How easily can neural networks learn relativity?

ACAT 2017

Kartik Chitturi
Dr. Peter Onyisi

August 24th, 2017

TEXAS
The University of Texas at Austin

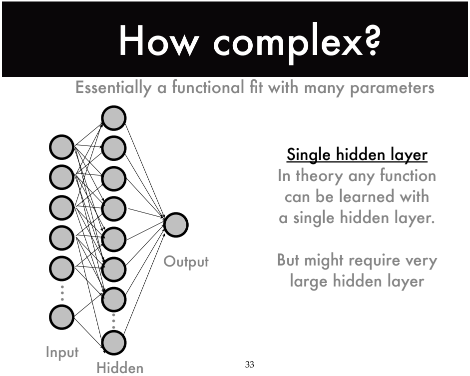
Introduction

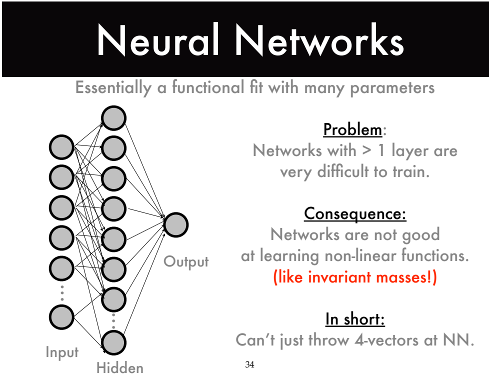
- Relativistic invariants are key variables in HEP problems and are believed to be learnt implicitly by deep learning approaches.

- We investigate the minimum network complexity needed to accurately extact such invariants. Doing so will help us understand how complex a neural net needs to be to obtain certain functions.

Software and Hardware

We used Keras (with a Tensorflow backend) on a NVIDIA Tesla K80.





Taken from Daniel Whiteson, *Deep Learning in Particle Physics*, ACAT, August 23rd, 2017

Neural Network Architectures

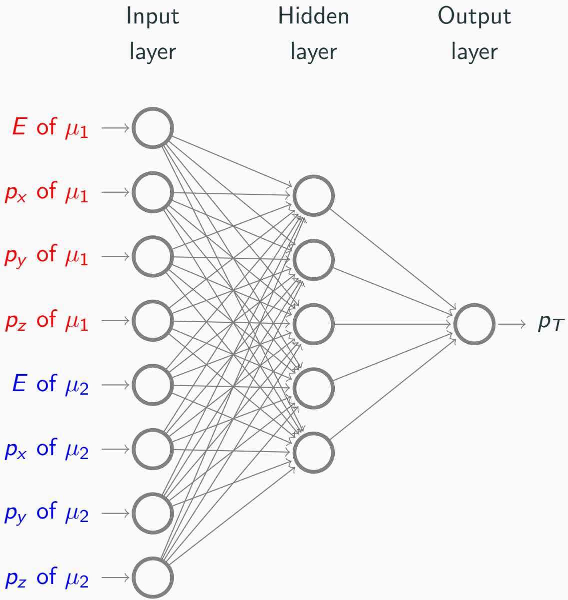
We take in 4-vectors of events and study the following problems using NNs.

- p_T of $Z \to \mu\mu$
- p_T of $t\bar{t} \to \mu\mu$
- Invariant mass of $t\bar{t} \to \mu\mu$

We adjusted these hyperparameters and studied how this affected accuracy:

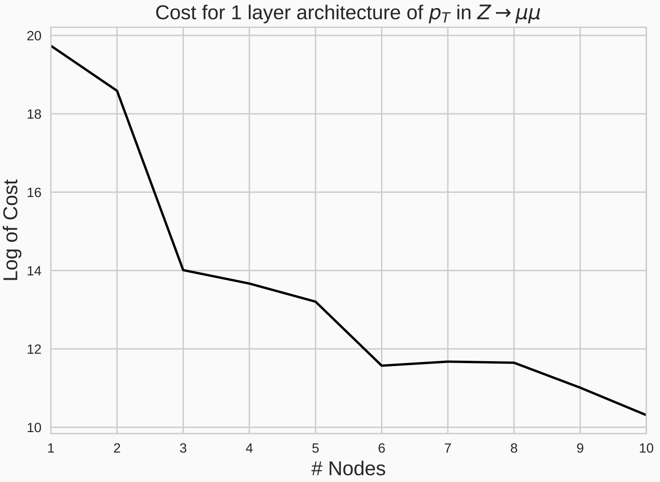
- # Nodes
- # Layer
- Activation Function

Finding p_T of $Z \to \mu\mu$ and $t\bar{t} \to \mu\mu$


Input layer
Hidden layer
Output layer
E of μ_1
p_x of μ_1
p_y of μ_1
p_z of μ_1
E of μ_2
p_x of μ_2
p_y of μ_2
p_z of μ_2
p_T

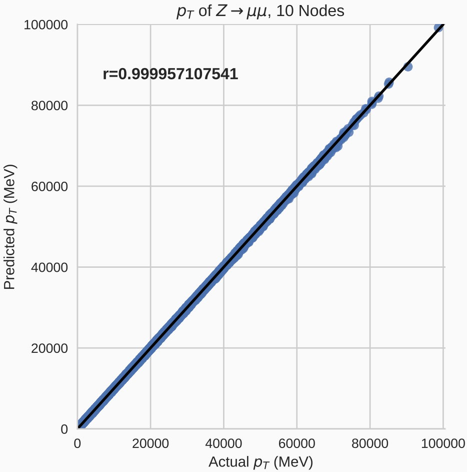
$Z \to \mu\mu$

We see that with a single layer, we train well.



$Z \to \mu\mu$

1 layer with 10 nodes is enough for almost perfect accuracy.



$Z \to \mu\mu$

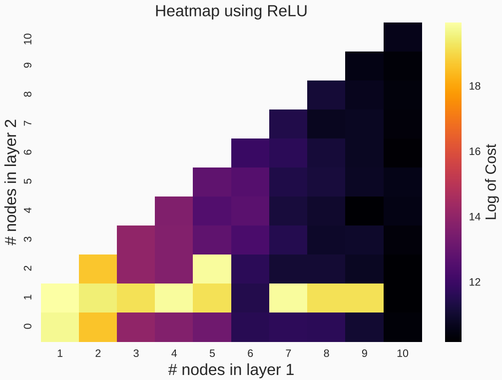
Looking at the weights generated for the hidden layer, we find:

- E and p_z are given weights of 0
- The weights for p_x and p_y are symmetric for both muons

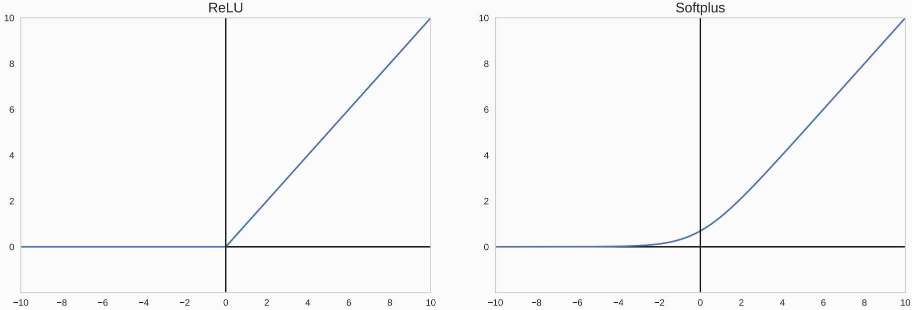
Thus neural network is learning a **non-linear** function of $(p_{x1} + p_{x2})$ and $(p_{y1} + p_{y2})$

$Z \to \mu\mu$

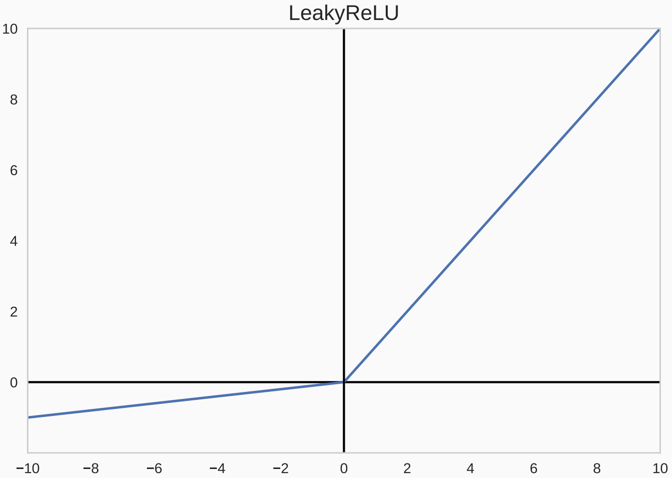
Adding an additional layer provided no increase in accuracy. However accuracy actually dipped with both softplus and ReLU activations for when we had a single node in the 2nd layer. This same problem occurs for p_T of $t\bar{t} \to \mu\mu$.



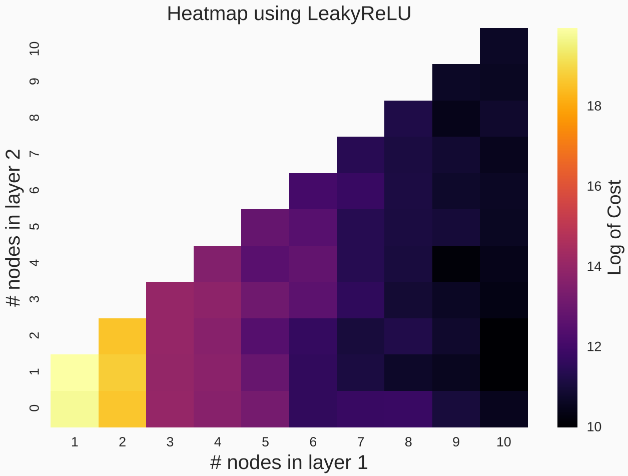
ReLU and Softplus suffer from having a zero gradient for negative inputs.
During training, neurons ‘die’ when they get put in a state where the output 0 for all inputs.



LeakyReLU was specifically made to fix this.

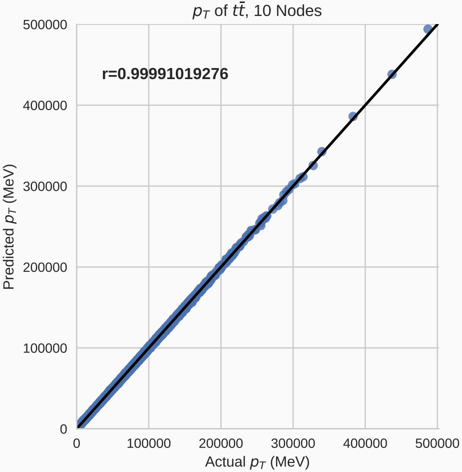


Thus using LeakyReLU made this problem go away
Dropout may have also worked.



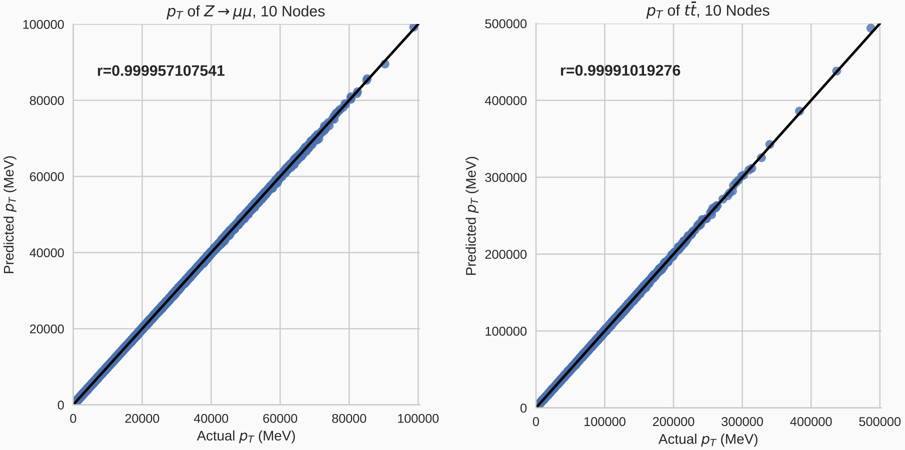
$t\bar{t} \to \mu\mu$

We see that the same accuracy occurs with $t\bar{t}$ confirming sample independence for finding p_T of dimuon production.



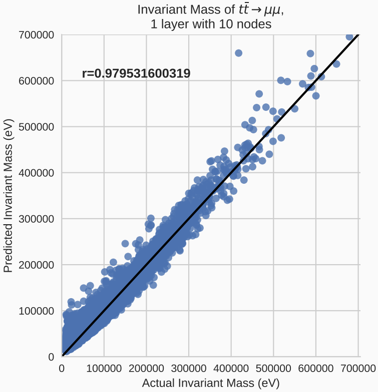
Finding Invariant Mass of $t\bar{t} \rightarrow \mu\mu$

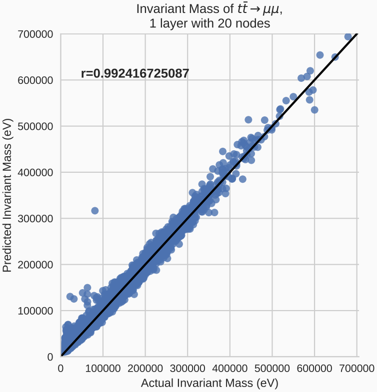
As seen before, a single layer with 10 nodes can very accurately predict p_T.



Invariant Mass of $t\bar{t} \to \mu\mu$

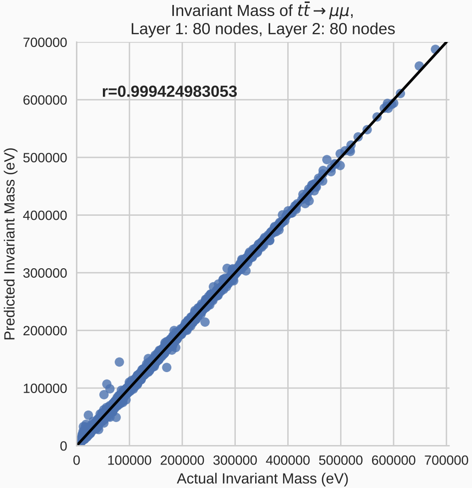
Using the same neural net architecture (1 layer with 10 nodes, LeakyReLU) we get a lower accuracy ($r = 0.97$).
We get a better accuracy with 20 nodes in a single layer





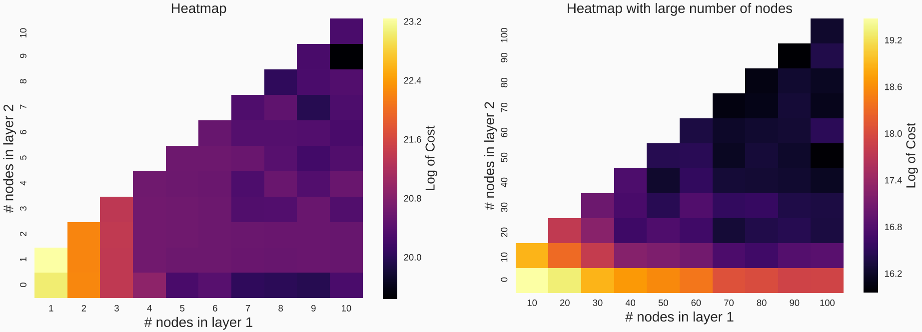
Invariant Mass of $t\bar{t} \to \mu\mu$

We finally achieve similar accuracy as p_T when given a large number of nodes in 2 layers.



Invariant Mass of $t\bar{t} \to \mu\mu$

We see little change happening across neural nets when they have a low number of nodes.



Summary

Used neural nets for regression of transverse momentum and invariant mass in two-body systems.

LeakyReLU was used in all neural nets (other activation functions were experimented with).

A single layer with 9/10 nodes has almost perfect accuracy for finding p_T (in a certain range) of dimuon production.

Sample-independence for finding p_T was shown through $t\bar{t}$.

Still trying to understand what features of the mass problem is causing the issues.

We plan to look into other invariants such as decay angles.