Applications of optimizer and ML algorithms at CERN by the *Beam Transfer*, *Machine Operators*, *Accelerator Physicists* and *IT Compute & Monitoring* groups

P. Van Trappen, V. Kain, S. Hirlander, B. Goddard, U. Schwickerath, et al.

ICALEPCS 2019: Data Science and Machine Learning Workshop
6 October 2019, Marriott NY, USA
https://indico.cern.ch/event/828418/

Problem statement

- Nowadays strong focus on reliability & availability of the machine (read ‘more beam-time with less money’)
- Increasingly more data generated (and stored) by all systems (experiments, accelerator equipment, beam operation, IT)
- This data can be used to model the system, on which these are used to provide better and faster solutions to all kinds of problems (classification, regression, feedback control, anomaly detection)
- Goal of the CERN ‘ML Coffee’ discussions is to collaborate on applying existing ML algorithms to our everyday problems; also identify future projects and share experience
- Indico site with presentations at https://indico.cern.ch/category/11178/

Discussions

<u>General theory & discussions:</u>

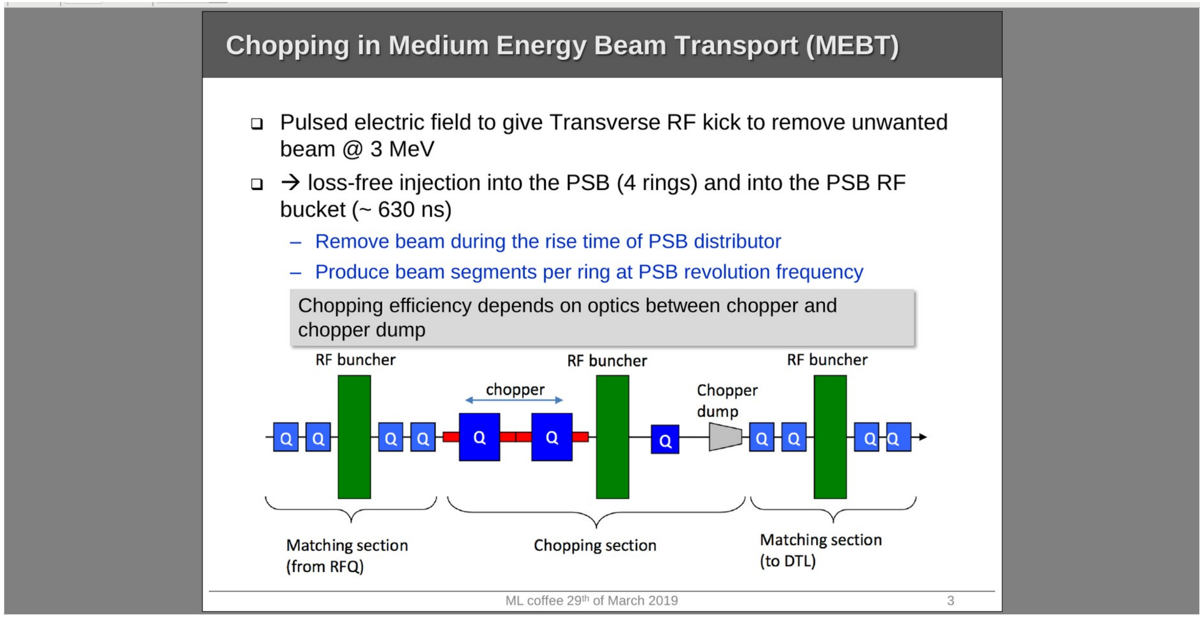
1. Neural Networks (NNs)
   1. Theory, existing libraries (TensorFlow, Keras)
   2. Sextupole surrogate model (design choices ref. trial and error)
2. Numerical optimization: Derative Free Optimization (DFO), Black Box Optimizatin (BBO), Powell’s method, COBYLA, BOBYQA
3. Reinforcement learning
   1. Theory (states, actions, reward, policies)
   2. Deep Q learning (DQN), Bellman, DDQN, Normalized Advantage Functions (NAF)
   3. Policy based, continues action space -> deterministic policy gradient (DPG)
4. Representational Learning with Variational AutoEncoder (VAE), disentanglement

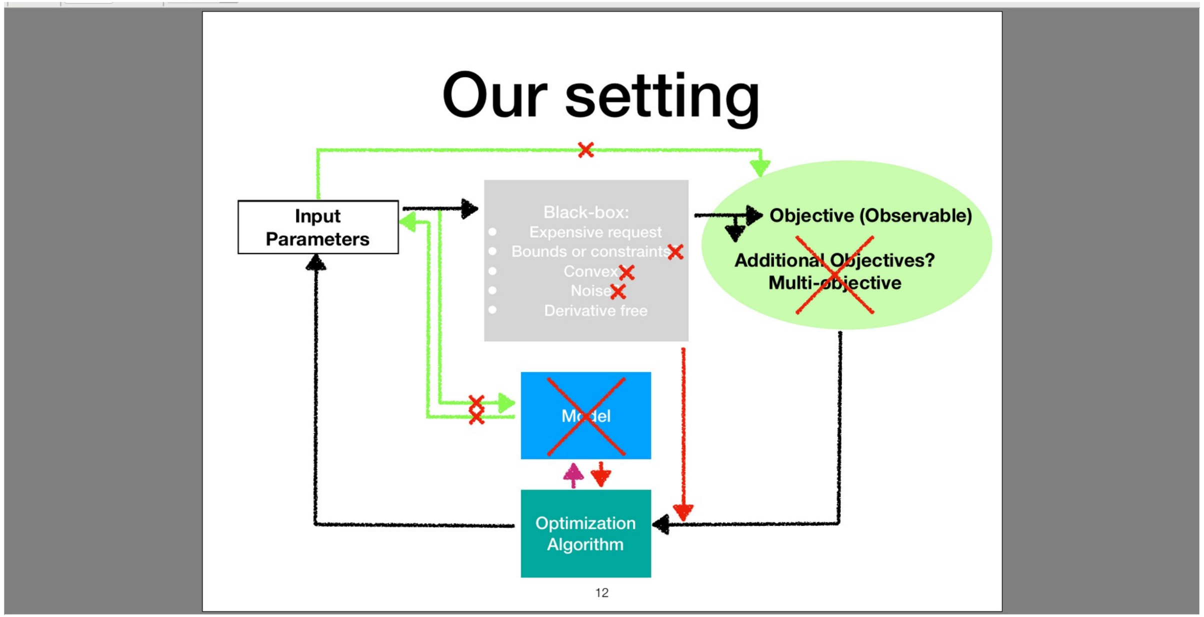
Discussions

Applications:

1. (Powell) numeric optimizer
    1. Linac4 optimizing transmission and chopping efficiency, including BOBYQA & COBYLA
    2. LEIR injected intensity optimization (including RL with DQN, DDPG demos)
    3. Automated ZS alignment (including surrogate model, RL for continuous action space)
2. Classifying LHC/SPS beam dump (images)
    1. With Deep Convolutional Neural Networks (DCNNs)
    2. Using Generative Adversarial Networks (GANs) to create images
    3. With Variational AutoEncoder (VAE)
3. LHC injection magnets anomaly detection
4. ElasticSearch Anomaly detection using LSTMs
5. Image reconstruction for beam profile measurements, using UNET architecture (CNN) and VAE

1.1 Linac4 transmission and chopping optimizer




Our setting
Input Parameters
Black-box:
Expensive request
Bounds or constraints
Convex
Noise
Derivative free
Objective (Observable)
Additional Objectives?
Multi-objective
Model
Optimization Algorithm
12

First implementation

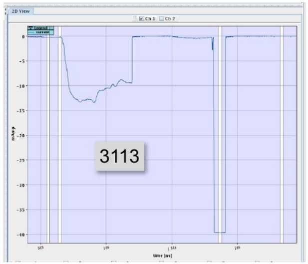
- Goal: optimized transmission AND chopping efficiency

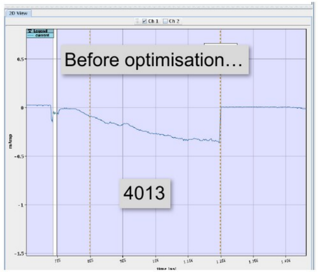
- The implementation:
  - Build reference = BCT 1 defines intensity for unchopped beam, chopping pattern from chopping timings
    - actions: quadrupole settings; observable: chi2 between BCT 2 and reference pattern
  - Powell numeric optimiser: goal minimise chi2.

- Powell: bi-directional line search along search vectors; search vectors are updated in the course of the optimization. No derivatives. Very robust. No prior knowledge of function needed.
  - conjugate direction method

Status from last LINAC4 3 MeV run

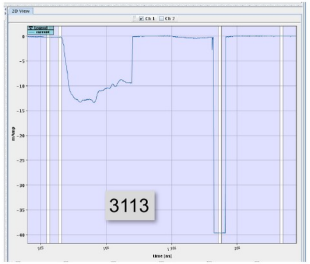
- BCT signals

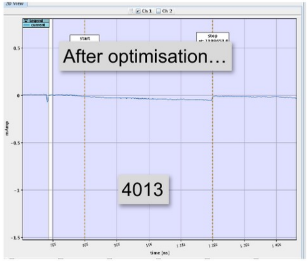




Status from last LINAC4 3 MeV run

- BCT signals

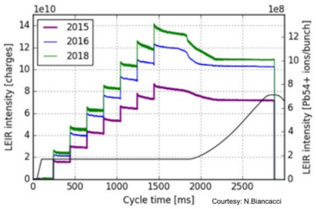




1.2 LEIR injected intensity optimization (Including RL with DQN, DDPG demos)

Classical Powell: LEIR revolution 2018

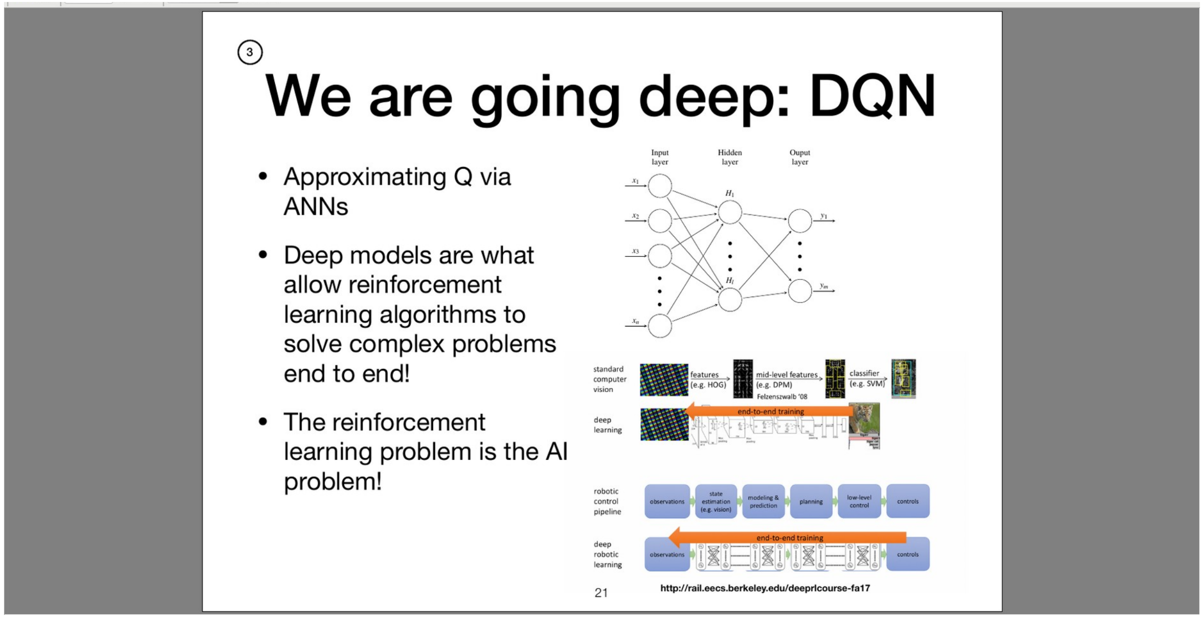
- 2018: record injected intensity into LEIR (and LHC)
- Fast recovery after LEIR machine stops and drifts
- Reproducible performance



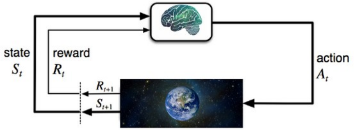
Result LHC 2018 for LEIR extracted intensity

75 ns	Mean $/10^{10}c$	Typical$/10^{10}c$	LIU$/10^{10}c$
LHC run	8.9	9.4	8.8

Iterative updates by using the Bellman equation yield an optimal policy p.
(The Q function is the function, which tells us (the computer) what the impact of taking a decision (a) in a situation (s) and following a tactic (p) afterwards in terms of achieving our goal is!)

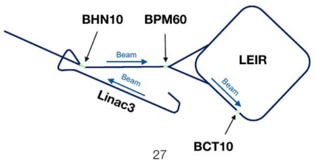


What we did:



- The reward: Intensity of BCT10
- The state: Position of the beam at BPM60
- The action: Change by $\pm\Delta$ or hold the value of dipole BHN10

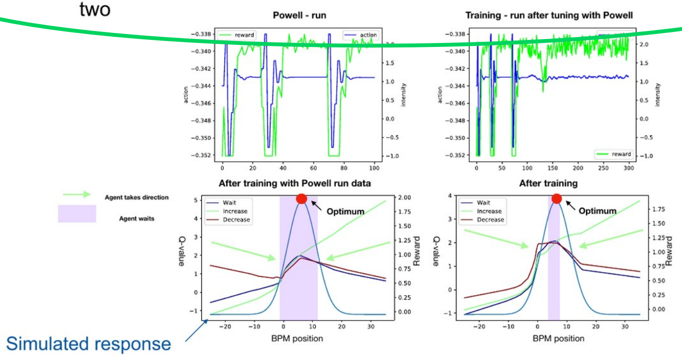
The position of the elements



DFO driven DQN

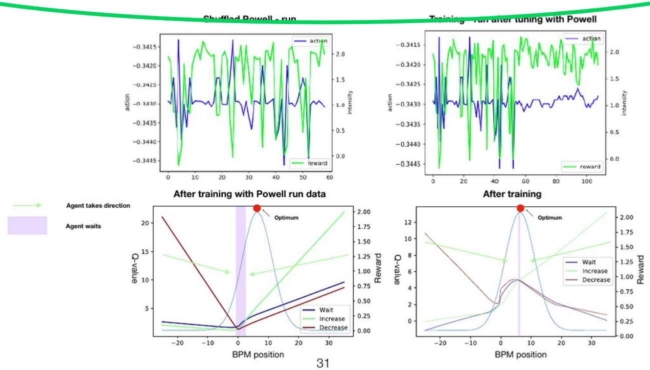
Using the data of a optimization as a pre-tuning of the Network

- Number of iterations reduced due to Powell training by a factor of two

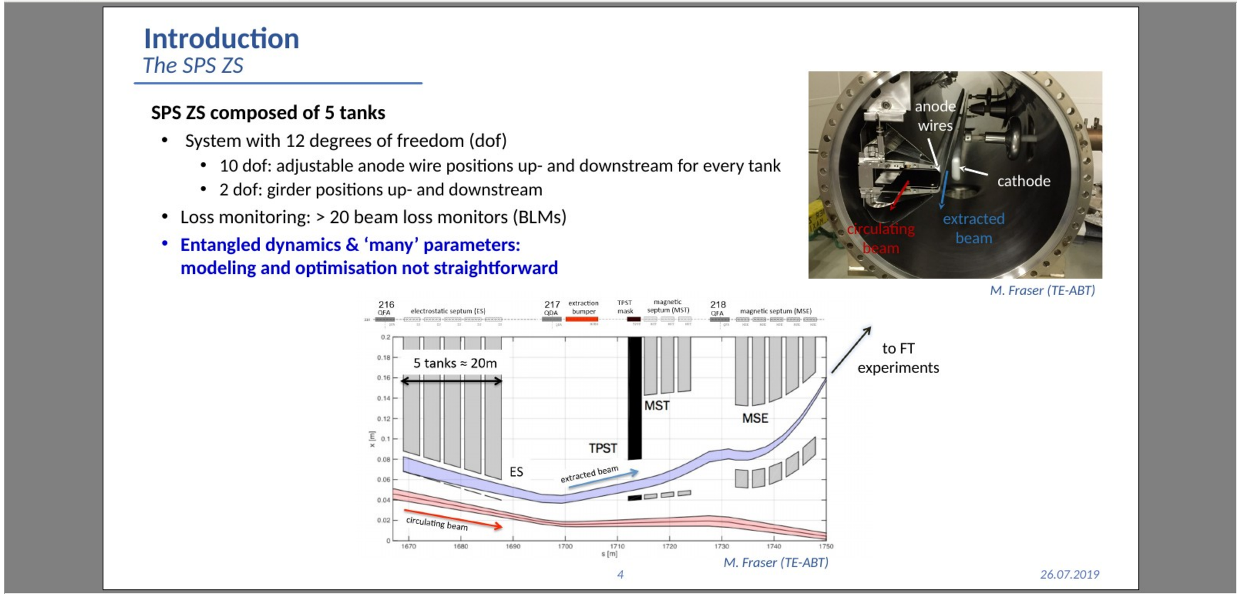


DDQN - results

- Number of iterations further reduced by a **factor of three**



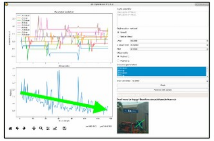
1.3 Automated ZS alignment (+ RL for continuous action space)



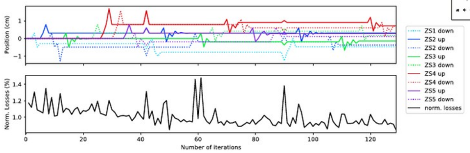
Modified Powell: Automatic ZS alignment

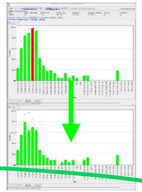
- A modification of Powell's method to obtain a bounded method:

$$\min_{x \in \mathbb{R}^n} \quad f(x)$$
$$\textbf{s.t.} \quad a \leq x \leq b$$



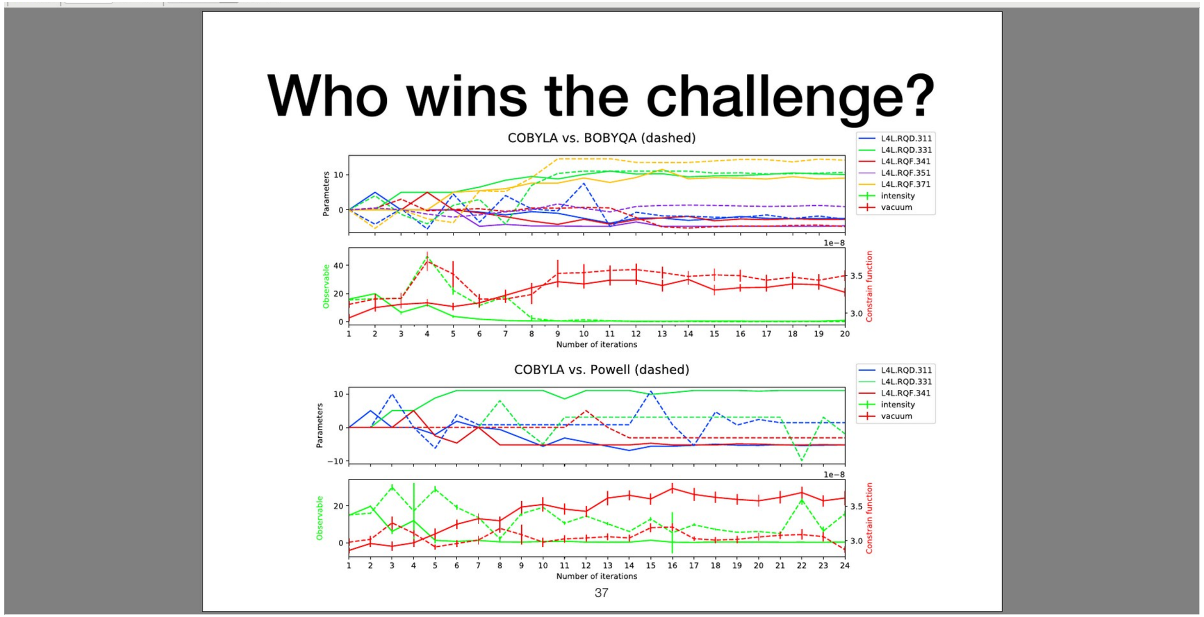
- Penalty method, avoid to move motors to far:





- ZS alignment with 9 DOF. All motors set to 0 - post LS2
- ~30% loss reduction with 125 iterations, 8h -> ~45 min!

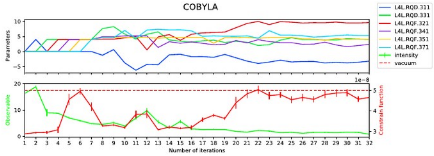
Using three of Powell’s derivative-free optimizers:
1/ classic Powell (Powell’s conjugate direction method)
2/ COBYLA (<u>Constrained</u> Optimization by Linear Approximation
3/ BOBYQA (<u>Bounded</u> Optimization by Quadratic Approximation)



COBYLA

Respects constraints!

- ✓ 6 Parameters
- ✓ The constrain is 5e-8
- ✓ **Does the job - controls the losses!**



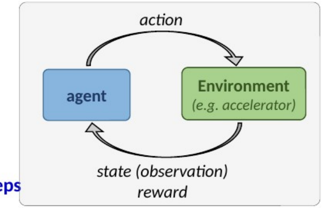
Motivation
Reinforcement learning

Can we do even better?

- **Numerical optimiser has *no memory*:** alignment from scratch every time
- **Reinforcement learning (RL)**
  - Agent interacts with environment and learns dynamics of the system
  - State *not* restricted to action space
  - Agent strategy / knowledge typically represented by neural network (Deep RL)

    => once trained, agent remembers and finds optimum in a few steps

Reinforcement learning

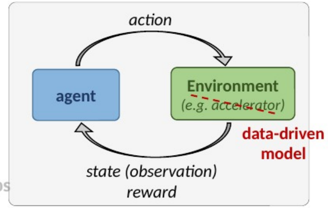


Motivation
Reinforcement learning

Can we do even better?
- **Numerical optimiser has *no memory*:** alignment from scratch every time
- **Reinforcement learning (RL)**
  - Agent interacts with environment and learns dynamics of the system
  - State *not* restricted to action space
  - Agent strategy / knowledge typically represented by neural network (Deep RL)

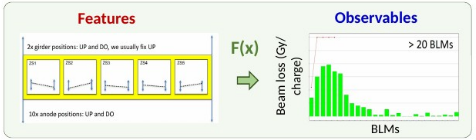
  => once trained, agent remembers and finds optimum in a few steps

Reinforcement learning



Reinforcement learning
- Sample efficiency is key as **machine time is expensive**
- **Idea:** Pre-train agent offline for **'warm start'** in accelerator

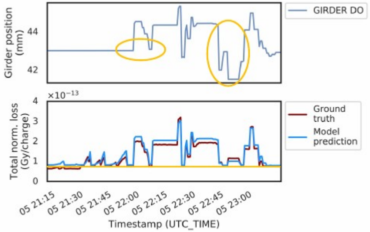
Offline training requires a model F(x) of the system
- Tracking simulation, data-driven model, …
- **Fast, cheap evaluation needed:**
  Pre-training may require hundreds to few thousand iterations

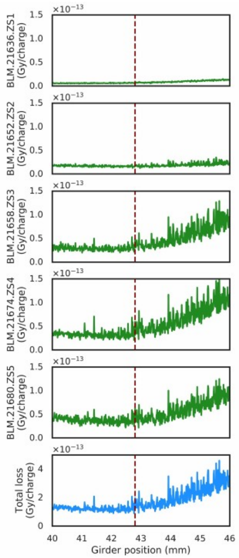


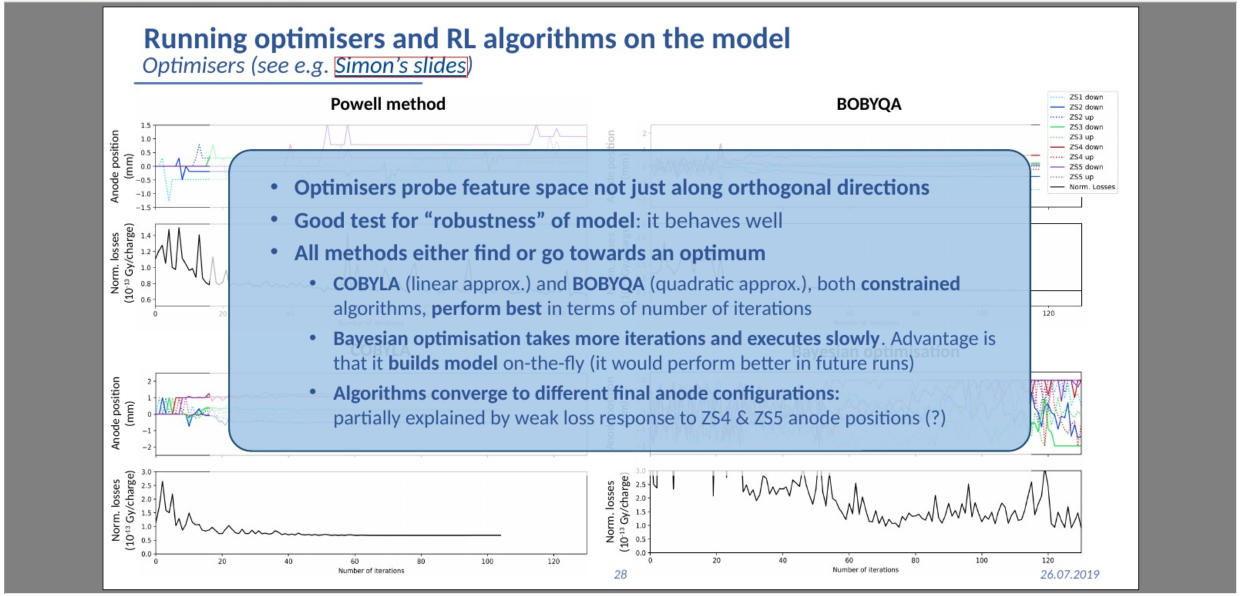
ZS data-driven model: *type B*

Validation: loss response to girder scan

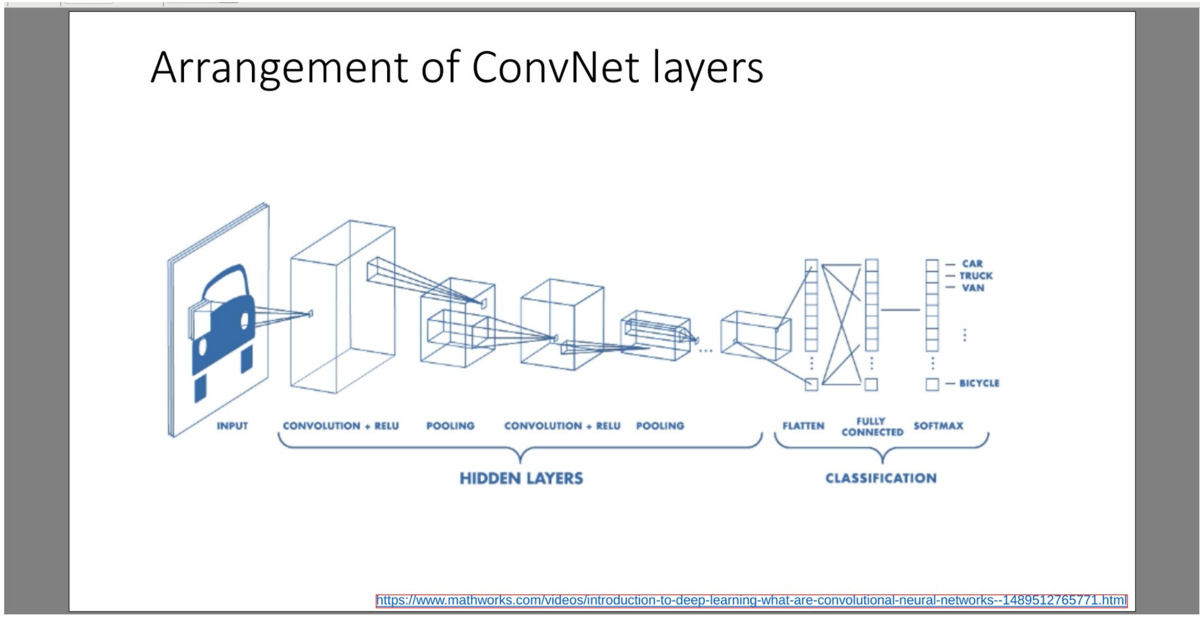
- **Initialise anodes to random positions** *(within +/- 2 mm)*
- **Scan girder DO** around optimum found in measurements, while keeping **initial anode positions fixed**
- Response is roughly **linear above optimum**, and **flattens out below it**
- **Is this reasonable behaviour?**
  At least we also see "clipping effect" in measurements when going below certain girder position






Running optimisers and RL algorithms on the model
Optimisers (see e.g. Simon's slides)
Powell method
BOBYQA
ZS1 down
ZS2 down
ZS2 up
ZS3 down
ZS3 up
ZS4 down
ZS4 up
ZS5 down
ZS5 up
Norm. Losses
Anode position (mm)
Norm. losses (10^-13 Gy/charge)
Number of iterations
Optimisers probe feature space not just along orthogonal directions
Good test for "robustness" of model: it behaves well
All methods either find or go towards an optimum
COBYLA (linear approx.) and BOBYQA (quadratic approx.), both constrained algorithms, perform best in terms of number of iterations
Bayesian optimisation takes more iterations and executes slowly. Advantage is that it builds model on-the-fly (it would perform better in future runs)
Algorithms converge to different final anode configurations: partially explained by weak loss response to ZS4 & ZS5 anode positions (?)
28
26.07.2019

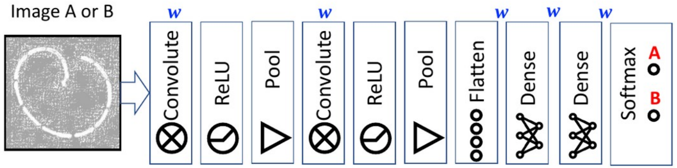
2.1 Classifying LHC/SPS beam dumps with Deep Convolutional Neural Networks (DCNNs)



Training the Network

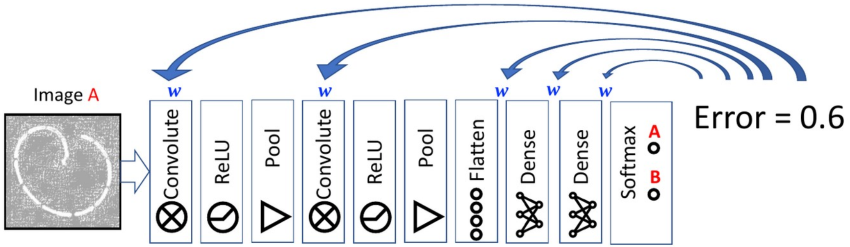
- Imagine we have 2 Convolution/activation layers each followed by pooling
  - 1st convolution layer is the input layer
- 1 Flattening layer
- 2 Dense (fully connected) layers
- 1 Softmax output layer

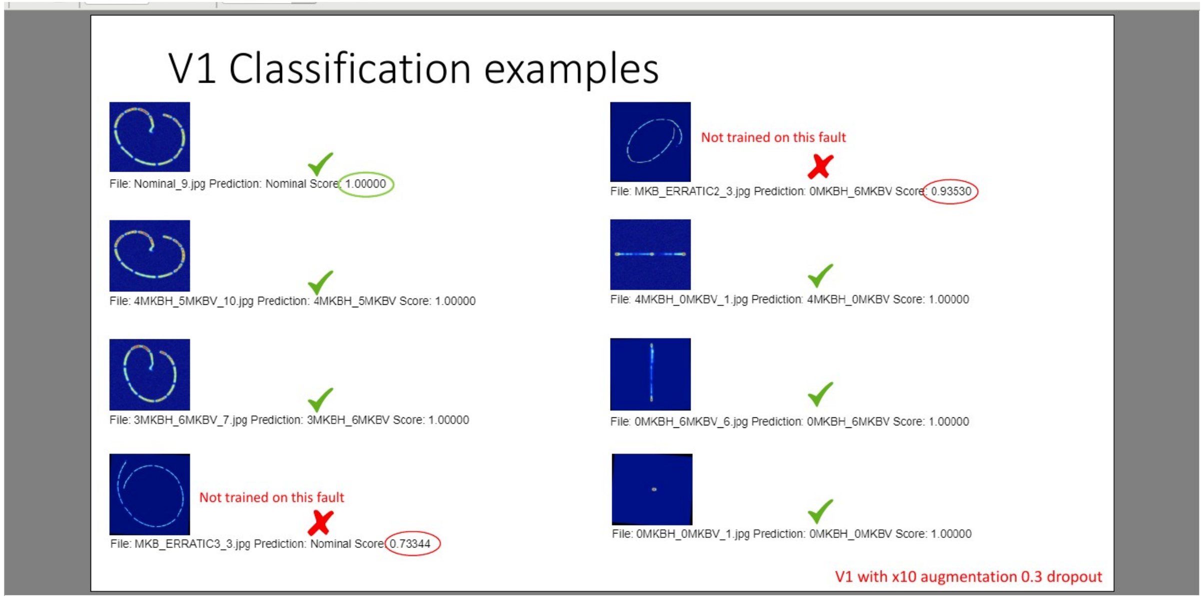
- Here 5 layers have trainable weights: convolution filters and dense layers



Network Training - Backpropagation

- Error E used to update training weights w_i, via calculation of partial derivative of each weight $\delta E/\delta w_\mathrm{i}$
- As an aside, training and fitting CNNs is very well adapted to GPUs….*much* faster than CPU

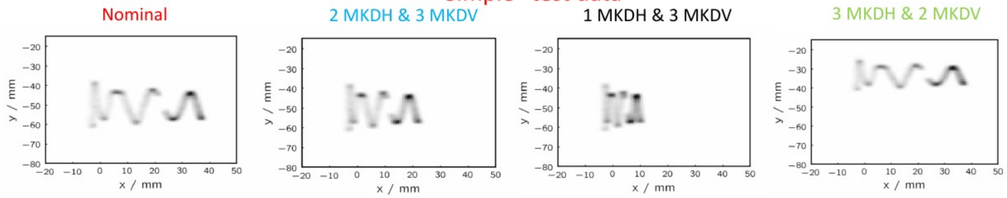



V1 Classification examples
File: Nominal_9.jpg Prediction: Nominal Score: 1.00000
File: 4MKBH_5MKBV_10.jpg Prediction: 4MKBH_5MKBV Score: 1.00000
File: 3MKBH_6MKBV_7.jpg Prediction: 3MKBH_6MKBV Score: 1.00000
Not trained on this fault
File: MKB_ERRATIC3_3.jpg Prediction: Nominal Score: 0.73344
Not trained on this fault
File: MKB_ERRATIC2_3.jpg Prediction: 0MKBH_6MKBV Score: 0.93530
File: 4MKBH_0MKBV_1.jpg Prediction: 4MKBH_0MKBV Score: 1.00000
File: 0MKBH_6MKBV_6.jpg Prediction: 0MKBH_6MKBV Score: 1.00000
File: 0MKBH_0MKBV_1.jpg Prediction: 0MKBH_0MKBV Score: 1.00000
V1 with x10 augmentation 0.3 dropout

V2 ConvNet for SPS dump

- Aim is try to train a NN using simulations
  - Produced 1000 simulations of expected BTVDD readings for 4 categories for the **training set**:
    - "Nominal", "3 MKDH & 2 MKDV", "2 MKDH & 3 MKDV", "1 MKDH & 3 MKDV"
  - Produced 1000 simulations for the same categories for the **validation set**
- Data produced for the SPS:
  - Dump for SFTPRO beam at 400 GeV → 2 batches with realistic length and batch spacing
  - Random CO in x and y at the BTV location → Gauss(0, 3 mm) (quite generous…)
  - Random emittance in x and y → Gauss_x(9 mm.mrad, 3 mm.mrad), Gauss_y(7 mm.mrad, 3 mm.mrad)
  - 1000 particles per 420x2x25 ns slots (simpler to simulate) all the same

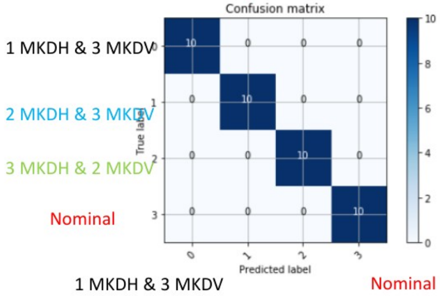
"Simple" test data

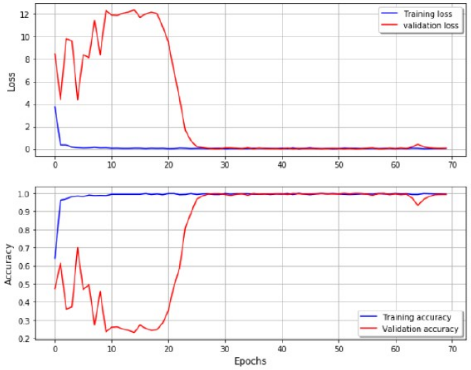


ConvNet for SPS dump

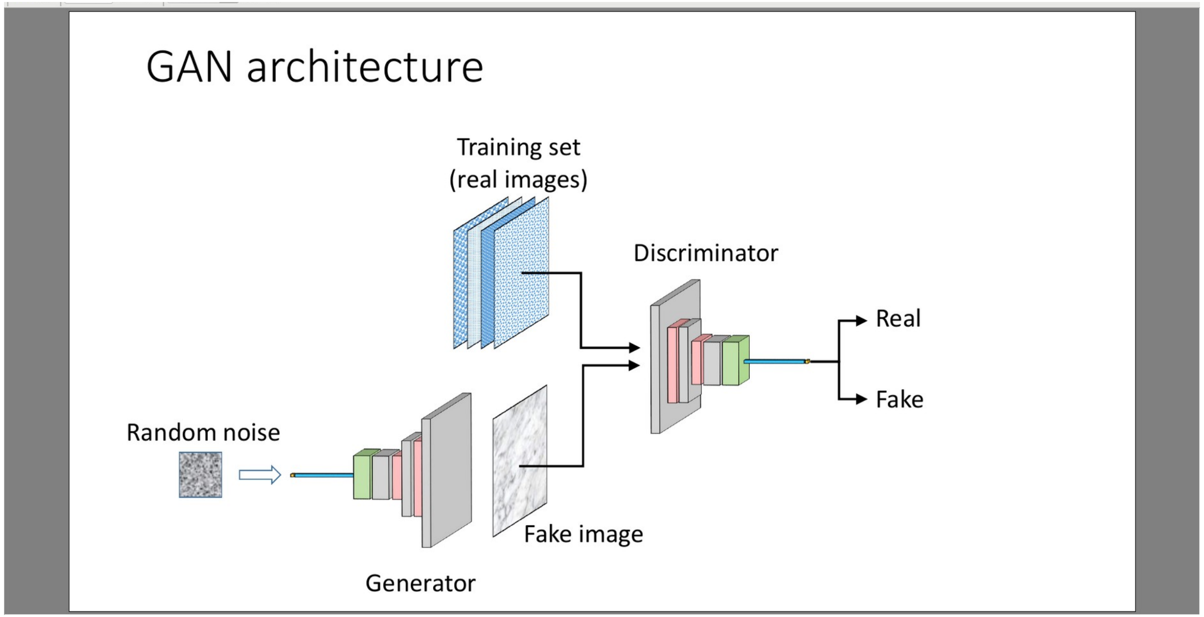
- Adding more training data with ImageDataGenerator augment from keras to increase the statistics → 4000x10 . Much better.

"Simple" test data

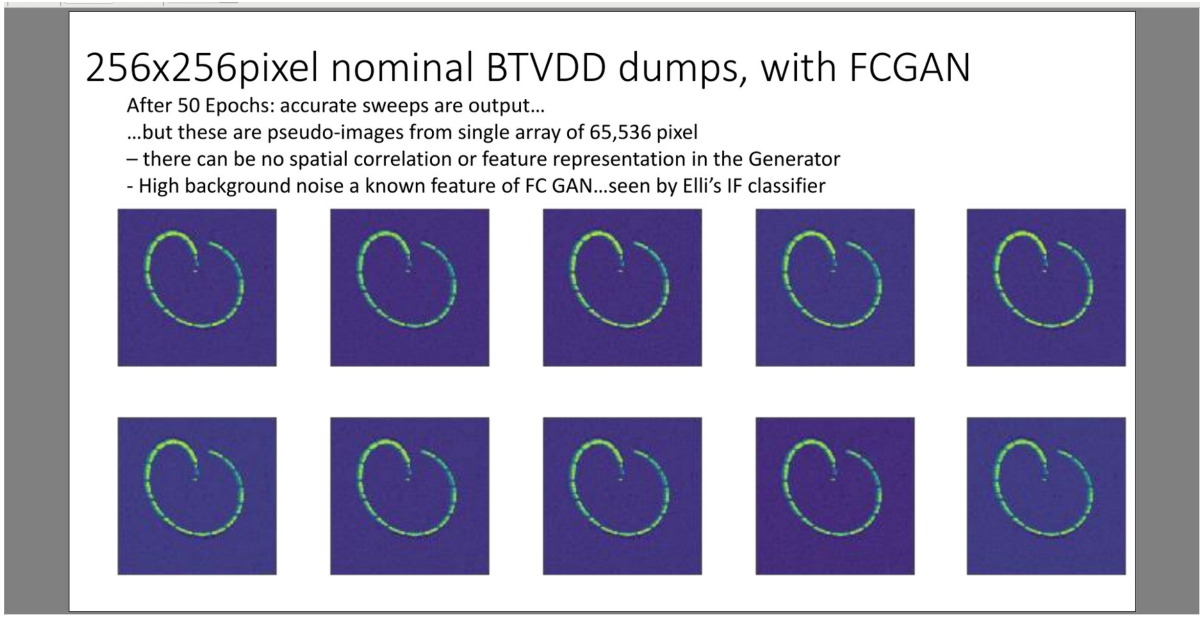




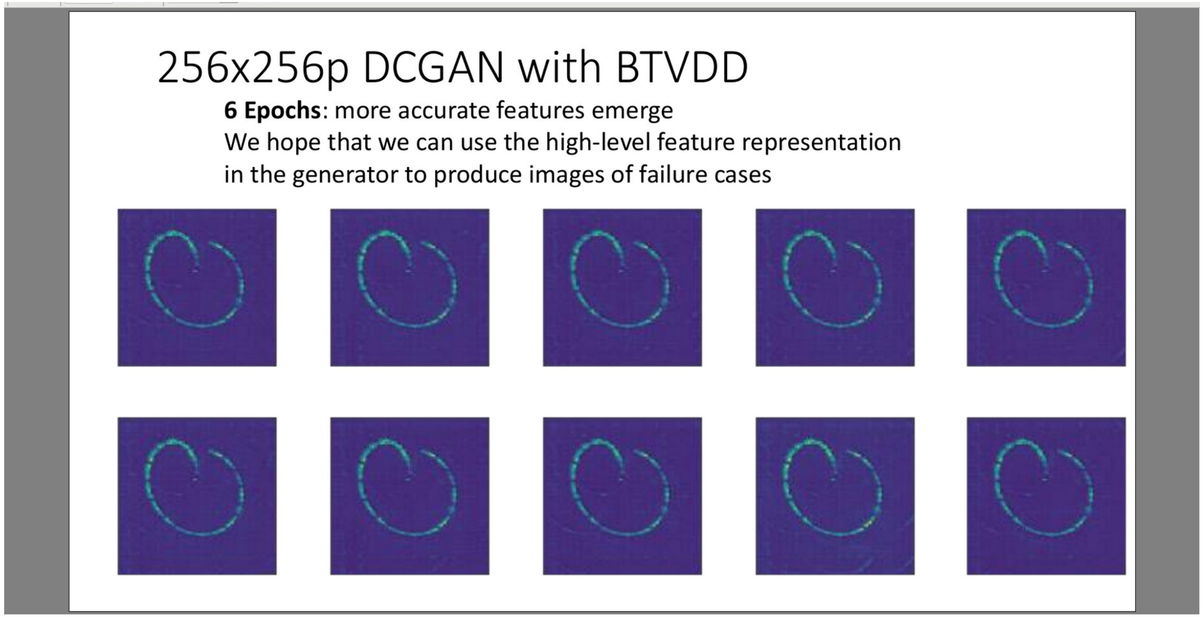
2.2 Classifying LHC/SPS beam dumps Using Generative Adversarial Networks (GANs) to create images



Fully-connected layers GAN



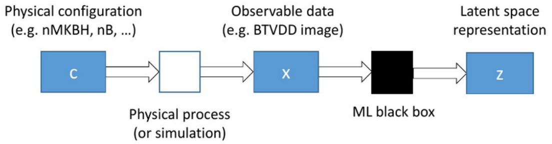
Deep convolutional GAN



2.3 Classifying LHC/SPS beam dumps with Variational AutoEncoder (VAE)

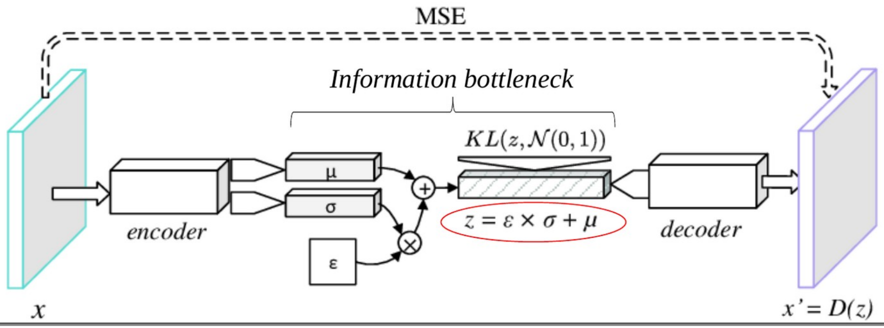
Latent space *z*

- Representational Deep Learning uses an accessible ‘latent’ space ***z*** where a representation of ***c*** (configuration, or generator) is deliberately enforced
- ***z*** values can be extracted corresponding to a specific observable ***x***
- The problem is then ‘reduced’ to mapping ***z*** to generators ***c***
- For a generative method (VAE, GAN, …) reconstructed observables ***x’*** can be produced by providing new samples of ***z***



Under the hood: VAE guts

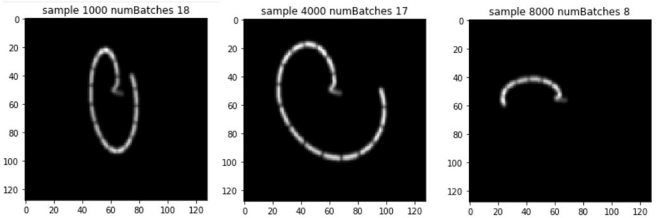
- Consists of Encoder, an information bottleneck, and decoder
- 2 parts to Loss function: MSE “***x*** reconstruction” and KL “***z*** relative entropy”
- ε adds randomness while allowing differentiation for back-propagation



CERN

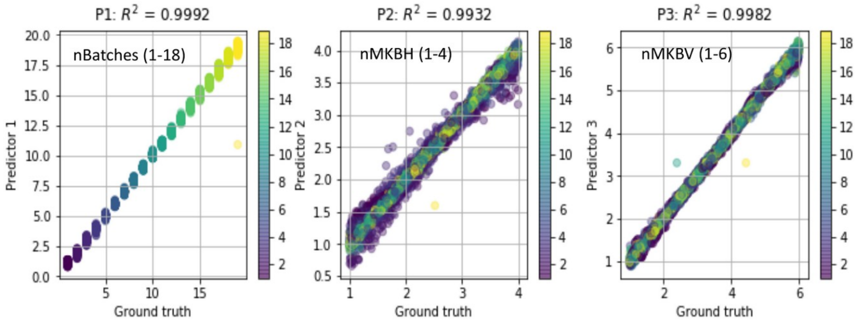
BTVDD datasets

- Produced synthetic BTVDD images ***x*** with tracking simulation
- Realistic physical configuration space ***c*** (kicker waveforms, tracking)
- Output closely resembles measured BTVDD traces
- 7 degrees of freedom, continuous or discrete
- Full control of physical configuration space ***c*** and of images ***x(c)***:
  - numBatches
  - batchLength
  - batchSpacing
  - nMKDH
  - nMKDV
  - energy
  - Sweep delay

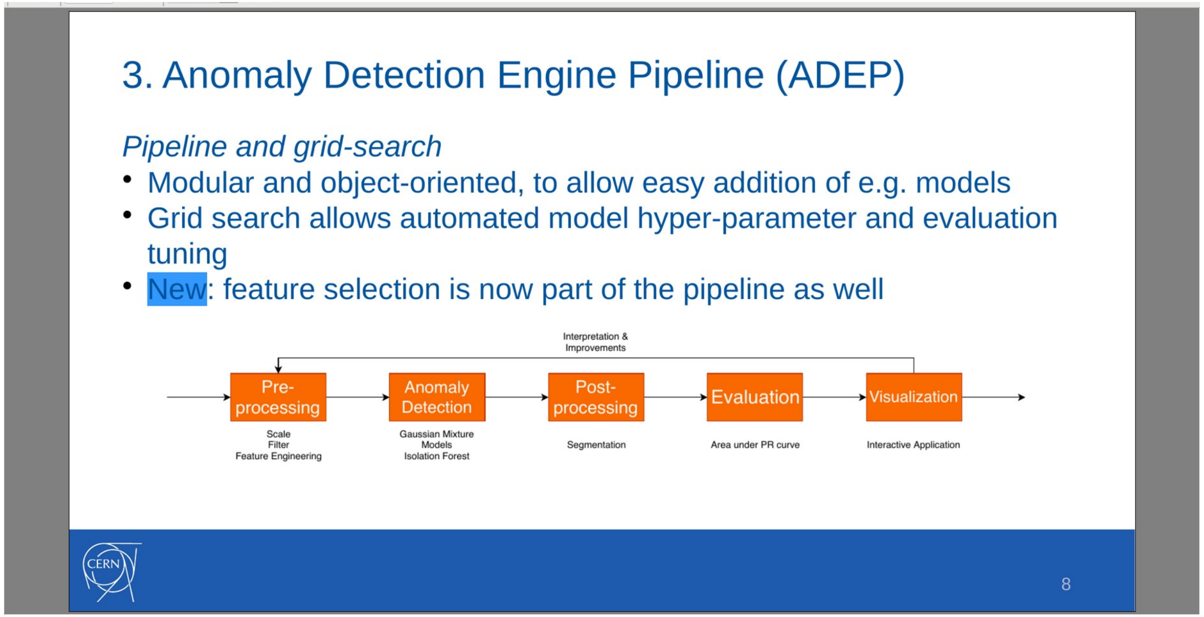


VAE + Predictor results

- Good accuracy in predicting (almost) all configuration space values
- Difficulty for nMKBH for short sweeps…



3. LHC injection magnets anomaly detection



3. Anomaly Detection Engine Pipeline (ADEP)

Visualization

using Plotly Dash, interactive data browser with live validation metrics

5. COBRAS for interactive clustering

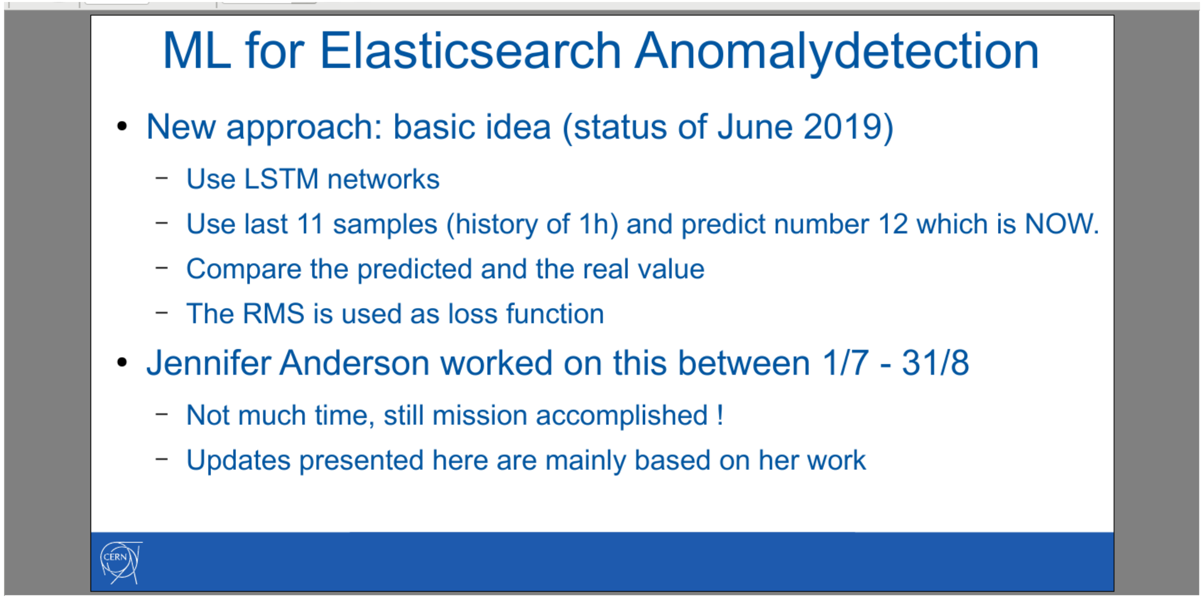
- https://dtai.cs.kuleuven.be/software/cobras/
- Goal: improve the segment predictions by clustering with user-input
- Currently through a terminal or a Jupyter notebook
- COBRAS requires fixed length segments, not our case, so we made an artifical subset of features, one is the number of timestamps
- Other features chosen: fourier-components of pressure and temperature, seemed most plausible to cluster the *false positives*
- Results from (non-interactive) clustering of 2018 data with the model from 2017 interactive COBRAS clustering:

Year 2018	Anomaly	Normal
Detected	TP = 4	FP = 33
Undetected	FN = 3	TN = 1407

→

Year 2018	Anomaly	Normal
Detected	TP = 6	FP = 2
Undetected	FN = 1	TN = 1437

4. Elasticsearch Anomaly detection using LSTMs

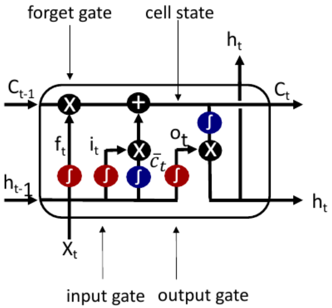


LSTM’s – what’s the purpose?

- Proposed in 1997 by Hochreiter and Schmidhuber
- Goal: Attack the decaying error back flow - introduce constant error flow through *constant error carrousels* in special units
- Complemented by Gers, Schmidhuber and Cummins in 1999
- Learn from sequenced data, time series with long time lags
- Solves the vanishing gradient problem

Colah’s blog post:
Undersalning LSTM Networks

Hochreiter-Schmidhuber:
Long Short-Term Memory (1997)



$$f_t = \sigma_s\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

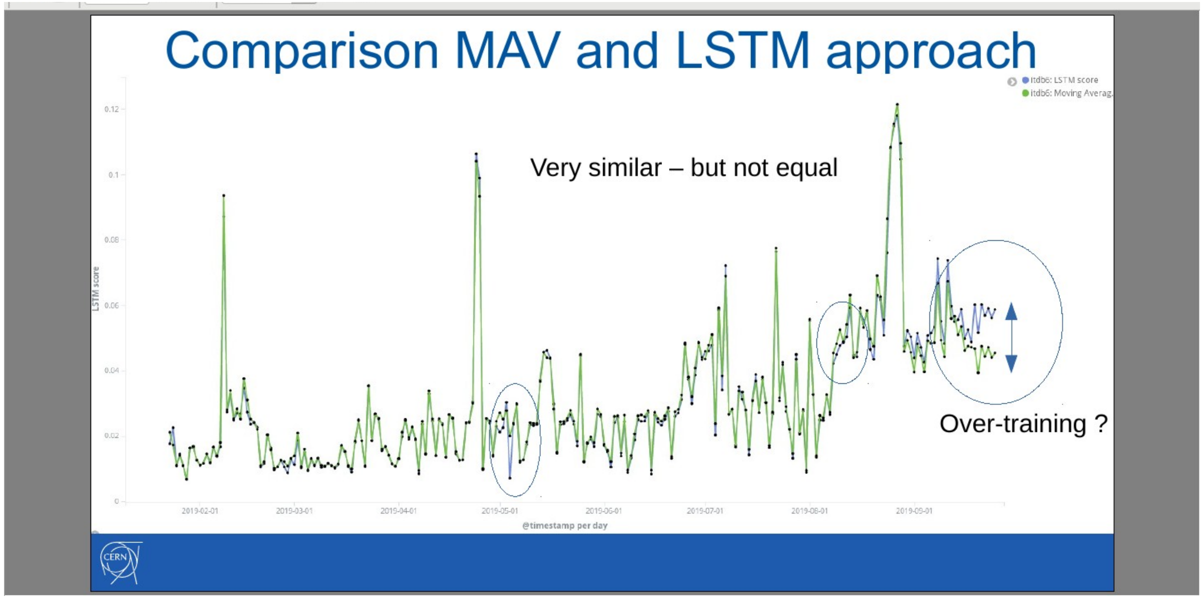
$$i_t = \sigma_s\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$

$$\bar{C}_t = \sigma_h\left(W_C \cdot [h_{t-1}, x_t] + b_C\right)$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \bar{C}_t$$

$$o_t = \sigma_s\left(W_o \cdot [h_{t-1}, x_t] + b_o\right)$$

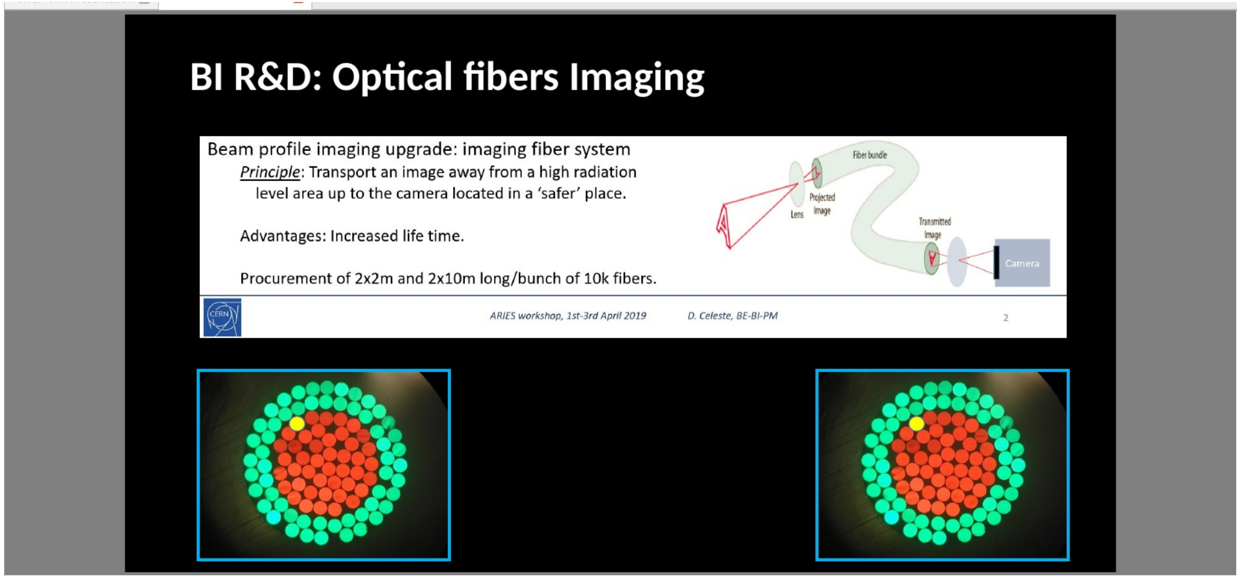
$$h_t = o_t \cdot \sigma_h(C_t)$$

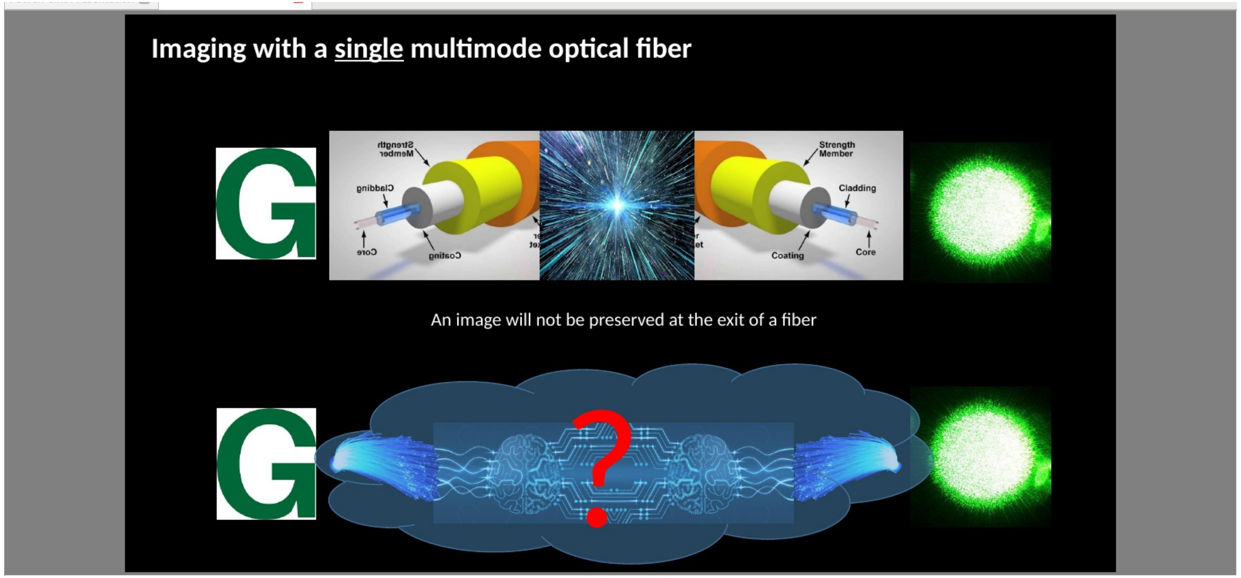

Comparison MAV and LSTM approach
itdb6: LSTM score
itdb6: Moving Averag
Very similar – but not equal
Over-training ?
LSTM score
0.12
0.1
0.08
0.06
0.04
0.02
0
2019-02-01
2019-03-01
2019-04-01
2019-05-01
2019-06-01
2019-07-01
2019-08-01
2019-09-01
@timestamp per day

ML for Elasticsearch Anomalydetection

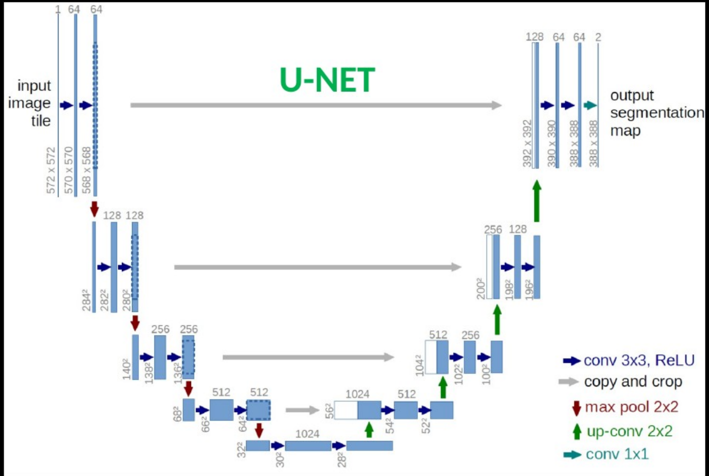
- Example:
  - View for Wednesday for one cluster
  - Normal monitoring clean until 3pm
  - Started investigating
  - One node over disk limit in ES
  - Caused by a new index class created by a user which is badly designed

5. Image reconstruction for beam profile measurements, using UNET architecture (CNN) and VAE

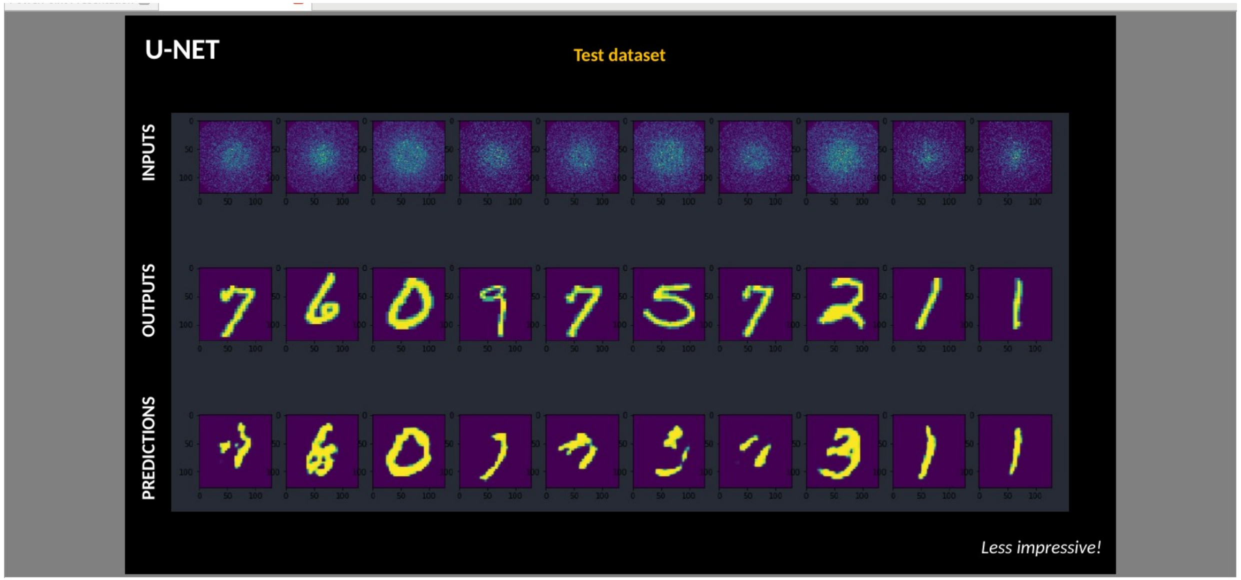



Imaging with a single multimode optical fiber
Strength Member
Cladding
Coating
Core
An image will not be preserved at the exit of a fiber
?

What network to use ?



https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

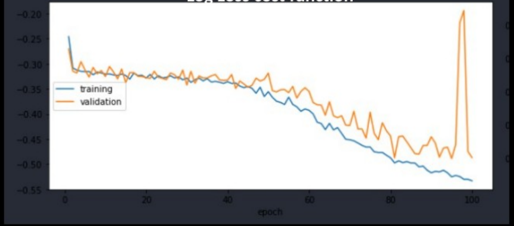


U-NET
Test dataset
INPUTS
OUTPUTS
PREDICTIONS
Less impressive!

U-NET

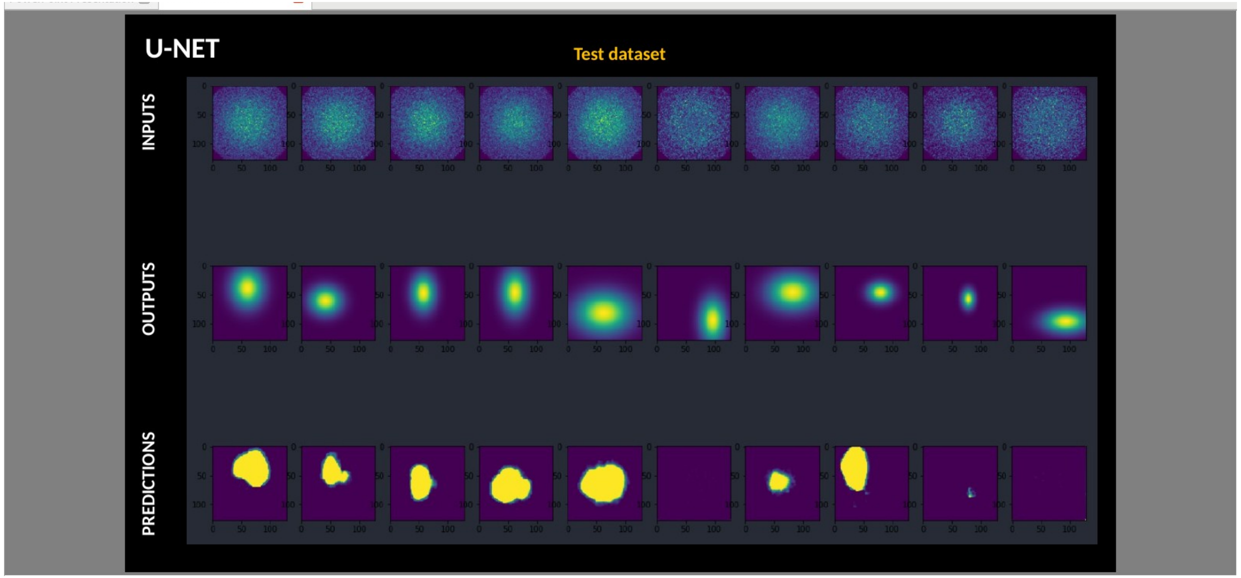
With “Beam” inputs

Log Loss cost function



batch_size=16,
steps_per_epoch=100,
epochs=100,
validation_split=0.1

Less Poor amount of data (1000 images)
Augmented by adding noise

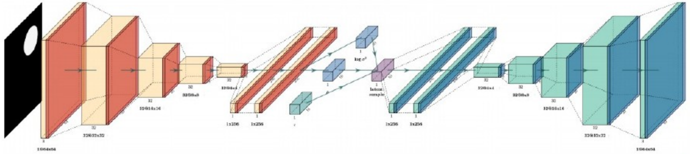


U-NET
Test dataset
INPUTS
OUTPUTS
PREDICTIONS

VAE?

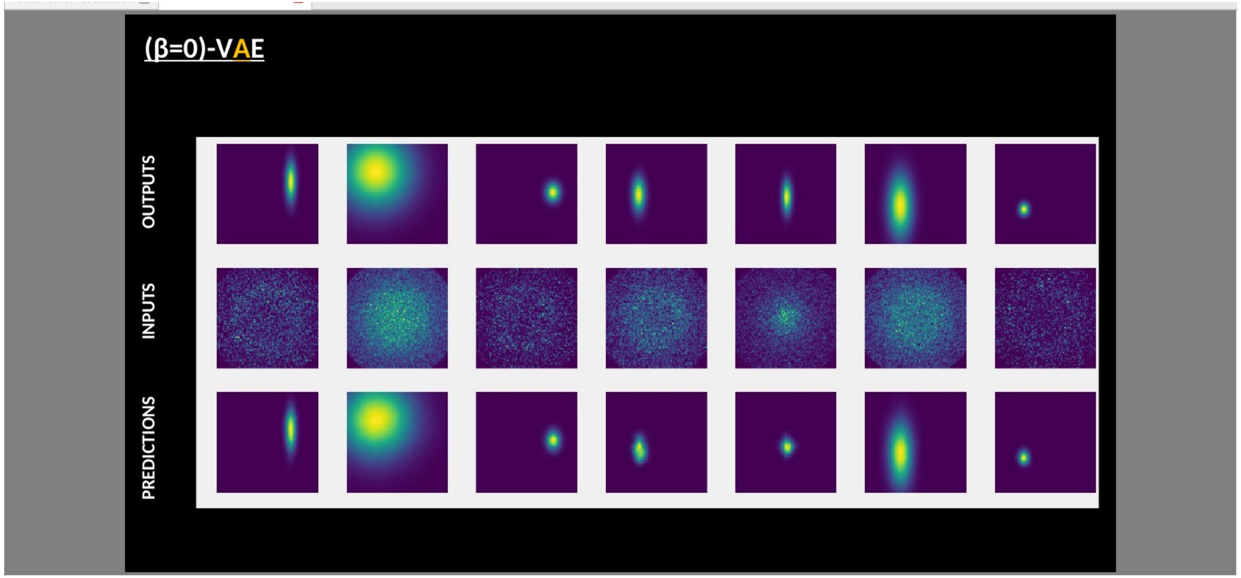
VAE configuration

- Images downsampled to 128x128 (eventually try 256x256, ≈real size)
- Convolutional VAE architecture, similar to below:
  - Additional 128x128x32 convolution layer and input/output dimension of 128x128
  - Convolution filter layer connected directly to loss layer/first filter layer (no dense)
  - Filter size always 3x3, no pooling layers (instead used stride = 2)



https://gitlab.cern.ch/goddard/beta-vae

https://keras.io/examples/variational_autoencoder_deconv/


(β=0)-VAE
OUTPUTS
INPUTS
PREDICTIONS