Computing Methods for Particle Physics

Neural Networks

Overview

- And 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 today, you may be able to use a network, but understanding them would mean...

Overview

- Today: **Neural Networks!**

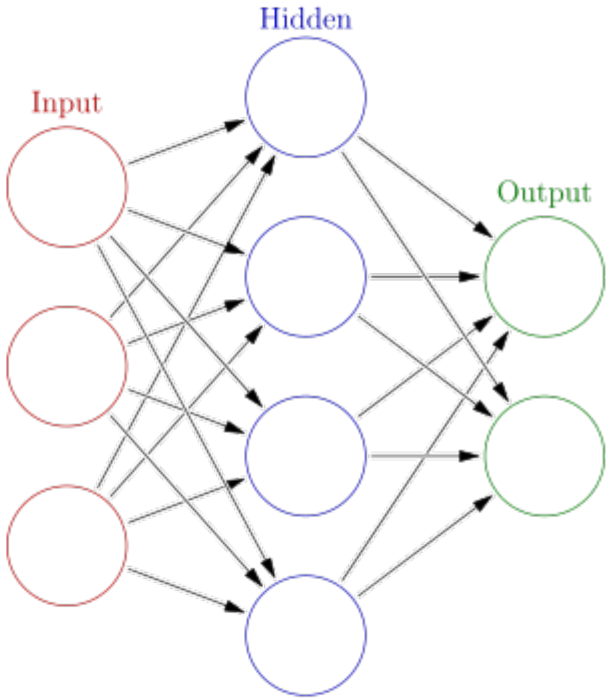
- Overview & terminology
- A very simple example in Python (without libraries)
- Deep learning
- Convolutional neural networks

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 today, you may be able to use a network, but understanding them would mean...

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**

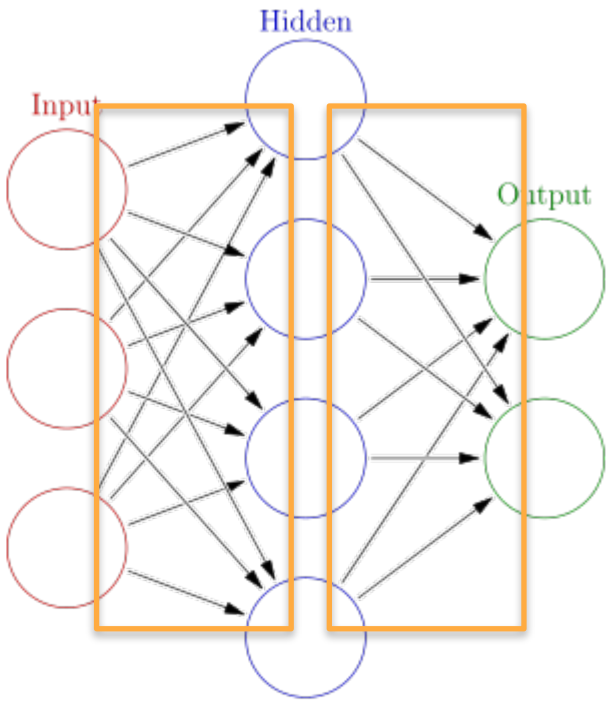
Outputs may be for **classification** (is this an apple or an orange) or **regression** (what is the energy of a particle?)



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**

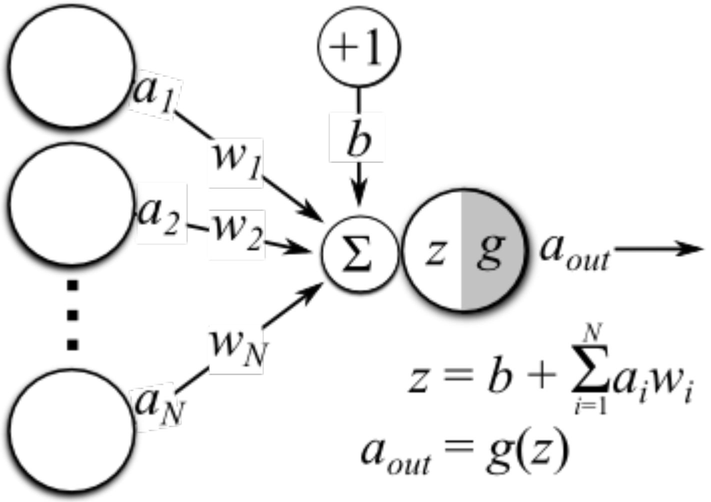
Outputs may be for **classification** (is this an apple or an orange) or **regression** (what is the energy of a particle?)



ANNs

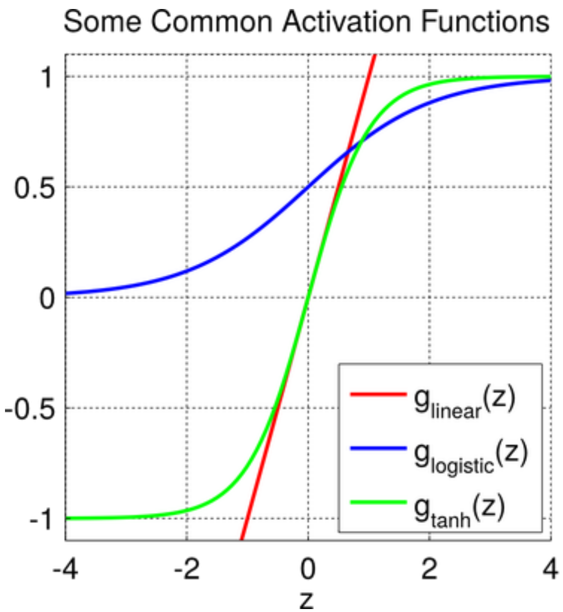
A simple model of a single layer ANN[ref]

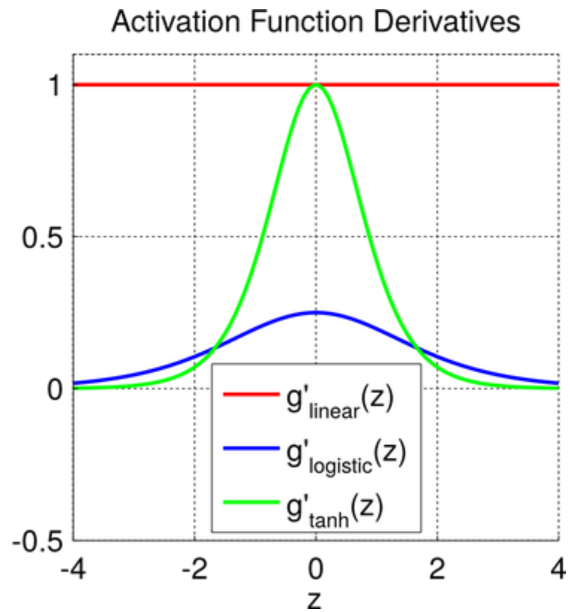
- **Nodes** (inputs): a_1, …, a_N
- **Weights**: w_1, …, w_N
- Neuron **input** value: z
- **Activation function**: g(z)
- **Bias**: b
- **Output**: a_{out} (e.g. {0,1}, {-1,1})

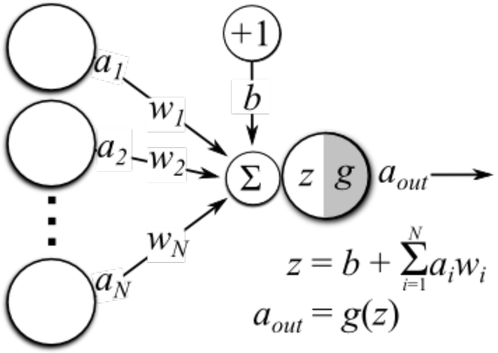


[ref]

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

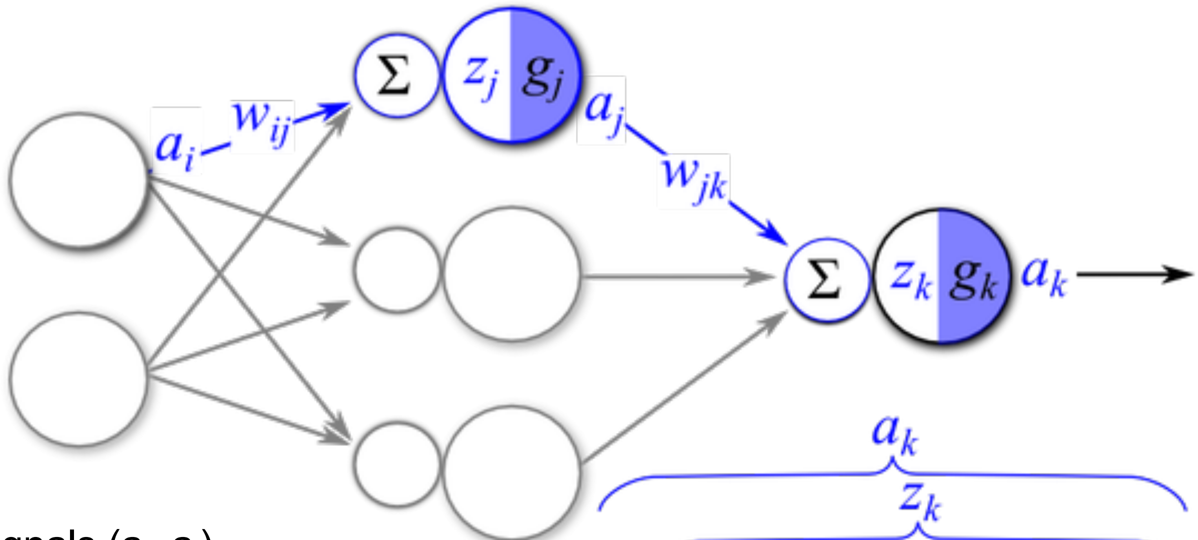
Linear regression: g_{linear}

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

I. Forward-propagate Input Signal



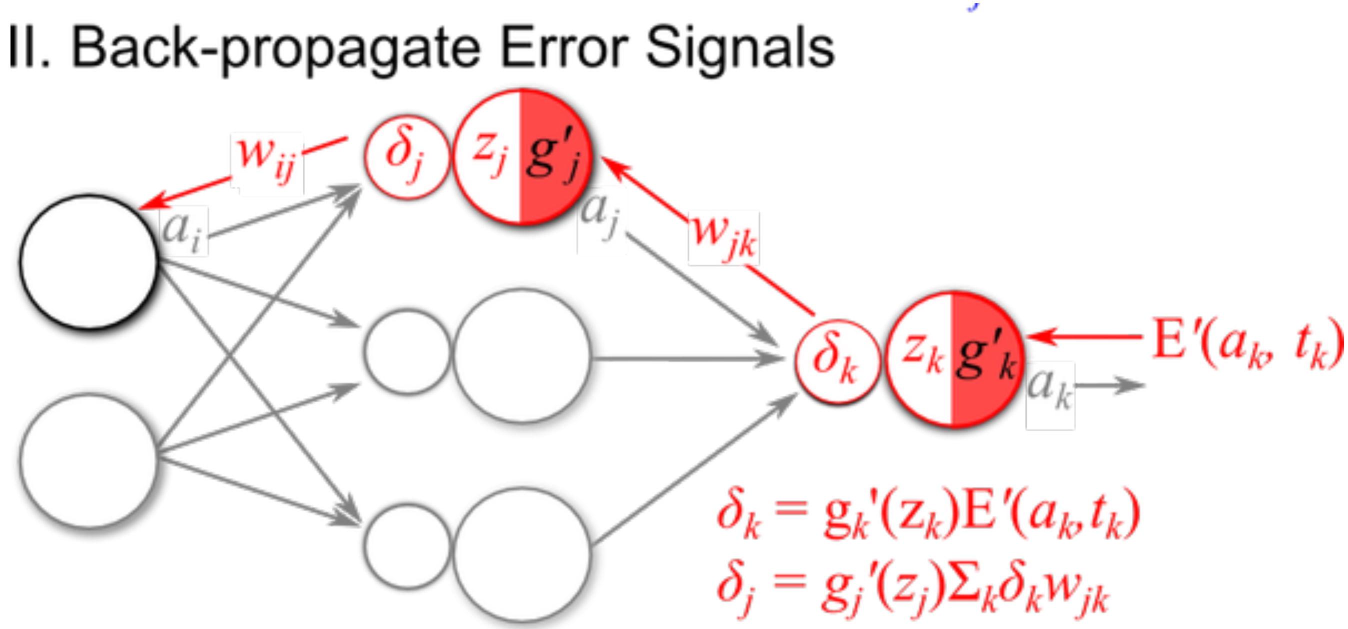
$$a_k = \overbrace{g_k(\overbrace{b_k + \Sigma_j \underbrace{g_j(\underbrace{b_j + \Sigma_i a_i w_{ij}}_{z_j})}_{a_j} w_{jk}}^{z_k})}^{a_k}$$

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).

Gradient Descent & Backpropagation

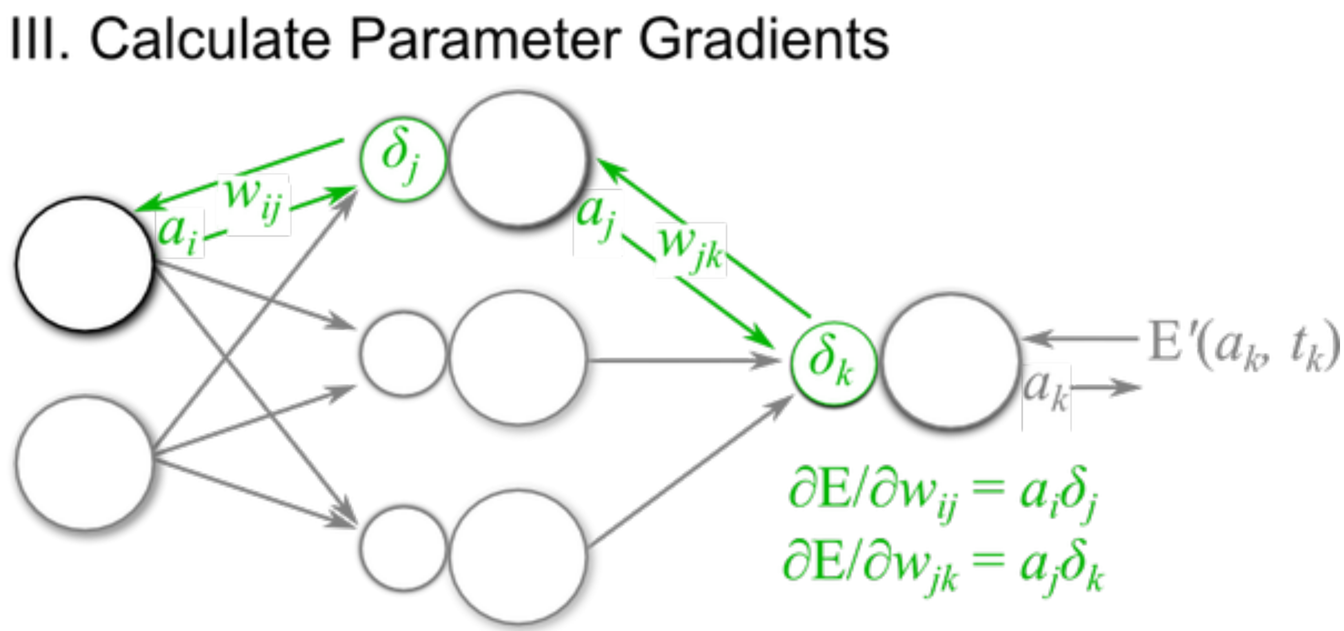


2. **Back propagating** error signals (δ_j, δ_k) through the network to compute differences between each weight.

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

$E=\frac{1}{2}(\text{output - expectation})^2$

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

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

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**.

[ref]

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.

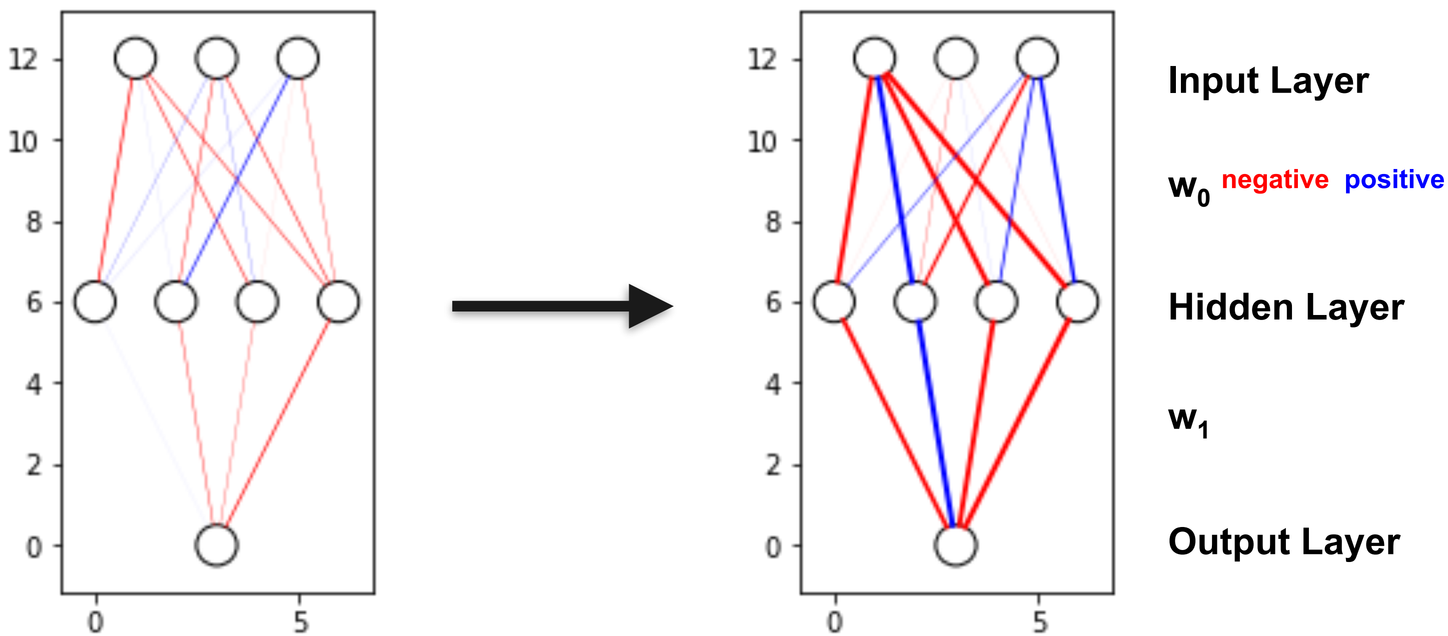
Train ANN to **partially** learn: **a && (b || c)**

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

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 binary classification or linear 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.

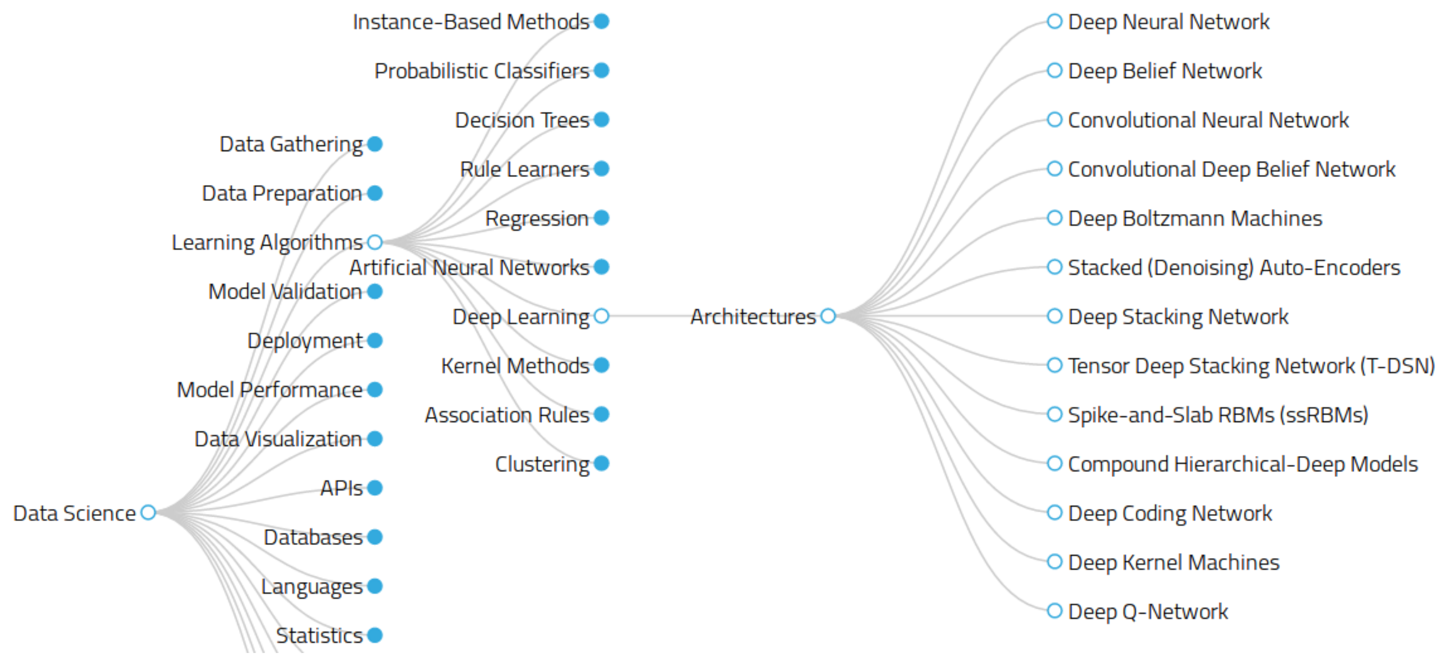
Generative NNs
Text processing or translation
Convolutional NNs (CNNs)
More...

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

GoogleNet

Convolution
Pooling
Softmax
Other

Classes of Deep Learning



CNNs

Machine Learning Approach

Deep Learning

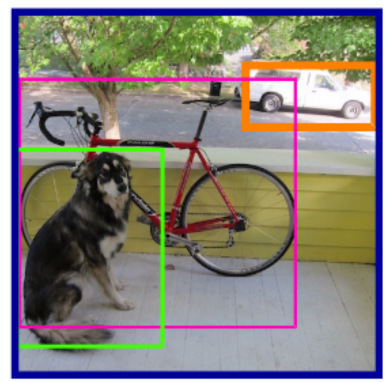
- Focus on a particular method: **Convolutional Neural Networks (CNN)**
  - powerful new technique developed for computer vision
  - *very good at image analysis!*

Image captioning

Car vision

Image classification

Defeating Humans at Go

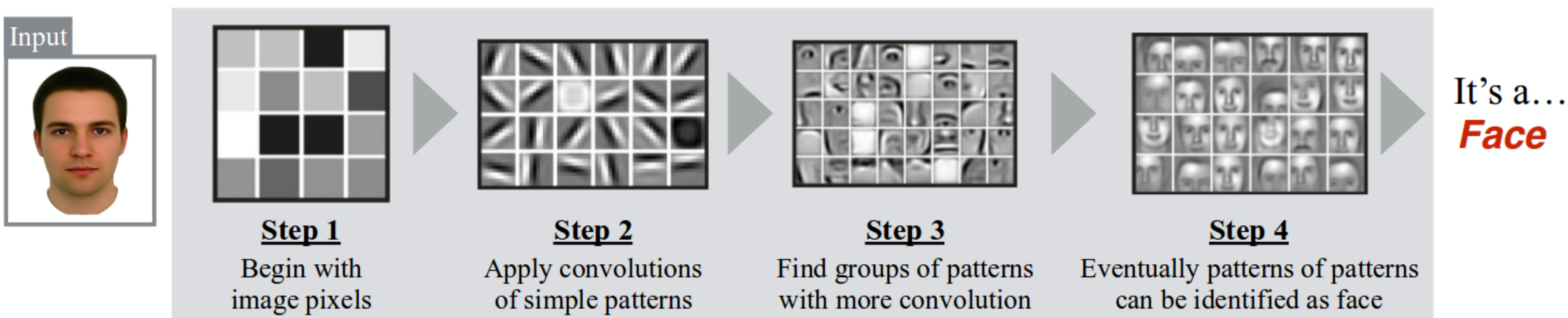


Object Detection

CNNs

Deep Learning - Convolutional Neural Networks

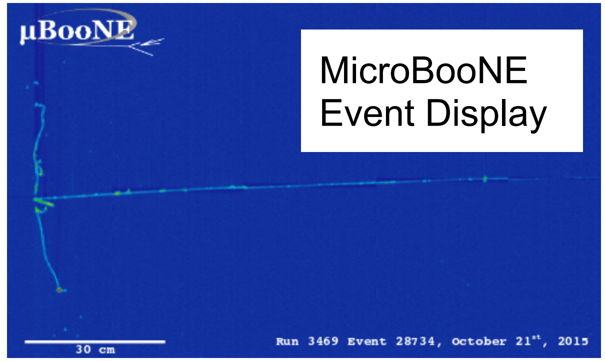
- CNNs can produce representations of high level objects based on low level features
- Consider the task of **facial recognition**…



- Involves training with **labeled data**

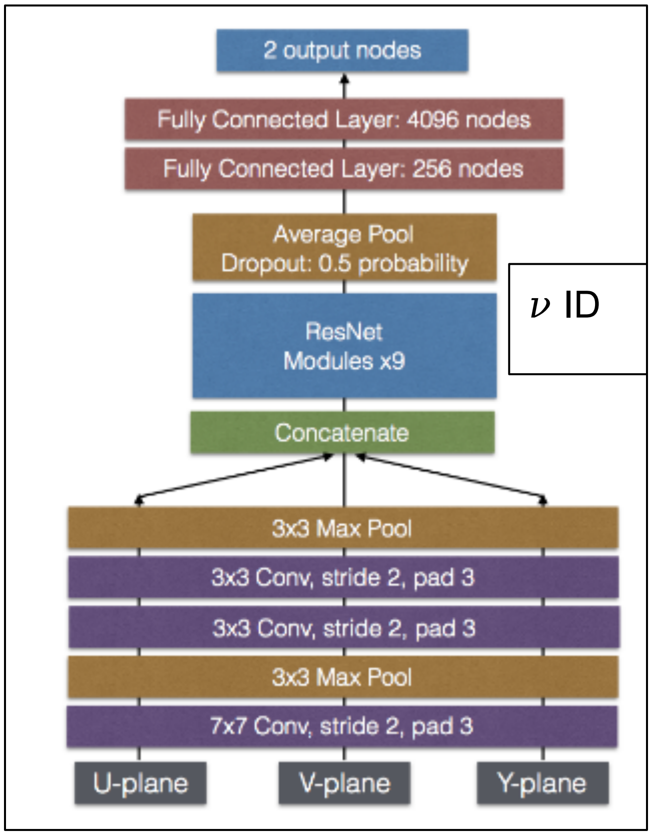
- Network learns features by itself
  - power to **generalize to new data**!

DL Applications: MicroBooNE



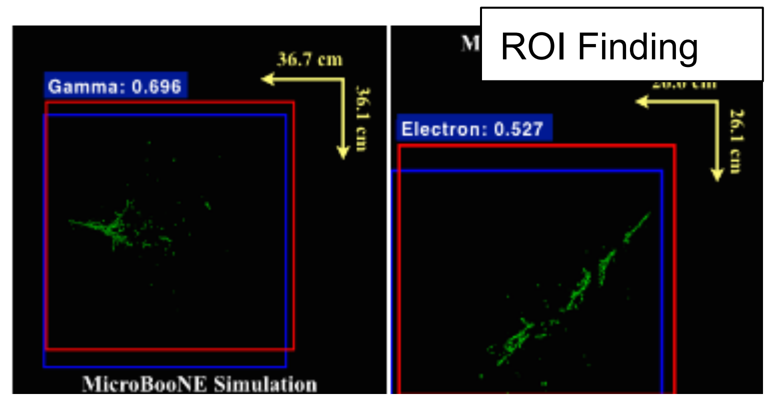
Convolutional Neural Networks Applied to Neutrino Events in a Liquid Argon Time Projection Chamber

MicroBooNE Collaboration



Classification

Image, Network	Classified Particle Type				
	e^- [%]	γ [%]	μ^- [%]	π^- [%]	proton [%]
HiRes, AlexNet	73.6 ± 0.7	81.3 ± 0.6	84.8 ± 0.6	73.1 ± 0.7	87.2 ± 0.5
LoRes, AlexNet	64.1 ± 0.8	77.3 ± 0.7	75.2 ± 0.7	74.2 ± 0.7	85.8 ± 0.6
HiRes, GoogLeNet	77.8 ± 0.7	83.4 ± 0.6	89.7 ± 0.5	71.0 ± 0.7	91.2 ± 0.5
LoRes, GoogLeNet	74.0 ± 0.7	74.0 ± 0.7	84.1 ± 0.6	75.2 ± 0.7	84.6 ± 0.6



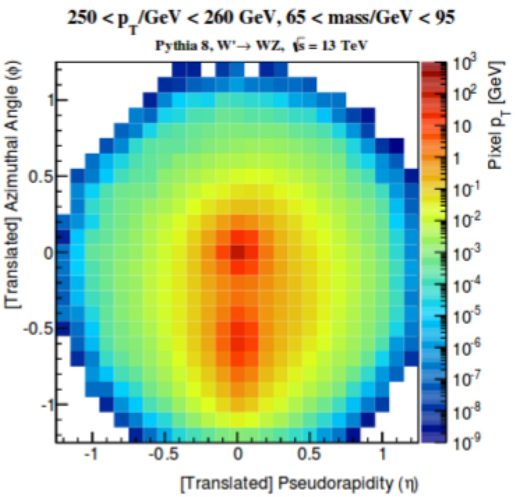
Study of using DL to:
- Classify single particle images
- Locate interactions in an image
- Differentiate neutrino interactions

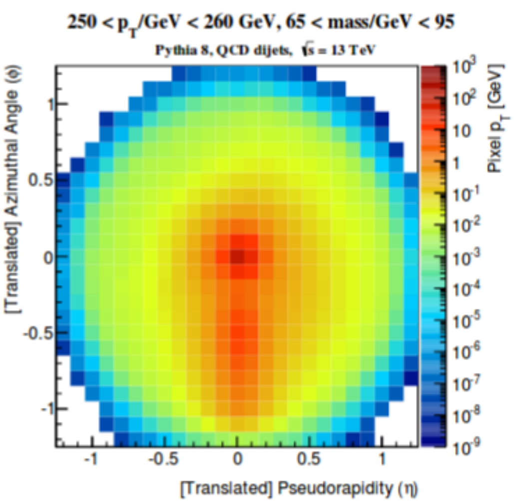
Compare performance of different network architectures on these problems. Published paper arXiv link

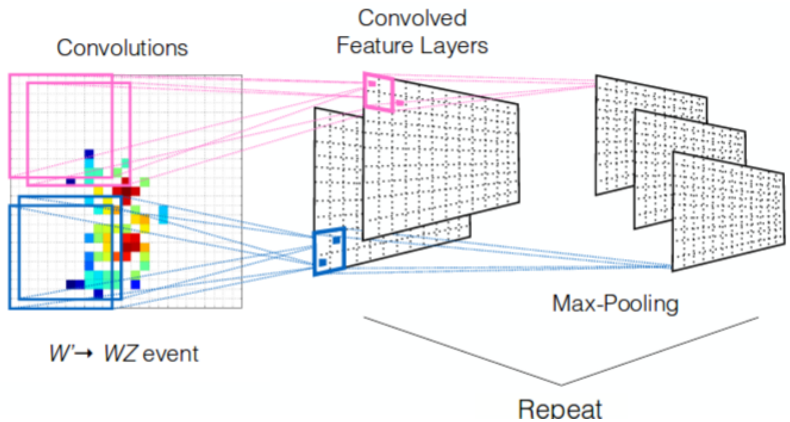
DL Applications: Jet-tagging

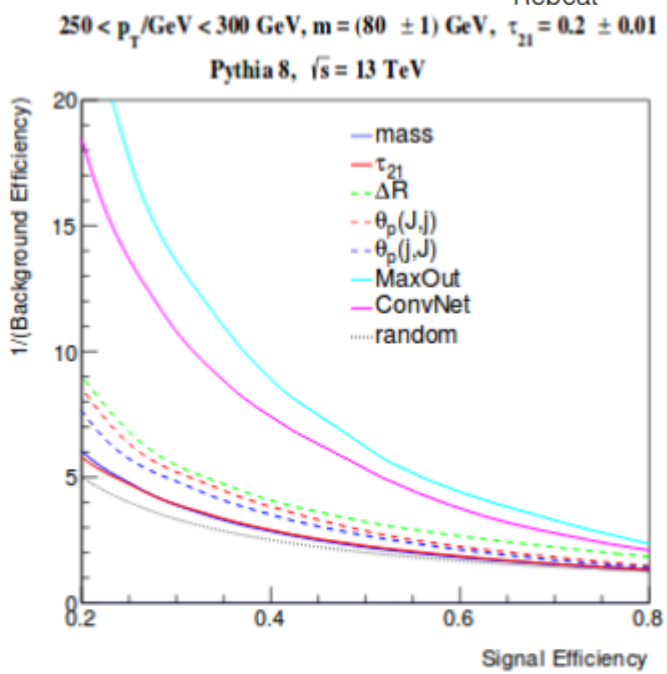
Jet-Images – Deep Learning Edition [link]

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









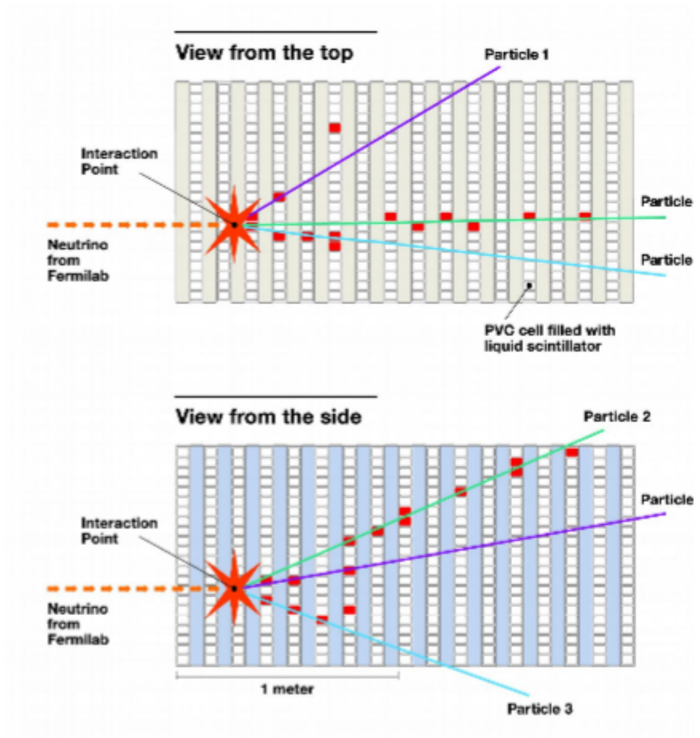
Study of DL methods to distinguish between **highly boosted W boson decays** and backgrounds.

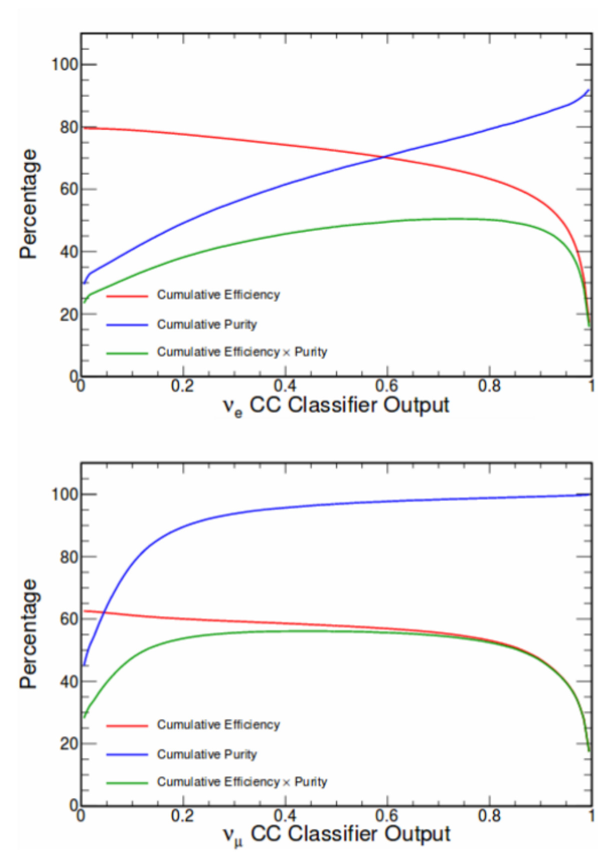
Compares performance of different network architectures on these problems.

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**.

Deep Learning Frameworks

We want to use a **framework** for our ANNs and deep learning for the same reasons we did before:

- 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.

We will use **TensorFlow**.

TensorFlow

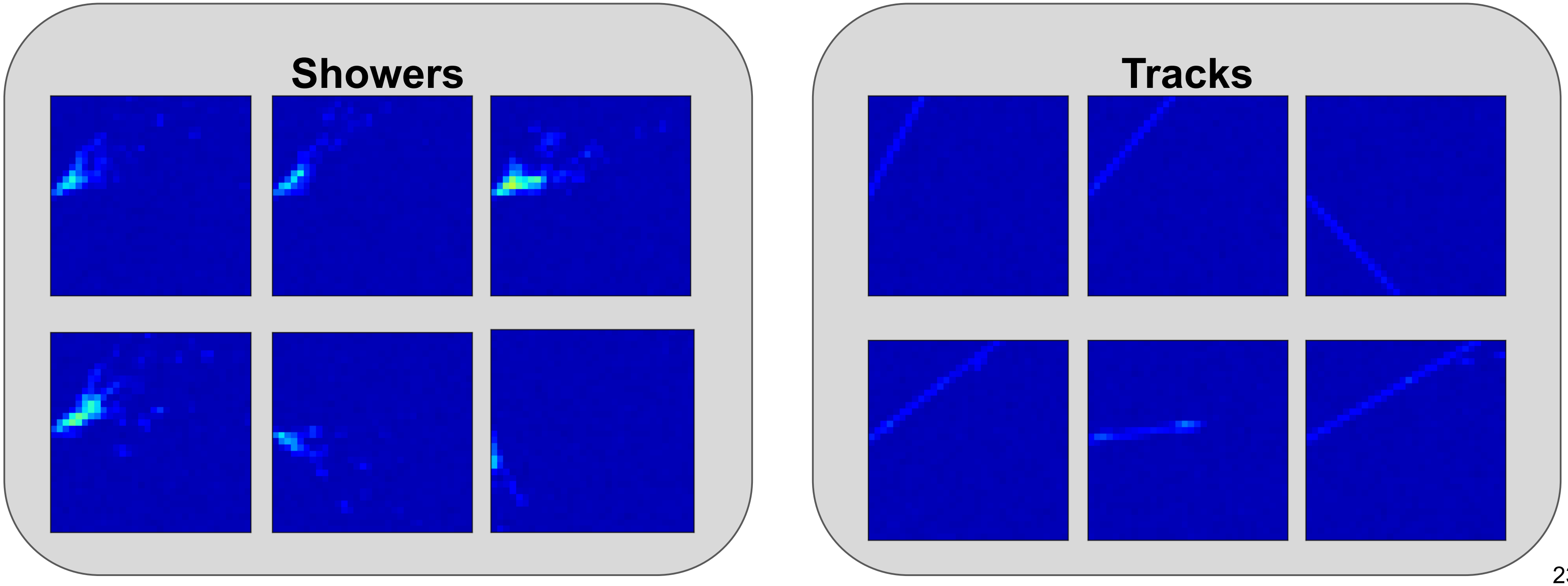
Tensorflow is Google's machine learning library.

- 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)

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

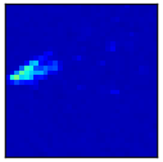
We will train a CNN to distinguish between these showers and track images.

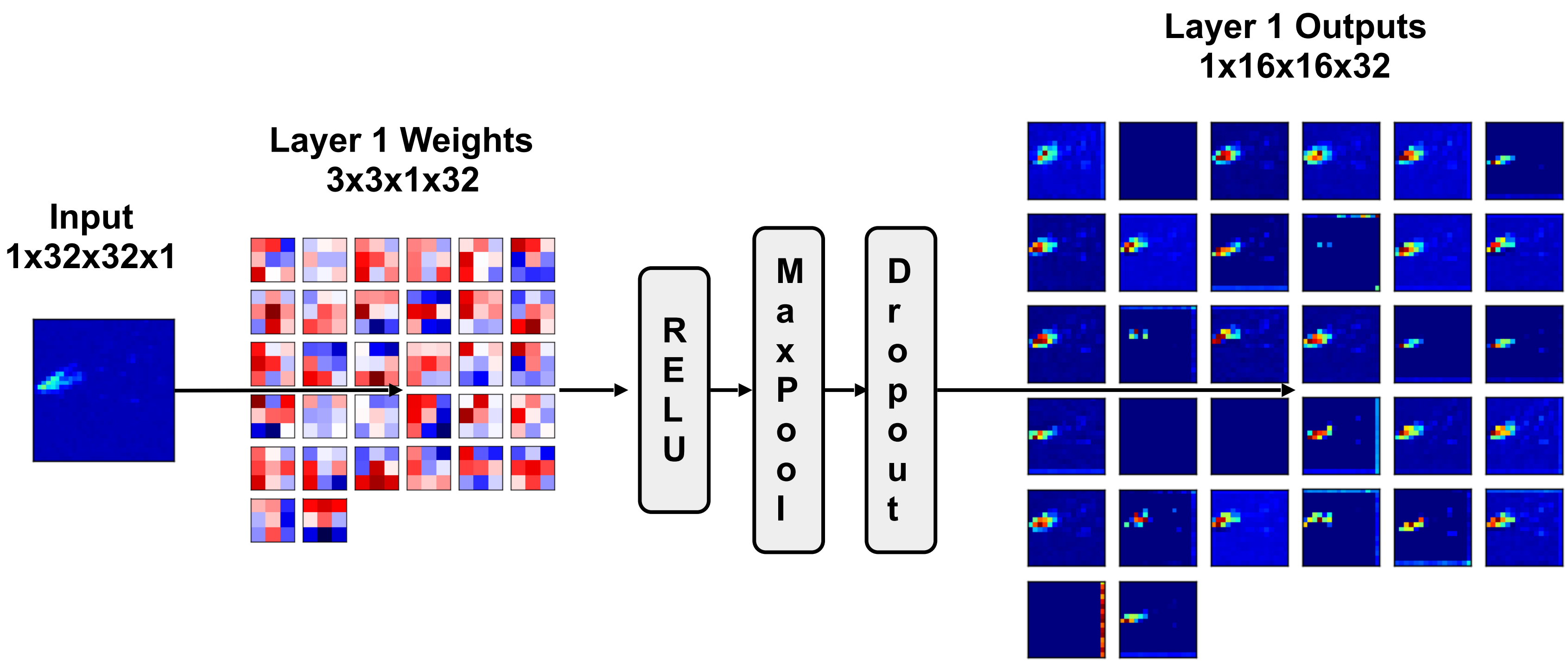
- This is the same network architecture as this MINST tensorflow example.The code for this example is based on this example, in fact.

The example is here.

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

Input
1x32x32x1

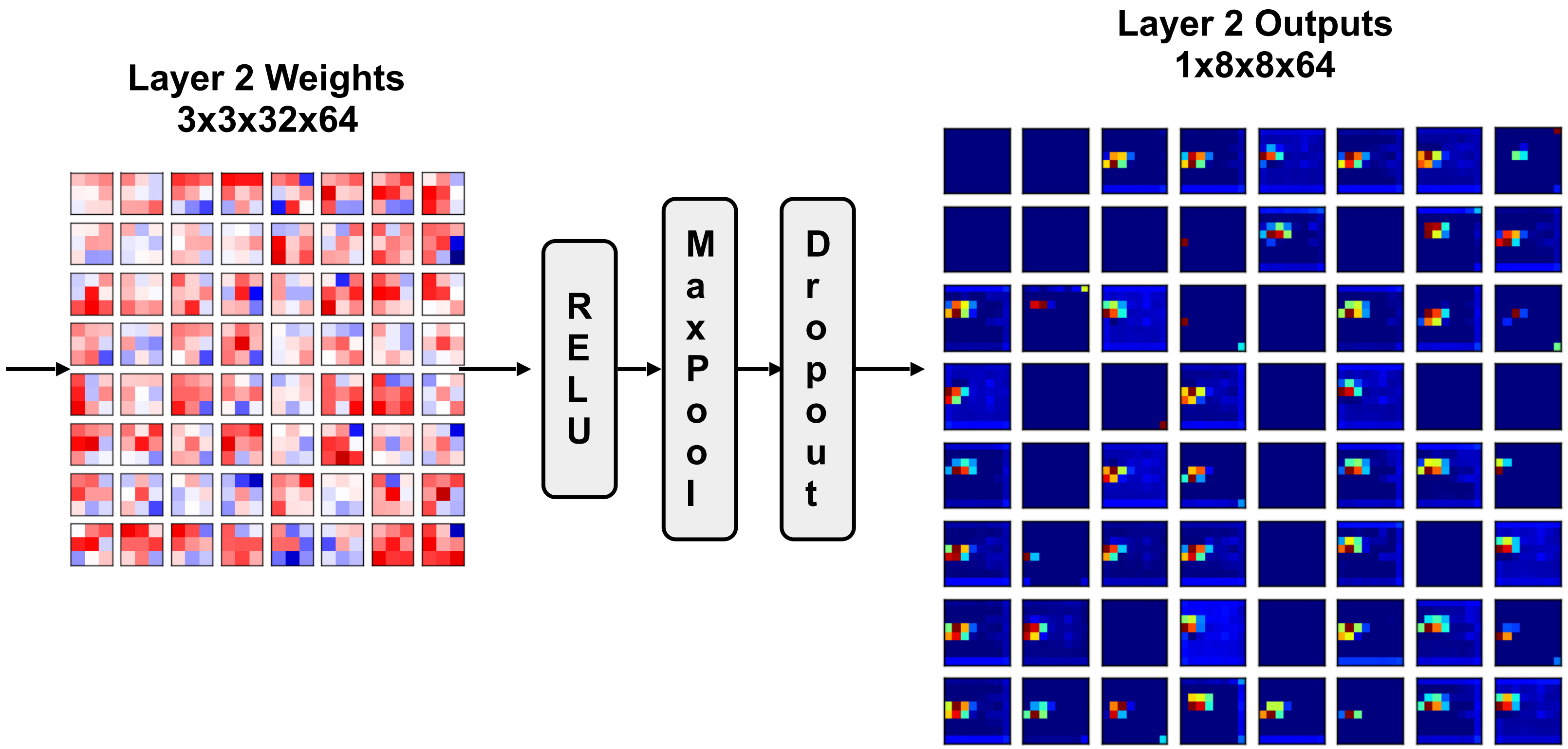




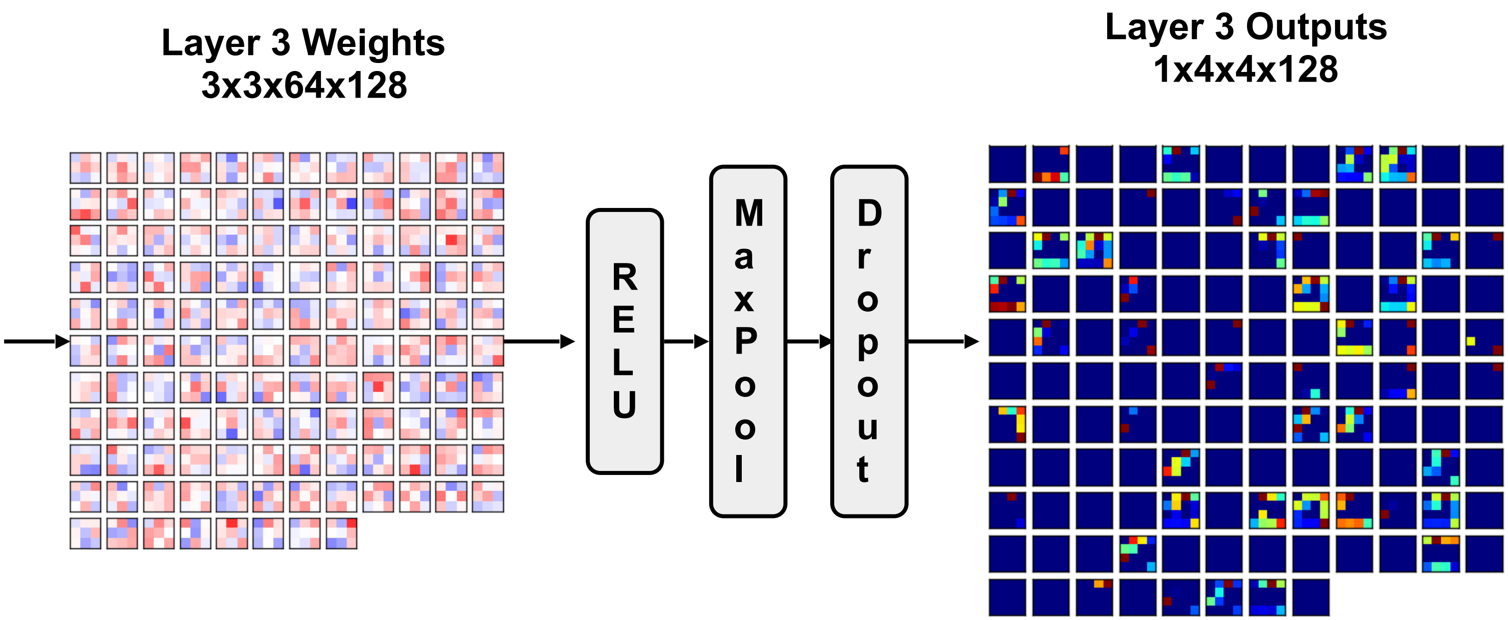
ReLU Activation Function: Rectifying Logistic Unit: g(x)=max(0,x)
MaxPool: Reduces granularity while taking maximum pixel values
Dropout: zeroes out some percentage of entries to prevent overfitting ref

Network Architecture - Layer 1

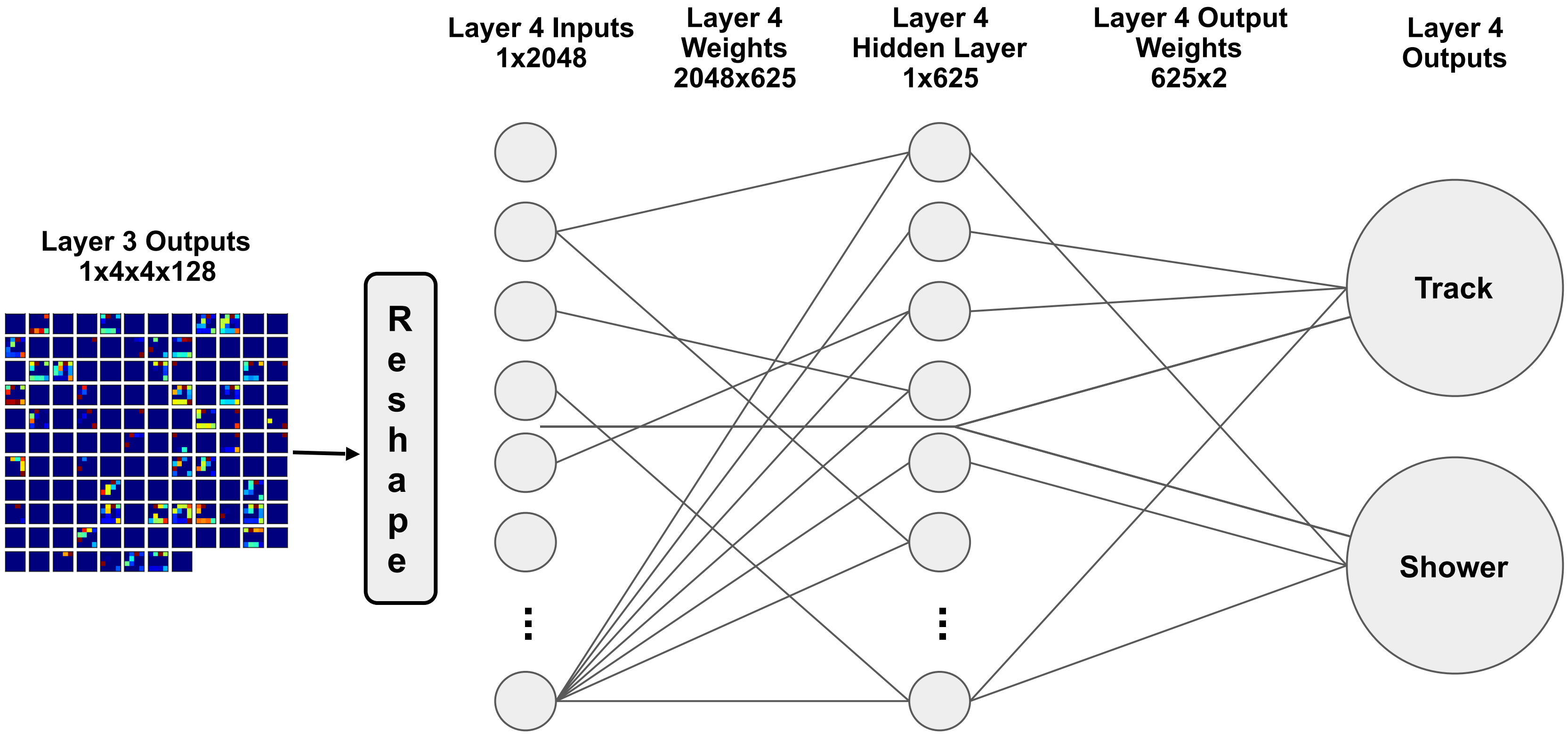
Input tensor format: [batch, in_height, in_width, in_channels]
Filter/kernel tensor format: [filter_height, filter_width, in_channels, out_channels]



Network Architecture - Layer 2



Network Architecture - Layer 3



Network Architecture - Layer 4
Fully Connected Layer

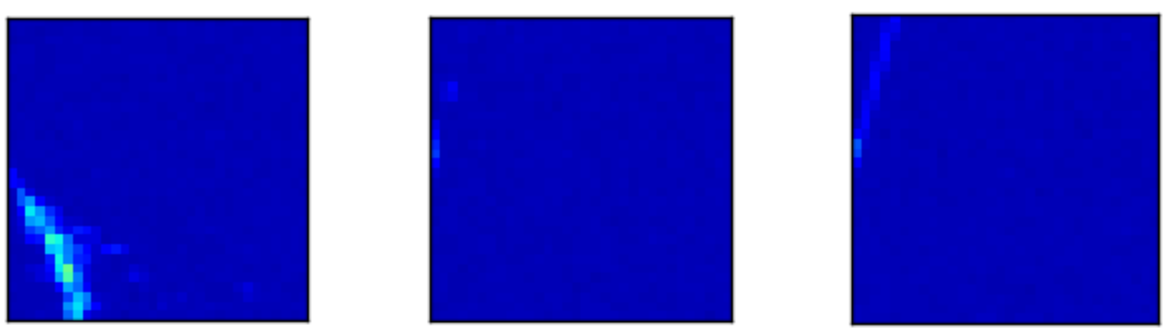
CNN Performance

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

The network is trained by minimizing the loss (error) using a **softmax function** (multi-dimensional logistic function).

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

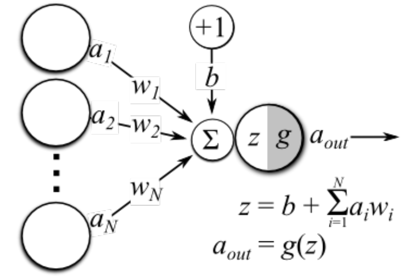
Tensorflow Installation

Follow the instructions here to install tensorflow in your anaconda environment.

Or, this worked for me: **conda install -c conda-forge tensorflow**

Note: you will need to reinstall matplotlib, seaborn, etc. if you create a new environment. You could opt to just install it in your default environment (this worked for me.)

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.

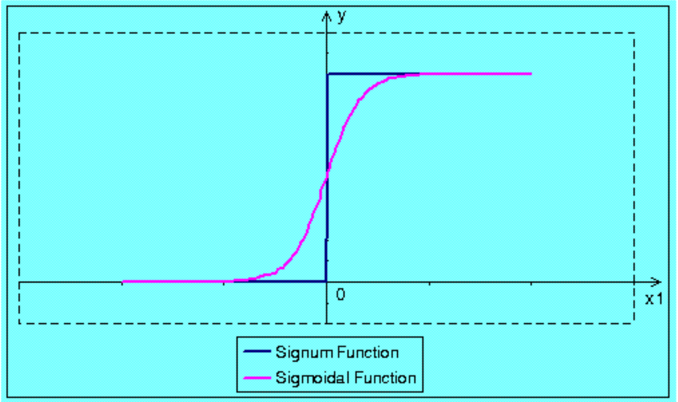


fig 1a. Neuron without bias activate function. Signum and sigmoidal function.

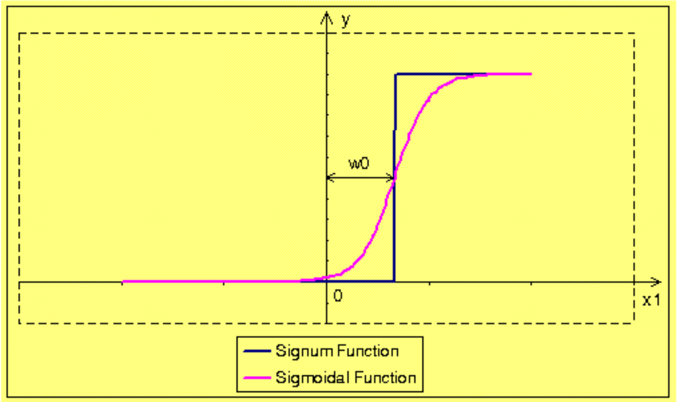


fig 1b. Neuron with bias activate function. Signum and sigmoidal function.