HUGS 2024

Machine Learning for Nuclear Physics

Lecture 2

Daniel Lersch

Tuesday, June 4, 2024

Jefferson Lab

U.S. DEPARTMENT OF ENERGY | Office of Science

JSA

On the Menu

- **Lecture 1**
  - Machine learning workflow
  - Neural networks
  - Deep learning
- **Lecture 2**
  - Network types and applications in Nuclear Physics
  - Methods and tools

AI ⊃ ML ⊃ DL

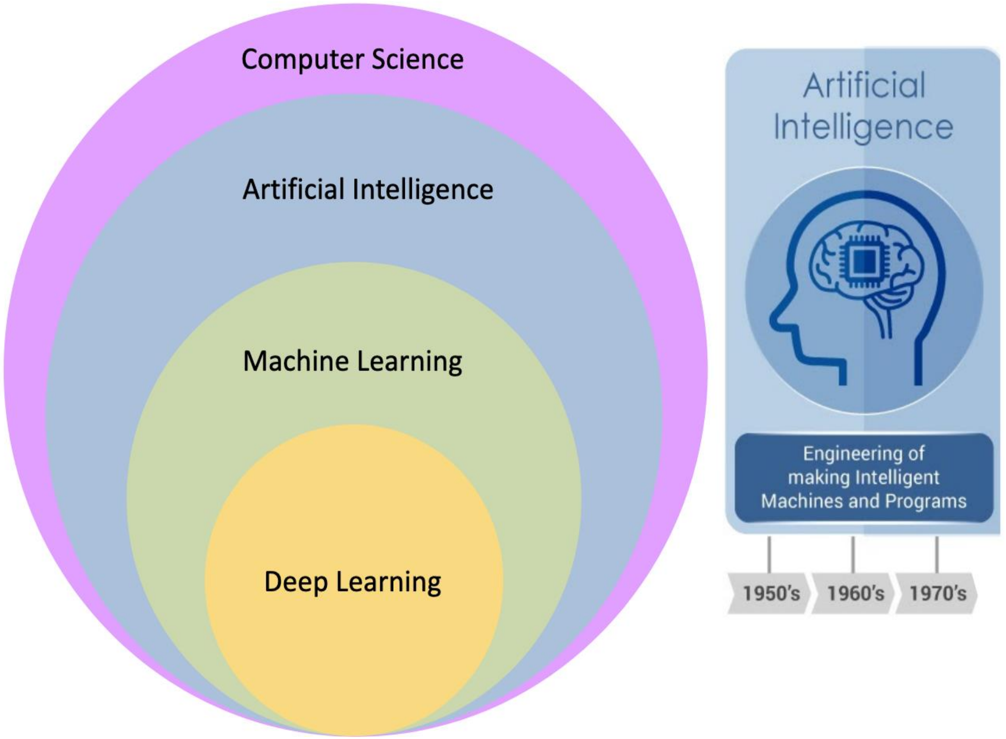


Image source: https://www.embedded-vision.com/industry-analysis/blog/artificial-intelligence-machine-learning-deep-learning-and-computer-visionwha

Plot taken from Brenda Ngs talk at deep learning for science school 2019

Network Types and Applications

- Dense neural networks **(already covered in part 1)**
- Convolutional neural networks
- Recurrent neural networks
- Graph neural networks
- Large language models
- Auteoncoder neural networks
- Generative Models

Convolutional Neural Network (CNN)

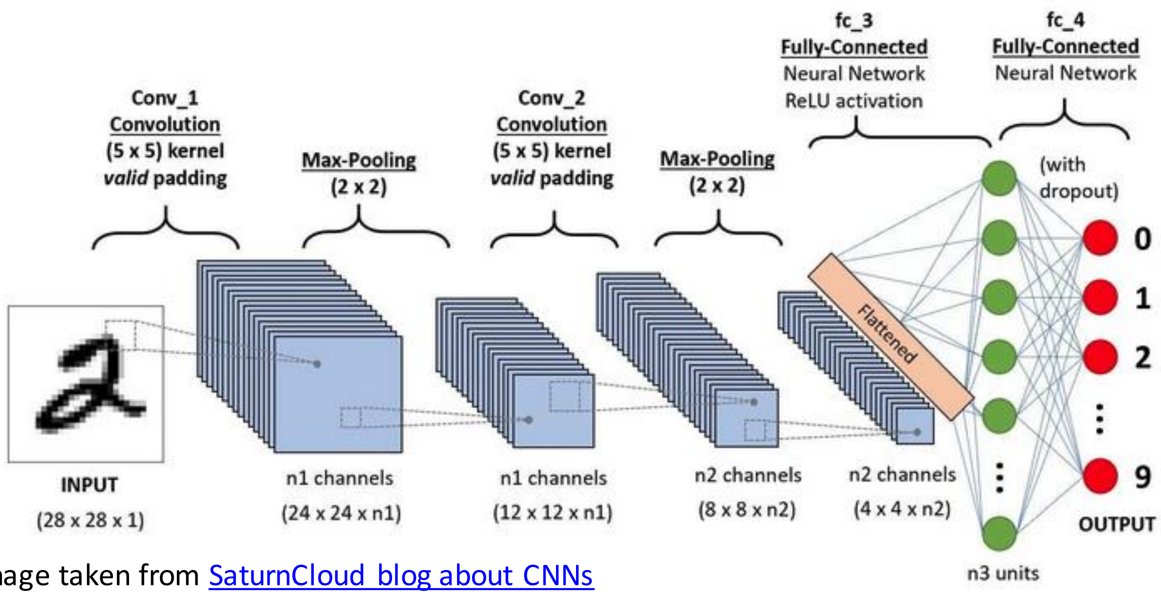


Image taken from [SaturnCloud blog about CNNs]

- Used for image recognition / computer vision
- Feature extraction via filters and subsampling
- **At JLab:** Support shift takers with online monitoring --> See **"Machine Learning for Nuclear Physics: Hands-On"**,Wed. 06/05/2024, Thomas Britton

Convolutional Layer:

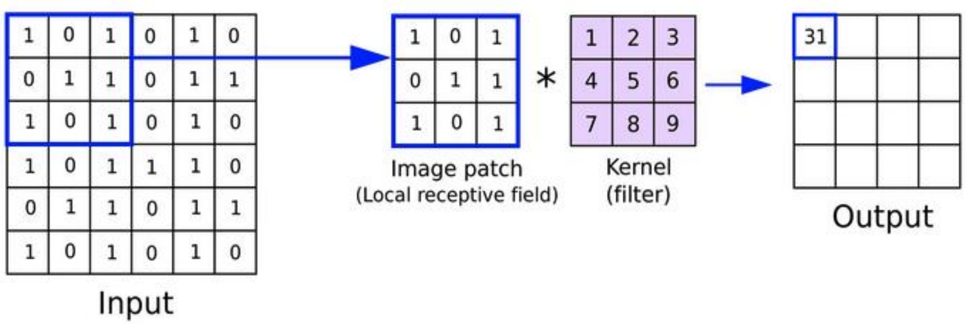


Image taken from [SuperAnnote blog about CNNs]

Pooling / Subsampling Layer:

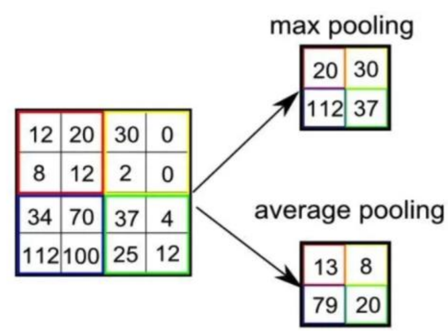
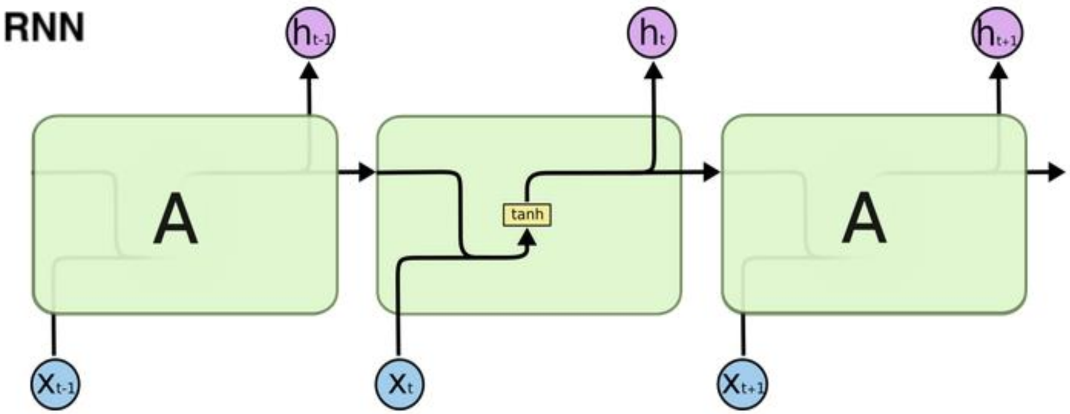
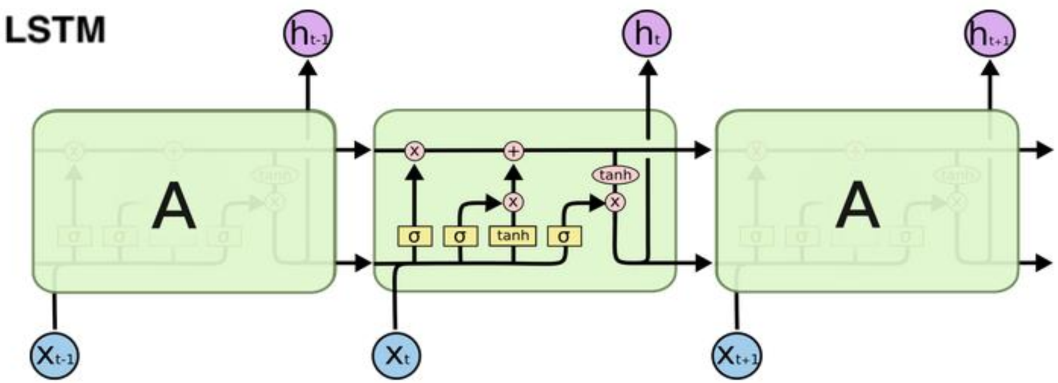


Image taken from [SaturnCloud blog about CNNs]

Recurrent Neural Network (RNN)

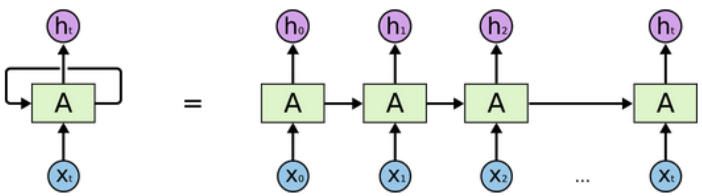




- RNNs may struggle with long-term dependencies
- Long Short Term Memory (LSTM) Networks are alternative to RNN
- Use gates to regulate information flow

- Used to analyze time series or sequential data **(e.g. hits in a forward drift chamber)**
- **Input:** (x_{t-1}, x_t, x_{t+1})
- **Output:** (h_{t-1}, h_t, h_{t+1})

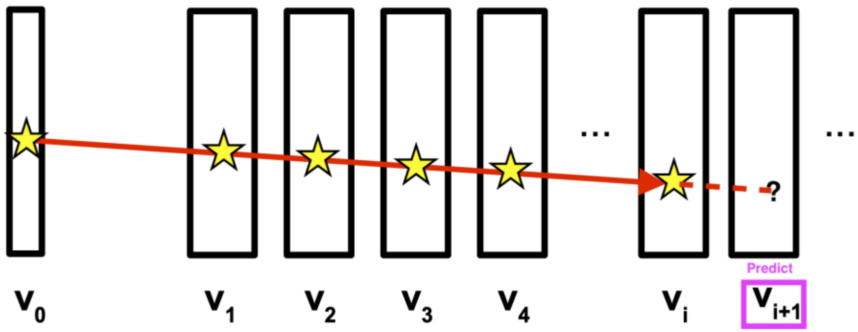
  $h_t(x_t, h_{t-1}) = \tanh[W \cdot (x_t, h_{t-1})]$
- Unroll recurrent loop

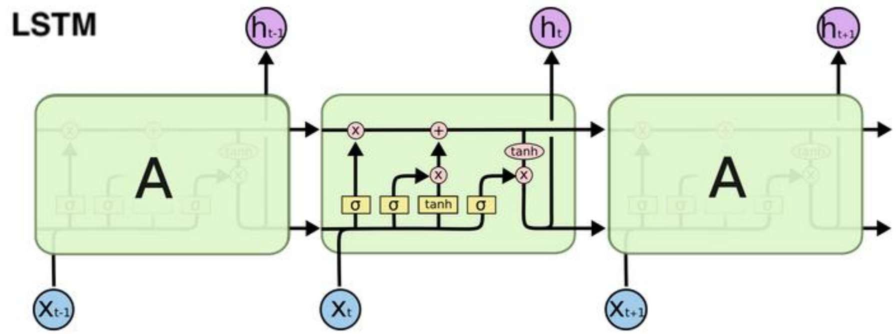


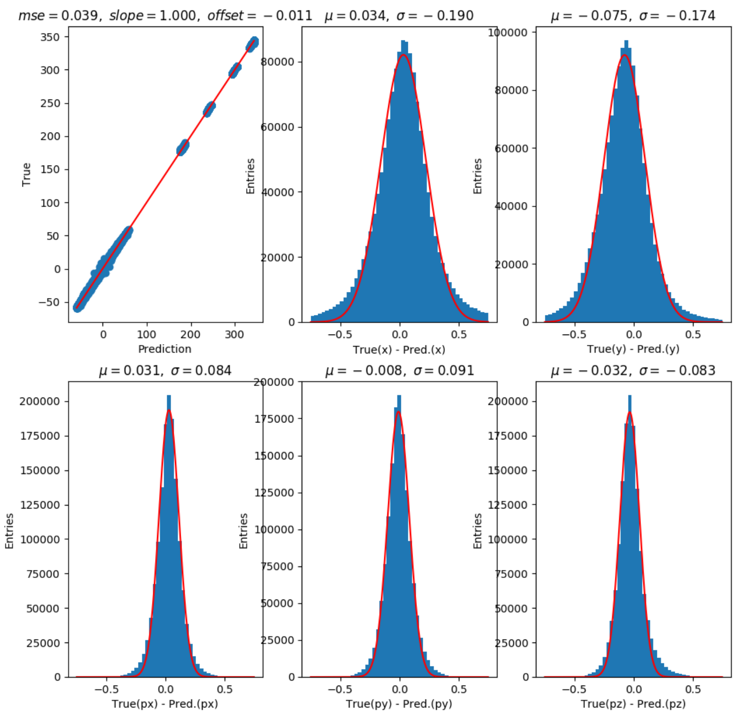
- Pictures taken from colah's blog
- This blog provides good and detailed explanation
- Did not talk about Gated Recurrent Unit (GRU) Networks

Recurrent Neural Network for Track Reconstruction

- Toy problem provided by David Lawrence and Thomas Britton
- Reconstruct particle tracks in GlueX forward drift chamber

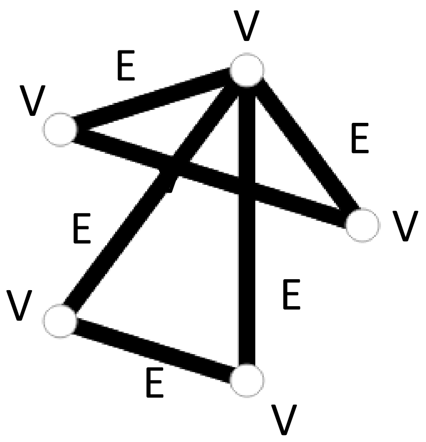




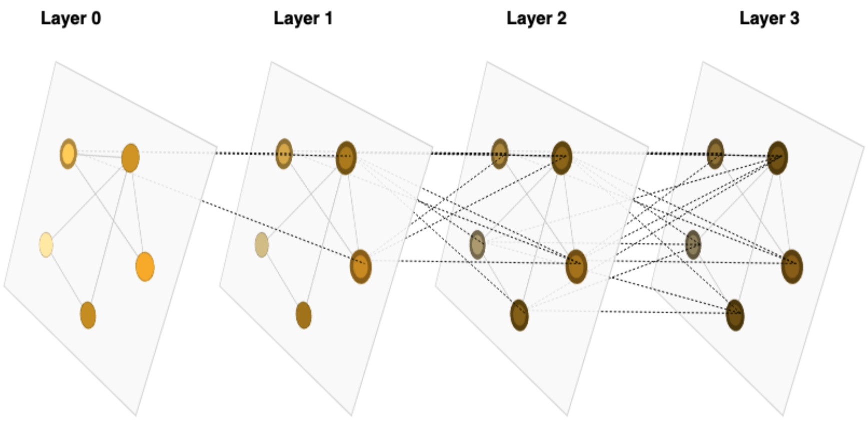


- Try to reconstruct particle properties (momentum and position in plane i when all previous planes fired
- Use LSTM + Dense layers to reconstruct particle tracks

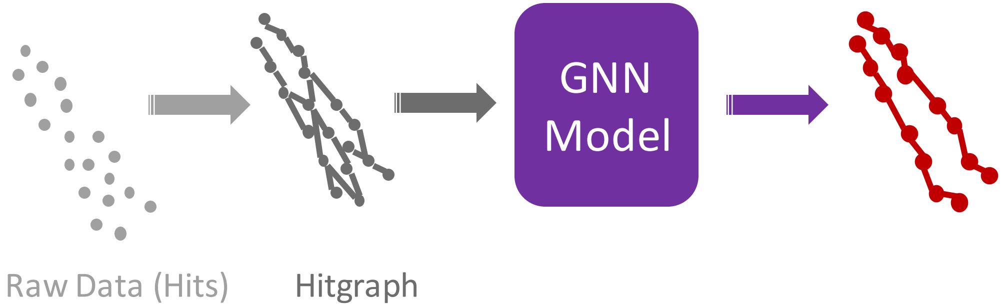
Graph Neural Network (GNN)

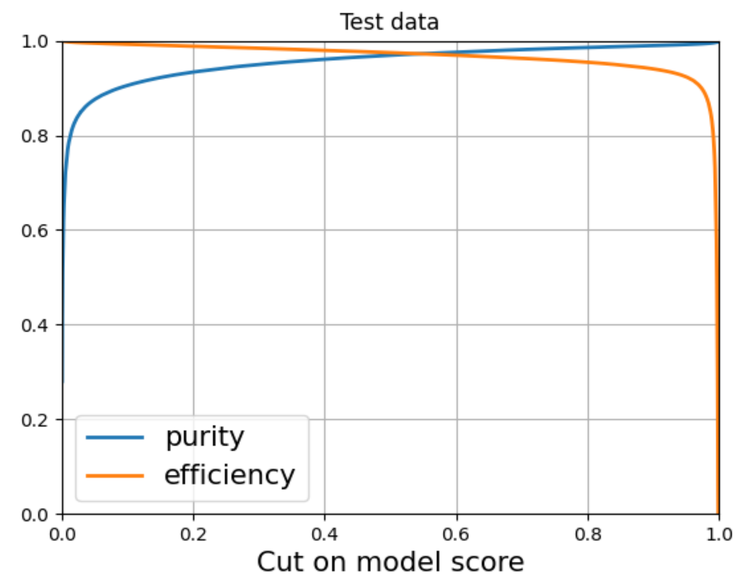


Node (Vertex V)
Edge E
Graph G(V,E)



Pictures taken from [distill.pub intro to GNNs]





- **At JLab:** Use GNNs for finding particle tracks in Hall D (Kishansingh Rajput & Ahmed Mohammed)
- Nodes: hits / Edges: Connect hits
- Form tracks via edge classification

Large Language Model (LLM)

- Natural language processing (e.g. language translation, analyze documents, answer questions,…)
- ChatGPT, Gemini, Meta's LLaMA
- Backbone is [transfromer network]() --> Replace RNN based [seq2seq model]()
- Encoder-Decoder structure
- Transformers capture distant / long-range contexts
- Crucial: Need to translate text to numbers --> Word Tokenizer

Many use-cases for LLMs in Nuclear Physics!

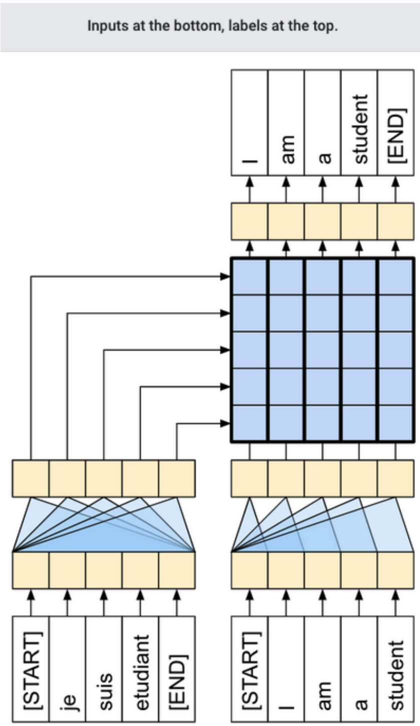
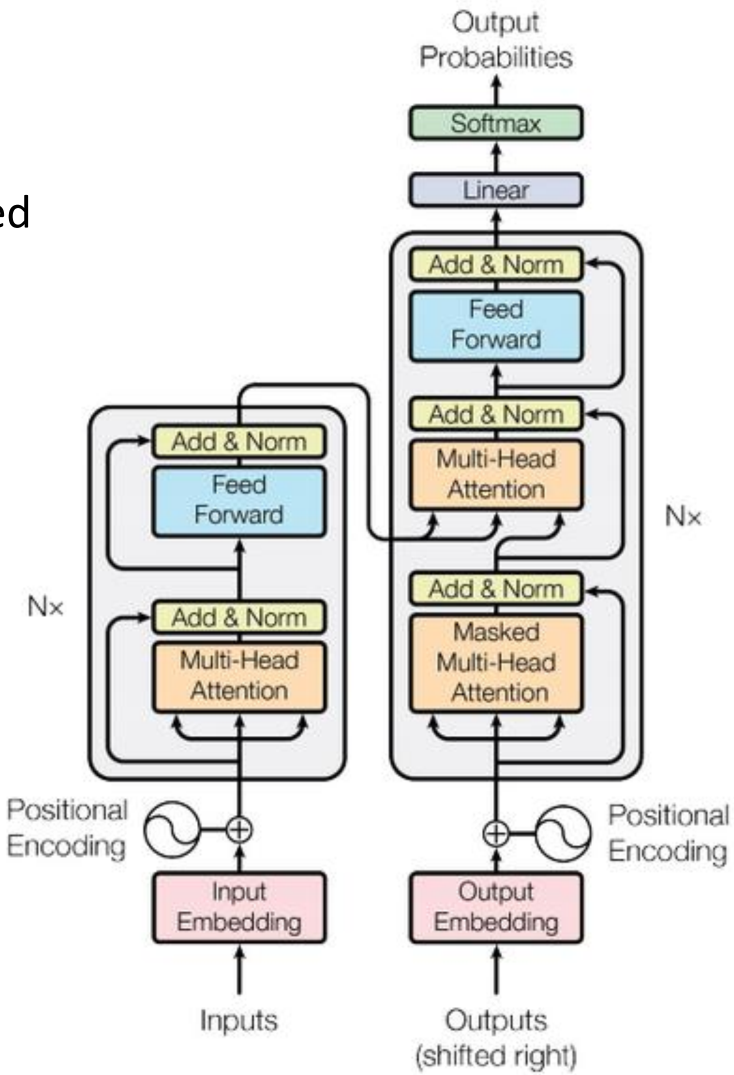
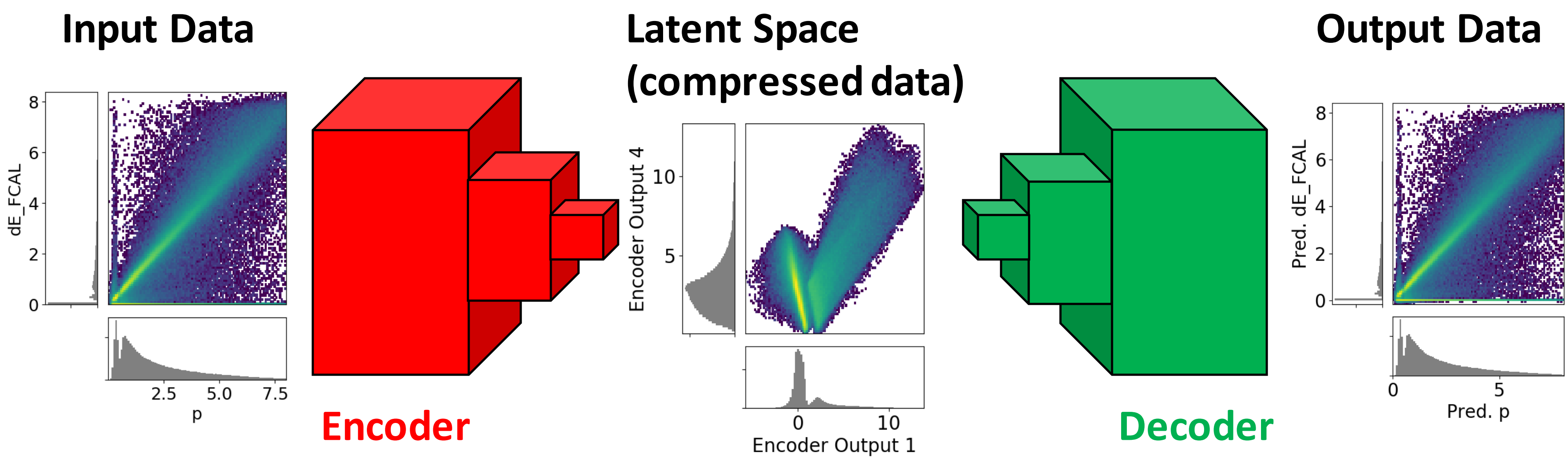


Image taken from [tensorflow webpage]()



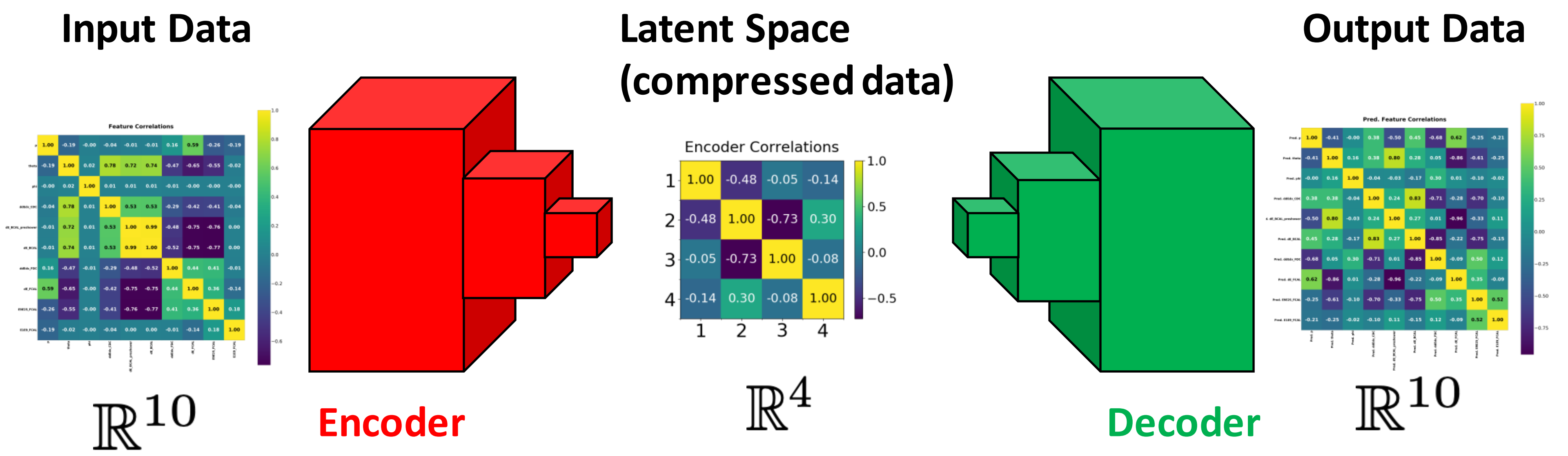
Encoder **Decoder**

Autoencoder Networks



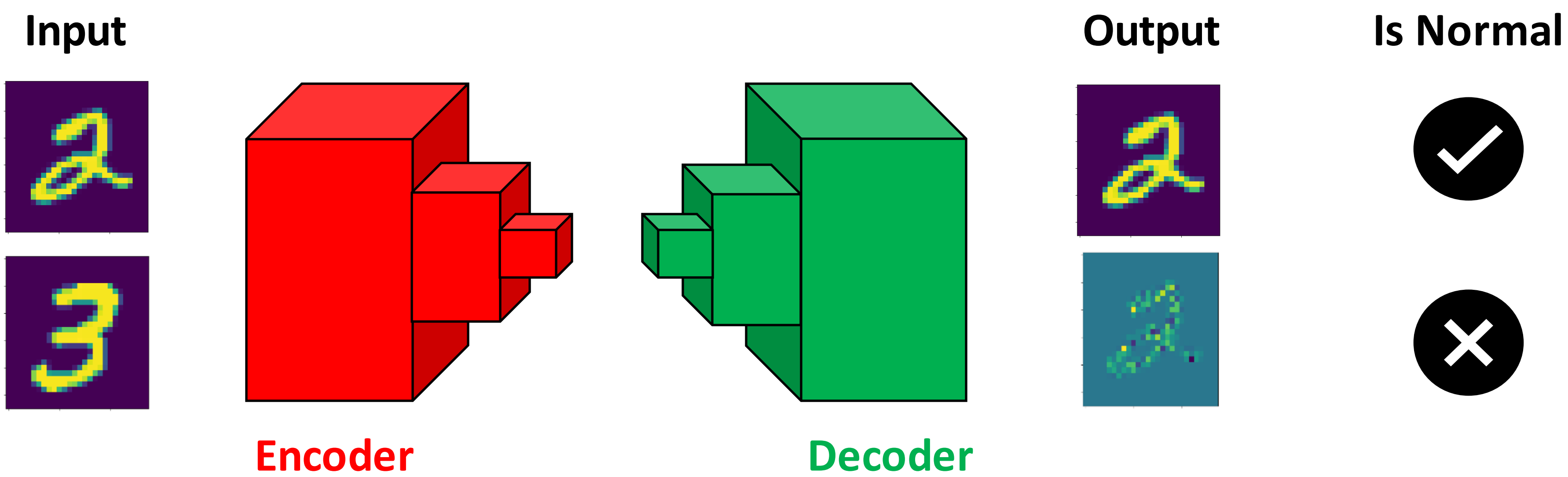
- **Encoder Network:** Compress input data to latent space
- **Decoder Network:** Translate latent space data back to original features space
- **Training:** Output Data = Decoder[Encoder(Input Data)] = Input Data
- **Typical loss functions:** Mean Squared Error, Mean Absolute Error, Huber
- Networks can be dense, convolutional, recurrent,…
- <u>Shown above:</u> Dense autoencoder on GlueX simulated lepton tracks

Autoencoder Networks



- **Encoder Network:** Compress input data to latent space
- **Decoder Network:** Translate latent space data back to original features space
- **Training:** Output Data = Decoder[Encoder(Input Data)] = Input Data
- **Typical loss functions:** Mean Squared Error, Mean Absolute Error, Huber
- Networks can be dense, convolutional, recurrent,…
- <u>Shown above:</u> Dense autoencoder on GlueX simulated lepton tracks

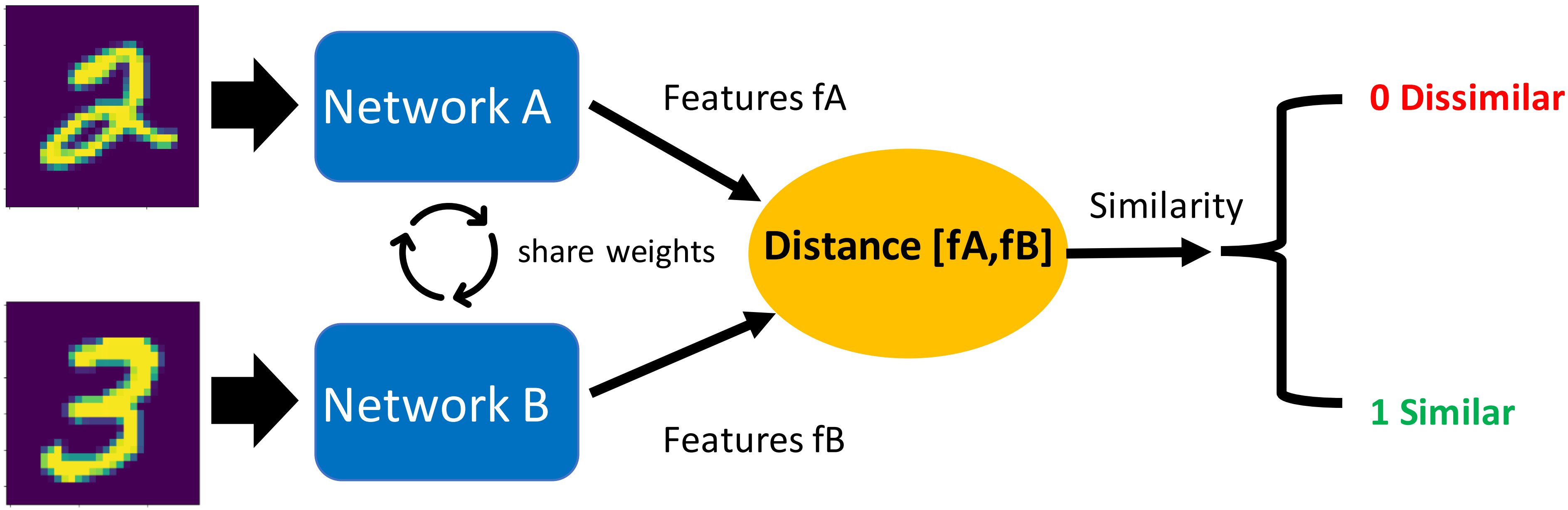
Autoencoders and Anomaly Detection



- Train Autoencoder: 2 = Decoder[Encoder(2)]
- **Anomaly: Input ≠ Output**
- Use loss as anomaly score: Loss ~ [Input - Output]

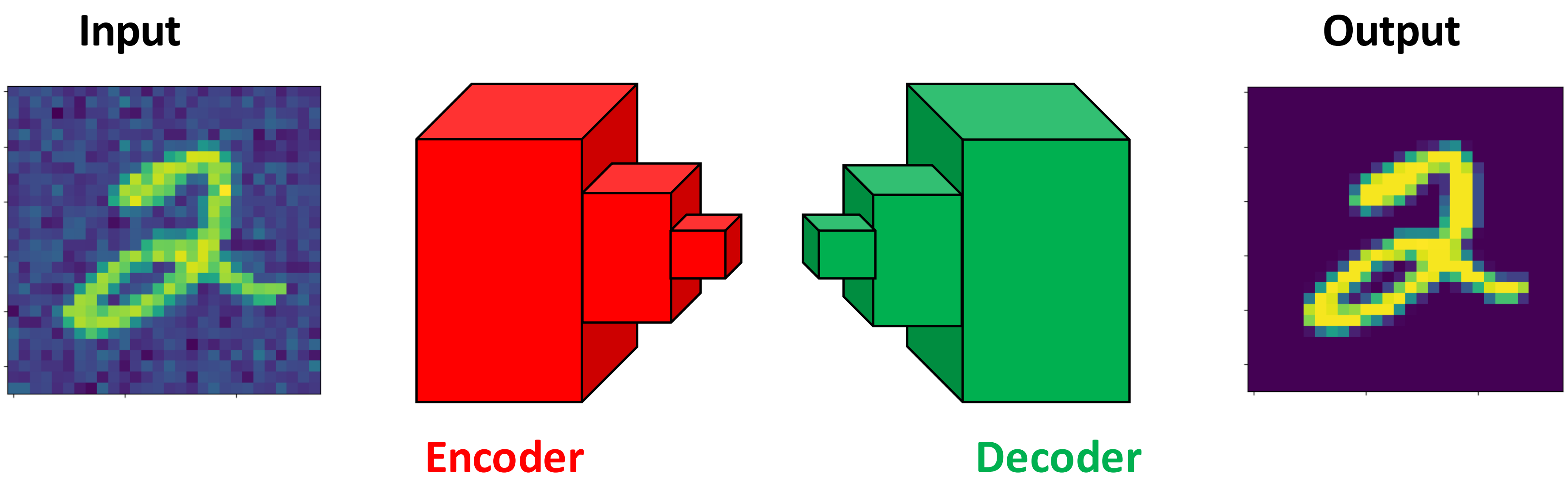
Input	Output = Decoder[Encoder(Input)]
2	2
Not a 2	?

Anomaly Detection with Siamese Models



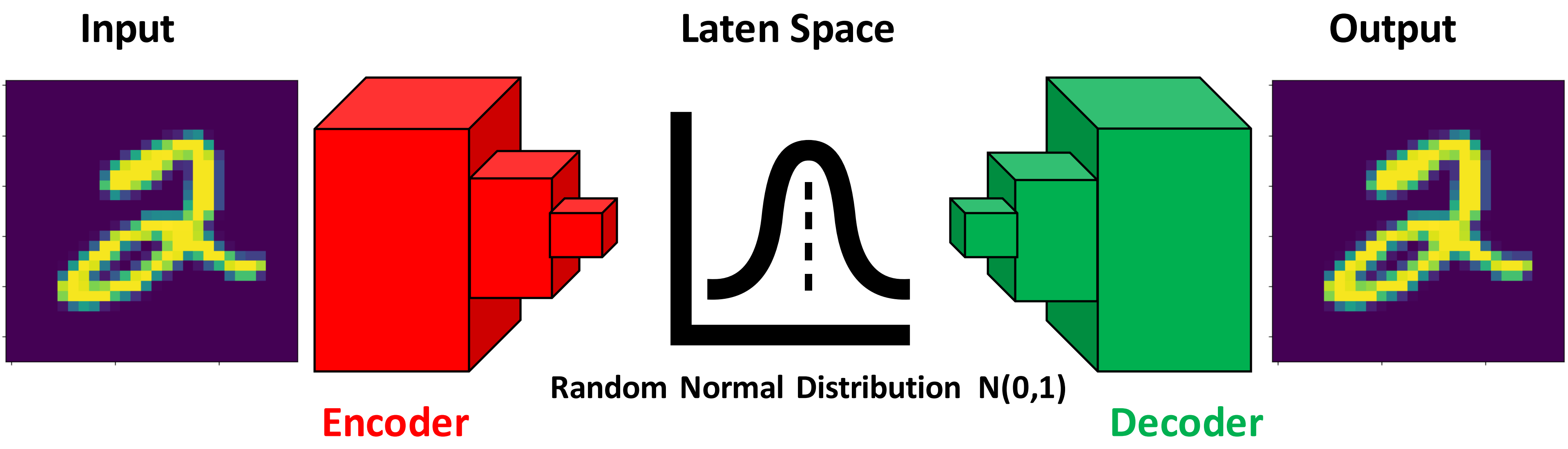
- Alternative to autoencoder networks
- Superior in handling unseen anomalies
- Siamese model focusses on similarity, rather than explicit classification
- Network A/B can be convolutional, dense,...
- **At JLab:** Anomality detection in particle accelerators (Kishan Rajput et al.)

Denoising with Autoencoders



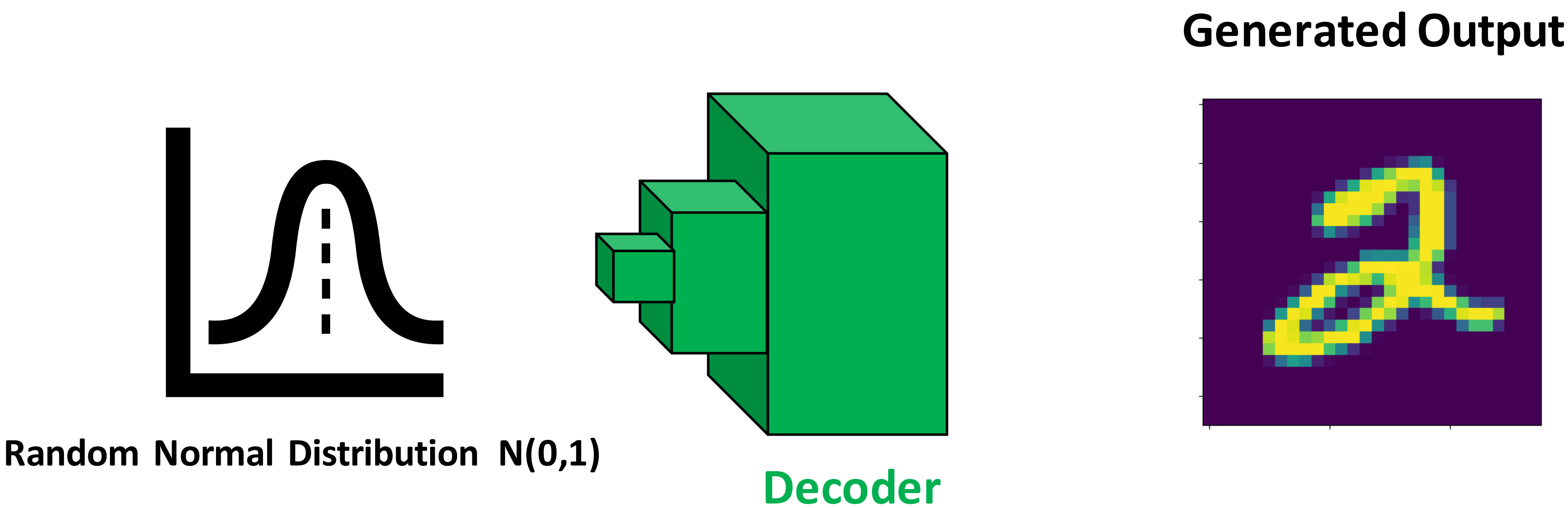
- Train Autoencoder: Image = **Decoder**[**Encoder**(Image + Noise)]
- Autoencoder translates noisy data to noise free data
- There is much more to Autoencoders --> See talk: **"Machine Learning for Nuclear Physics: Lecture 3"**, Thu. 06/06/2024, Gagik Gavalian at el.

Variational Autoencoder (VAE)



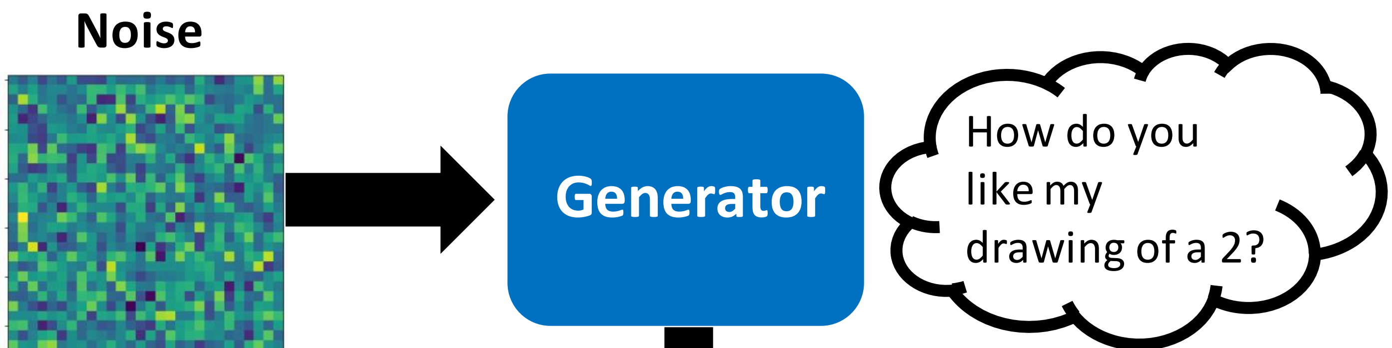
- Like autoencoder, but constrain latent space to follow a normal distribution
- Once trained, generate data from decoder: **Decoder**[N(0,1)]
- Generative model
- Used as anomaly detector
- **At JLab:** Mainly used for PID and / or physics data generation

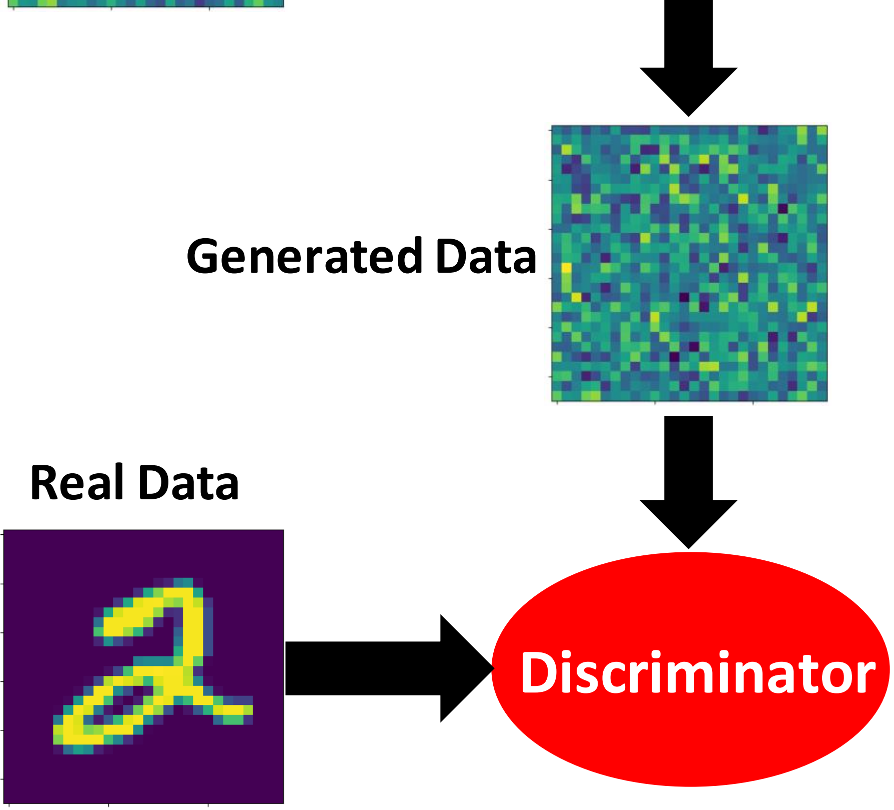
Variational Autoencoder (VAE)



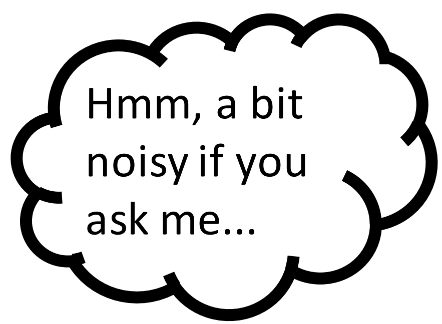
- Like autoencoder, but constrain latent space to follow a normal distribution
- Once trained, generate data from decoder: **Decoder**[N(0,1)]
- Generative model
- Used as anomaly detector
- **At JLab:** Mainly used for PID and / or physics data generation

Generative Adverserial Network (GAN) (1)



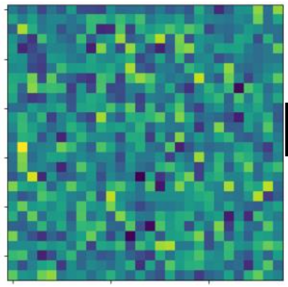


- Generative Model
- **Generator Network:** Generates data and tries to fool the discriminator
- **Discriminator Network:** Tries to tell real and generated data apart
- Train both networks competitively



Generative Adverserial Network (GAN) (1)

Noise

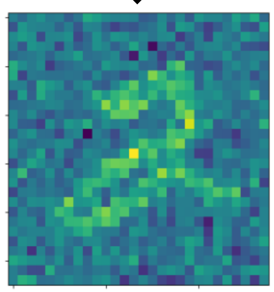


Generator

and now?

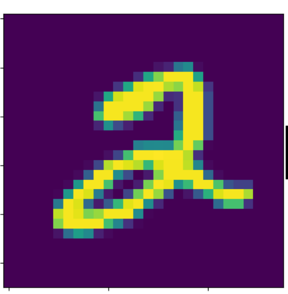
A few training iterations later...

Generated Data



- Generative Model
- **Generator Network:** Generates data and tries to fool the discriminator
- **Discriminator Network:** Tries to tell real and generated data apart
- Train both networks competitively

Real Data

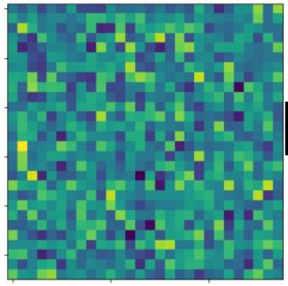


Discriminator

No, not really...

Generative Adverserial Network (GAN) (1)

Noise

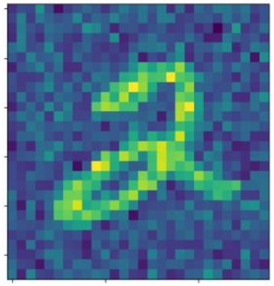


A few more training iterations later...

Generator

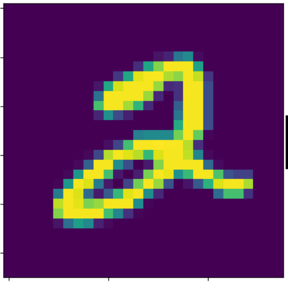
How about now?

Generated Data



- Generative Model
- **Generator Network:** Generates data and tries to fool the discriminator
- **Discriminator Network:** Tries to tell real and generated data apart
- Train both networks competitively

Real Data

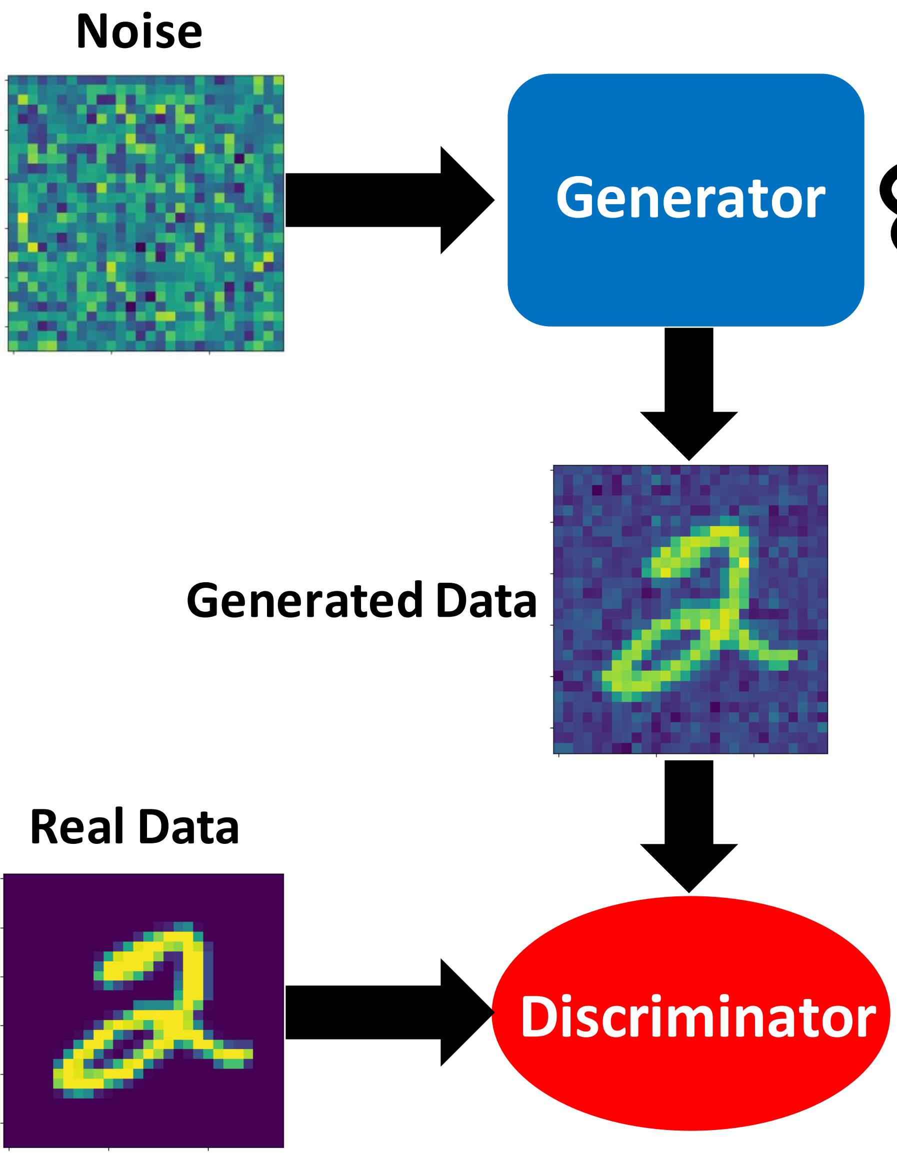


Discriminator

I still don't feel it...

Generative Adverserial Network (GAN) (1)

Many more training iterations later…

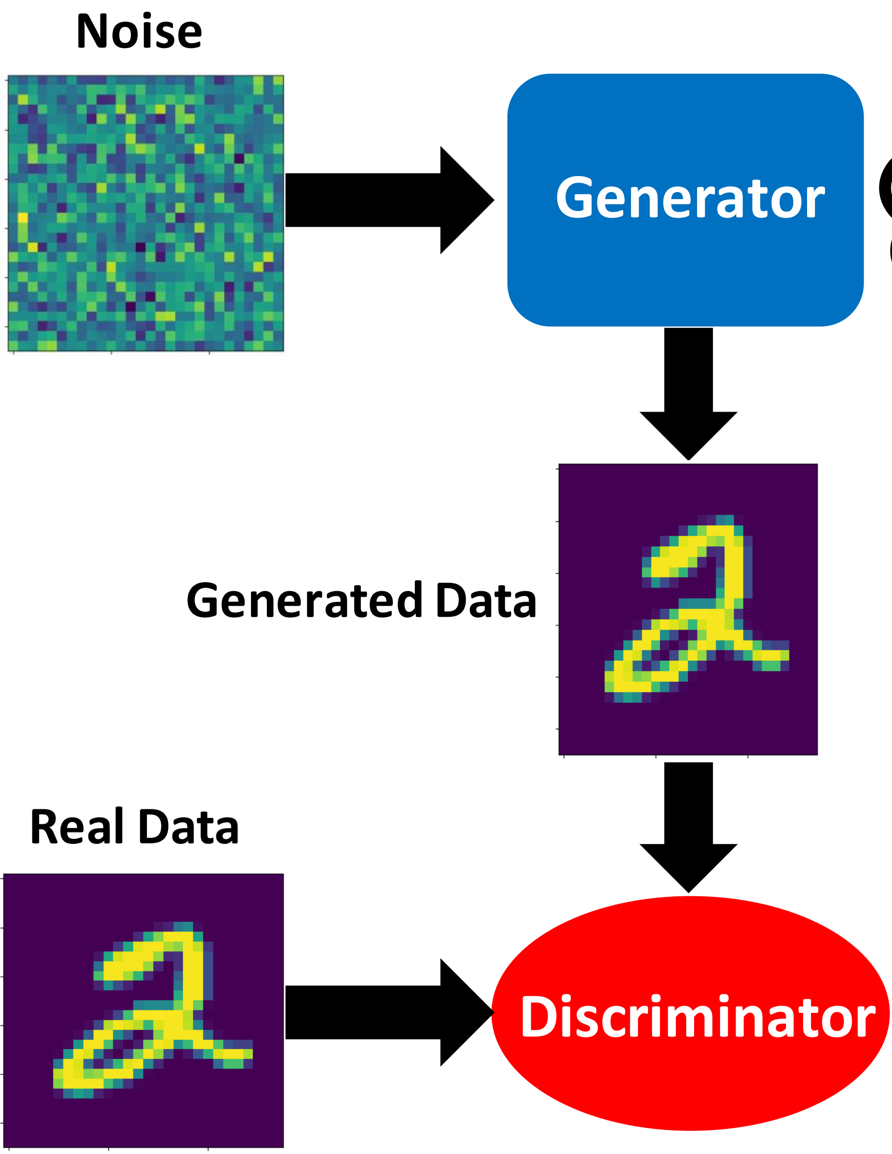


What about this one?

- Generative Model
- **Generator Network:** Generates data and tries to fool the discriminator
- **Discriminator Network:** Tries to tell real and generated data apart
- Train both networks competitively

Reminds me of a 2…

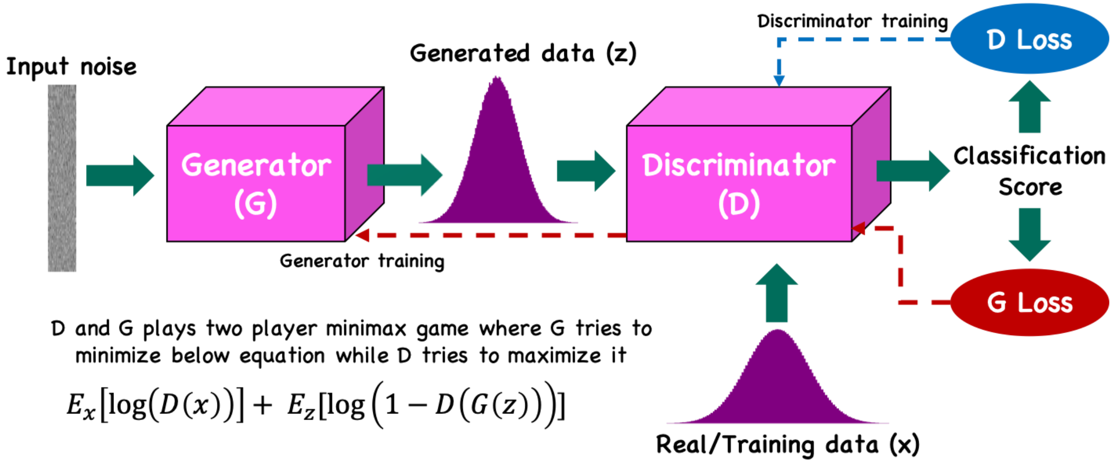
Generative Adverserial Network (GAN) (1)

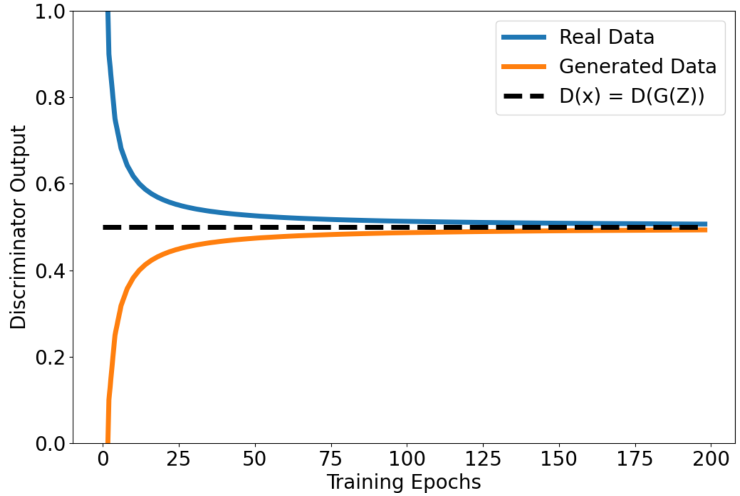


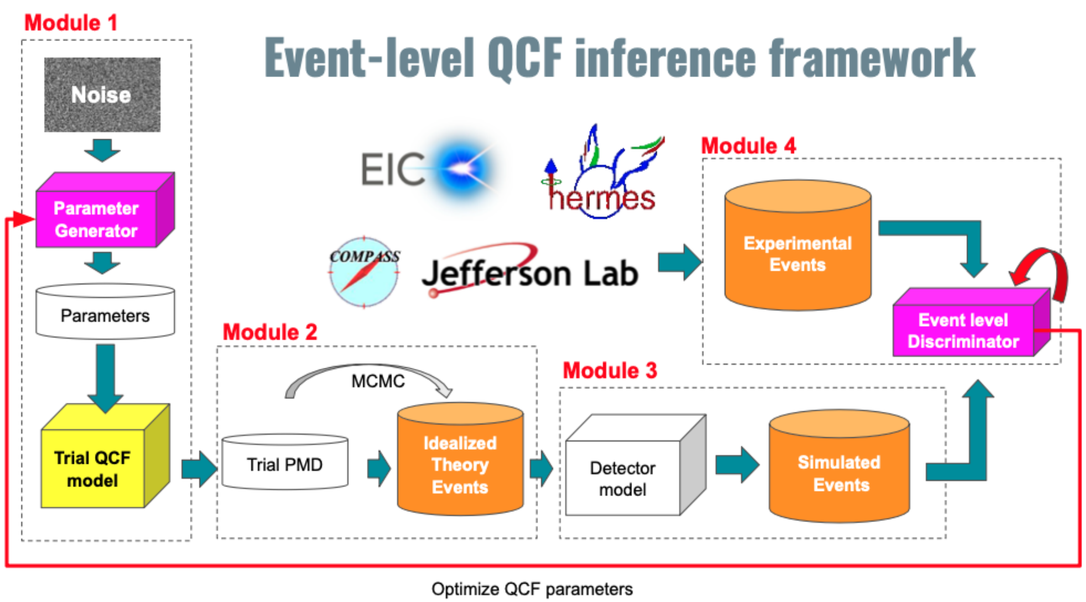
GAN training finished

- Generative Model
- **Generator Network:** Generates data and tries to fool the discriminator
- **Discriminator Network:** Tries to tell real and generated data apart
- Train both networks competitively

Generative Adverserial Network (GAN) (2)







- GAN training is successful <--> Discriminator output similar for real / generated data
- **At JLab:** Use GAN to solve inverse nuclear physics problems

Generative Models – A brief Overview

	Anomaly Detection	Data Generation	Training Difficulty	Image Quality & Diversity
Generative Adverserial Network (GAN)	⊘	✓		★★★★
Variational Autoencoder (VAE)	✓	✓		★★★
Diffusion Models	✓	✓		★★★★★
Energy Based Models (EBM)	✓	✓		★★★★★

Methods and Tools

- Uncertainty Quantification
- Distributed Training
- Hyper Parameter Optimization (HPO)
- Software packages

Uncertainty Quantification

A single model, without specific modifications, has no uncertainty!

What is often quoted: mean squared error, confusion matrix,.. ROC-Curve, …
- Deduced from data with known truth (or something close to it)
- No applicable to single prediction

Example: Mean Squared Error

$$\mathrm{MSE}= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2,\ y_i : \text{Known truth for } x_i,\ \hat{y}_i = \mathrm{model}(x_i)$$

==> Gives an idea how good / bad the model performs on the entire data set

$$\hat{y}_i = \mathrm{model}(x_i) \text{ holds NO information about uncertainty of } \hat{y}_i$$

Uncertainty Quantification

A single model, without specific modifications, has no uncertainty!

What is often quoted: mean squared error, confusion matrix,.. ROC-Curve, …

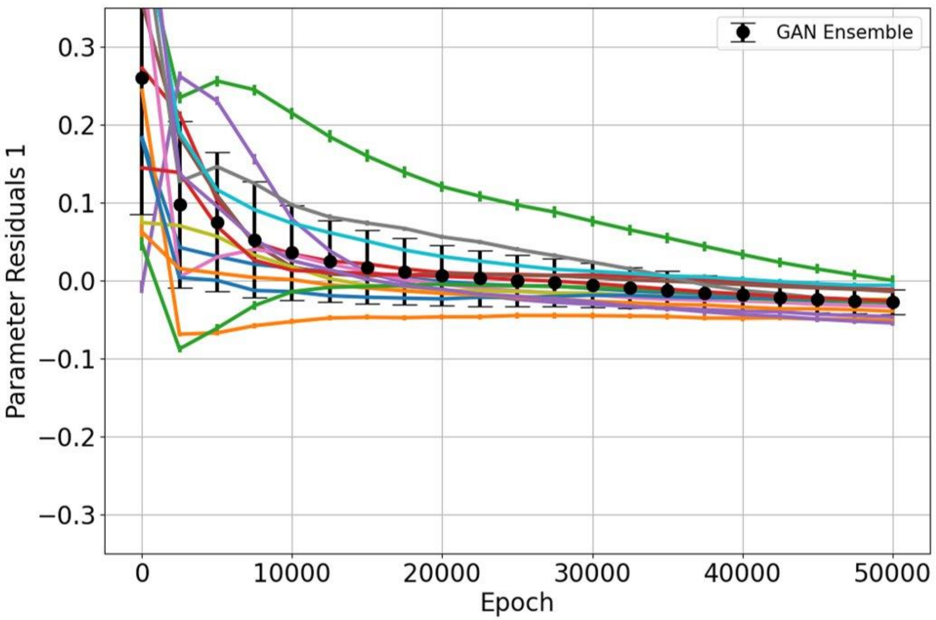
- Deduced from data with known truth (or something close to it)
- No applicable to single prediction

Common Techniques (just 2 out of many techniques)

1.) Ensemble: M models, independently trained on same data, but different initialization for internal parameters

$$\hat{y}_i = \frac{1}{M}\sum_{k=1}^{M} \mathrm{model}_k(x_i)$$

$$\sigma_i = \sqrt{\frac{1}{M}\sum_{k=1}^{M} (\mathrm{model}_k(x_i) - \hat{y}_i)^2}$$



Jefferson Lab

Uncertainty Quantification

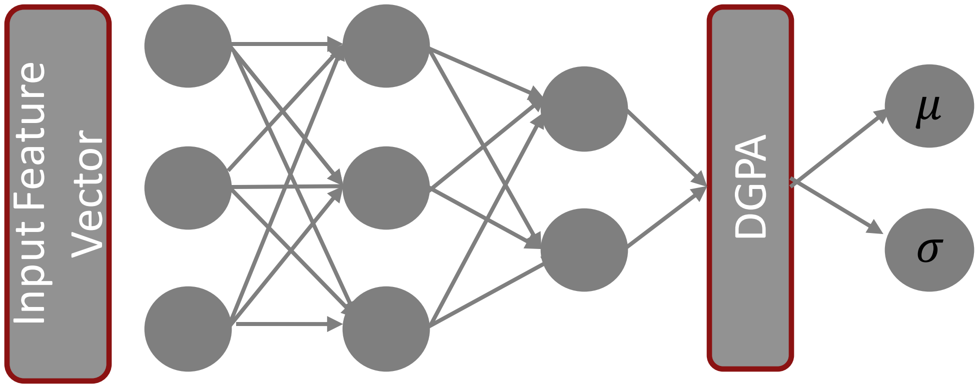
A single model, without specific modifications, has no uncertainty!

What is often quoted: mean squared error, confusion matrix,.. ROC-Curve, …

- Deduced from data with known truth (or something close to it)
- No applicable to single prediction

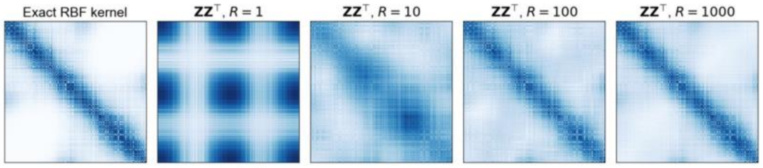
Common Techniques (just 2 out of many techniques)

2.) Deep Gaussian Process Approximation (DGPA): Approximate kernel k(x,y) to reduce computational cost. Model directly predicts uncertainty.



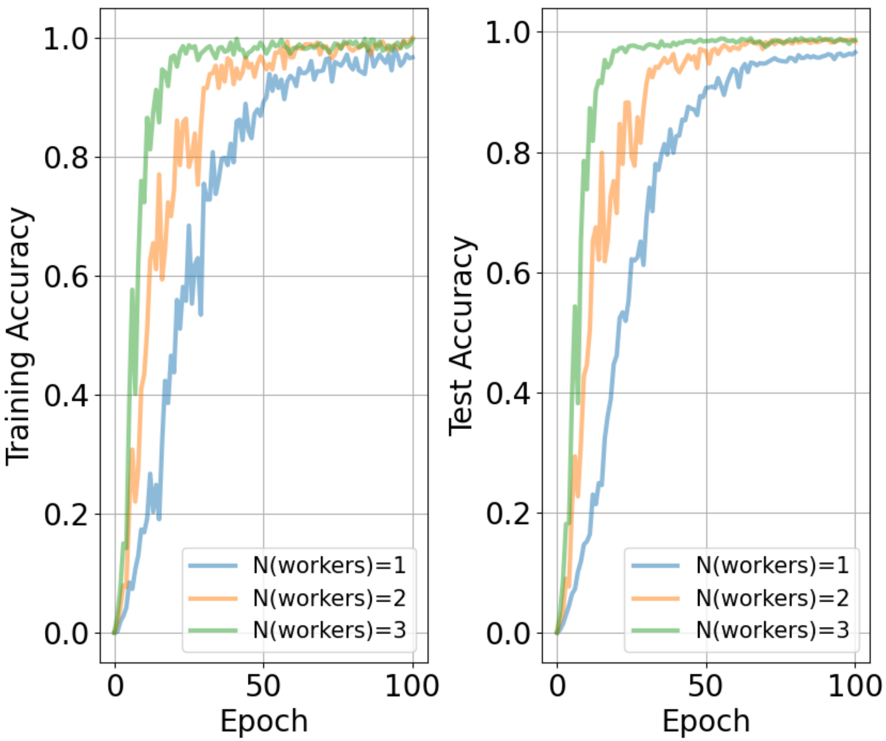
Allows to formulate uncertainties

$$\boxed{k(x,y)} \approx z^T(x)z(y)$$



Distributed Training and do I need it?

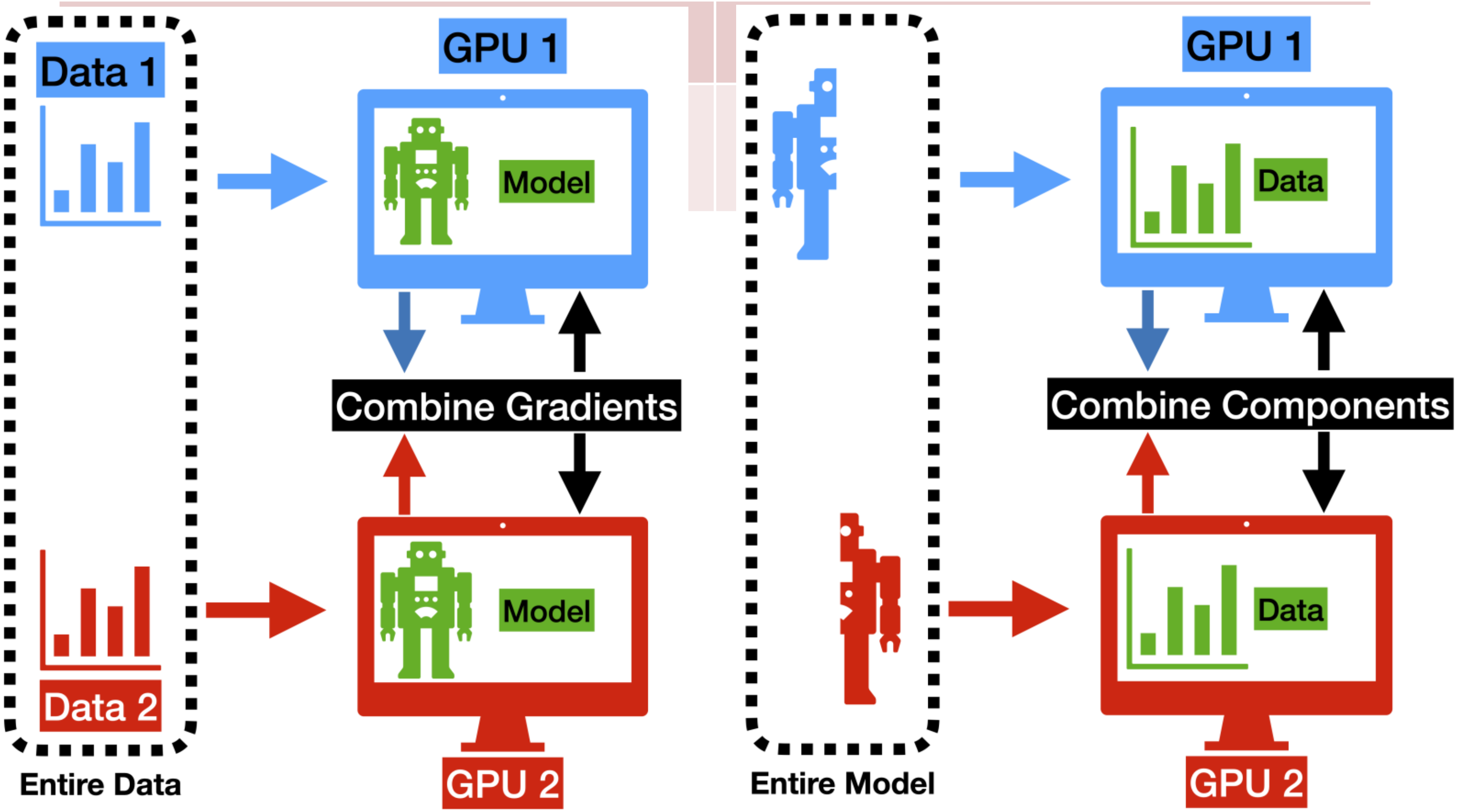
Data Format	Model Complexity (Number of trainable Parameters)
Digits	~1k - 100k
Images & Videos	~100k - 10000k
Text & Language	>> 10000k



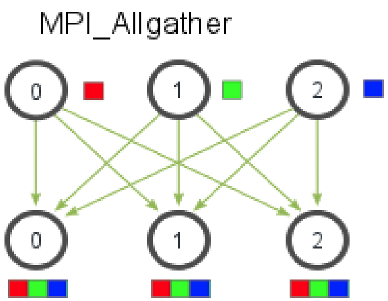
- Depending on the model complexity, a single GPU is not suitable for training (Unless you are fine waiting months for your publication results)
- To speed up training time: Run your analysis across multiple GPUs
- **Scaling:** Total training time / Model performance vs. Number of GPUs
- **Example on the left:** MNIST Classifier trained on JLab GPUs, training times nearly identical for all runs

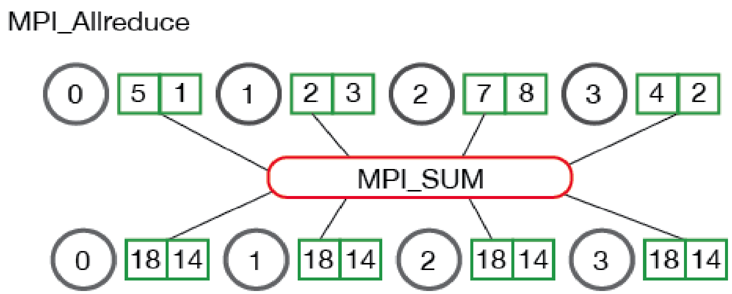
Basic Distributed Training Strategies

Training Method	Explanation
Data Parallel	Shard data across GPUs, each GPU sees full model --> Distribute gradients
Model Parallel	Shard model across GPUs, each GPU sees fraction of the model and full data

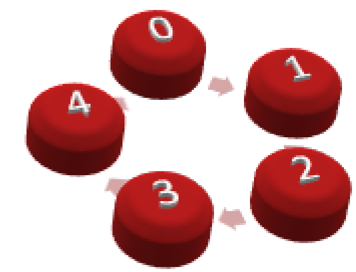


Distributed Training Tools and Methods





(asynchronous) ring all-reduce

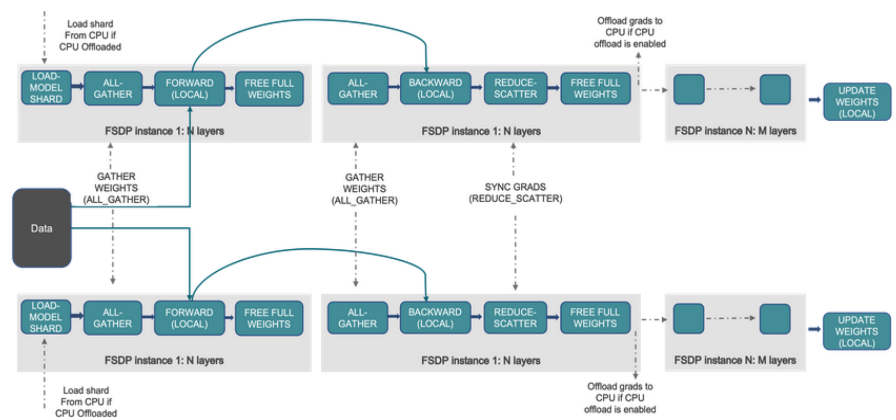


[Horovod]

- Supports many deep learning frameworks (PyTorch, Tensorflow, Keras,…)
- Based on data parallel training

[PyTorch Distributed Training Packages]

- Supports various training strategies: data parallel, model parallel, fully sharded data parallel (FSDP),…
- Works for PyTorch models only



- Asynchronous ring all-reduce is common method to distribute gradients
- Even faster: [double binary trees]

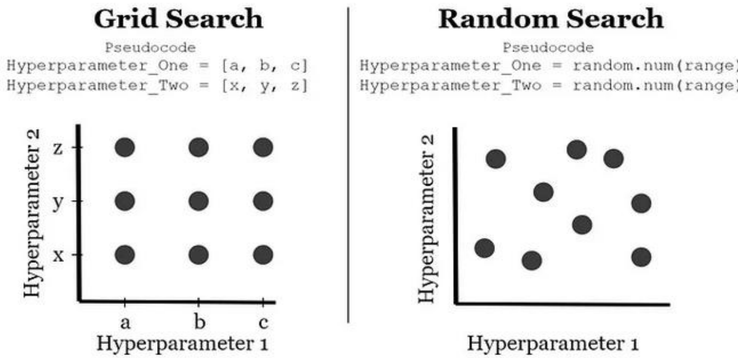
Hyper Parameter Optimization (HPO)

Training

Neural Network Parameters	Neural Network Hyper Parameters
Weights, biases, filters,..	Number of hidden layers, batch size, filter size, activation functions, learning rate, number of learning epochs,...

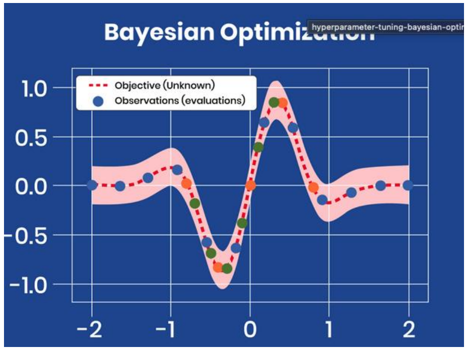
HPO

- Various optimization algorithms on the market
  - Grid search
  - Random search
  - Bayesian optimization



Taken from [Jack Stalforts blog]()

- Many software packages supporting HPO
  - [KerasTuner]()
  - [Optuna]()
  - [Weights & Biases]()
  - [Ray Tune]()
  - ...



Taken from [Juan Navas blog]()

Software Packages

Software Package	Suited for	Language
sciki-learn	Machine learning with off the shelf models; Provides all tools to set up an entire ML workflow	python
tensorflow	Customize deep learning models; Supports variety of diagnostic tools, e.g. tensorboard	python
PyTorch	Customize deep learning models; High flexibiltiy for user to define own training / evaluation routines	python
keras	Customize deep learning models; Supports tensorflow and pytorch; HPO tools	python
ROOT TMVA	Machine learning with off the shelf models + Deep Learning with keras / PyTorch	C / C++ / python

Which one to choose? --> Depends on what you want to do and personal taste...

Summary & Outlook

- Deep learning models in nuclear physics
  - Anomaly detection / classification
  - Tracking & reconstruction
  - Event level analysis
  - Monitoring
  - …

- Practical Tips
  - Try to use a baseline analysis for comparison
  - Use HPO to tune your model
  - Speed up training with distributed strategies
  - Further reading: Andrej Kaparthy blog, distill.pub

- Not covered
  - Reinforcement Learning
  - Fairness and ethnics in AI
  - Continual learning
  - …