Reconstruction Using Machine Learning Technologies

Emmanuel Olaiya

IOP2021

Joint APP, HEPP and NP Conference

15th April 2021

Outline

- What is AI and Machine Learning?
- Why the sudden boom?
- Types of machine learning
- Examples of Neural Networks
    - Multilayered Perceptron (MLP)
    - Convolutional Neural Net (CNN)
    - Graph Neural Net (GNN)
- Hardware that drives machine learning
- Reconstruction in High Energy Physics using machine learning

What is AI/Machine learning

- AI (Artificial Intelligence) is a broad term that includes machine learning. AI is the idea that a machine can be similar to the human brain. Machine learning is a product of AI's evolution
- Machine learning is the science of enabling machines to learn algorithms which are not explicitly programmed
- This talk focuses on machine learning!
- Machine learning has been around for decades. So why is it taking off now?

Why is Machine Learning taking off now

- Feasibility and accessibility
  - Data
    - Data is at the core of machine learning. Recent advances in data capture, data management and data storage have resulted in an exponential increase in data. Also disks are becoming faster, cheaper with larger capacities
  - Hardware
    - Once you have the data you still need a large amount of computing power to do the number crunching. For a long time the required computing power was unavailable.
      - Enter the GPU (Graphics Processing Unit). Invented by Nvidia in 1999, GPUs can speed up significantly the training of neural nets (more about this in later slides)
  - Algorithms
    - With the hardware and data we are now seeing rapid development of sophisticated algorithms

Types of Machine Learning

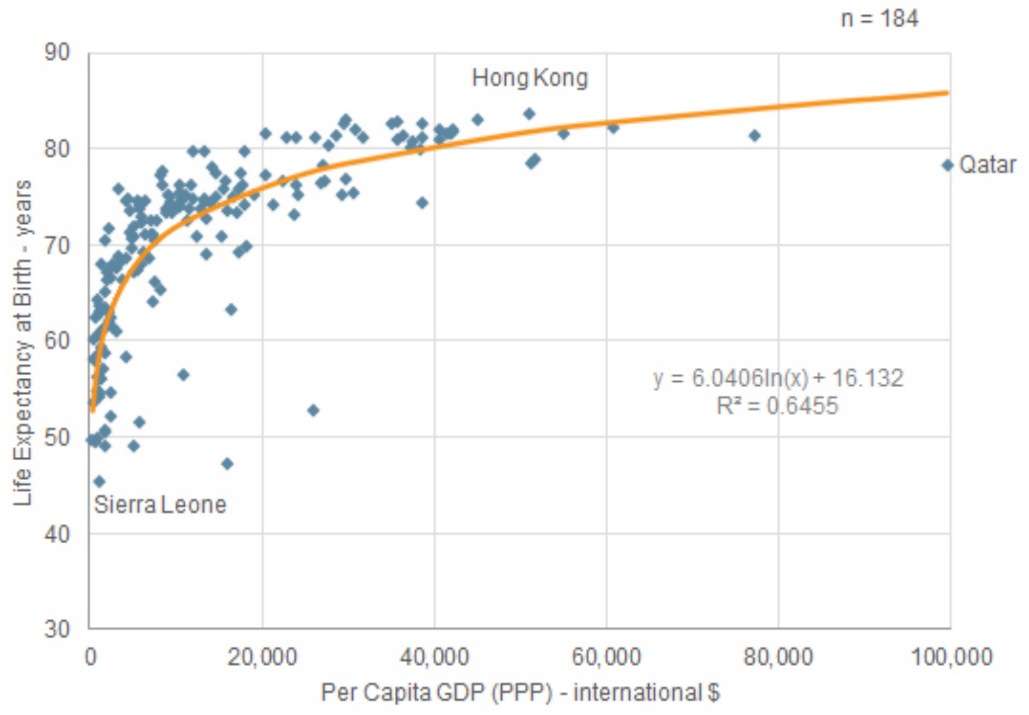
- Consider the types of Machine Learning
  - Supervised v Unsupervised
    - Supervised: the data you train your algorithm with also contains the solutions, called labels
      - Examples:k-Nearest Neighbors, Linear Regression, Logistic Regression, Decision Trees, …
    - Unsupervised: the data you train your algorithm on is unlabeled
      - Examples: K-Means, DBSCAN, t-SNE, ...
  - Batch v Online Learning
    - Batch: your algorithm is incapable of learning incrementally. All the data is used. If more data is acquired updating the algorithm requires retraining and then relaunching. This is offline learning.
    - Online: your algorithm is capable of learning incrementally. It can learn on the fly using new data. Online learning is great for systems that need to adapt or change rapidly eg. stock price modeling

Types of Machine Learning

- Instance v Model-based Learning
  - Instance: the system learns the examples by heart and then generalises to new cases eg. t-SNE
  - Model-based: build a model of these examples and the use that model to make predictions eg. Linear Regression, etc.

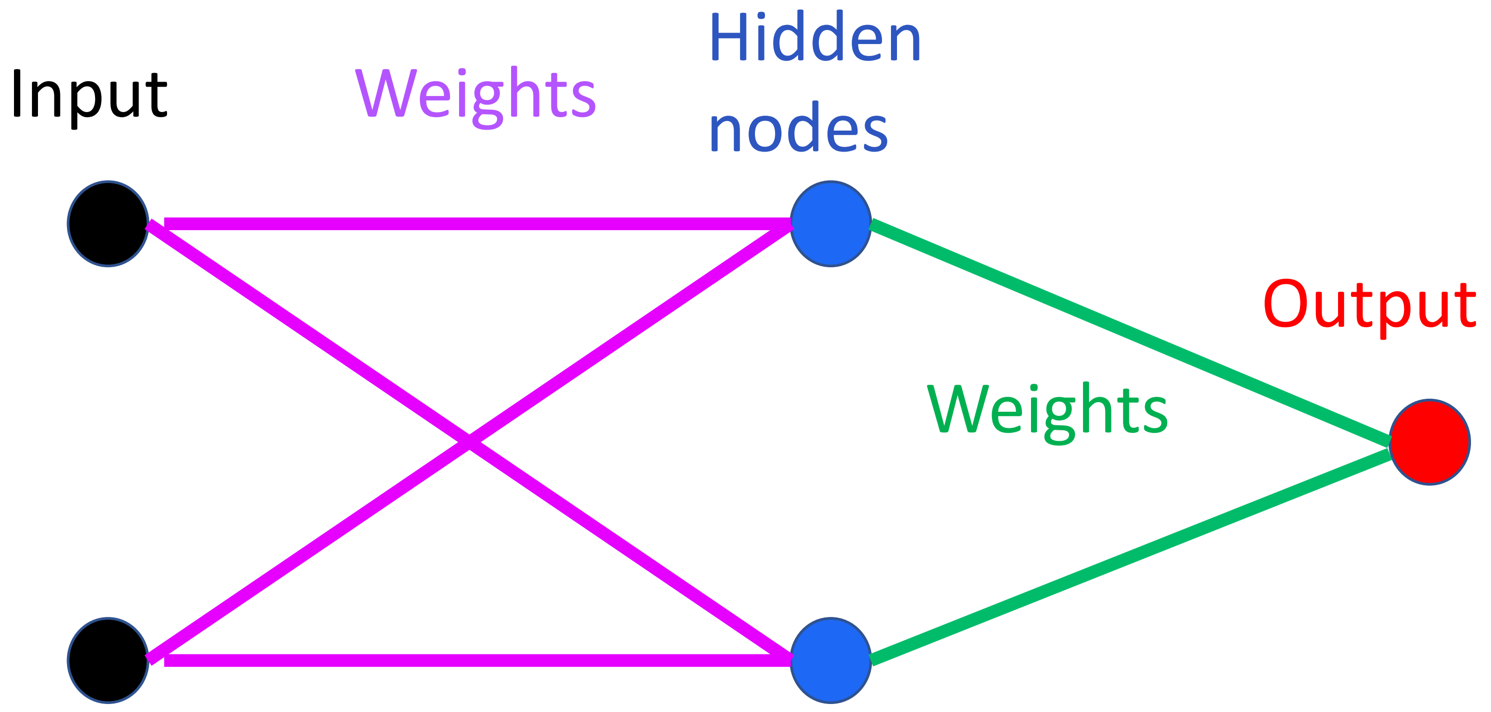
Model-based learning: Data of life expectancy v GDP. Fit a function to the data. Function can be used to make predictions

- Neural nets are model based
- Let's look at the architecture of Deep Neural Nets (DNNs) by taking Multilayered Perceptrons (MLPs) and Convolutional Neural Nets (CNNs) and Graph Neural Nets (GNNs) as examples



A Simple Neural Network

- A Multilayer Perceptron (MLP) neural net is inspired by the biological neuron
- A simple neural net consist of
  - Inputs
  - Weights and bias
  - Nodes
    - Weights are summed at the nodes
    - Activation function is applied after the weights are summed
  - Output
- The more nodes and layers, the more sophisticated your neural net can be. The additional layers are what makes the neural net "deep"

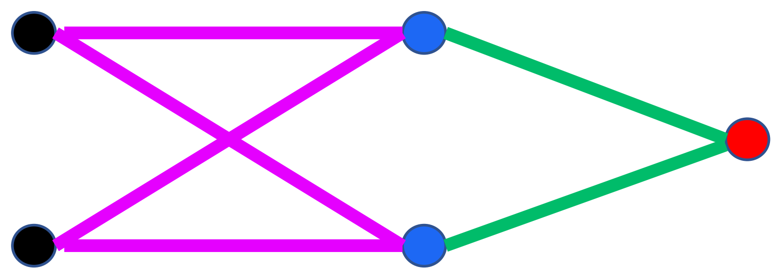


Neural Network f(x)

- A neural network is not a black box!
- It is a function that can be used to model your data!

$f_{NN} = \sigma(b_2 + W_2\sigma(b_1 + W_1x))$

Multilayered Perceptron



$f_{NN} = \sigma(b_2 + W_2\sigma(b_1 + W_1 x))$

$$f_{NN} = \sigma_2\left(b_{2[1,1]} + \begin{bmatrix} W_{2[1,1]} & W_{2[1,2]} \end{bmatrix}\sigma_1\left(\begin{bmatrix} b_{1[1,1]} \\ b_{1[2,1]} \end{bmatrix} + \begin{bmatrix} W_{1[1,1]} & W_{1[1,2]} \\ W_{1[2,1]} & W_{1[2,2]} \end{bmatrix}\begin{bmatrix} x_{1,1} \\ x_{2,1} \end{bmatrix}\right)\right)$$

Input: $x = \begin{bmatrix} x_{1,1} \\ x_{2,1} \end{bmatrix}$

Weight: $W_1 = \begin{bmatrix} W_{1[1,1]} & W_{1[1,2]} \\ W_{1[2,1]} & W_{1[2,1]} \end{bmatrix}$

Bias: $b_1 = \begin{bmatrix} b_{1[1,1]} \\ b_{1[2,1]} \end{bmatrix}$

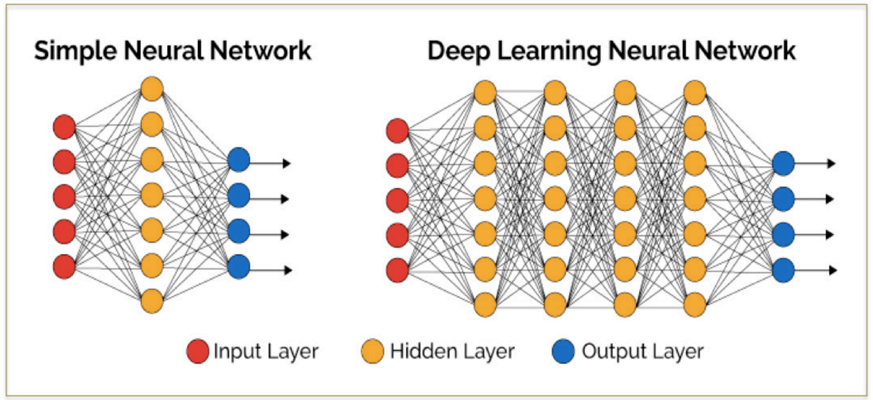
Activation (classification): $\sigma(x) = \tanh(x)$, Sigmoid $\left({}^{1}/_{1+e^{-x}}\right)$

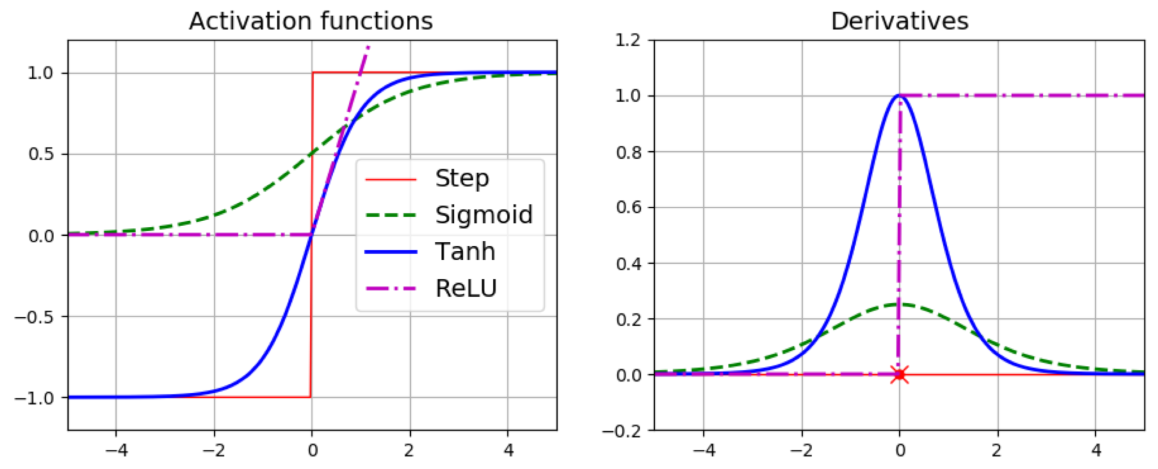
Activation (regression):RELU

Activation functions are preferably differentiable!

- Train the Neural net on data to obtain optimal parameters for the weights.
  - Will mention training later!

- More layers and node can make neural net more sophisticated

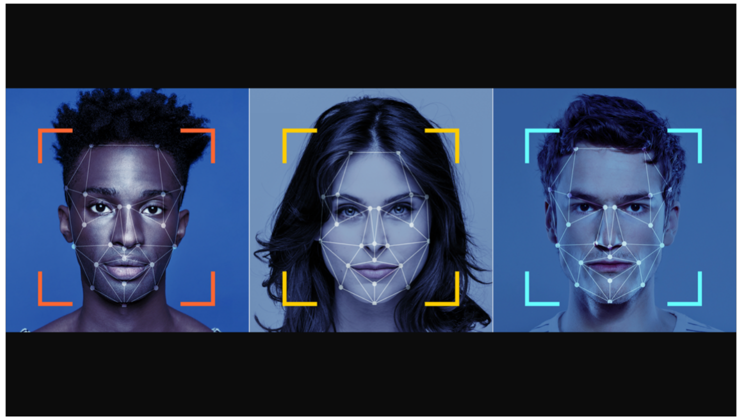




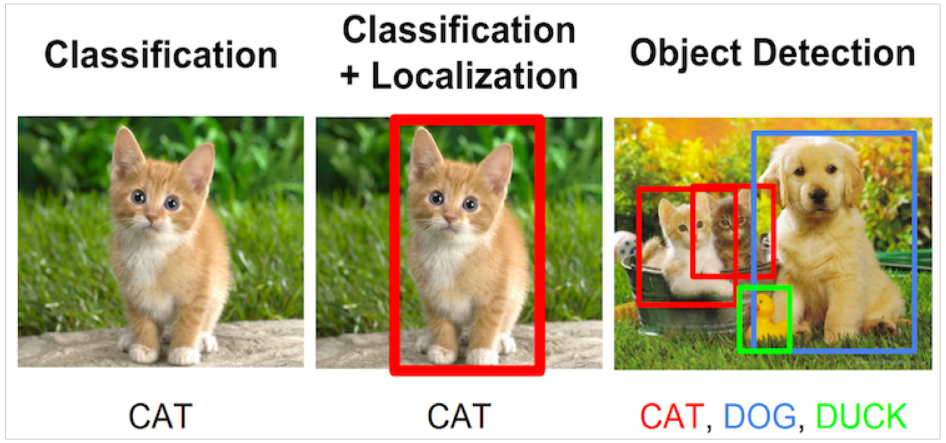
Convolutional Neural Net (CNN)

- A Convolutional Neural Net is a class of Neural Network mainly applied to analysing visual images
- We may be familiar with their use with

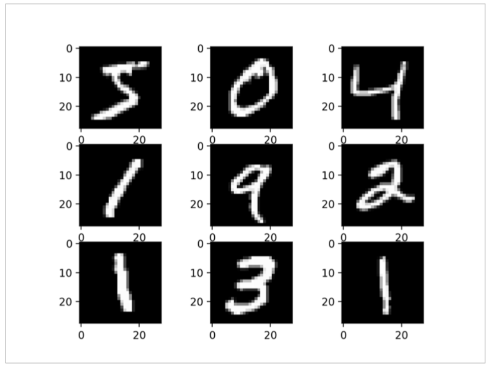
- Face recognition



- Object detection

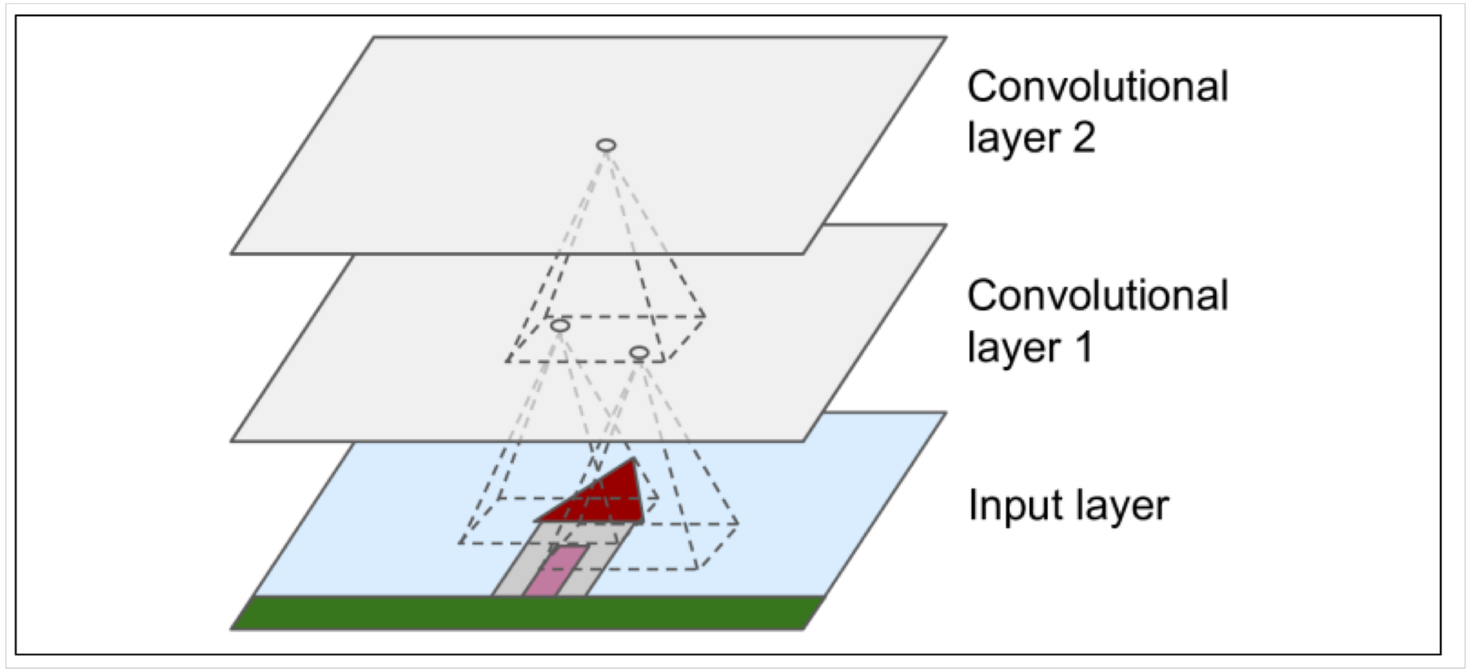


- Identifying numbers/text in images



Convolutional Neural Net (CNN)

- How do CNNs work?

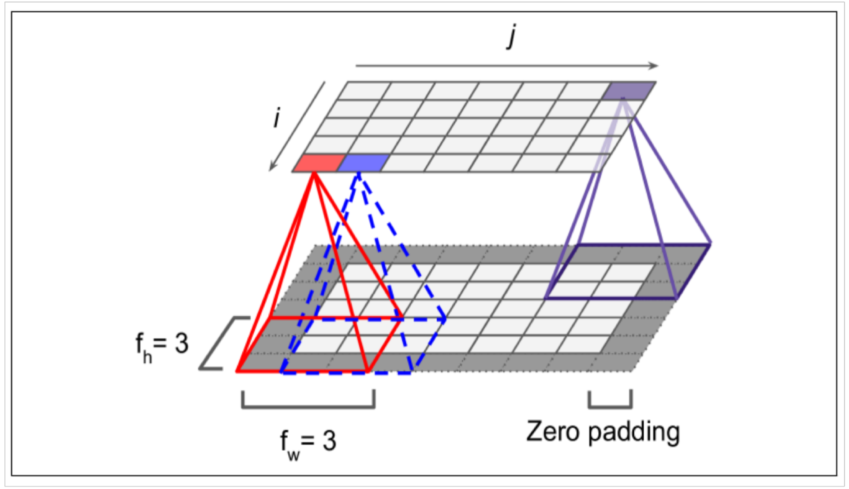


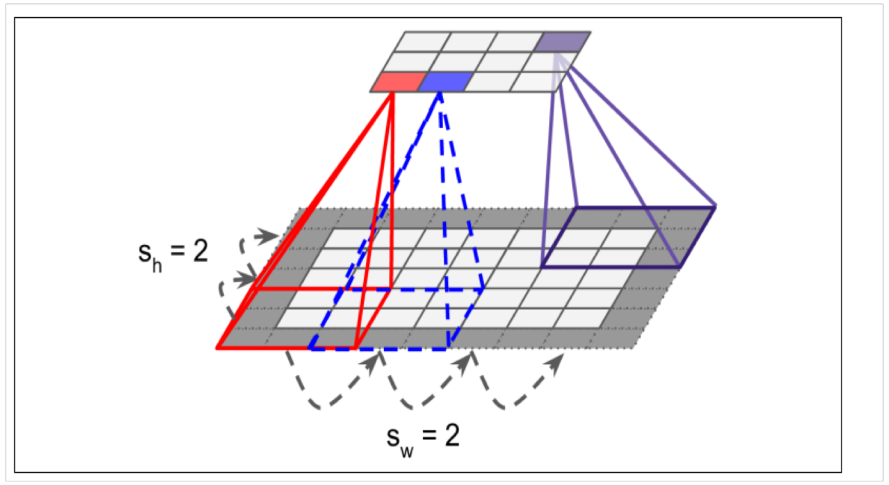
- An n dimension filter (n=2 for the image above) scans across your data/image taking the product. The output forms the first convolutional layer. This process is repeated with different filters on the first convolutional layer to output a second convolutional layer. We can proceed producing as many convolutional layers as we see fit

- What is the point of this?
  - Reducing the model's complexity

Filters

- How do filters work?
  - Converting low level feature into higher level features





- With zero padding (adding zeros around the layer) and a step size of 1 the next convolutional layer will same size. Without zero padding the next convolutional layer is reduced in size

- You can also specify the step size (stride) of the scan. Increasing the step size will reduce the size the next convolutional layer

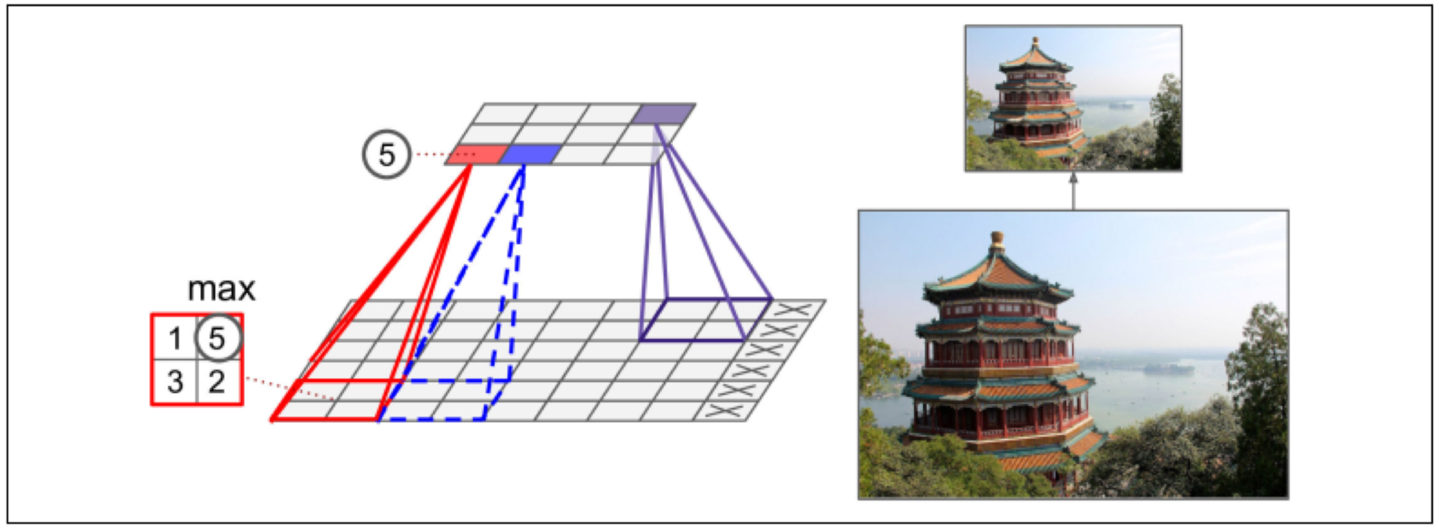
- The filters are the weights and the contents of the convolutional layers are referred to as the neurons. Unlike in a MLP you have a set of weights (filter) that is applied to all neurons in the layer. The filter values are set during training

- You can work with a less weights compared to a MLP

Pooling

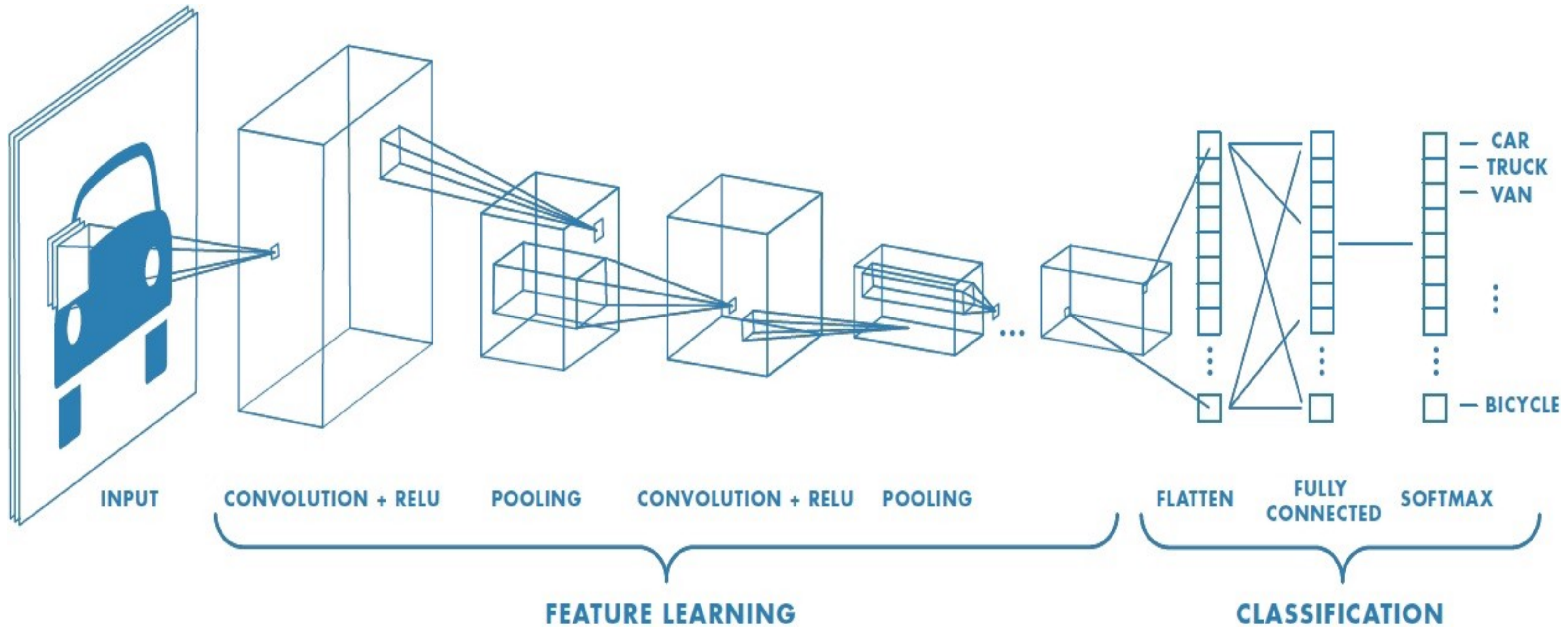
- You can reduce the load even further by pooling
  - This reduces the computational load, memory usage
  - The principle is the same as a filter. You specify a field and scan over the layer applying pooling and output the result to another layer
  - There no weights for pooling. For example you have:
    - Max pooling: Output the maximum value
    - Ave pool: Output the average value

Max pooling example



CNNs Putting it all together

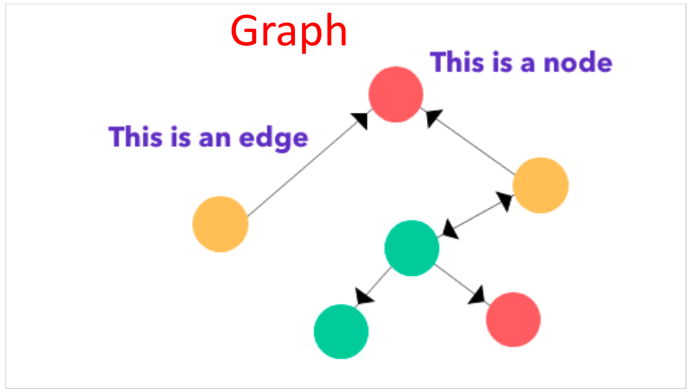
- The last layer is then fed to a MLP and trained on data where the output can be used for regression or classification

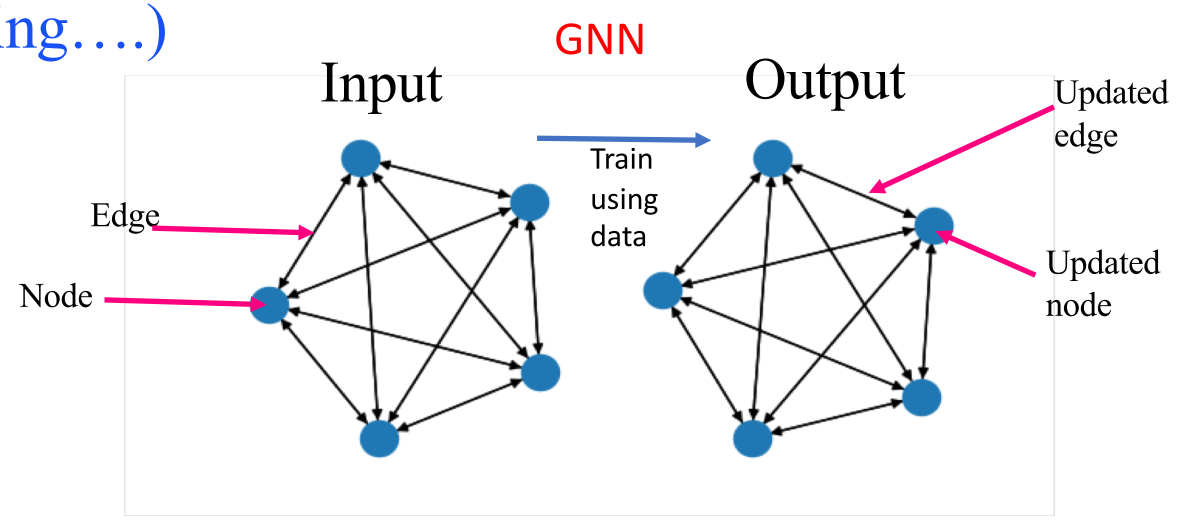


- Apply filters to the RGB components of the image to produce the 1st convolutional layer
- Apply pooling to the 1st convolutional layer to reduce the layer size
- Apply filters to the pooling output to produce the 2nd convolutional layer
- Apply pooling and then feed the output to a MLP with outputs for classification

Graph Neural Nets (GNNs)

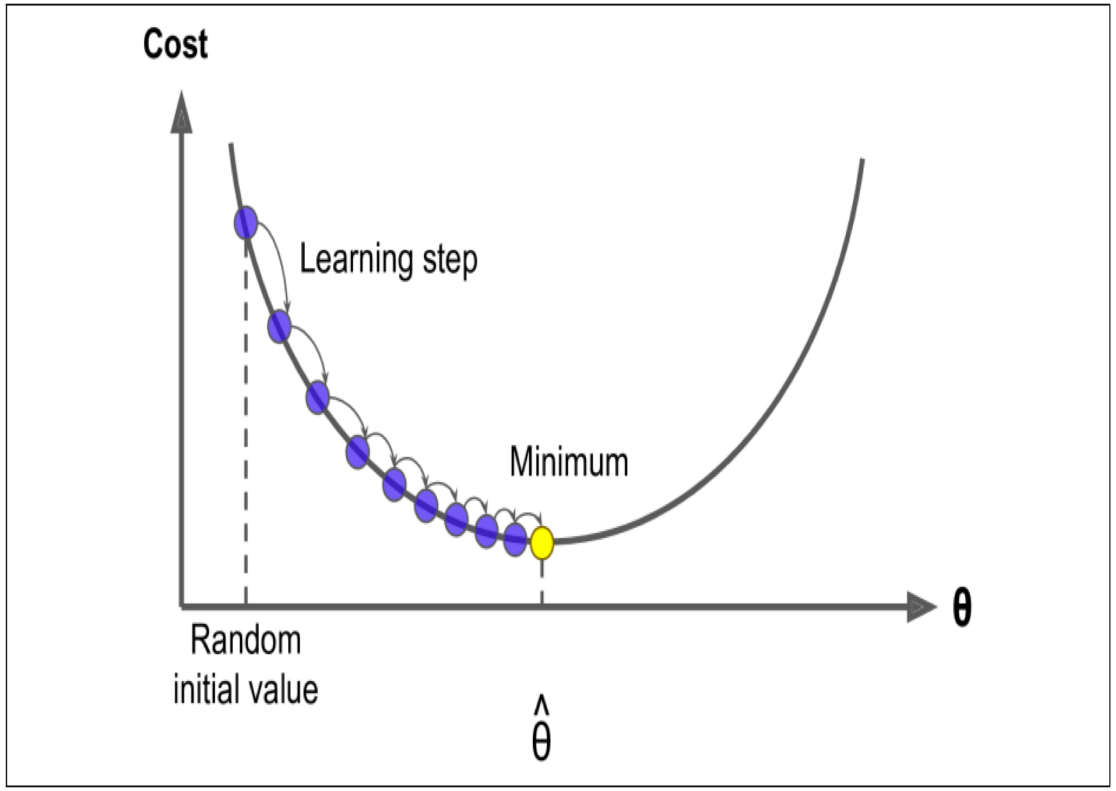
- What is a GNN?
- First a graph is a structure comprising of nodes and edges
    - All nodes occupy arbitrary positions
- Each node has a set of features defining it
- Edges may connect nodes with similar features
- A GNN takes a graph as an input, and information is passed along edges and aggregated on the nodes. A graph is returned as an output
- Graphs are good for problems that can be described via their entities and their relations
- GNNs are growing in popularity and we are seeing them being used to tackle many reconstruction problems (vertexing, tracking….)





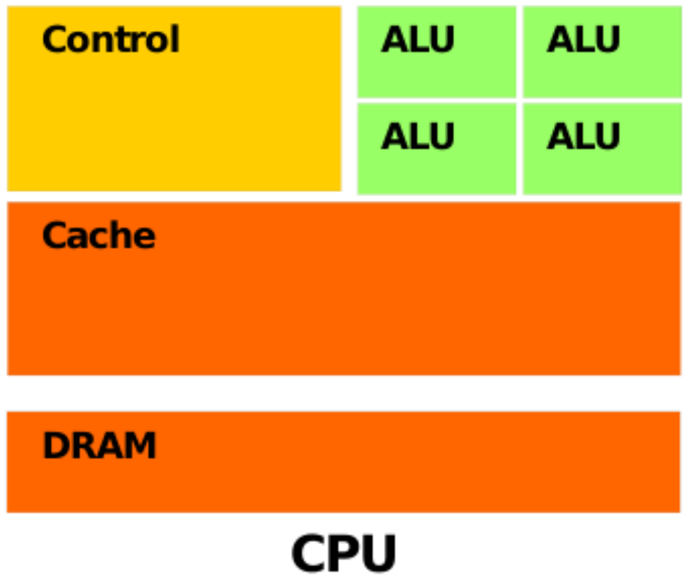
Training Neural Nets

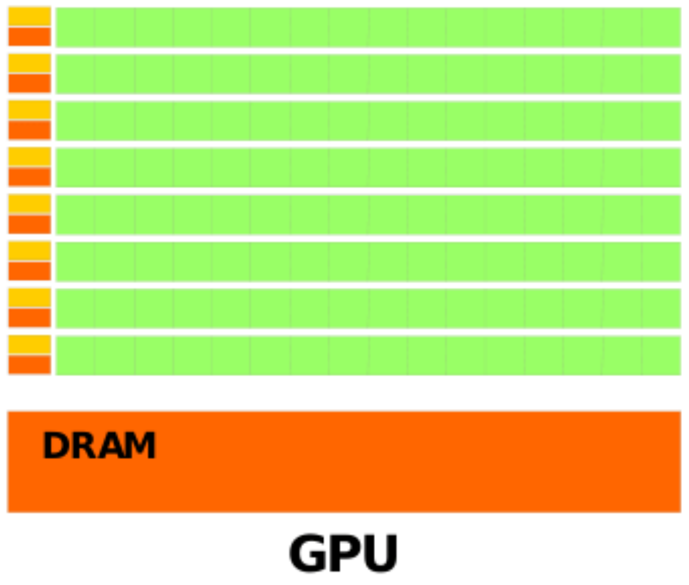
- The Loss (Cost) function measures the performance of a Machine Learning model for given data
- The loss function quantifies the error between the model's predicted values and the data's true values in the form of a single number
- You can define the loss function, for example $\frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$
- Calculate the loss function using all the events in your training sample, update the weights, repeat until you have minimized the loss function
  - How do you minimise the loss function
- One option is to use Gradient Decent. Take the local loss gradient with respect to the parameters and step (learning step) in the direction of the negative gradient. Repeat until the gradient is 0.
- Evaluating the loss function for all the events and repeating until you get to the minimum is very computationally expensive!
- This is where GPUs really help!

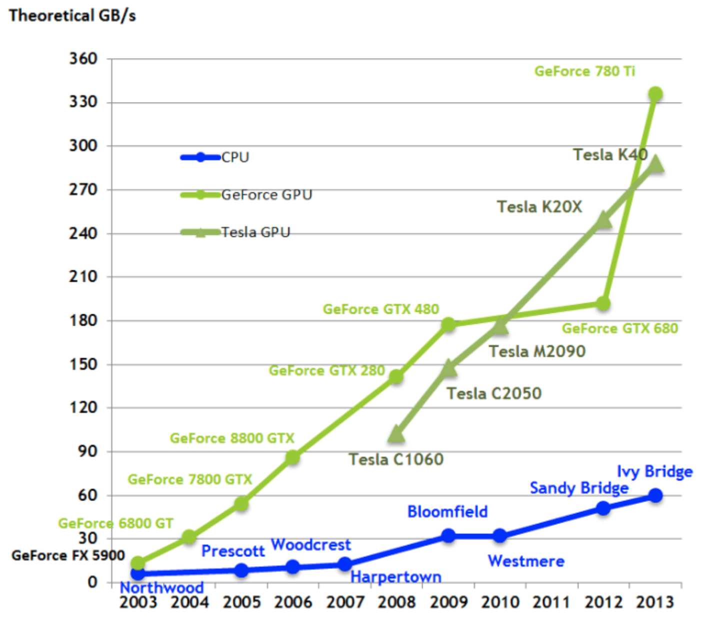


GPU v CPU

Arithmetic Logic Unit (ALU)





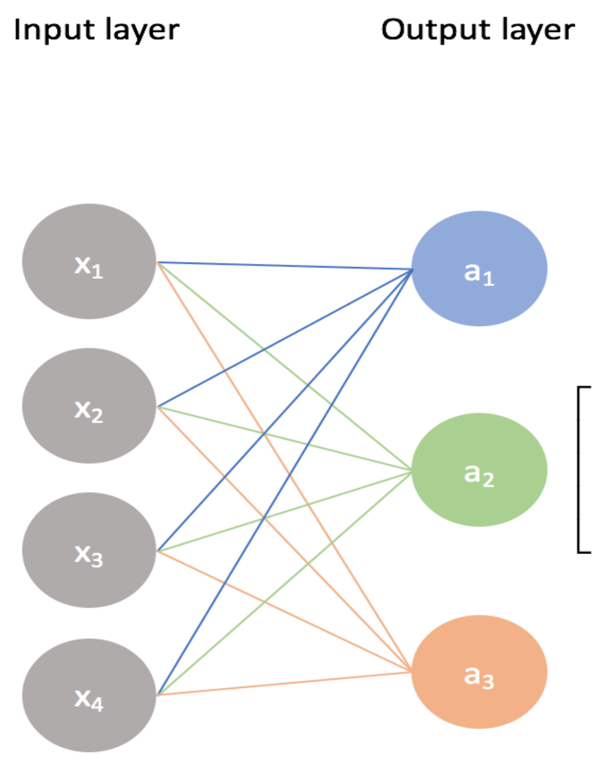


- Small number of compute cores
- MIMD (Multiple Instruction Multiple Data)
- Optimised for serial operations
- Low latency

- Large number of compute cores
- SIMD (Single Instruction Multiple Data)
- Built for parallel operations
- High throughput

Let's look at an MLP to demonstrate the high throughput of a GPU!

GPU v CPU



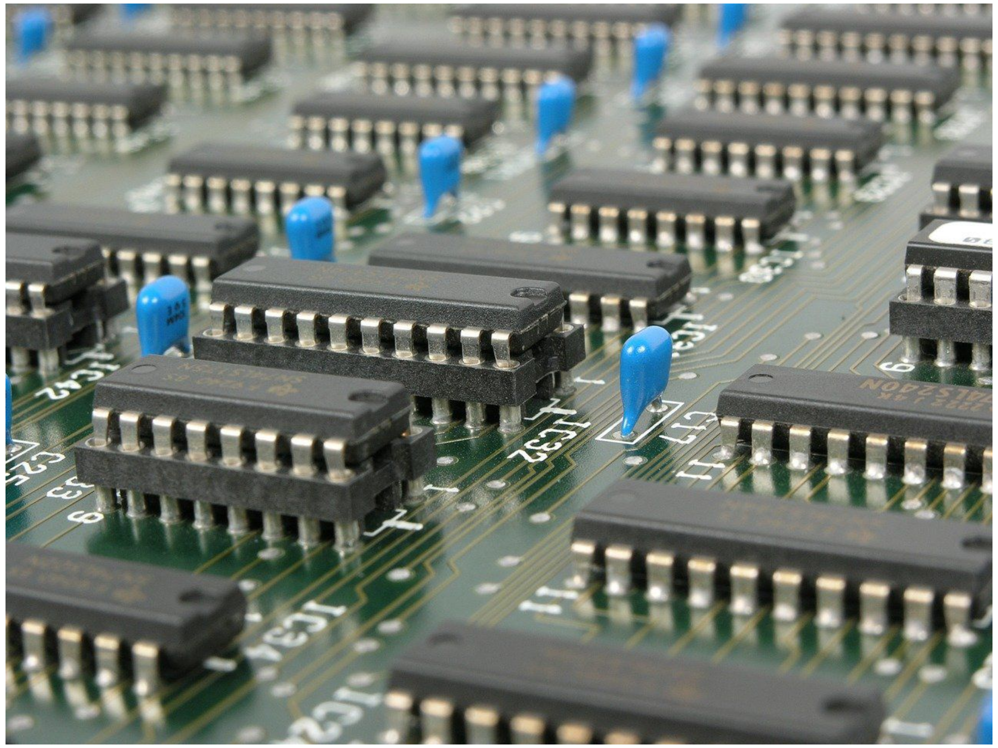
A simple neural network

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \end{bmatrix} \xrightarrow[\textit{activation}]{} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

- Consider this output layer of the neural net
- Output is a vector calculated from the matrix equation on the LHS
- Each element of the vector can be calculated on a computing core
- In this simple case this can be done easily on a CPU
- However repeating this process sequentially over many data points is very slow on a CPU
- Because we want to run the same instruction, it is much faster to run in parallel on the many GPU cores
- GPUs were not designed with machine learning in mind! Can we come up with anything more specialised?

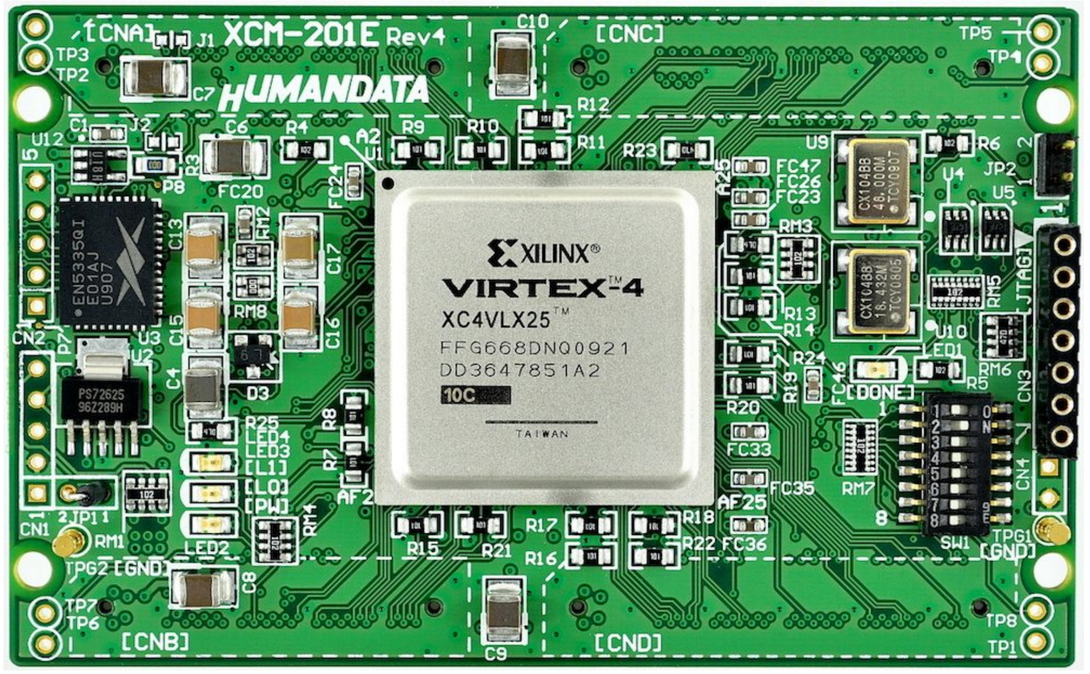
ASIC

- GPUs is a massively parallel processor with thousands of computational units capable of execute many different algorithms
- ASICs (Application Specific Integrated Circuit) are more specialised, designed to be capable of doing a very small set of computations (say, only matrix multiplications). But do so extremely well.
- You can build an ASIC to execute the neural net algorithm
    - Performance much better than a running algorithm on a GPU
    - ASIC can only run the algorithm it was built for

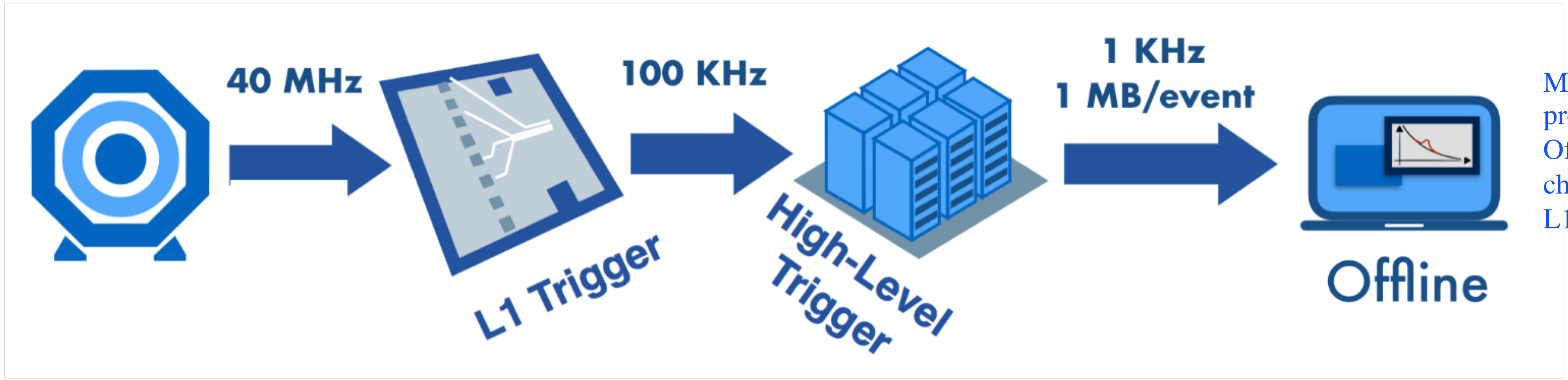


FPGAs

- An Field Programmable Gate Array (FPGA) is a customizable hardware device that can be programmed to perform the functionality of a desired application
  - They are different to ASICs, which are custom manufactured for specific design tasks.
- FPGAs can produce circuits with thousands of memory units for computation, so work similarly to GPUs
- They consume less power (an important consideration for many reasons) than GPUs and can be optimized for an increase in throughput
- FPGAs can implement custom data types whereas GPUs are limited by architecture
- The tricky part is implementing ML frameworks which are written in high level languages such as Python in a Hardware Description Language (HDL) used to program an FPGA
  - More software available, HLS4ML, Microsoft Azure …etc.



Reconstruction Using Machine Learning



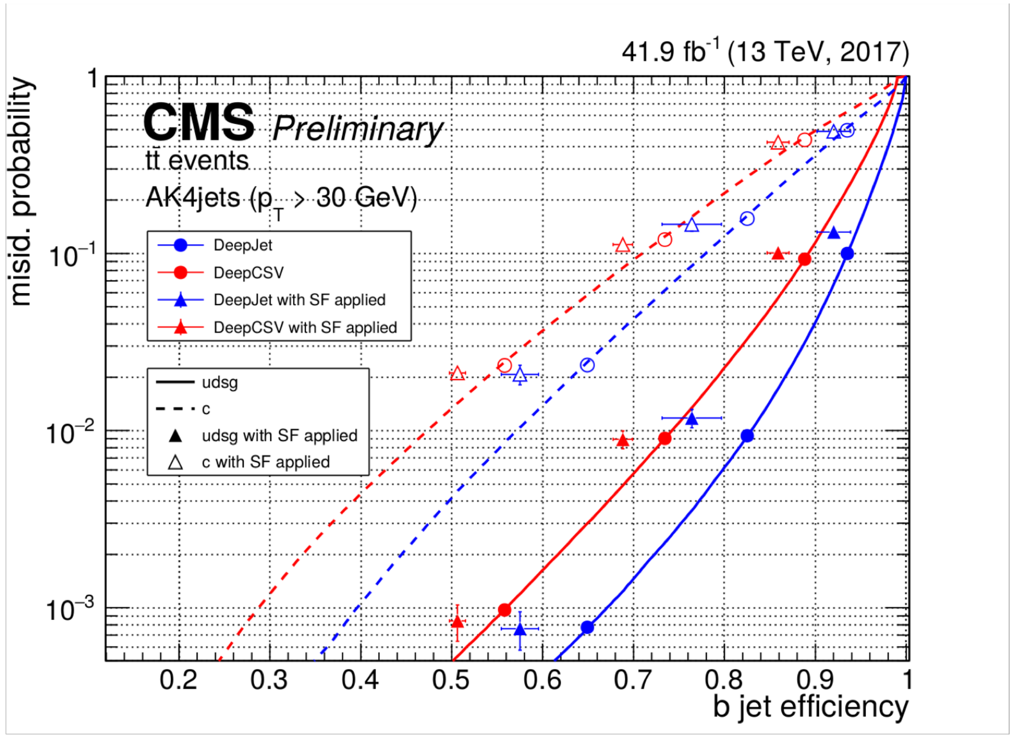
Machine learning prevalent in Offline. The challenge is at the L1 and HLT stage

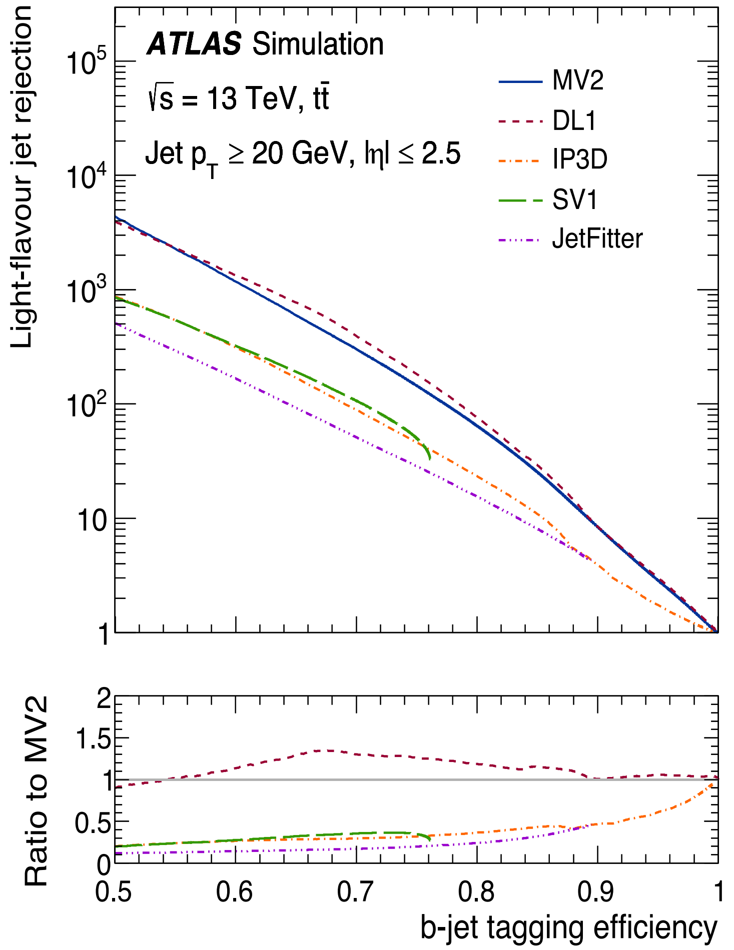
- All the components for successful development of machine learning are not new to the field of high energy physics
  - Actually, we as High Energy Physicist have expertise in all these fields!
- Data
  - Huge datasets
    - From detectors
    - Simulation
- Software
  - Python, VHDL, ….
- Hardware
  - GPUs, ASICs and FPGAs
- It is no surprise we are also seeing a growth of machine learning in high energy physics
- Let’s look at a few current and future uses of machine learning in high energy physics

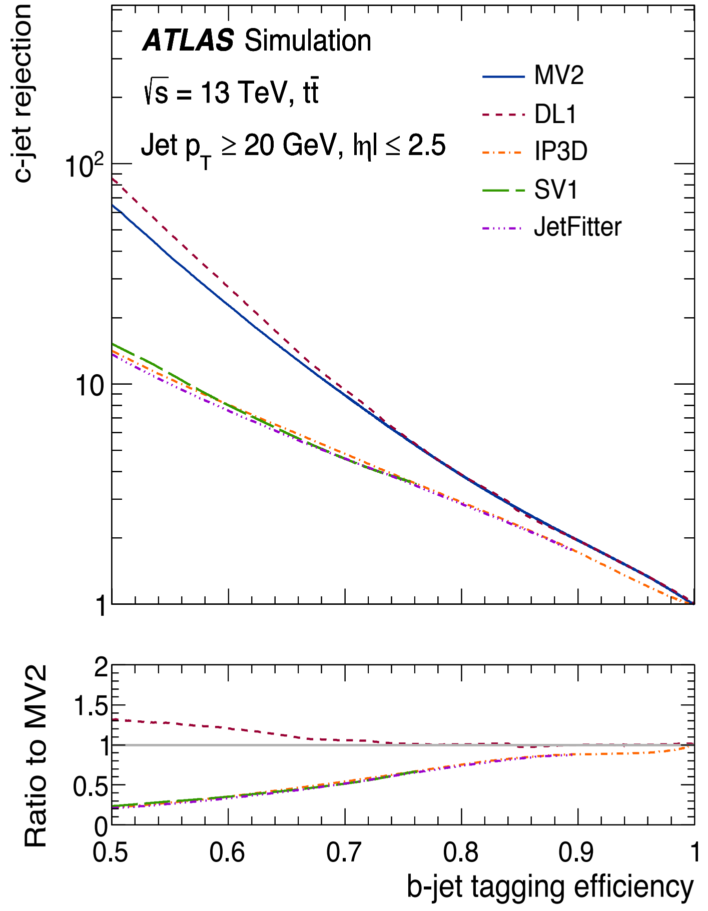
B-tagging

- Physics analysis was one of the first areas where machine learning for was deployed and has been around for years now
- A popular example is B-tagging

- In the plots below we see the strong performance of B-jet identification using Deep Neural Nets

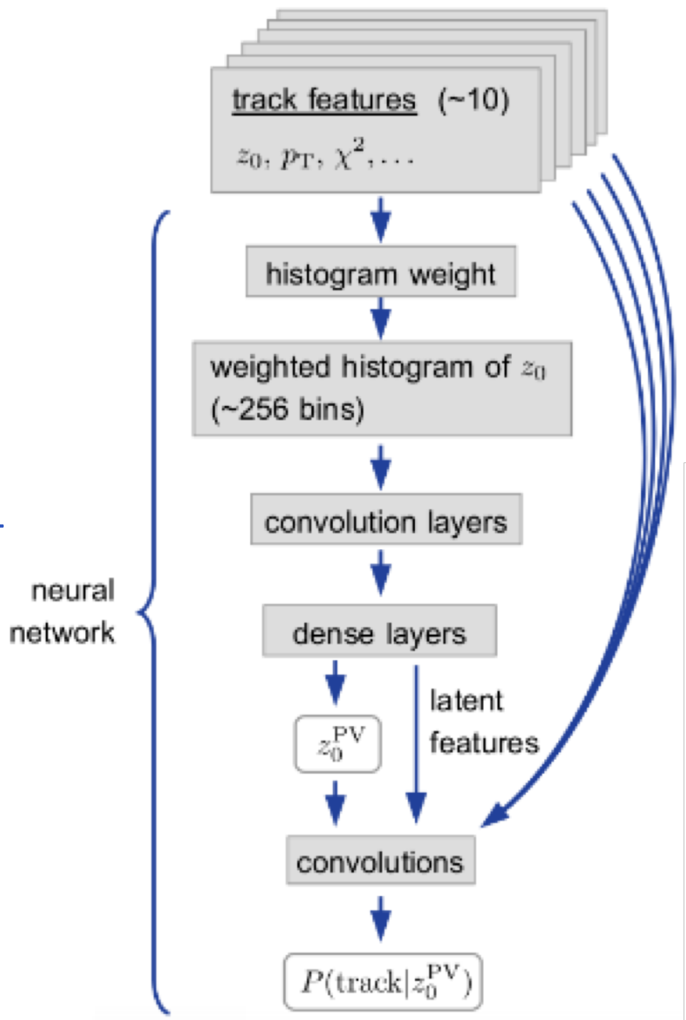




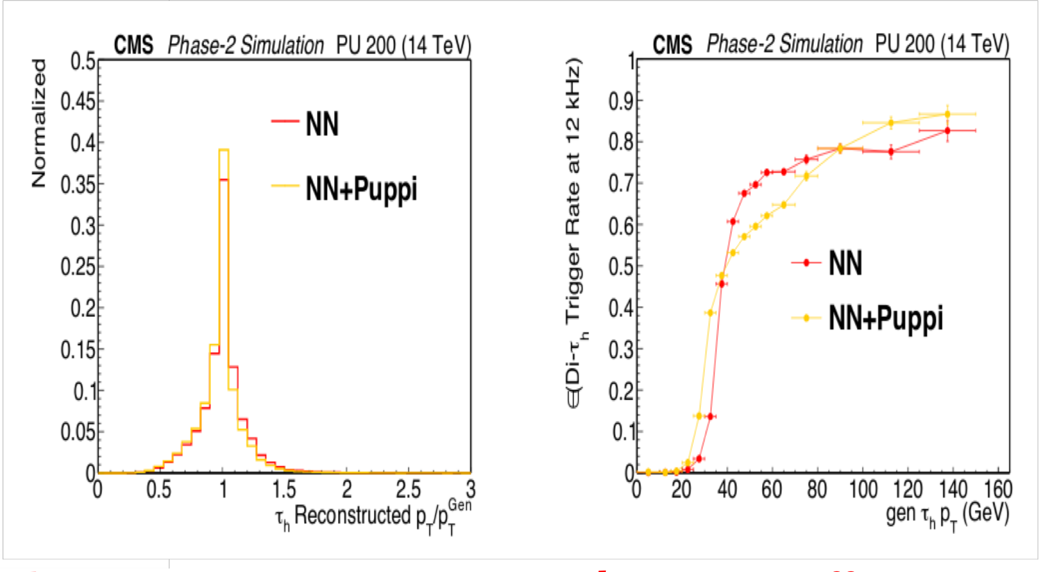


Vertex Reconstruction

- Now we are developing ML algorithms for online reconstruction!
- Identifying the primary vertex is crucial for mitigation of pileup and event reconstruction
- Tricky! Need to find primary vertex amongst tens of vertices and hundreds of tracks
- CMS plan to implement and end-to-end neural net
    - Use a CNN to feed in tracks and output the primary vertex. Also output the probability track is associated with primary vertex
    - Useful information in the L1 which can be fed downstream for further reconstruction



- Downstream in the trigger a deep neural net is used for the tau ID
- PUPPI events used for reconstruction rely heavily on the primary vertex

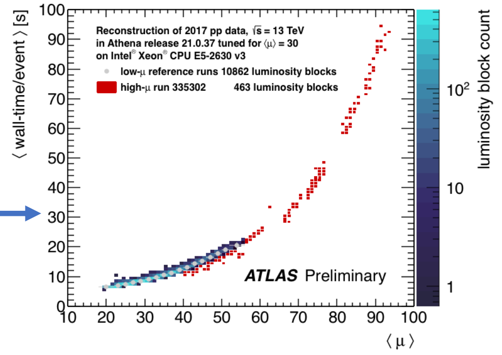


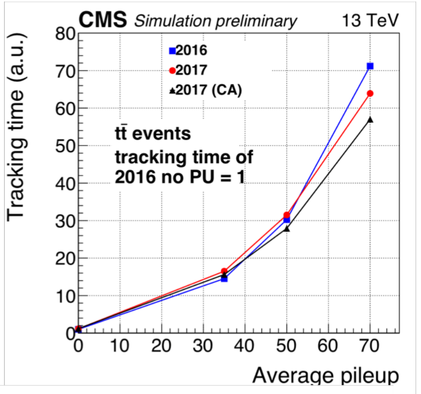
$h \rightarrow \tau_h \tau_h$ energy scale $h \rightarrow \tau_h \tau_h$ efficiency

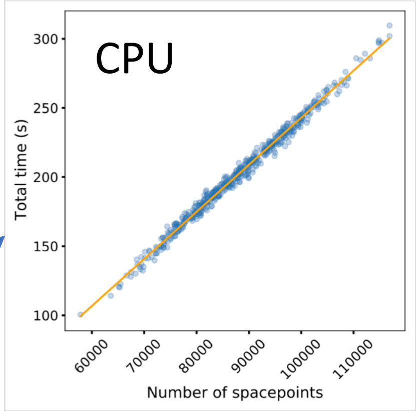
cds:2714892

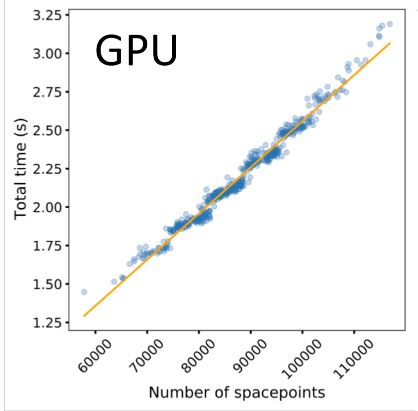
Tracking

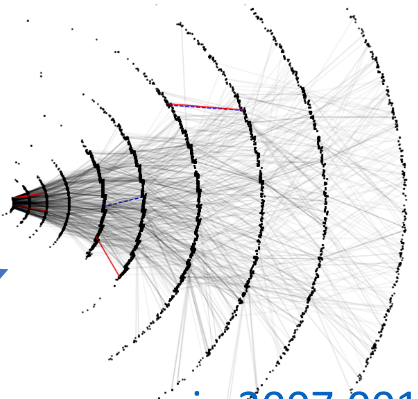
- The TrackML challenge has been a real boost to the exploration of using machine learning in particle tracking in HEP. arxiv :1904.06778
- Computational costs of typical tracking algorithms (Hough transforms etc.) grows more than linearly as a function of beam intensity due to increase in track hit combinatorics
  - Could be a limit on physics reach!
- Exa.TrkX project arxiv:2103.06995 uses Geometic Deep Learning (GDL) and Graph Neural Networks for particle tracking
- Suitable for running on GPUs, unlike typical tracking algorithms
- Using GPUs see a linear computational cost with beam intensity.
  - Inference time much faster than CPU
- High levels of efficiency and purity

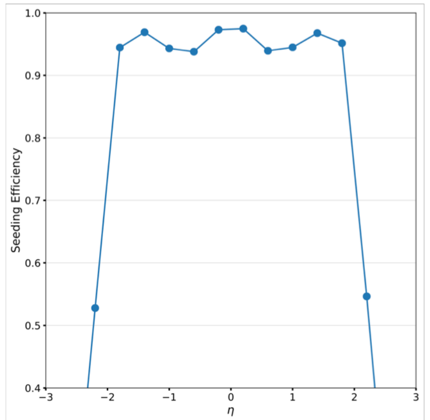


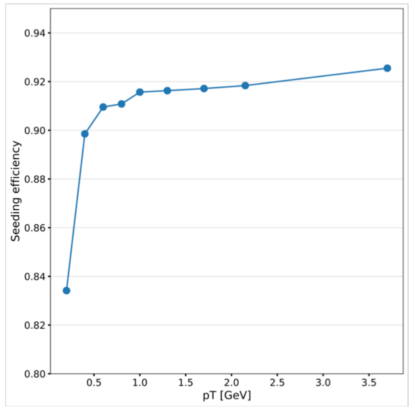








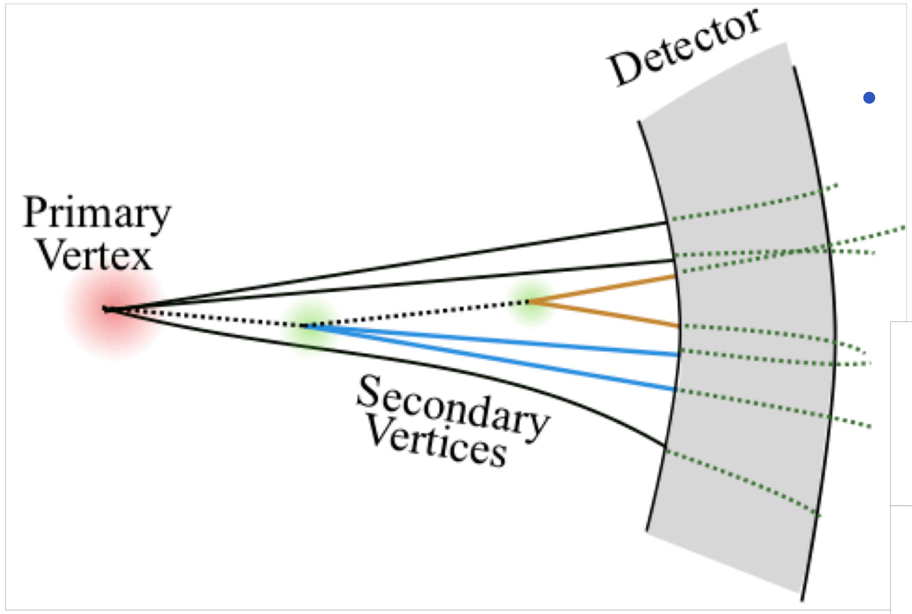




Black lines: true positive, blue lines: false negative and red lines: false positive

arxiv:2007.00149

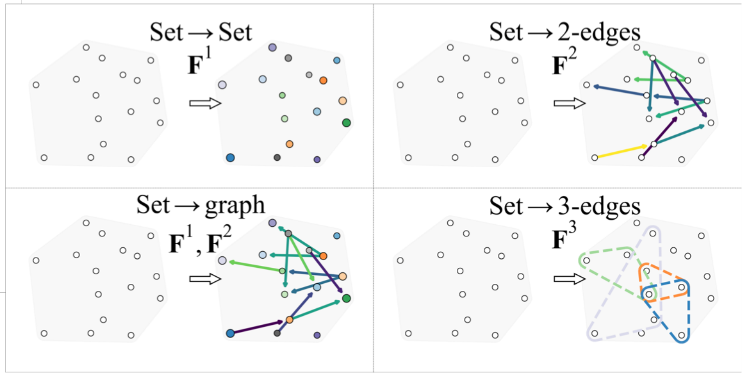
Secondary Vertex Finding in Jets with Neural Networks

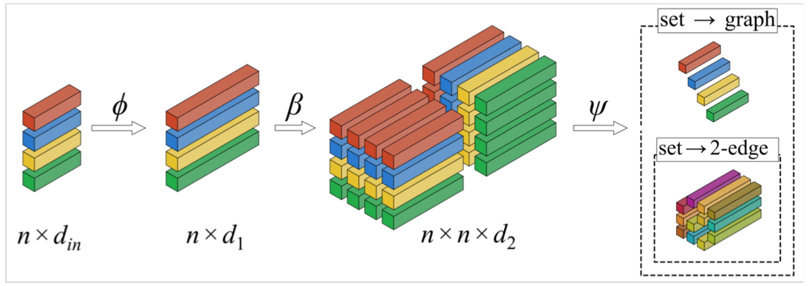


- An important component to b-taggers is secondary vertex (SV) information
- Look at comparing methods for SV calculations
  - Adaptive Vertex Reconstruction (AVR)
  - Track Pair (TP) classifier
  - Recurrent Neural Net
  - Set2Graph

arxiv:2008.02831

- Set2Graph learns taking sets of vectors to graphs
  - A graph consists of nodes linked by edges

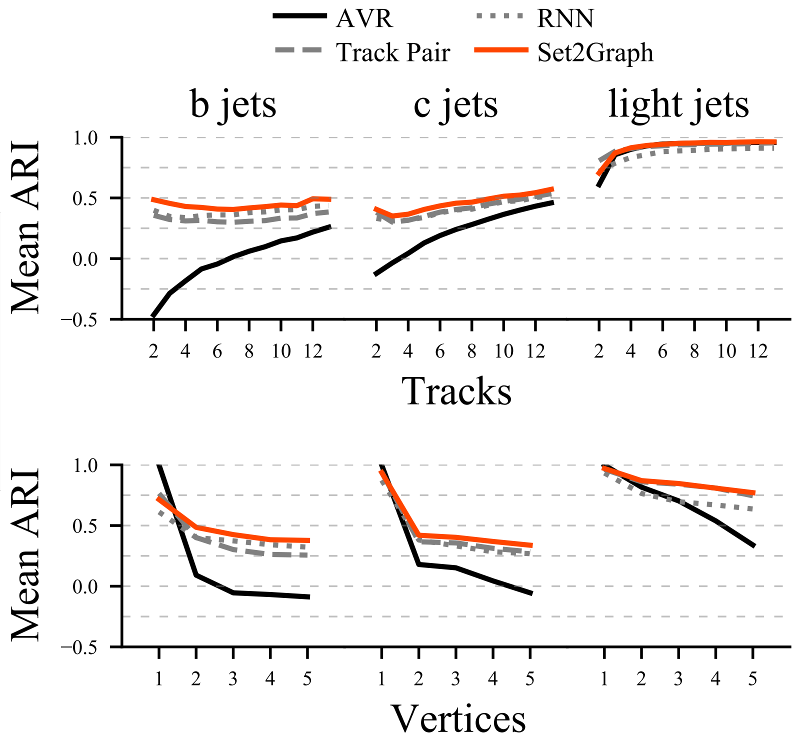




- Model architecture for set-to-graph and set-to-edge functions

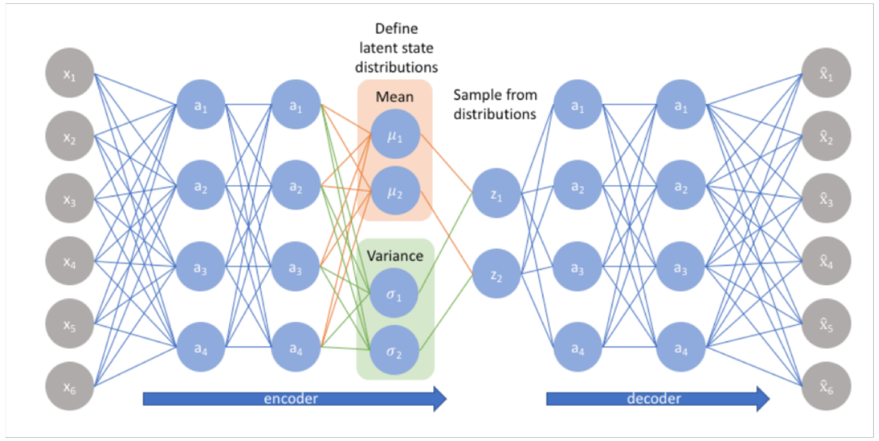
arxiv:2002.08772

- Use a metric called the ARI
- Essentially a score which reflects how well the vertex was reproduced
- Find set2Graph is the best performing algorithm



Identifying Events

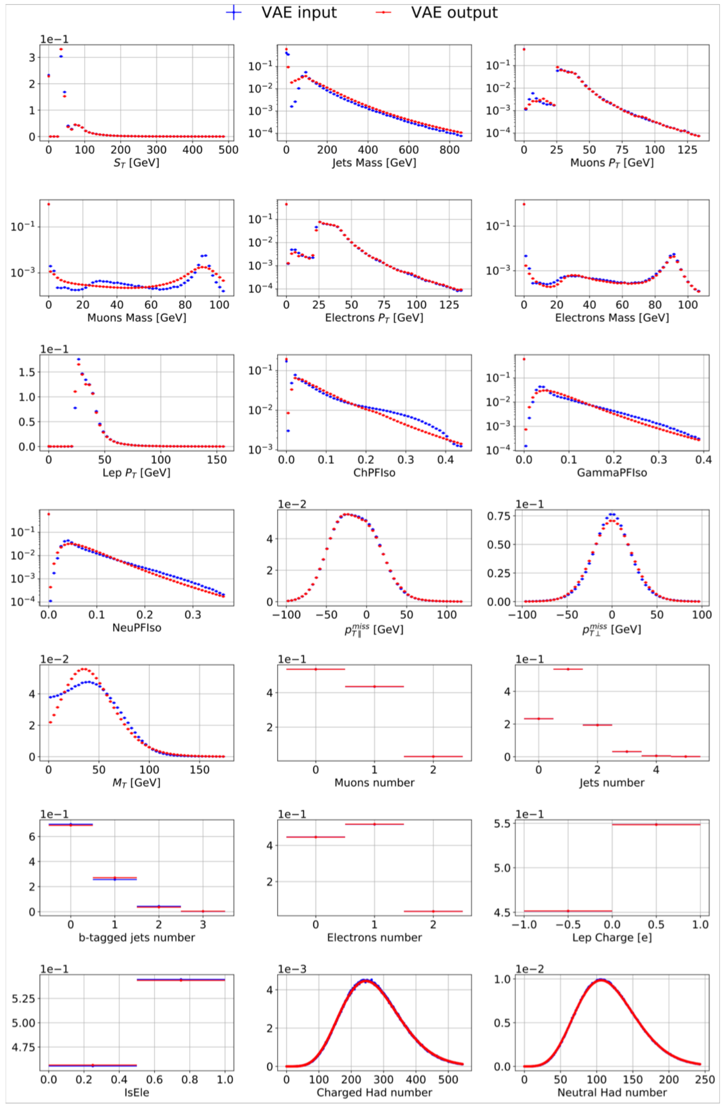
Variational Autoencoder (VAE)



- An Autoencoder (AE) is a DNN where the number of nodes per layer decrease the deeper the layer up to the midpoint and increase to the output
- A VAE applies a random sampling from a Gaussian distribution with a mean μ and a standard deviation σ of the midpoint nodes

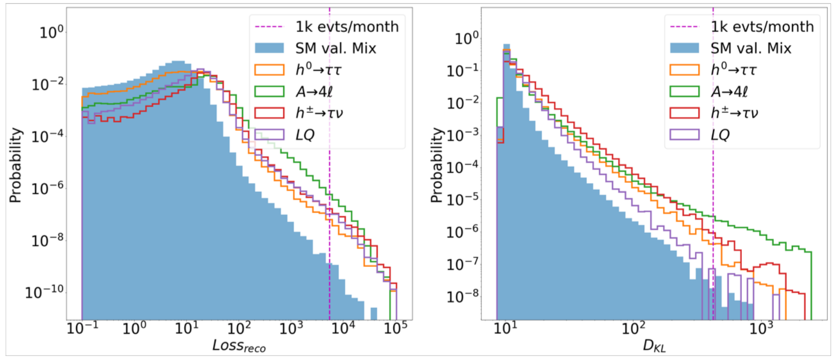
arXiv:1811.10276

VAE Input v Output



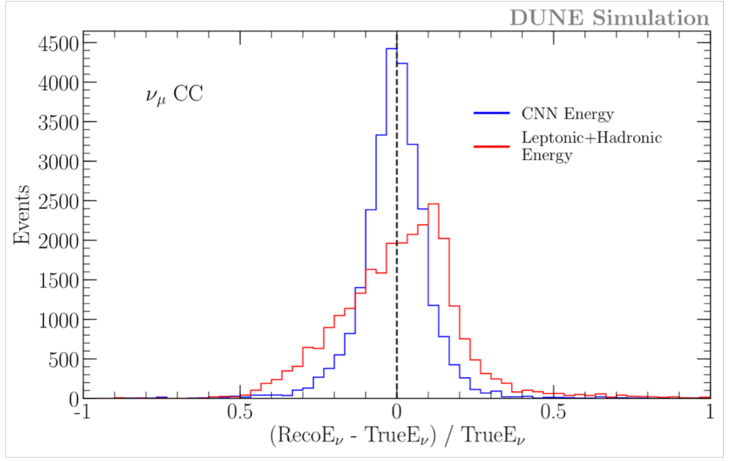
- CMS developing trigger to identify interesting events
- Train the VAE to reproduce the inputs
  - After training, running the VAE produces sample events
- Use VAE as trigger for BSM events
- Train VAE on SM data
- Compare events (using the loss function) with the VAE output (SM like). Trigger on events where the difference is large!

Loss greater for BSM events so significantly more likely to be selected by trigger



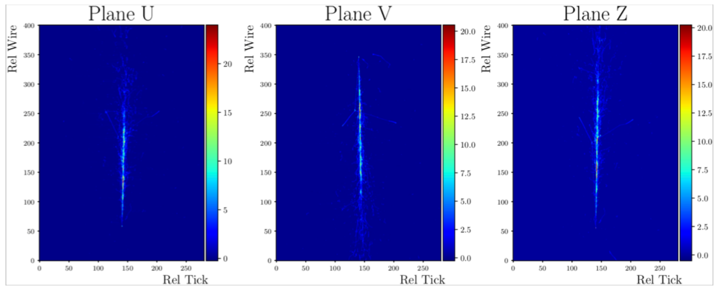
Neutrino Kinematic Reconstruction on DUNE

- DUNE will measure neutrino oscillation parameters through the detection of $\upsilon_\mu/\bar{\upsilon}_\mu$ disappearance and $\upsilon_e/\bar{\upsilon}_e$ appearance
- Precise energy reconstruction of the neutrino required
  - Reconstruct charged-current interaction in the detector
    - Charged lepton produced in detector identifies neutrino flavour
  - Traditionally kinematics are calculated directly from detector with calibration factors
- Look to use CNN to calculate kinematics (regression rather than classification)

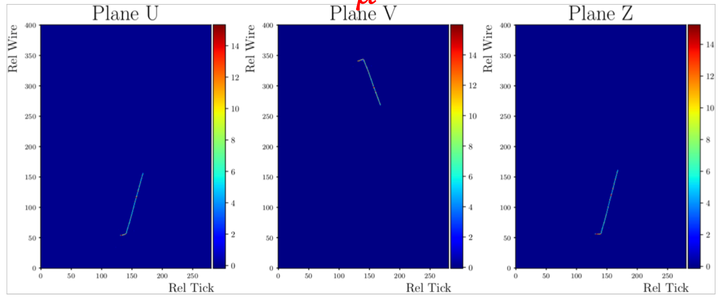


2D pixelmaps hits. Inputs for the 2D CNN for energy reconstruction

υ_e

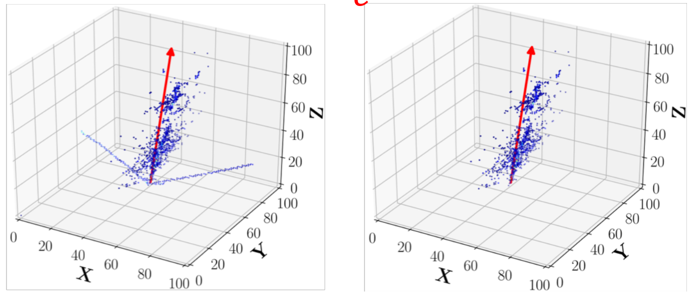


υ_μ

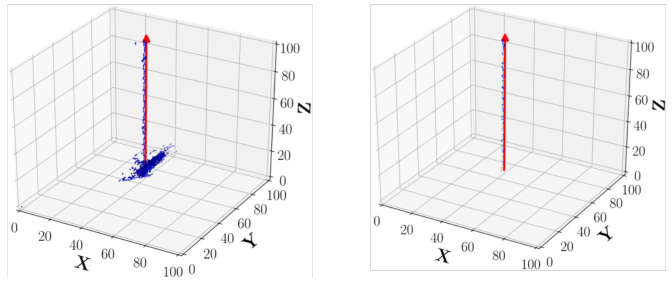


3D pixelmaps hits. Inputs for the 3D CNN for direction reconstruction

υ_e



υ_μ



arxiv:2012.06181

Summary

- Machine learning is a rapidly growing field
    - We are seeing this growth in High Energy Physics
        - We have the data, software and hardware experience
- Machine learning is prevalent in offline physics analysis but we are now seeing swift development of online machine learning reconstruction algorithms
- As well as becoming more popular, machine learning is evolving
    - Important to keep abreast with the latest developments
- A lot has been done, but there is still a lot to do
- Future of reconstruction in particle physics using machine learning looks promising!