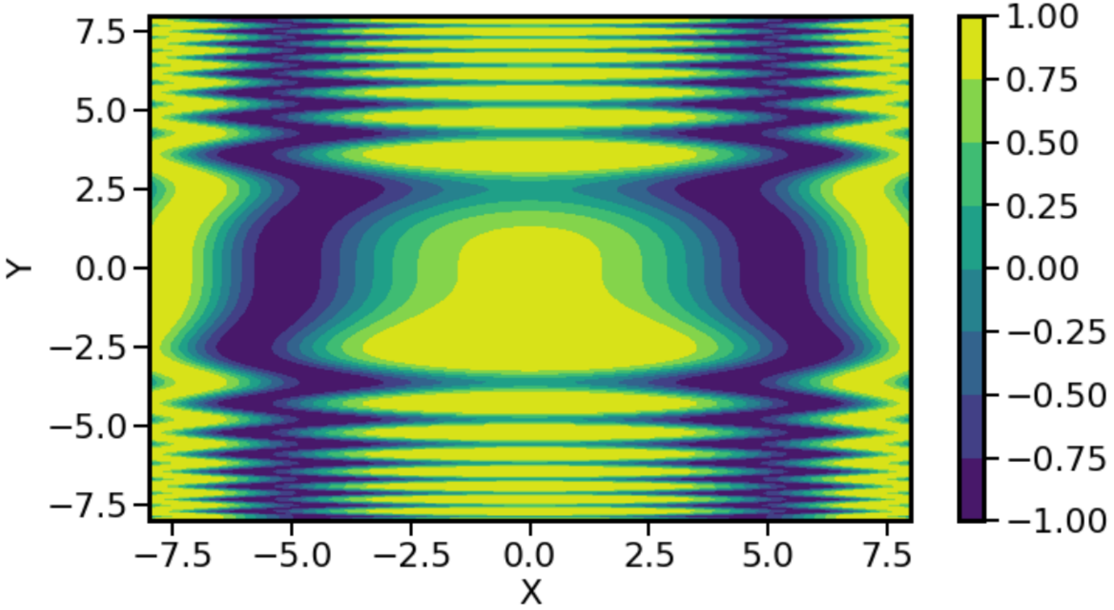
Only one layer

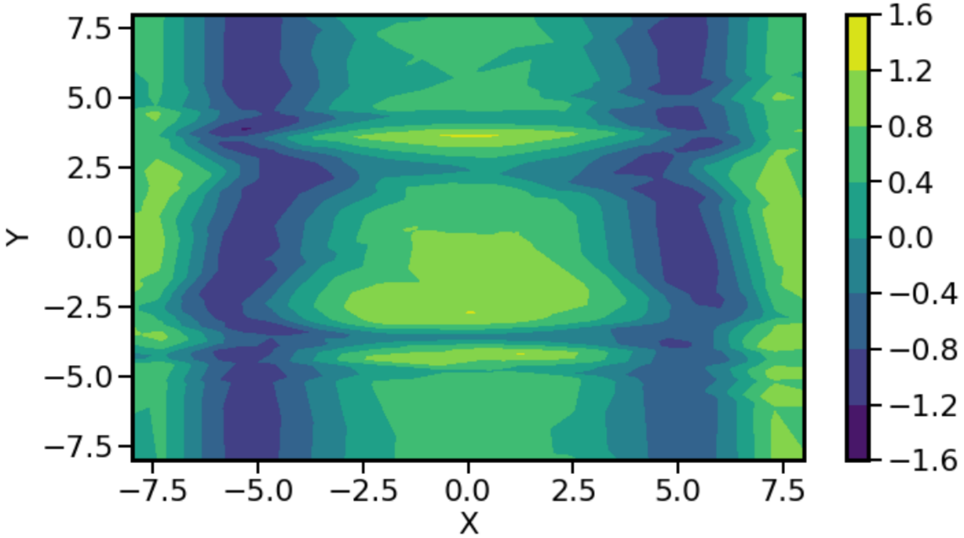


Argonne NATIONAL LABORATORY

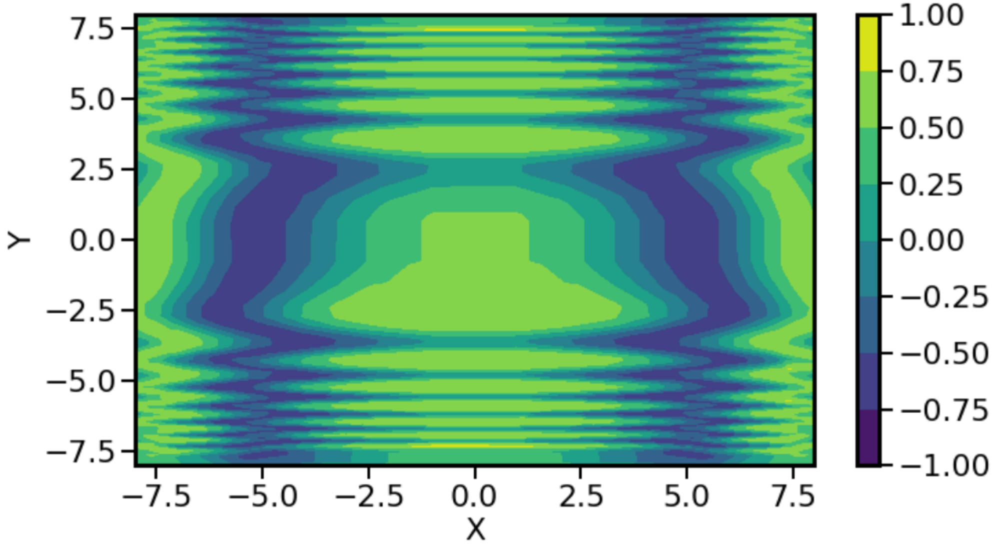
Multiple layers: function composition g(f(x))

Z = **cos**(X^2*0.1 +**sin**(Y^3*0.1)+0.5)





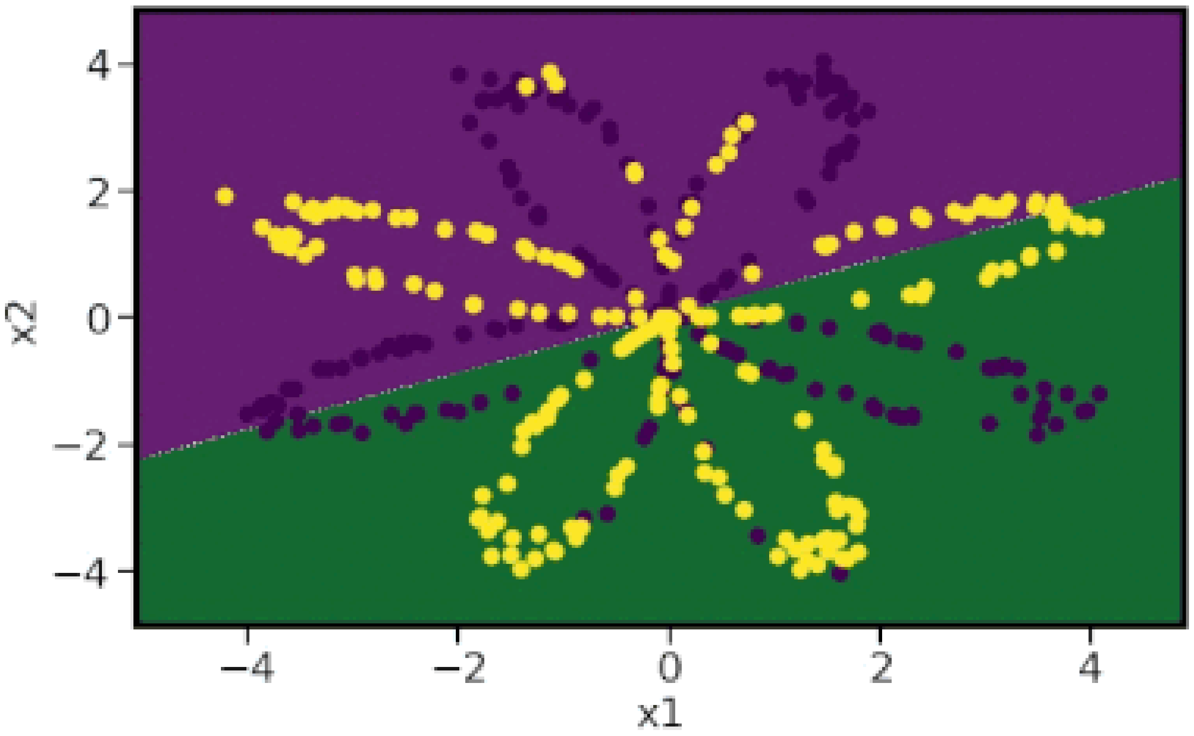
One layer (top); two layers (bottom)

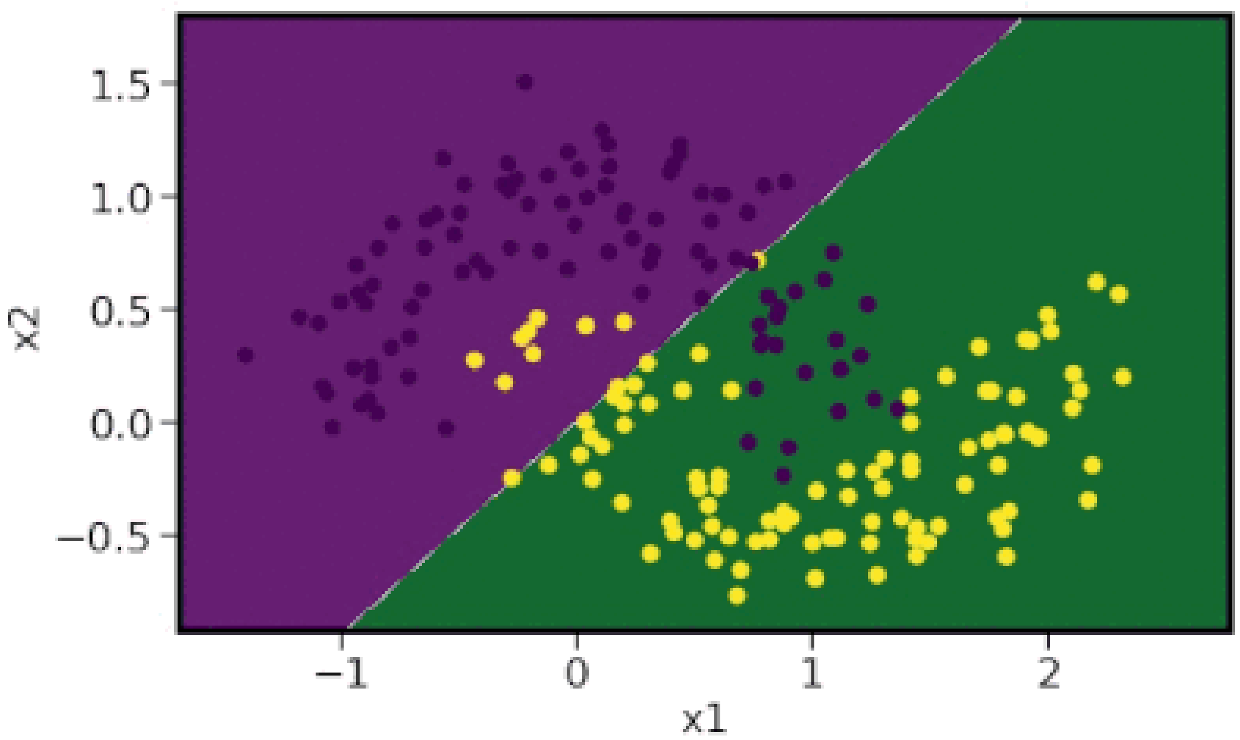


Python codes in development for neural network

- Motivation: to understand the algorithms and apply to our work
- Why Python
  - ‘default’ choice of deep learning: computation speed, user community, existing algorithms (TensorFlow, Keras, Caffe, PyTorch...)
- What has been done:
  - Initialization (0, random, random normalised)
  - Forward and backward propagation (activations, gradients)
  - Cost functions (MSE, MAE, CE)
  - Optimization algorithms (SGD, Rprop, momentum, ImpliMom, AdaGrad, RMSprop, **adam**)
  - Variational auto encoder, generative adversarial networks
- To be developed:
  - More activation and cost functions
  - regularization algorithms (dropout...), CNN, RNN...

Application: rose function etc.



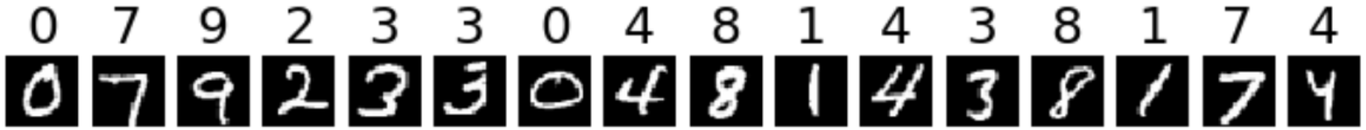


Training data: red-blue dots from rose function
Prediction: contour plot
Animation on iterations

Training data: sklearn.datasets.make_moons
Prediction: contour plot
Animation on iterations

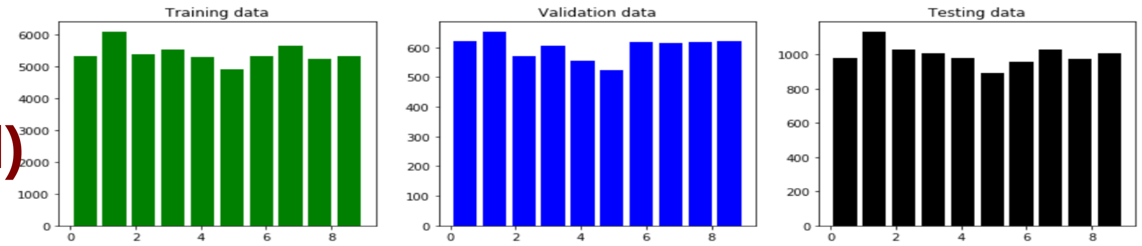
Higher than 90% ‘accuracy’

Application: Hand writing, 10-class [1]

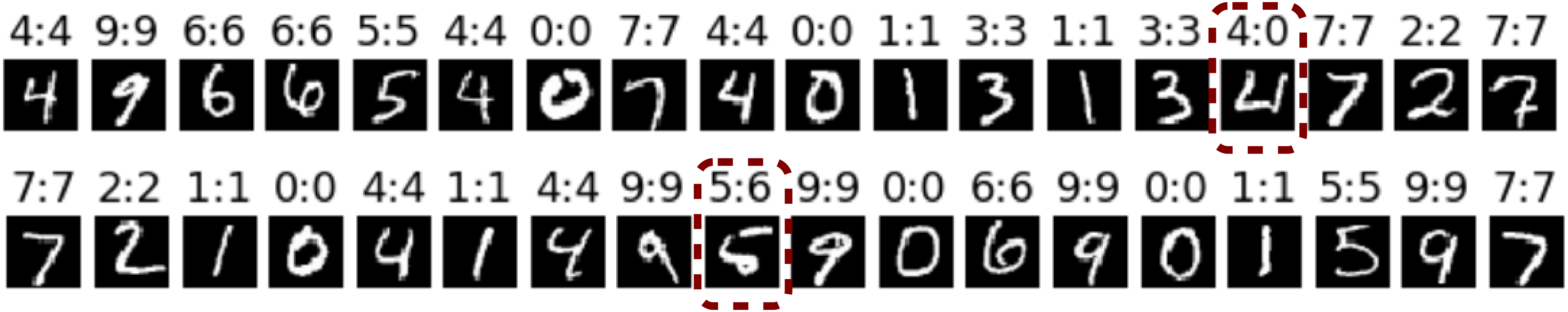


[1] MNIST database, http://yann.lecun.com/exdb/mnist/

Training/val data: 60k images (28 by 28 pixels)
Test data: 10k images
Accuracy: 99% (training data), ~98% (test/val)

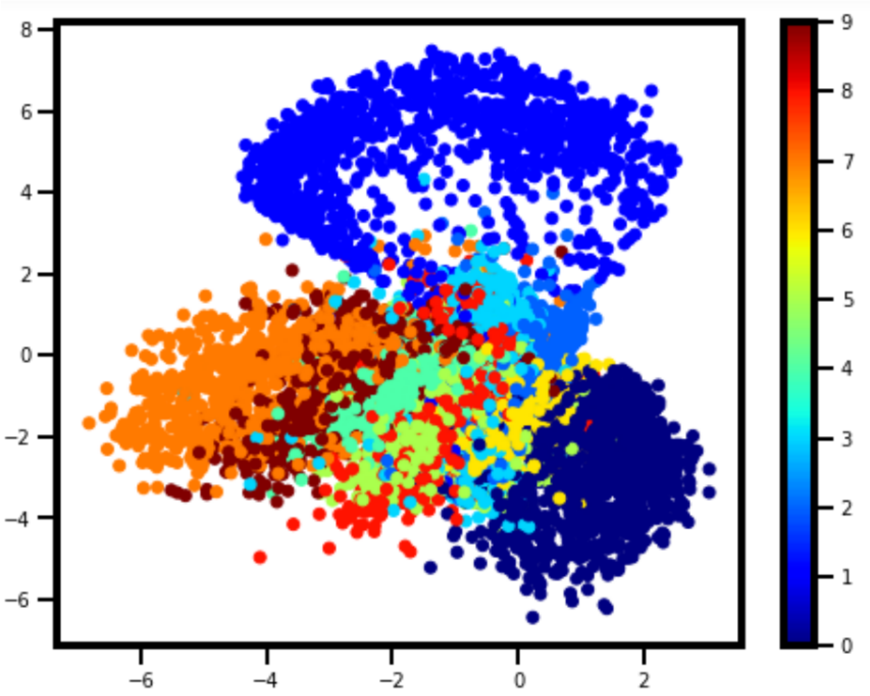


Predictions (selected blocks with wrong labels)

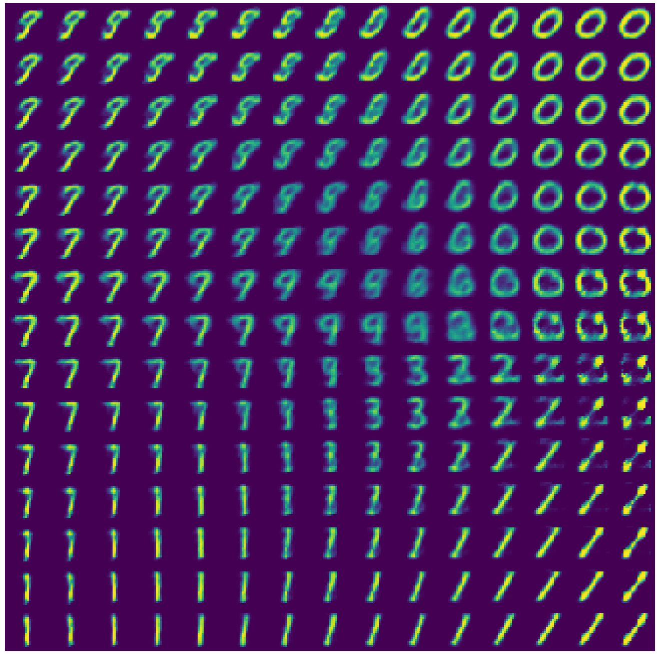


Generative models: variational auto encoder

2D latent (hidden) space, Z



2D latent space, Y^

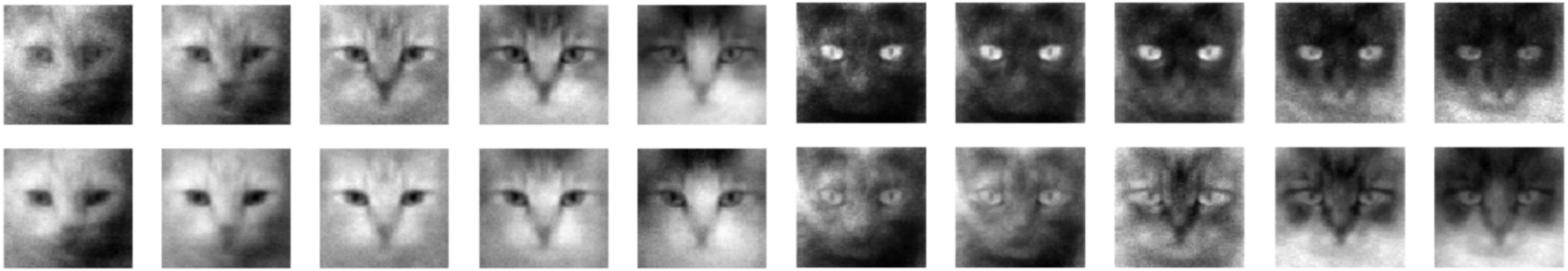


Training with MNIST database, http://yann.lecun.com/exdb/mnist/

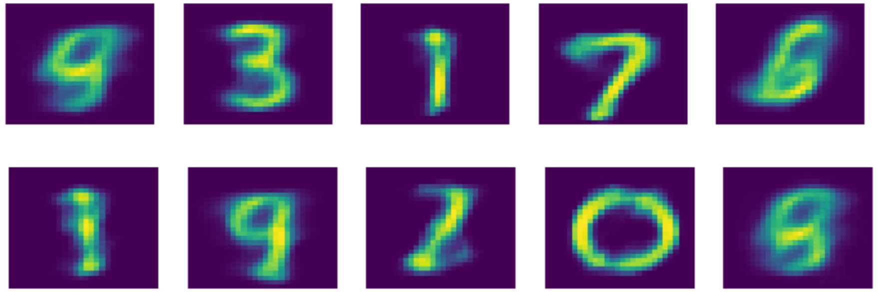
Generative models: variational auto encoder

Training data

Generated synthetic cat images



Generated numbers

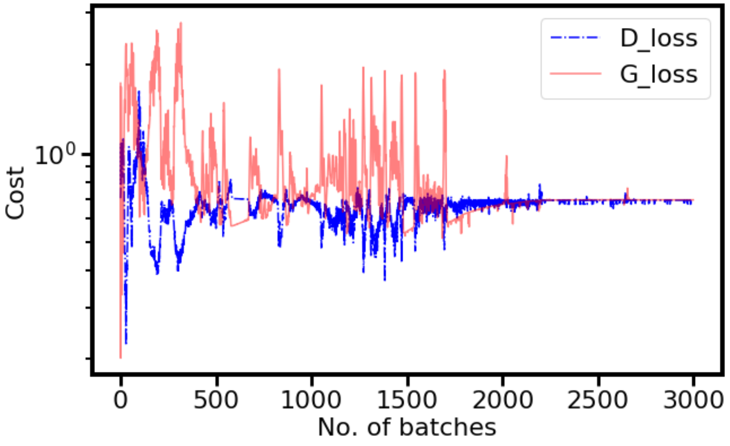


Generated cats animation

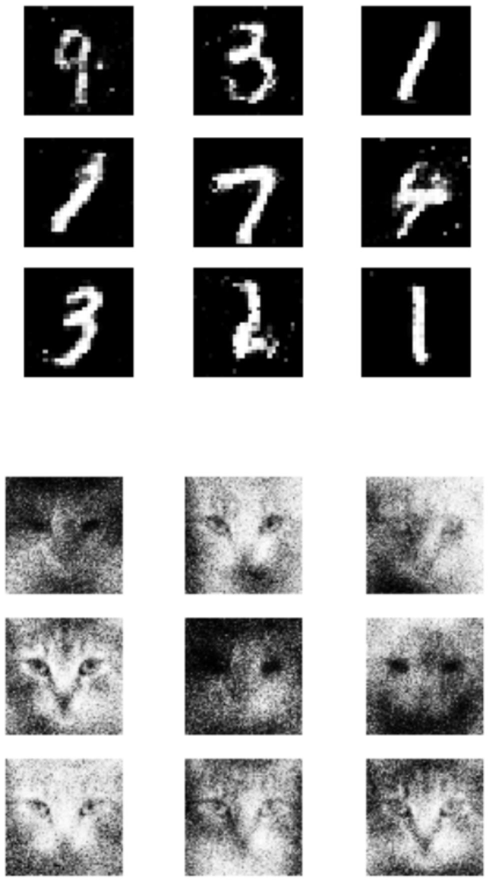
* Cat data, https://ajolicoeur.wordpress.com/cats/

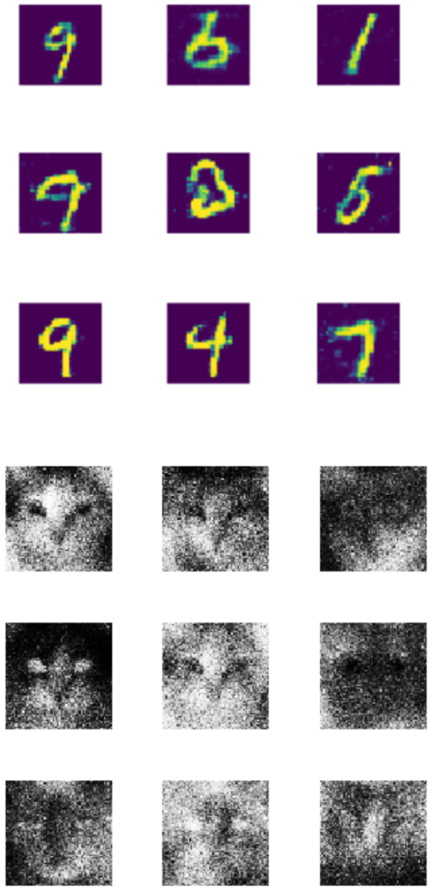
GAN: Generative Adversarial Networks

Training: discriminator and generator



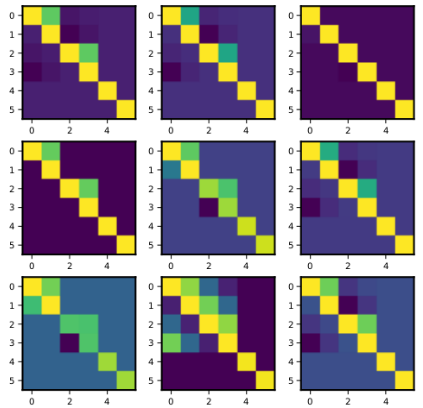
GAN Generated synthetic images
This code (left); Keras/TF (right)

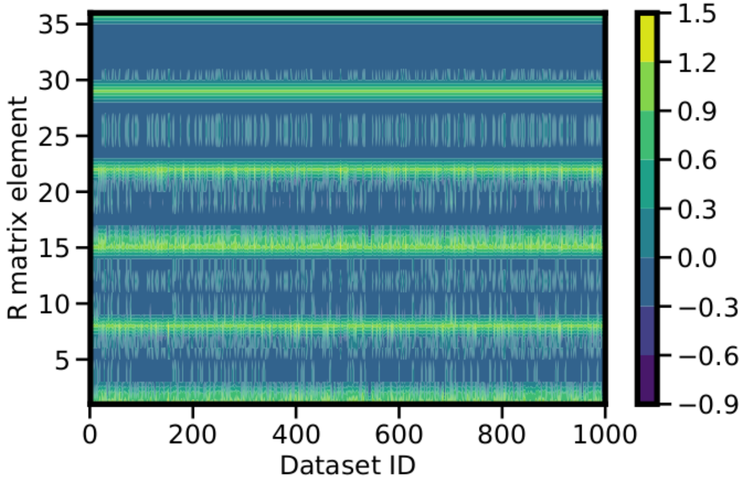


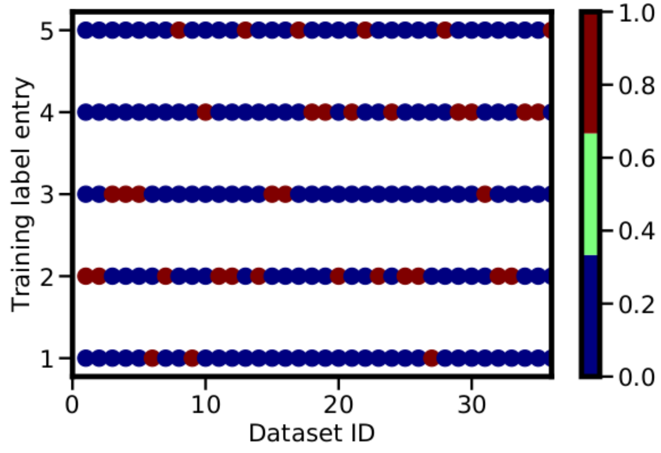


Application: R matrix for accelerator elements*

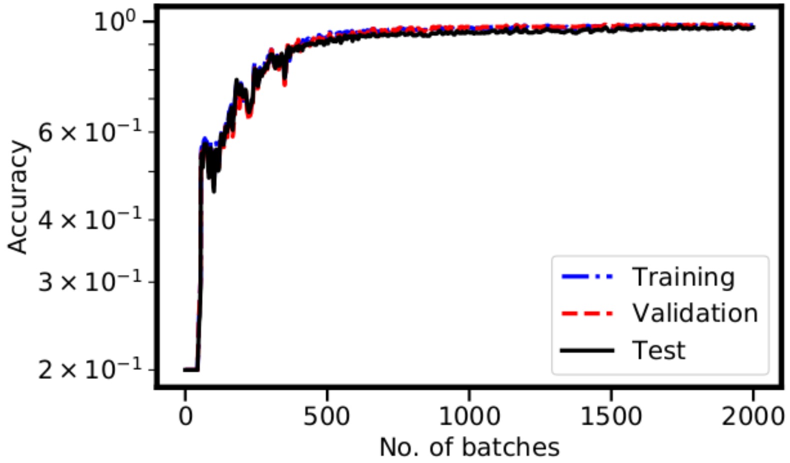
5-class: sbend, quad, drift, skew quad, bendEdge

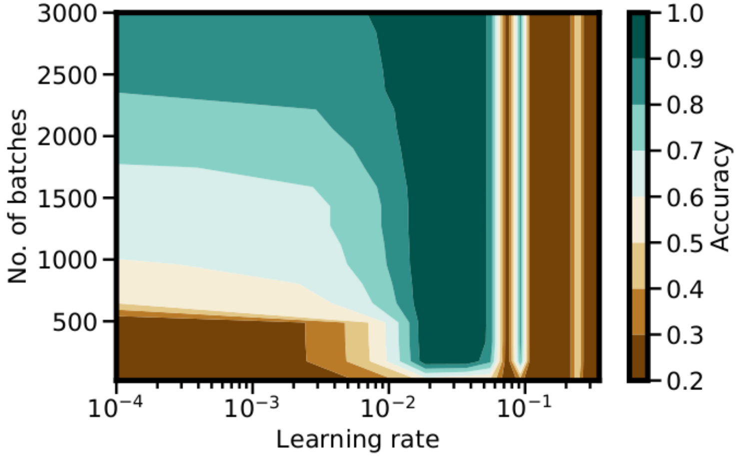


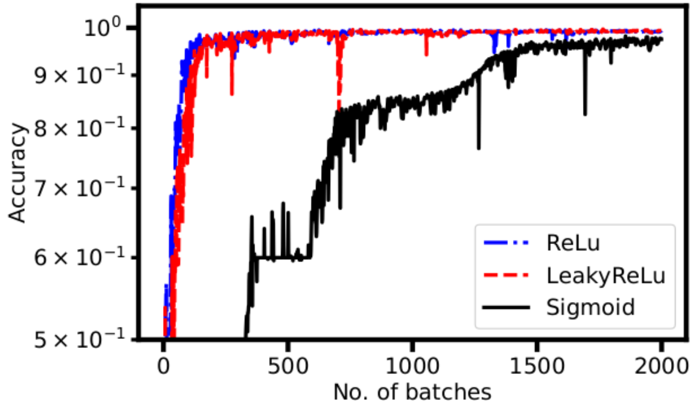




(Accuracy: Training 99.8% validation 99.6% test 99.1%)

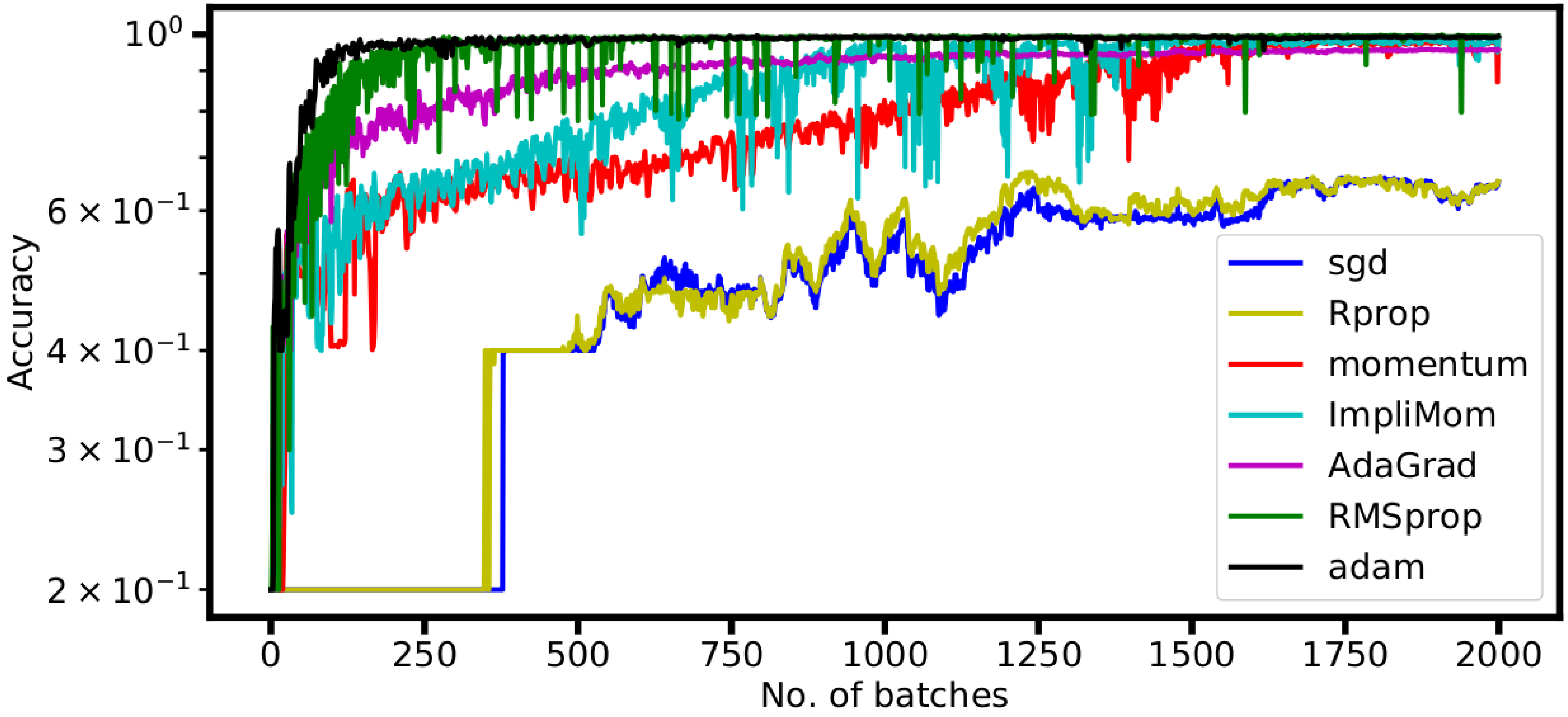






*Yipeng Sun, NAPAC19

Benchmark on optimization algorithms*

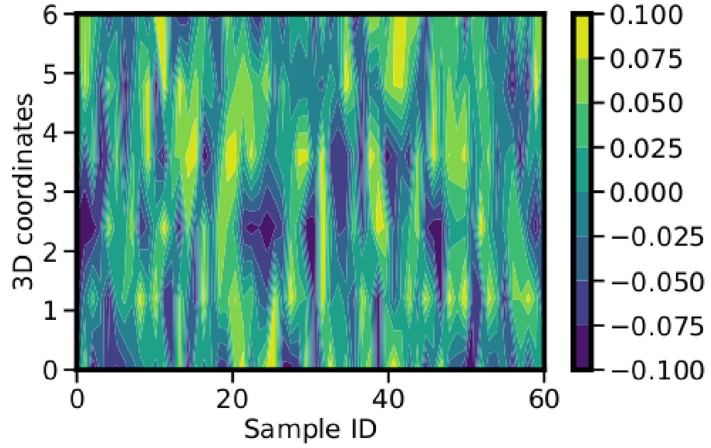


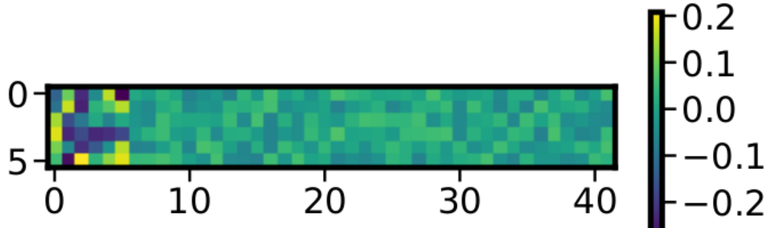
'adam' seems to be most robust
example with R matrix classifications, showing average accuracy

*Yipeng Sun, NAPAC19

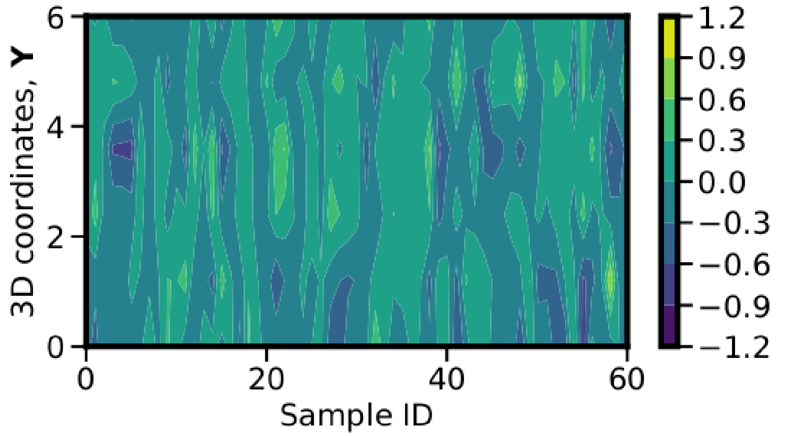
Application: 2nd order map*

Training data input

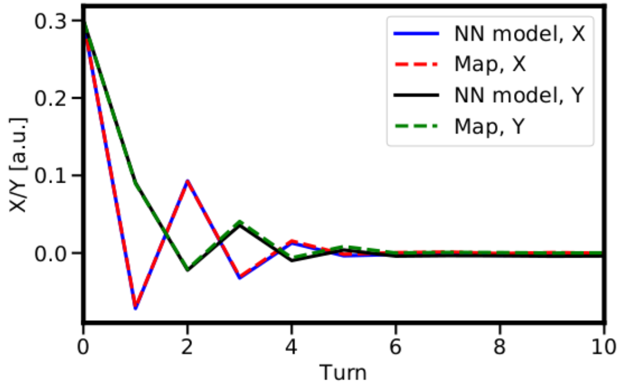




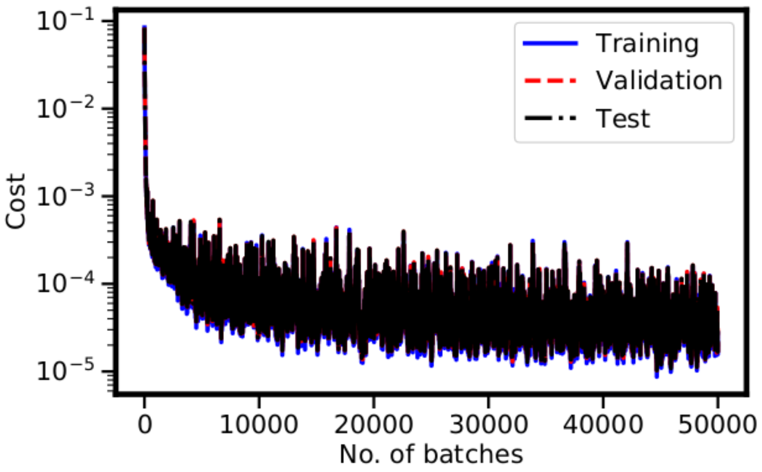
Training data output



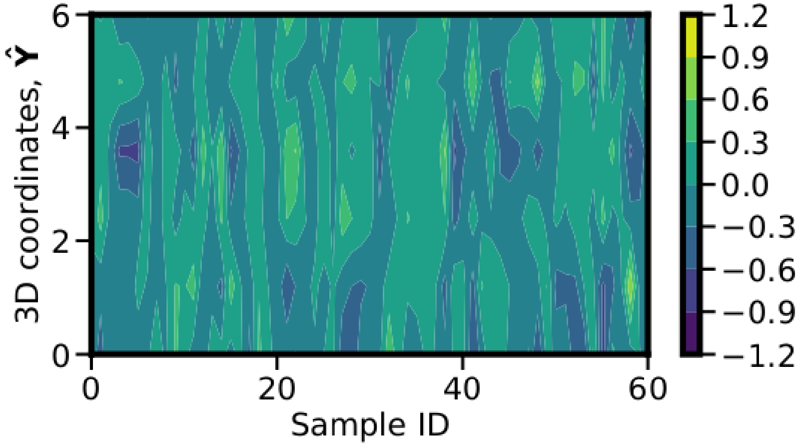
ANN particle tracking



ANN training



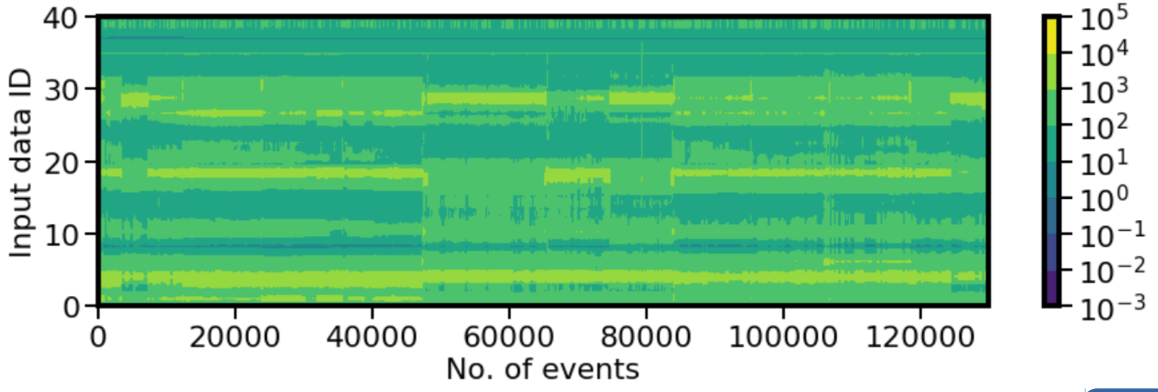
ANN predictions

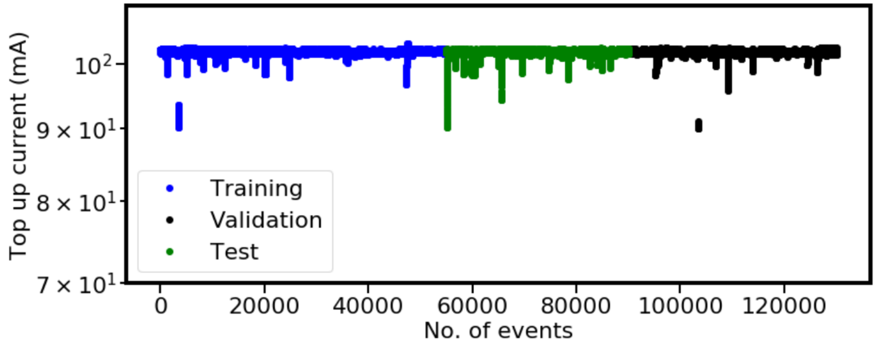


*Yipeng Sun, NAPAC19

Top up injection efficiency v.s. loss monitors etc.

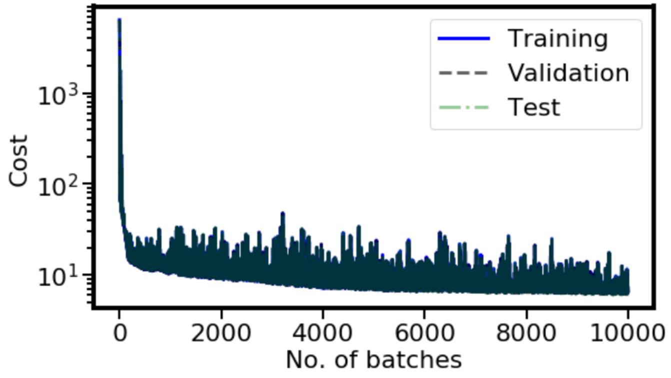
Input: APS operation data: 01-22-2019 to 06-28-2019

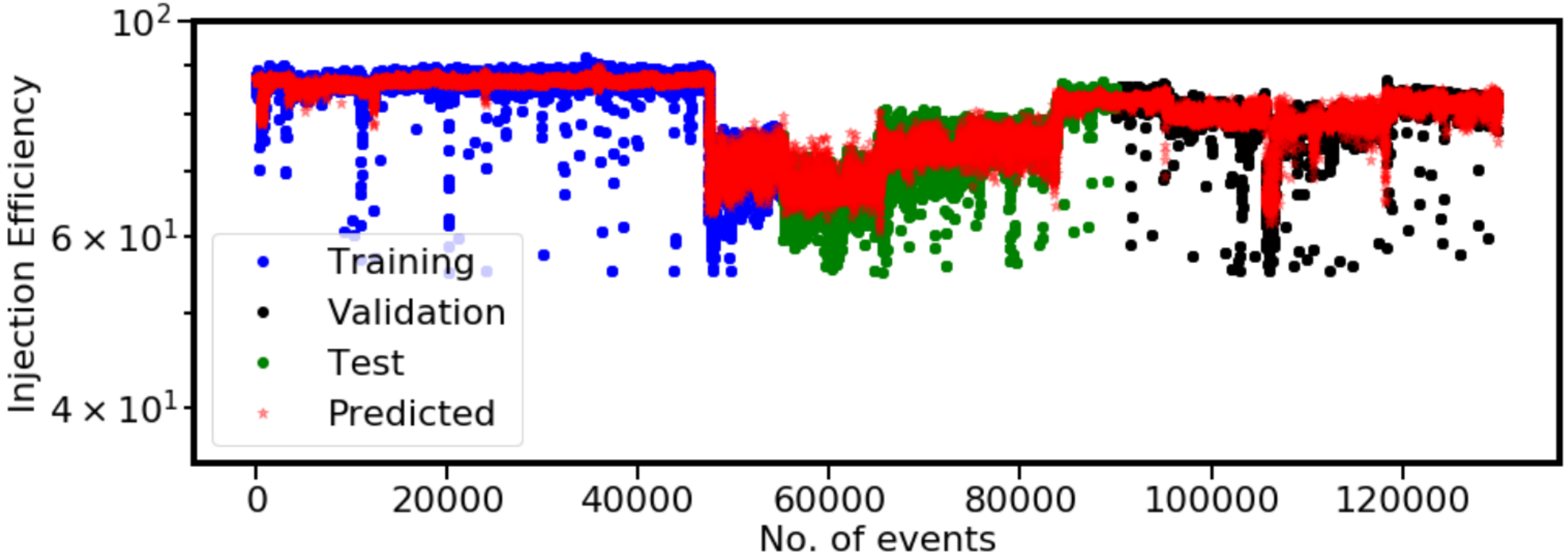




Predictions on injection efficiency (output)

Training / Cost 2%



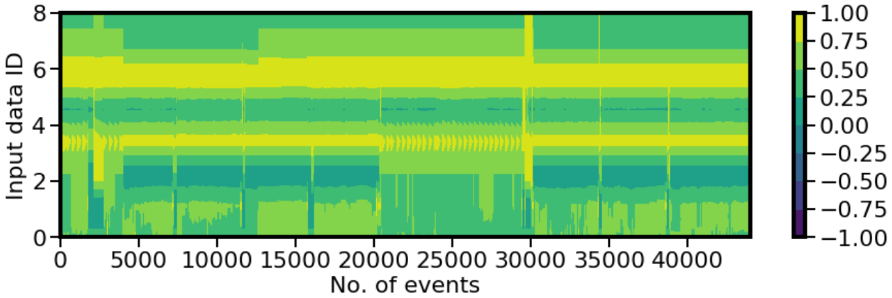


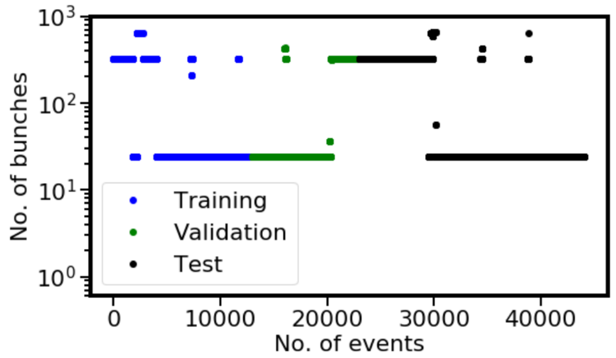
Lifetime

Depends on:
- Beam current
- No. of bunches
- Coupling
- Chromaticity
- Bunch length
  - RF voltage
  - Mom. compaction
- ID energy loss
- Physical apertures
  - ID4 orbit

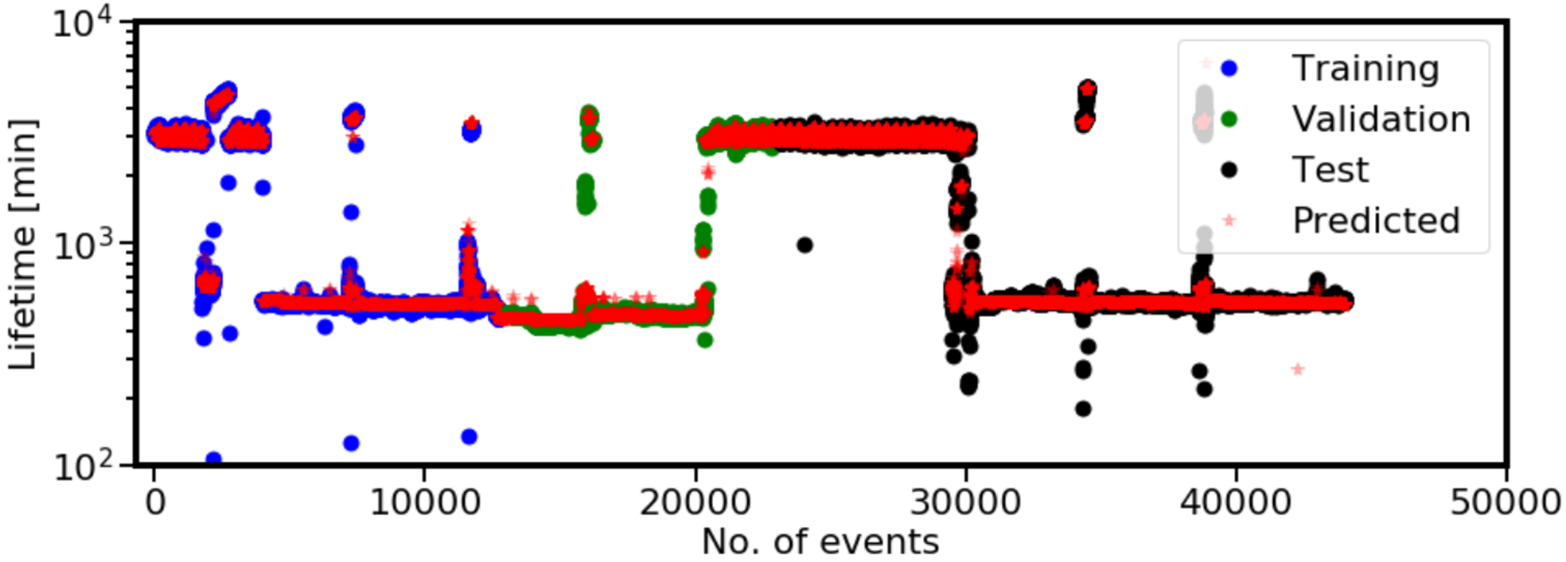
Results in:
- Beam loss monitors

Training data: APS operation 01-2019 to 06-2019



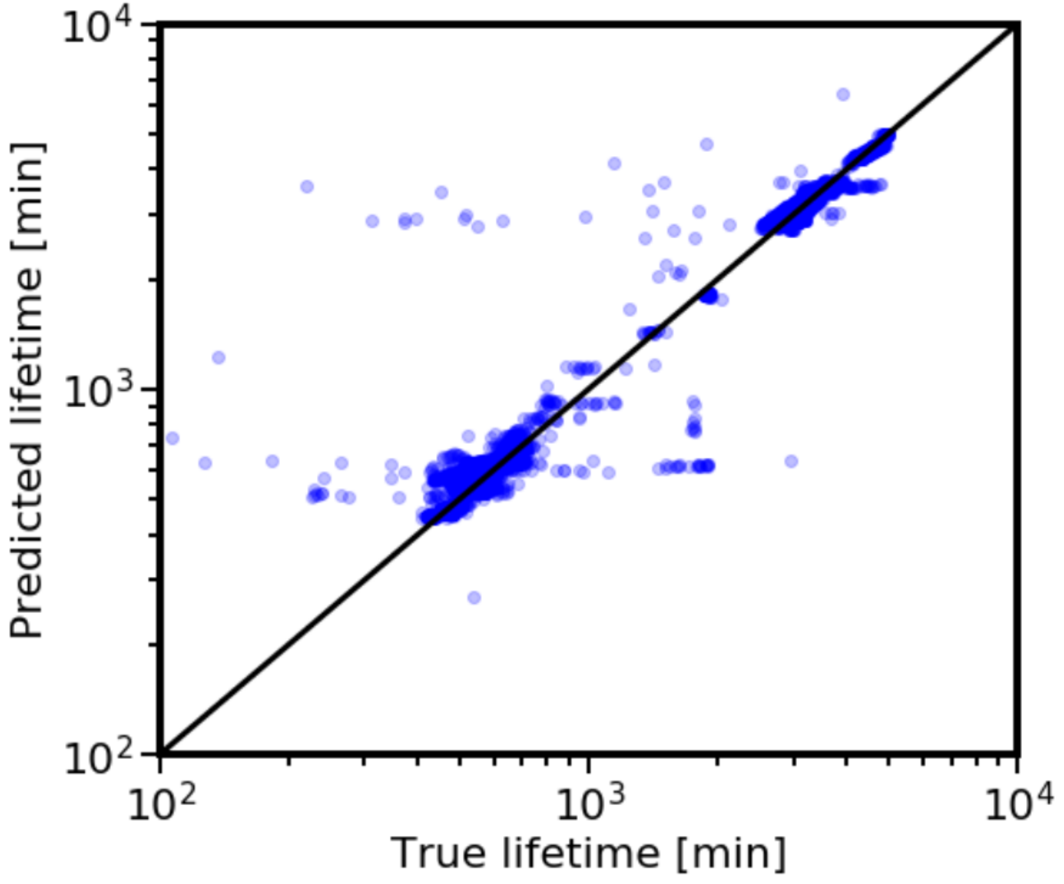


Predictions on lifetime (bottom). Training data (top)

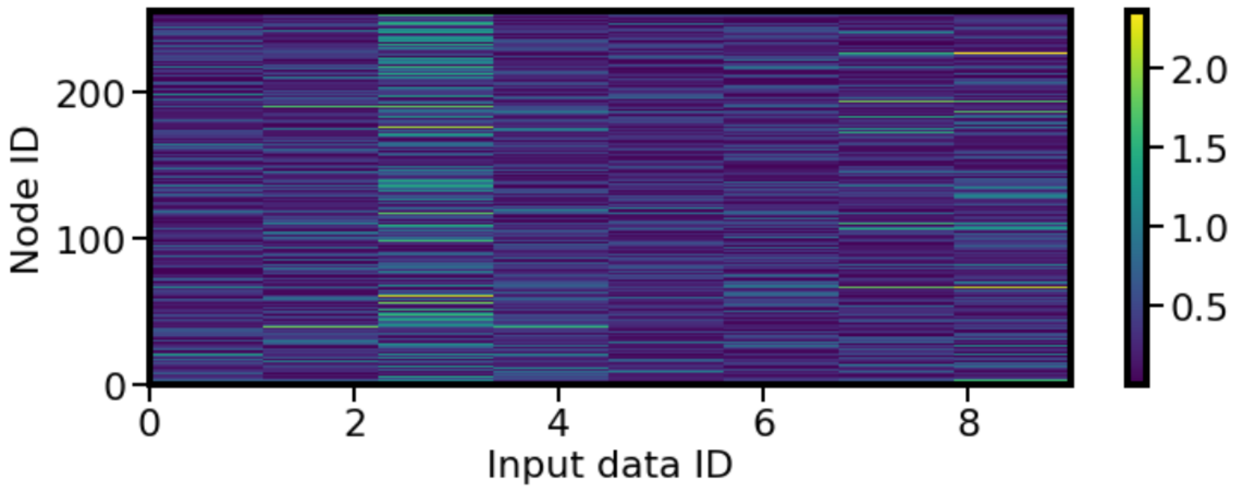


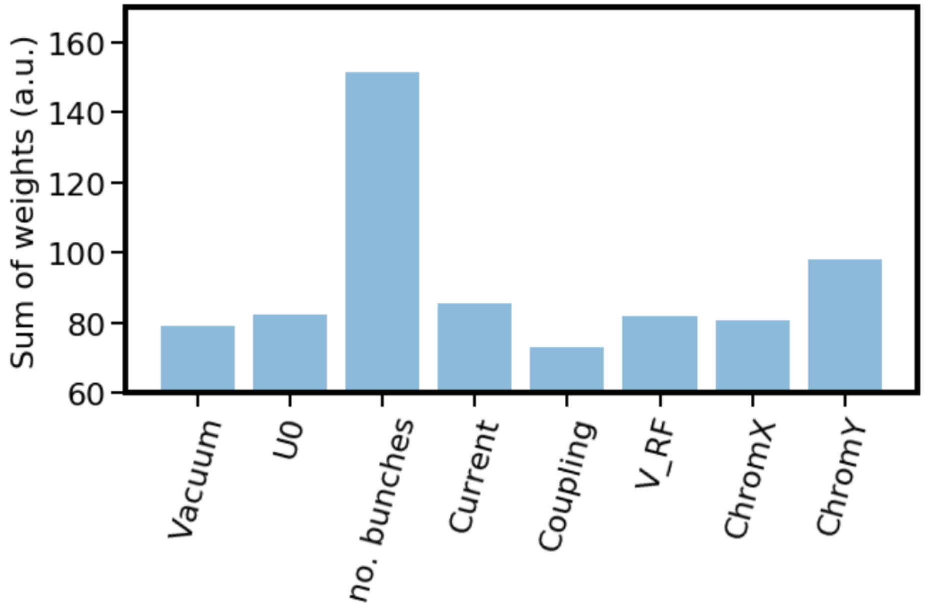
Lifetime

Predictions on lifetime



Analyzing trained NN model





Copyright