Predicting Neural Networks' accuracies from their architectural characterizations.

Duc Hoang

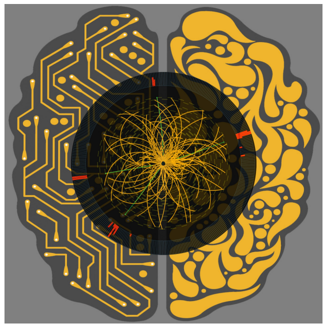
Supervisor: Dr. Gabriel N. Perdue

SIST – Introduction Presentation

10 June 2019

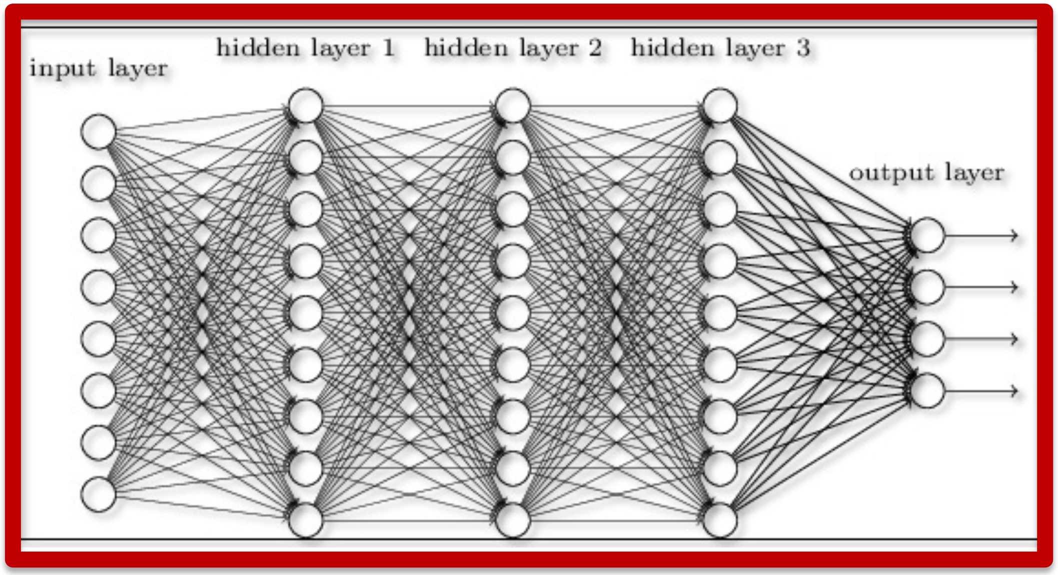
Artificial Neural Networks

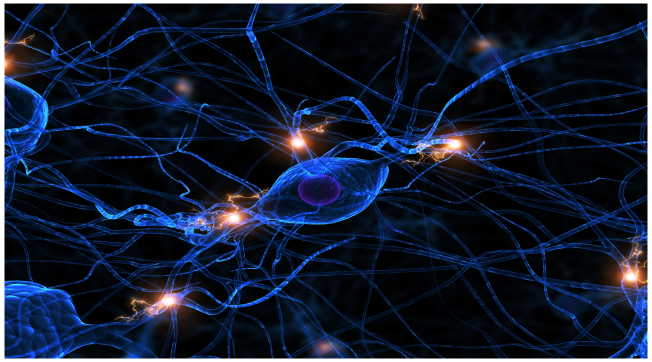
- One of the main tools used in Machine Learning, which is a sub-field of Artificial Intelligence that uses statistical techniques and pattern recognition to train computers to "learn" without being explicitly programmed.
- It's extremely efficient in analyzing large datasets!!!
- But why is it relevant to Fermilab?



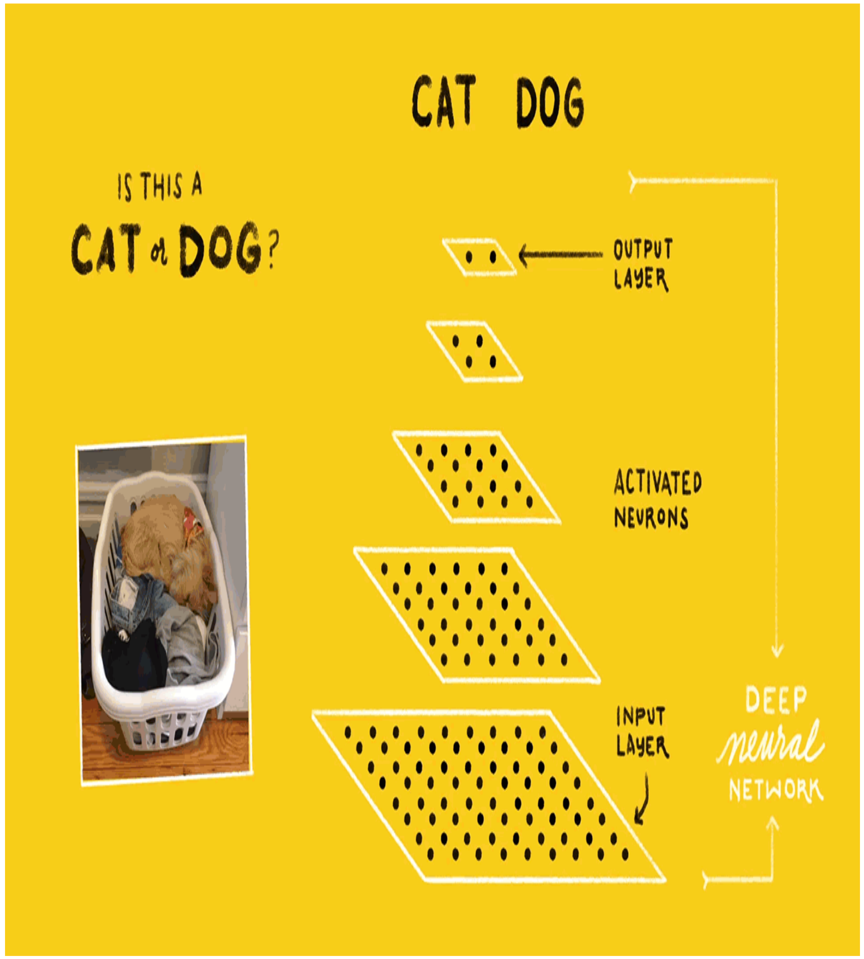
Artificial Neural Networks

- Common depictions of Artificial Neural Networks that you will see on the internet:

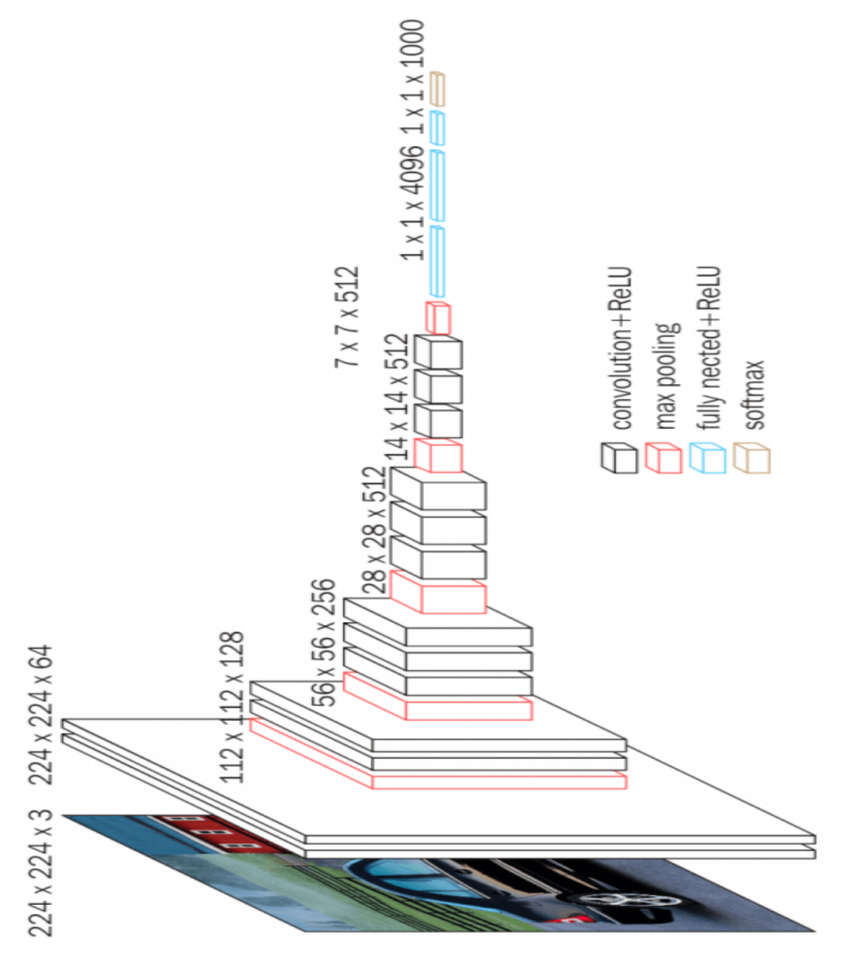




Convolutional Neural Networks (CNNs)



Source



Source

Designing CNNs is similar to constructing a building

Source

- The actual buildings (or CNNs models in this case) are based on pre-designed architecture.
- However, Neural Networks' architecture is designed based on a trial-and-error process. Build -> test accuracy -> tuning and run it again until it works. (Imagine doing it for a building)
- Time consuming and computationally expensive. It's also hard to predict the impact of changing one parameter in the network on its final performance.

My Project

- I'm going to pick up a grad student's project last summer:
  - He wrote some codes to parse records of the networks structures and compute some features. I am extending that code and doing statistical analysis (maybe doing some machine learning) on those features.
  - The dataset currently has ~141k samples (these networks are originally designed to solve MINERvA vertex finding problem). This dataset will also extend in the future.
  - Study the relationships between different architectural characterizations (depth, number of convolutional layers, …) of the networks and their accuracies.

Back up: feature list

1. average depth [net depth avg]
2. number of convolutional layers [avg num conv layers]
3. number of pooling layers [avg num pooling layers]
4. average number of neurons of fully-connected layers [avg IP neurons]
5. average number of weights of fully-connected layers [avg IP weights]
6. average number of feature maps in convolutional layers [num conv features]
7. proportion of convolutional layers followed by a pooling layer [prop conv into pool]
8. proportion of pooling layers followed by a pooling layer [prop pool into pool]
9. proportion of convolutional layers with 1×1 kernels [prop 1x1 kernels]
10. proportion of convolutional layers with square kernel-shapes [prop square kernels]
11. proportion of convolutional layers with horizontally-oriented kernels [prop horiz kernels]
12. proportion of convolutional layers with vertically-oriented kernels [prop vert kernels]
13. number of ReLU-activated convolutional layers [num relu]
14. number of sigmoid-activated convolutional layers [num sigmoid]
15. average percent reduction in activation grid area/height/width between consecutive convolutional layers [avg grid reduction area/height/width consecutive]
16. average percent reduction in activation grid area/height/width between input layers and final con- volutional layers [avg grid reduction area/height/width total]
17. proportion of convolutional layers using non-overlapping stride [prop nonoverlapping]
18. average convolutional stride height/width [avg stride h/w]
19. average ratio of features to depth of convolutional layers [avg ratio features to depth]
20. average of features to kernel area/height/width of convolutional layers [avg ratio features to kerArea/Height/Width]
21. average ratio of kernel area/height/width to depth of convolutional layers [avg ratio kerArea/Height/Width to depth]