Computing Methods for Particle Physics

Neural Networks

Overview

- Now: **Neural Networks!**

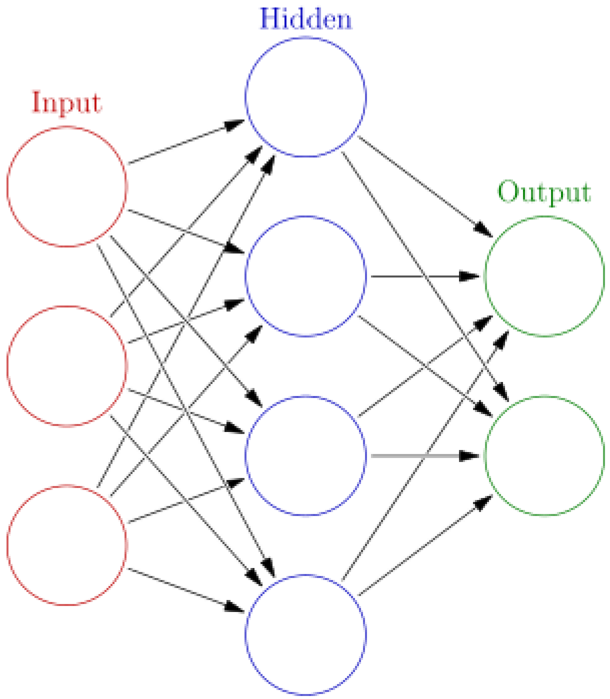
- Overview & terminology
- A very simple example in Python (without libraries)
- Deep learning
- Convolutional neural networks

Goal is to give you an overview of NNs and a taste of deep learning. These are deep, complex topics that take a lot of work to understand and implement. After this lecture, you may be able to use a network, but fully understanding them will most likely require more study!

Artificial Neural Networks

In analogy with their biological counterparts, **Artificial Neural Networks** (ANNs), process **inputs** through layers of artificial neurons, via **weights**, to produce an **output** or **outputs**

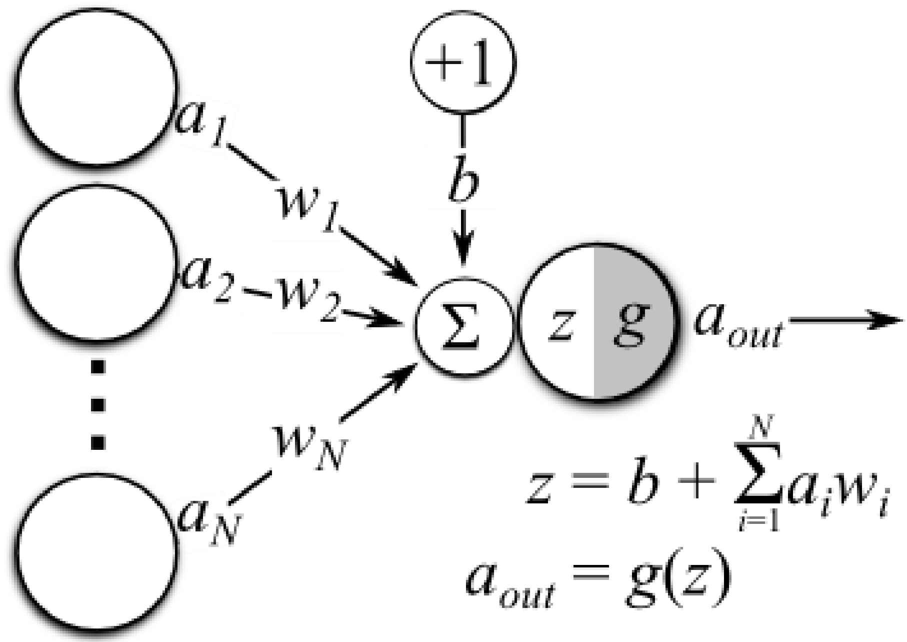
Outputs may be for **classification** (is this an apple or an orange) or **regression** (what is the energy of a particle?)



ANNs

A simple model of a single layer ANN[ref]

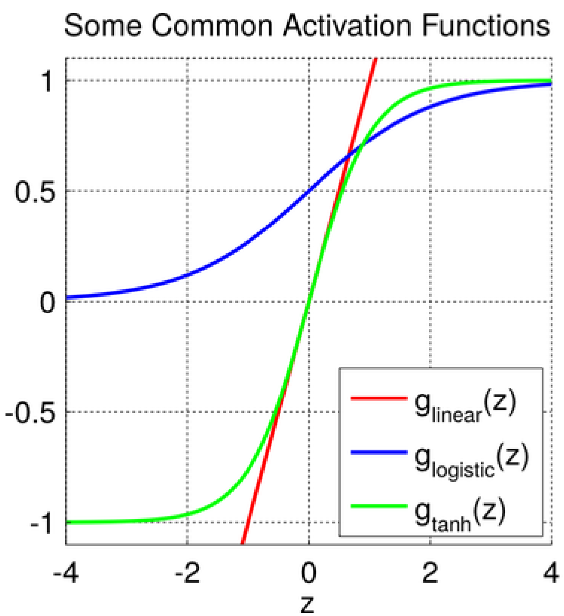
- **Nodes** (inputs): a_1, …, a_N
- **Weights**: w_1, …, w_N
- Neuron **input** value: z
- **Activation function**: g(z)
- **Bias**: b
- **Output**: a_{out} (e.g. values in [0,1], [-1,1])

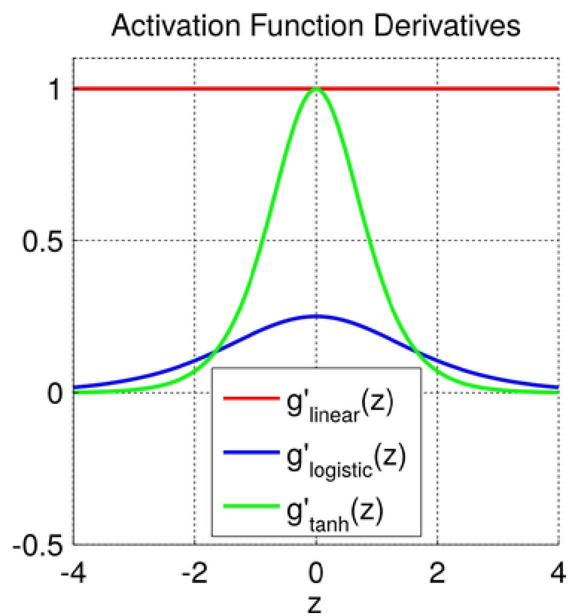


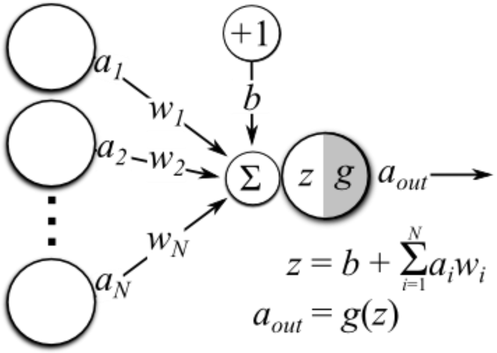
$$z = b + \sum_{i=1}^{N} a_i w_i$$

$$a_{out} = g(z)$$

Activation Function







$$g_{\text{linear}}(z) = z$$

$$g_{\text{logistic}}(z) = \frac{1}{1+e^{-z}}$$

$$g_{\tanh}(z) = \tanh(z)$$

Activation function determined by type of problem to be solved:

Binary (**logistic**) classification: g_{logistic} or $g_{\tanh}$ ref

Linear regression: g_{linear}

Function must be **differentiable** in order to train the network (e.g. with gradient descent) so these are good choices since their **derivatives** are defined in terms of the **initial function** (computationally efficient).

[ref]

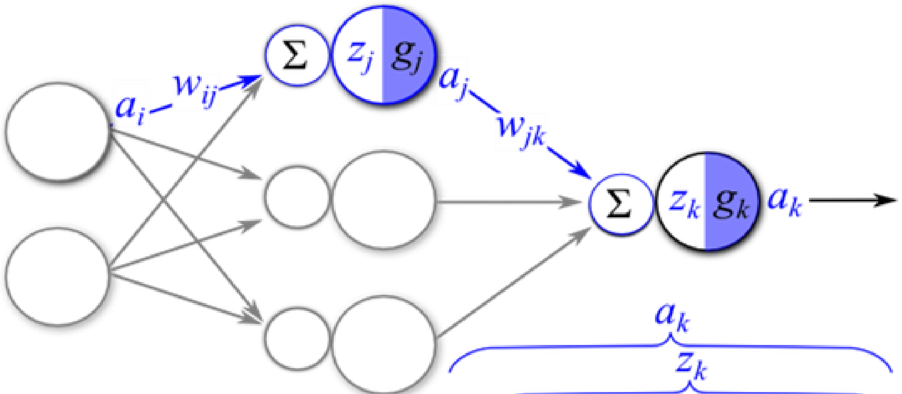
Gradient Descent & Backpropagation

The network is trained by:

1. Forward propagating the signals (a_i, a_j) from the input nodes via **randomly initialized weights** (w_{ij}, w_{jk}) to produce an output value (a_k).

Output at each layer (a_j, a_k) is computed from the computed value (z_j, z_k) passed through the activation function (g_j, g_k).

I. Forward-propagate Input Signal



$$\overbrace{a_k = g_k(\overbrace{b_k + \Sigma_j \underbrace{g_j(\underbrace{b_j + \Sigma_i a_i w_{ij}}_{z_j})}_{a_j} w_{jk}}^{z_k})}^{a_k}$$

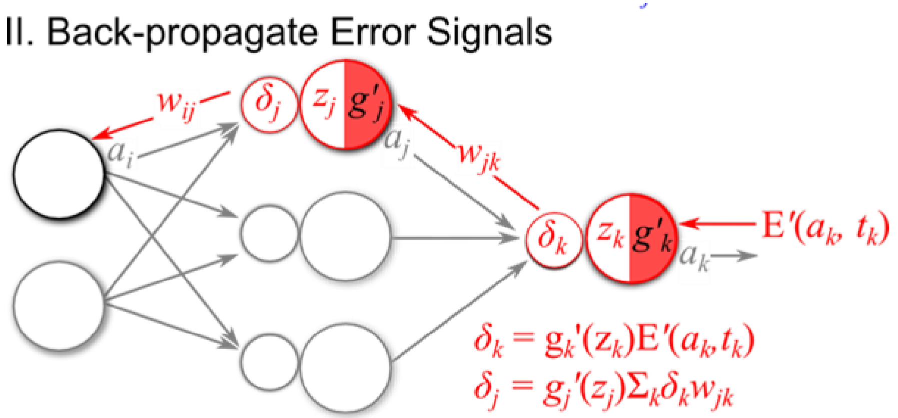
[ref]

Gradient Descent & Backpropagation

2. **Back propagating** error signals (δ_j, δ_k) through the network to compute differences between each weight.

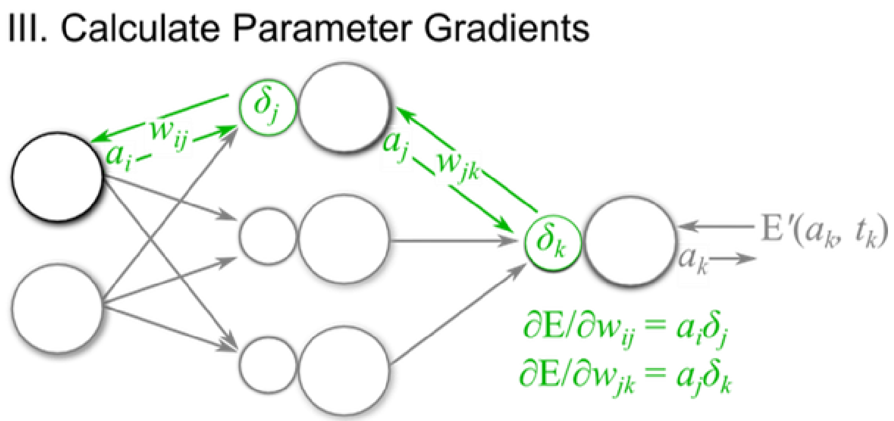
The error function (E') computes the difference between the network output and the expected output given the input.

E=½(output - expectation)^2



Gradient Descent & Backpropagation

3. **Numerically** computing the **gradients** of the error function with respect to the **weights**, cascading back through the layers



Gradient Descent & Backpropagation

4. **Updating** the **weights** in each layer in the direction of the derivative from the previous step. The magnitude of this update is typically weighted by a tunable **learning rate**.

IV. Update Parameters

$$w_{ij} = w_{ij} - \eta(\partial E / \partial w_{ij})$$

$$w_{jk} = w_{jk} - \eta(\partial E / \partial w_{jk})$$

for learning rate η

Three Layer Feed-forward Network

We will use a simple example to put these ANN principles into practice in a way that is easy to dissect.

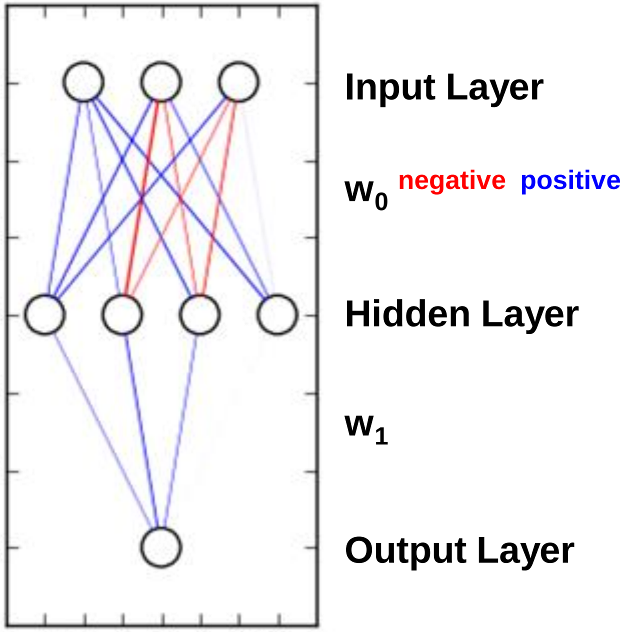
Train ANN to **partially** learn: a && (b || c)

Three **inputs**
Weights - fully connected
Hidden layer - 4 nodes
Output - single output layer
Gradient descent & backpropagation

a	b	c	out
0	0	1	0
0	1	1	0
1	0	1	1
1	1	1	1

To the code...

Example: Training in Action



ANN Summary

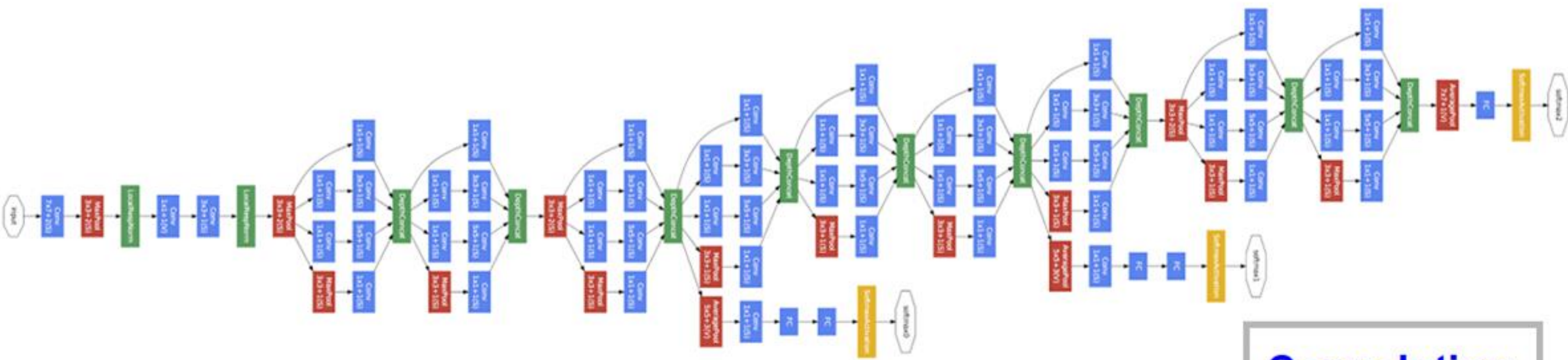
Artificial neural networks can be trained to perform classification or regression. We saw how to train a simple example using **gradient descent**.

These are the basic building blocks for more complex networks. Which we will discuss now...

Deep Learning

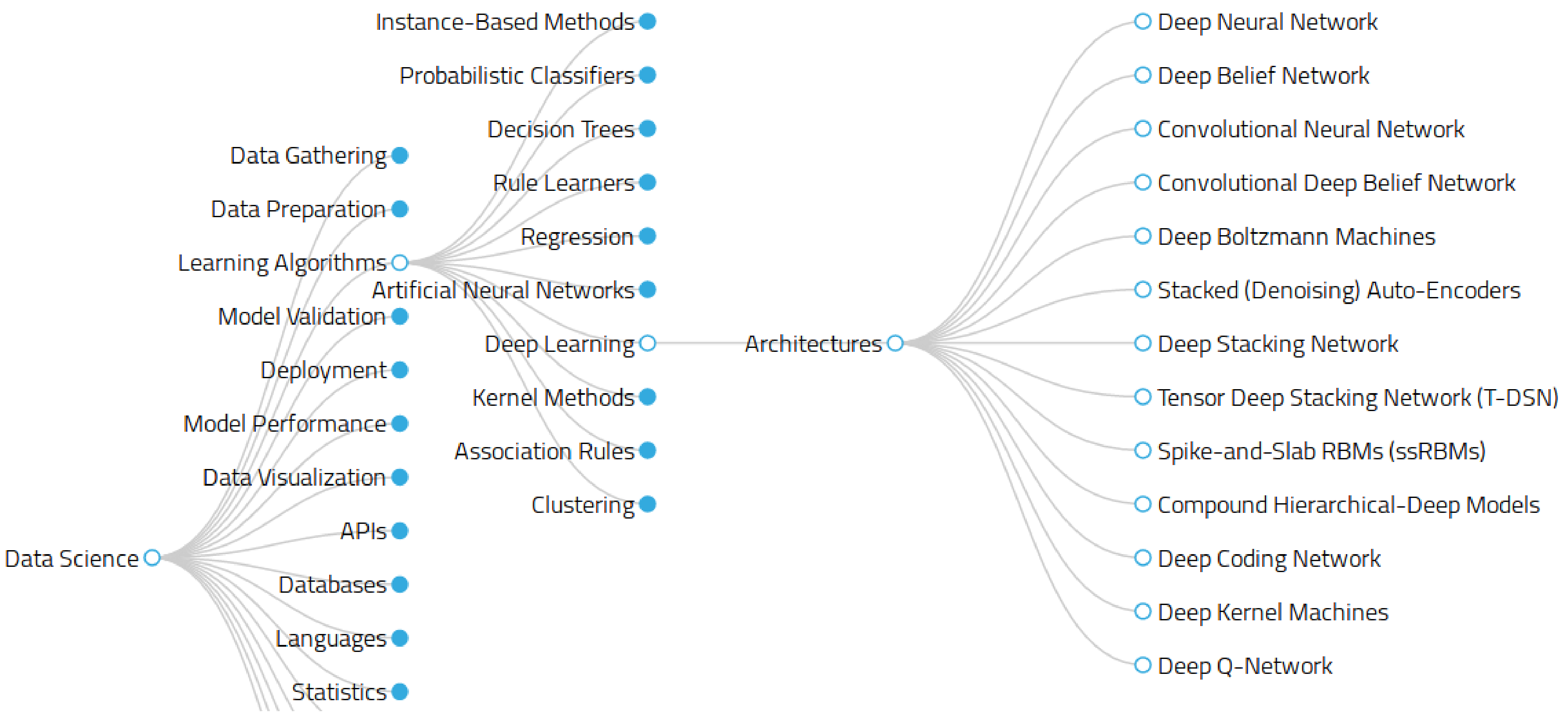
Deep learning refers to a **family of ML algorithms** that have **many layers** of neural networks, of various types, chained together.

Significant research goes into building sophisticated DL **network architectures** for particular tasks



e.g. GoogLeNet – winner if the 2014 ImageNet challenge

Classes of Deep Learning



Convolutional Neural Networks (CNN)

Machine Learning Approach

Deep Learning

- Focus on a particular method: **Convolutional Neural Networks (CNN)**
  - powerful new technique developed for computer vision
  - *very good at image analysis!*

Image captioning

Car vision

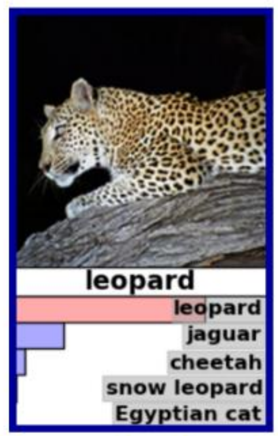
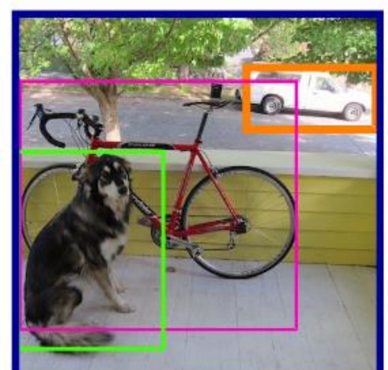


Image classification

Defeating Humans at Go

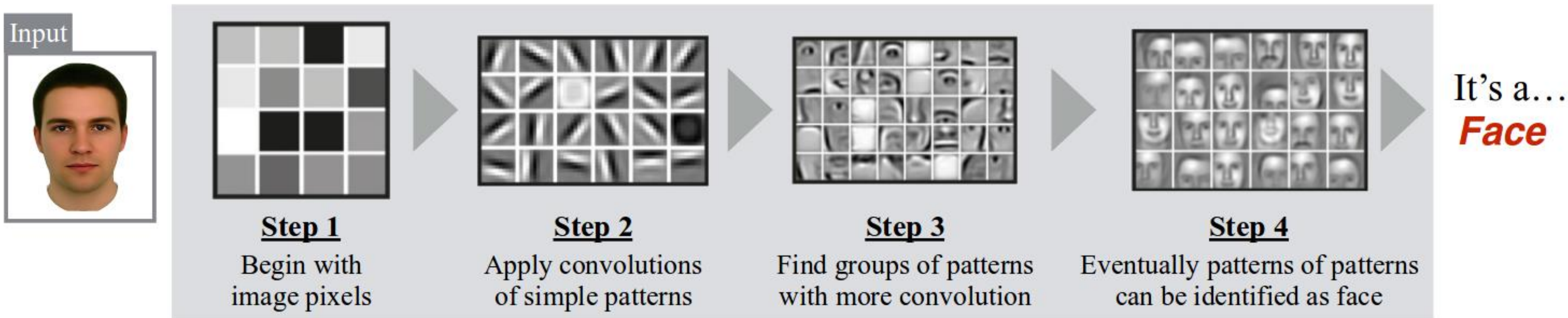


Object Detection

Convolutional Neural Networks (CNN)

Deep Learning - Convolutional Neural Networks

- CNNs can produce representations of high level objects based on low level features
- Consider the task of **facial recognition**…



- Involves training with **labeled data**
- Network learns features by itself
  - power to **generalize to new data**!

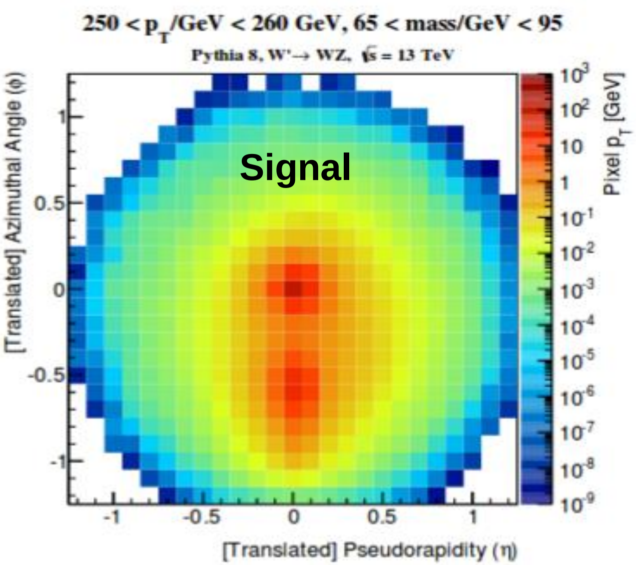
Example DL Applications (more in backup)

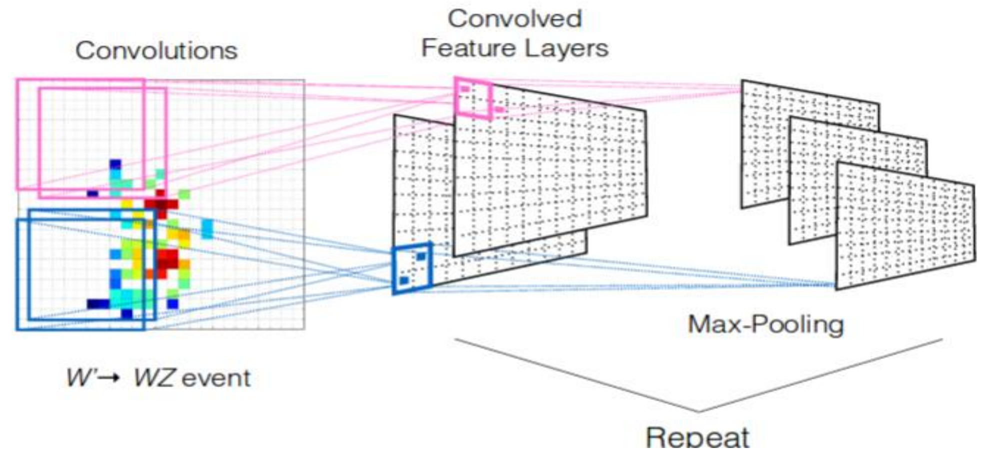
Jet-tagging

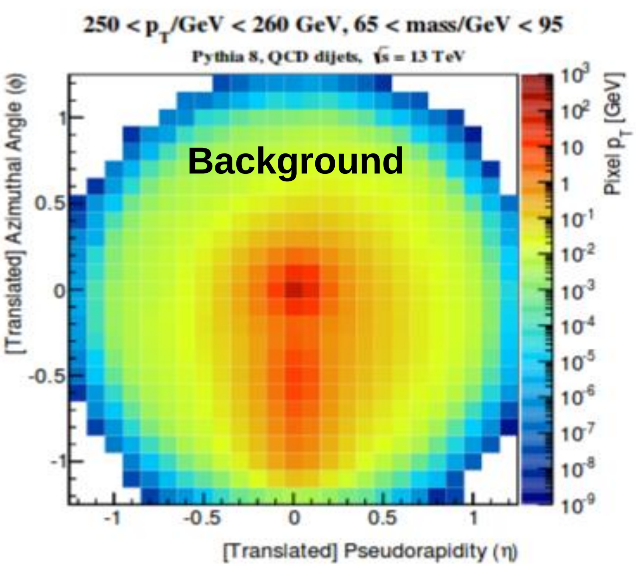
Jet-Images – Deep Learning Edition

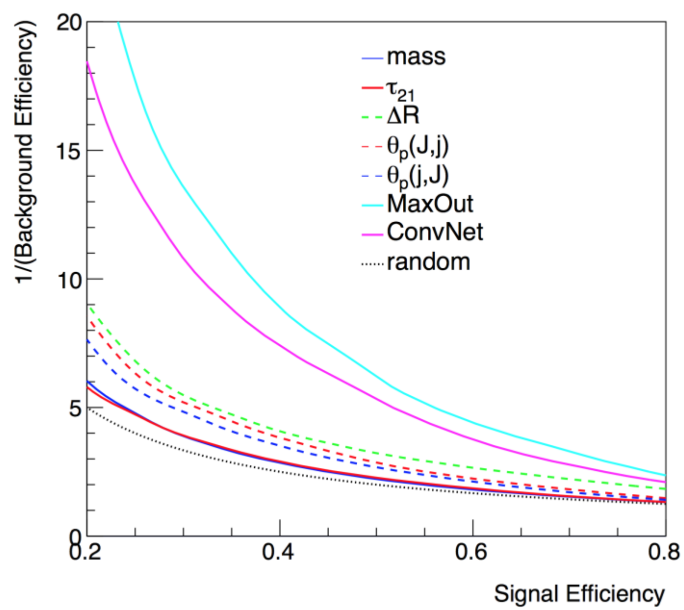
link

Luke de Oliveira,[a] Michael Kagan,[b] Lester Mackey,[c] Benjamin Nachman,[b] and Ariel Schwartzman[b]









Study of DL methods to distinguish between **highly boosted W boson decays** and backgrounds.

Compares performance of different network architectures on these problems.

Deep Learning Frameworks

Why use a deep learning framework?

- Take advantage of **optimized** implementations
- Multiple **CPU** and **GPU** support (important for **scalability**)
- Support for **batch** operations
- Allows us to start from a well formulated **example**
- **Focus** on network **architecture** and **optimization**

There are many **frameworks** for deep learning and/or CNNs. Wikipedia has a nice comparison of the different frameworks:

Deep learning software by name [edit]

Software	Creator	Software license[a]	Open source	Platform	Written in	Interface
Caffe	Berkeley Vision and Learning Center	BSD license	Yes	Linux, macOS, Windows[2]	C++	Python, MATLAB
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	Apache 2.0	Yes	Linux, macOS, Windows, Android (Cross-platform)	C++, Java	Java, Scala, Clojure, Python (Keras), Kotlin
Dlib	Davis King	Boost Software License	Yes	Cross-Platform	C++	C++
Intel Data Analytics Acceleration Library	Intel	Apache License 2.0	Yes	Linux, macOS, Windows on Intel CPU[10]	C++, Python, Java	C++, Python, Java[10]
Intel Math Kernel Library	Intel	Proprietary	No	Linux, macOS, Windows on Intel CPU[11]		C[12]
Keras	François Chollet	MIT license	Yes	Linux, macOS, Windows	Python	Python, R
MATLAB + Neural Network Toolbox	MathWorks	Proprietary	No	Linux, macOS, Windows	C, C++, Java, MATLAB	MATLAB
Microsoft Cognitive Toolkit	Microsoft Research	MIT license[21]	Yes	Windows, Linux[22] (macOS via Docker on roadmap)	C++	Python (Keras), C++, Command line,[23] BrainScript[24] (.NET on roadmap[25])
Apache MXNet	Apache Software Foundation	Apache 2.0	Yes	Linux, macOS, Windows,[31][32] AWS, Android,[33] iOS, JavaScript[34]	Small C++ core library	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl
Neural Designer	Artelnics	Proprietary	No	Linux, macOS, Windows	C++	Graphical user interface
OpenNN	Artelnics	GNU LGPL	Yes	Cross-platform	C++	C++
PyTorch	Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan	BSD license	Yes	Linux, macOS	Python, C, CUDA	Python
Apache SINGA	Apache Incubator	Apache 2.0	Yes	Linux, macOS, Windows	C++	Python, C++, Java
TensorFlow	Google Brain team	Apache 2.0	Yes	Linux, macOS, Windows[39]	C++, Python	Python (Keras), C/C++, Java, Go, R[40]
Theano	Université de Montréal	BSD license	Yes	Cross-platform	Python	Python (Keras)
Torch	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet	BSD license	Yes	Linux, macOS, Windows,[50] Android,[51] iOS	C, Lua	Lua, LuaJIT,[52] C, utility library for C++/OpenCL[53]
Wolfram Mathematica	Wolfram Research	Proprietary	No	Windows, macOS, Linux, Cloud computing	C++	Wolfram Language

Software	OpenMP support	OpenCL support	CUDA support	Automatic differentiation[1]	Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)
Caffe	Yes	Under development[3]	Yes	Yes	Yes[4]	Yes	Yes	No	?
Deeplearning4j	Yes	On roadmap[5]	Yes[6][7]	Computational Graph	Yes[8]	Yes	Yes	Yes	Yes[9]
Dlib	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Intel Data Analytics Acceleration Library	Yes	No	No	Yes	No		Yes		Yes
Intel Math Kernel Library	Yes[13]	No	No	Yes	No	Yes[14]	Yes[14]		No
Keras	Only if using Theano as backend	Under development for the Theano backend (and on roadmap for the TensorFlow backend)	Yes	Yes	Yes[15]	Yes	Yes	Yes	Yes[16]
MATLAB + Neural Network Toolbox	No	No	Train with Parallel Computing Toolbox and generate CUDA code with GPU Coder[17]	No	Yes[18][19]	Yes[18]	Yes[18]	No	With Parallel Computing Toolbox[20]
Microsoft Cognitive Toolkit	Yes[26]	No	Yes	Yes	Yes[27]	Yes[28]	Yes[28]	No[29]	Yes[30]
Apache MXNet	Yes	On roadmap[35]	Yes	Yes[36]	Yes[37]	Yes	Yes	Yes	Yes[38]
Neural Designer	Yes	No	No	?	?	No	No	No	?
OpenNN	Yes	No	Yes	?	?	No	No	No	?
PyTorch	Yes		Yes	Yes	Yes	Yes			Yes
Apache SINGA	No	No	Yes	?	Yes	Yes	Yes	Yes	Yes
TensorFlow	No	On roadmap[41] but already with SYCL[42] support	Yes	Yes[43]	Yes[44]	Yes	Yes	Yes	Yes
Theano	Yes	Under development[45]	Yes	Yes[46][47]	Through Lasagne's model zoo[48]	Yes	Yes	Yes	Yes[49]
Torch	Yes	Third party implementations[54][55]	Yes[56][57]	Through Twitter's Autograd[58]	Yes[59]	Yes	Yes	Yes	Yes[60]
Wolfram Mathematica	No	No	Yes	Yes	Yes[61]	Yes	Yes	Yes	Yes

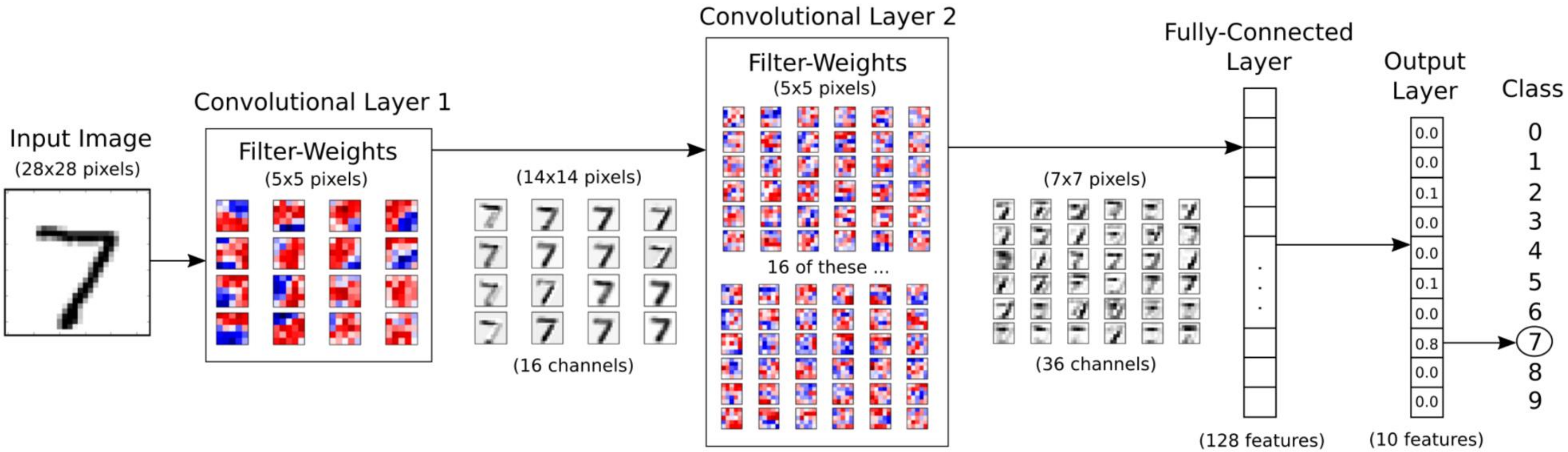
→ We will use **TensorFlow**.

TensorFlow

Tensorflow is Google's machine learning library.

- Used internally at Google for speech recognition, GMail, photo search, etc.
- Can run across multiple CPUs and GPUs
- Has **APIs for Python and C++** (we will use Python)

Convolution Neural Network in practice



In addition after each convolutional step:

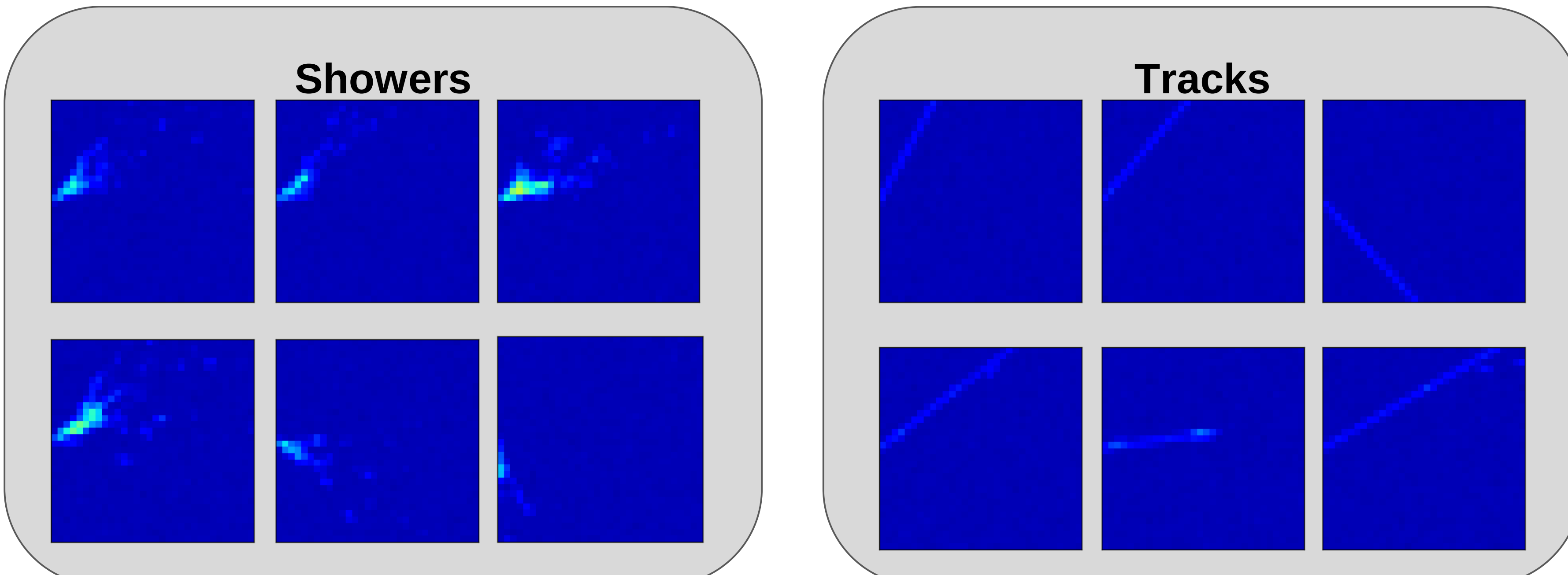
- **Rectified linear unit (ReLU)**: Set all negative values to zero
- **Max pooling:** Downsample image taking only max value in a given pixel window

Check out this excellent video for a complete explanation: https://www.youtube.com/watch?v=HMcx-zY8JSg

CNN Example Dataset

We will use a toy MC dataset to train a CNN to distinguish between **shower-like** and **track-like** images. A separate toy-MC script generates the dataset.

The output of the script is a (pickled) set of testing & training datasets (with labels):



CNN Example

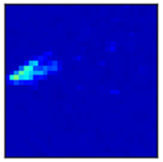
We will train a CNN to distinguish between these showers and track images.

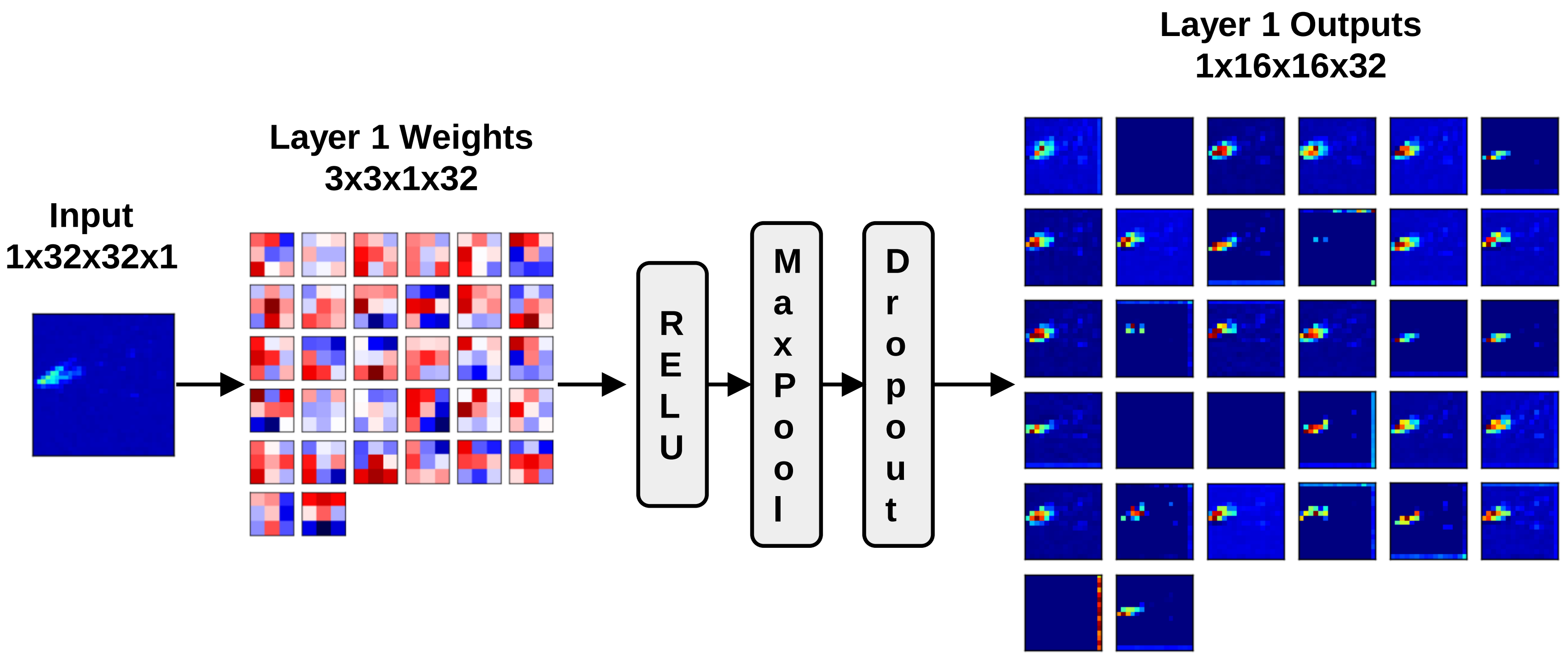
- This is the same network architecture as this [MNIST tensorflow example].The code for this example is based on this example, in fact.

The example is [here].

We will follow **an input image** through the network to explore its **architecture**.

Input
1x32x32x1

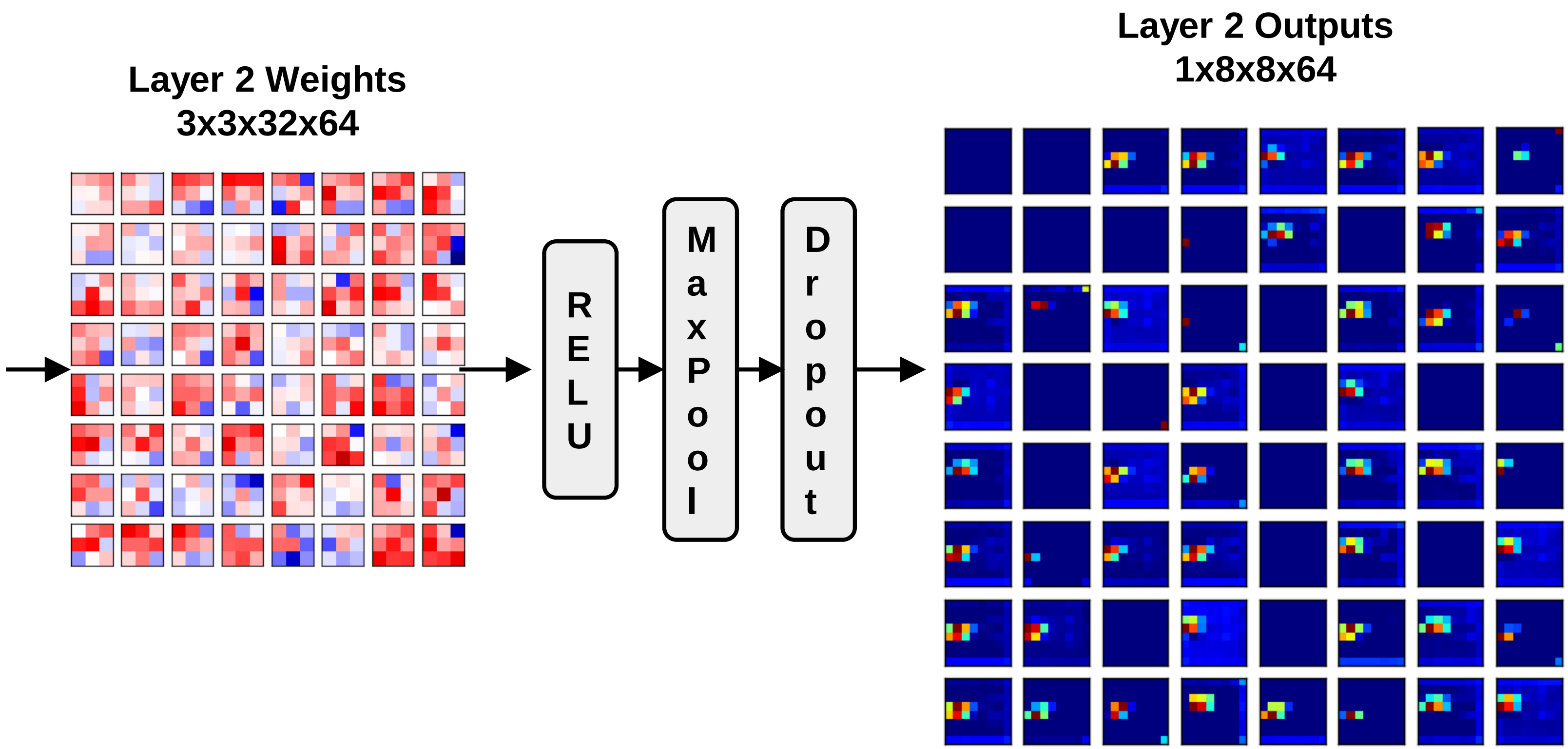




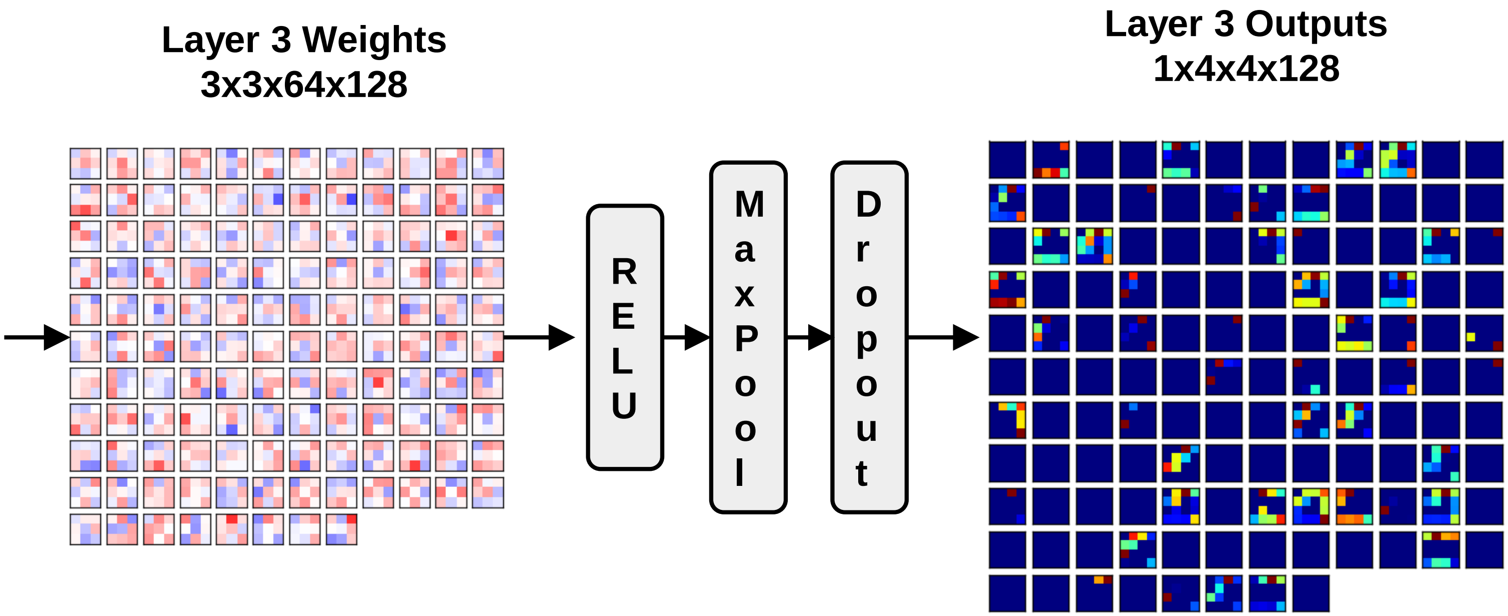
ReLU Activation Function: Rectifying Linear Unit: g(x)=max(0,x)
MaxPool: Reduces granularity while taking maximum pixel values
Dropout: zeroes out some percentage of entries to prevent overfitting ref

Network Architecture - Layer 1

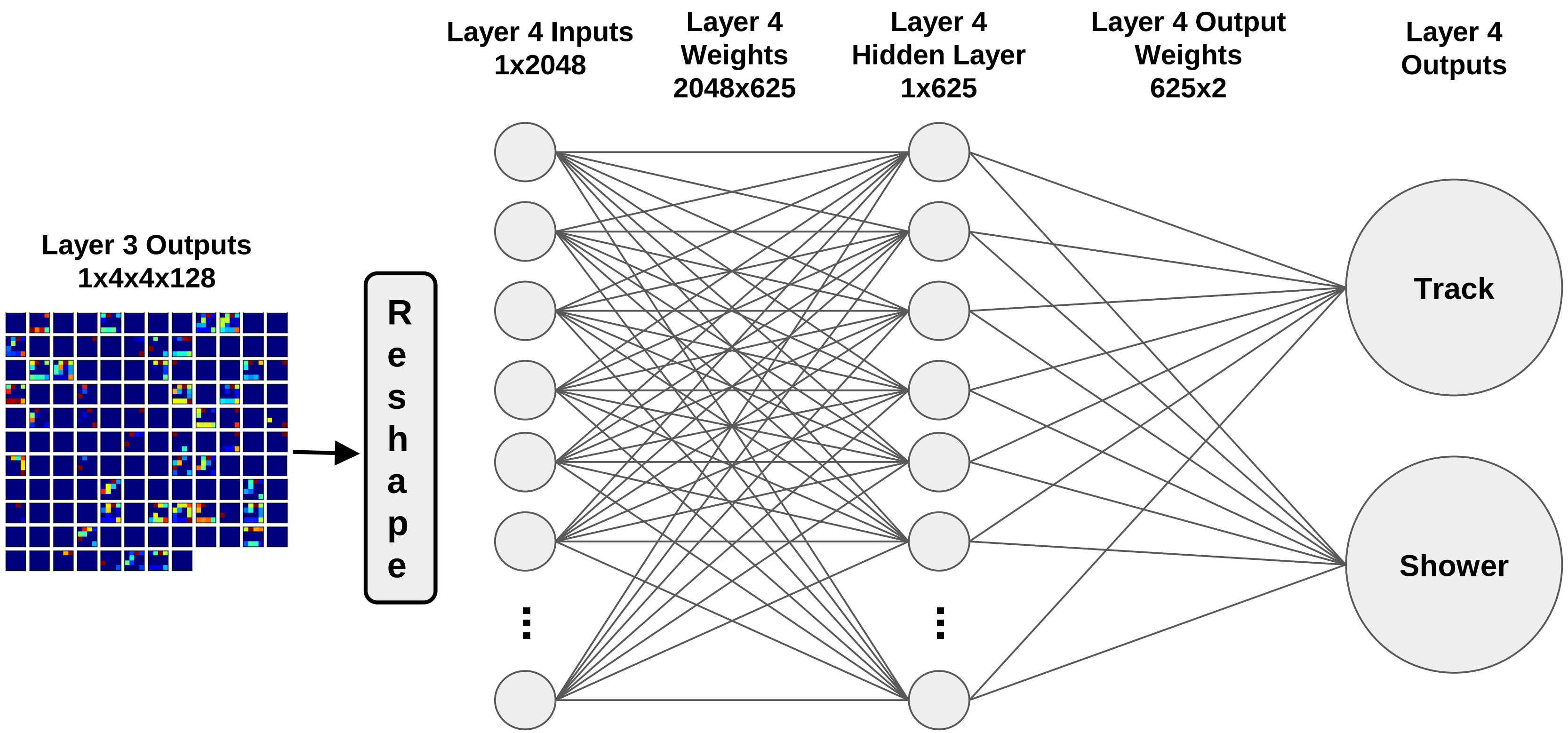
Input tensor format: [batch, in_height, in_width, in_channels]
Filter/kernel tensor format: [filter_height, filter_width, in_channels, out_channels]



Network Architecture - Layer 2



Network Architecture - Layer 3



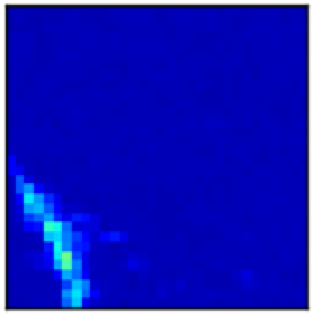
Network Architecture - Layer 4
Fully Connected Layer

CNN Performance

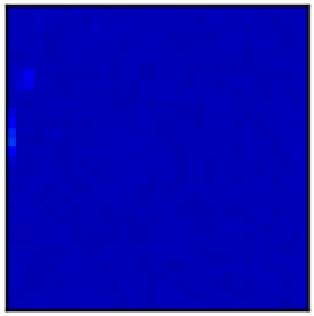
The network is trained by minimizing the loss (error).

Result: It does well, correctly identifying 98-100% of images.

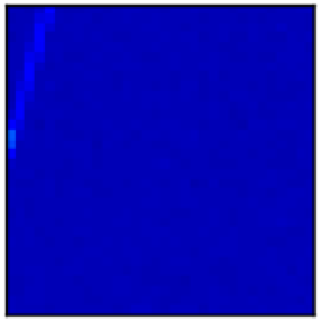
Incorrect cases:



True: S Pred: T



True: S Pred: T



True: T Pred: S

Summary

We talked about **neural networks**, their **categories** and associated **jargon**.

We saw simple a **simple neural network**.

We discussed CNNs and their recent application in a few **experiments**.

We saw an example of classifying tracks and showers using **tensorflow**.

Suggested Exercises & Further Reading

Exercises:

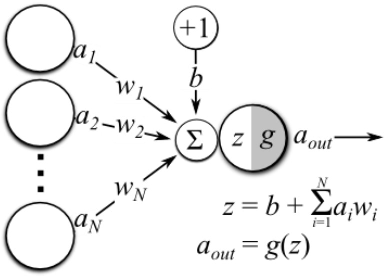
1. Use an ANN in scikit learn and/or **TMVA**
2. Modify the simple **three layer feed forward example**
   - Train on the full truth table, does it work?
   - Use a different activation function
   - Add another hidden layer
   - Add a bias term
3. Run the Track/Shower example in **Tensorflow**
   - Modify the network

Further reading:

- Deep Learning, Goodfellow et al, http://www.deeplearningbook.org/
- CS231n (Stanford) Website
- Tensorflow tutorials
- Siraj Raval on YouTube

Extra Slides

Bias in an ANN



The bias term in an ANN effectively allows the activation function to be shifted. This can allow the network to learn features more effectively.[ref]

Can add one **bias** node that is connected to every neuron with an independent weight.

Serves the same role as the **y-intercept** in a linear regression model. See z equation, top-right.

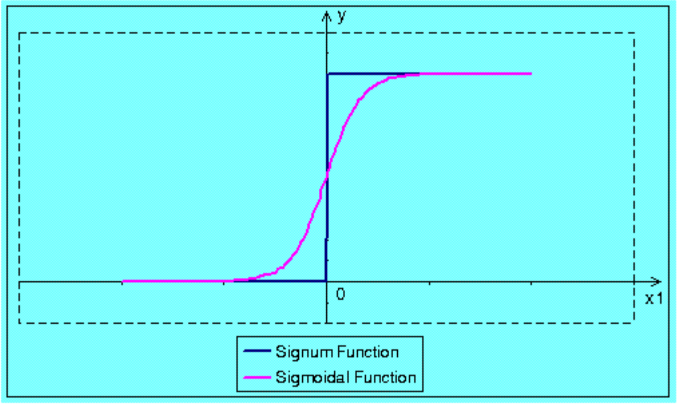


fig 1a. Neuron without bias activate function. Signum and sigmoidal function.

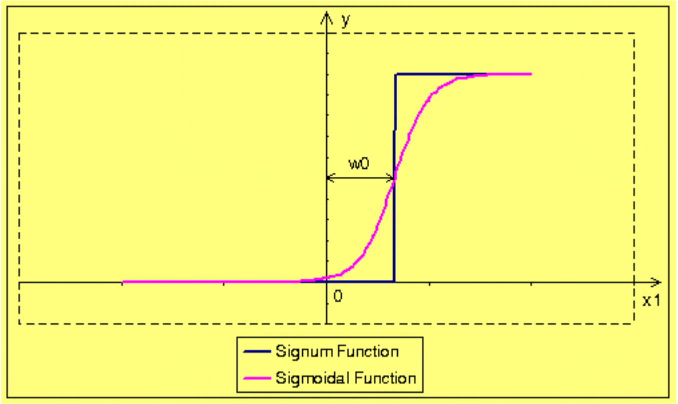


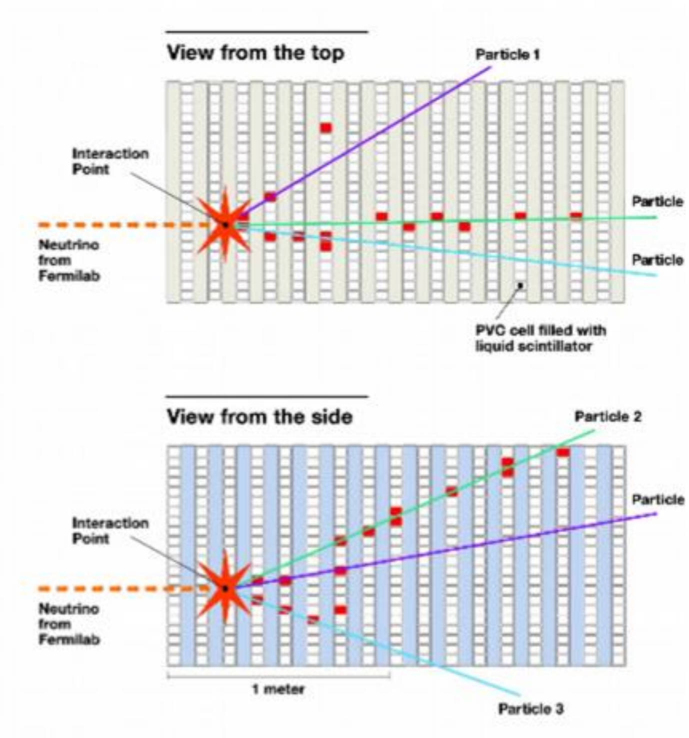
fig 1b. Neuron with bias activate function. Signum and sigmoidal function.

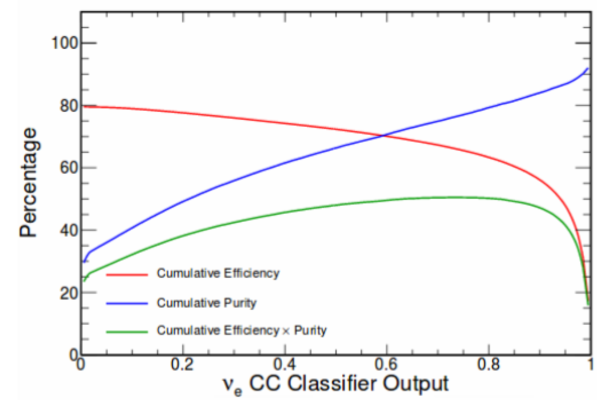
DL Applications: NOvA

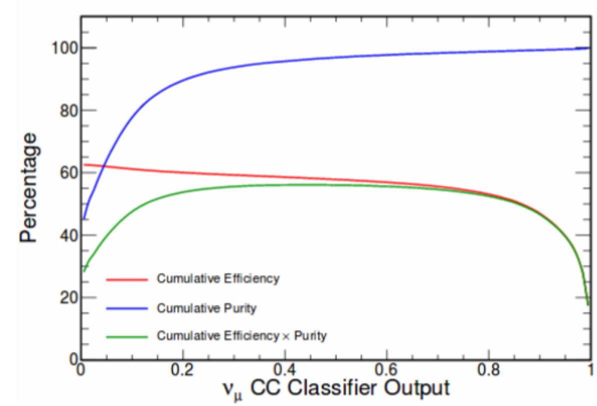
A Convolutional Neural Network Neutrino Event Classifier

link

A. Aurisano,[a,1] A. Radovic,[b,1] D. Rocco,[c,1] A. Himmel,[d] M.D. Messier,[e] E. Niner,[d] G. Pawloski,[c] F. Psihas,[e] A. Sousa[a] and P. Vahle[b]







Study of DL methods to distinguish between ν_e **and** ν_μ **events in event displays**.