ARTIFICIAL NEURAL NETWORKS

BY: MICHAEL STANCZYK

WHAT ARE NEURAL NETWORKS?

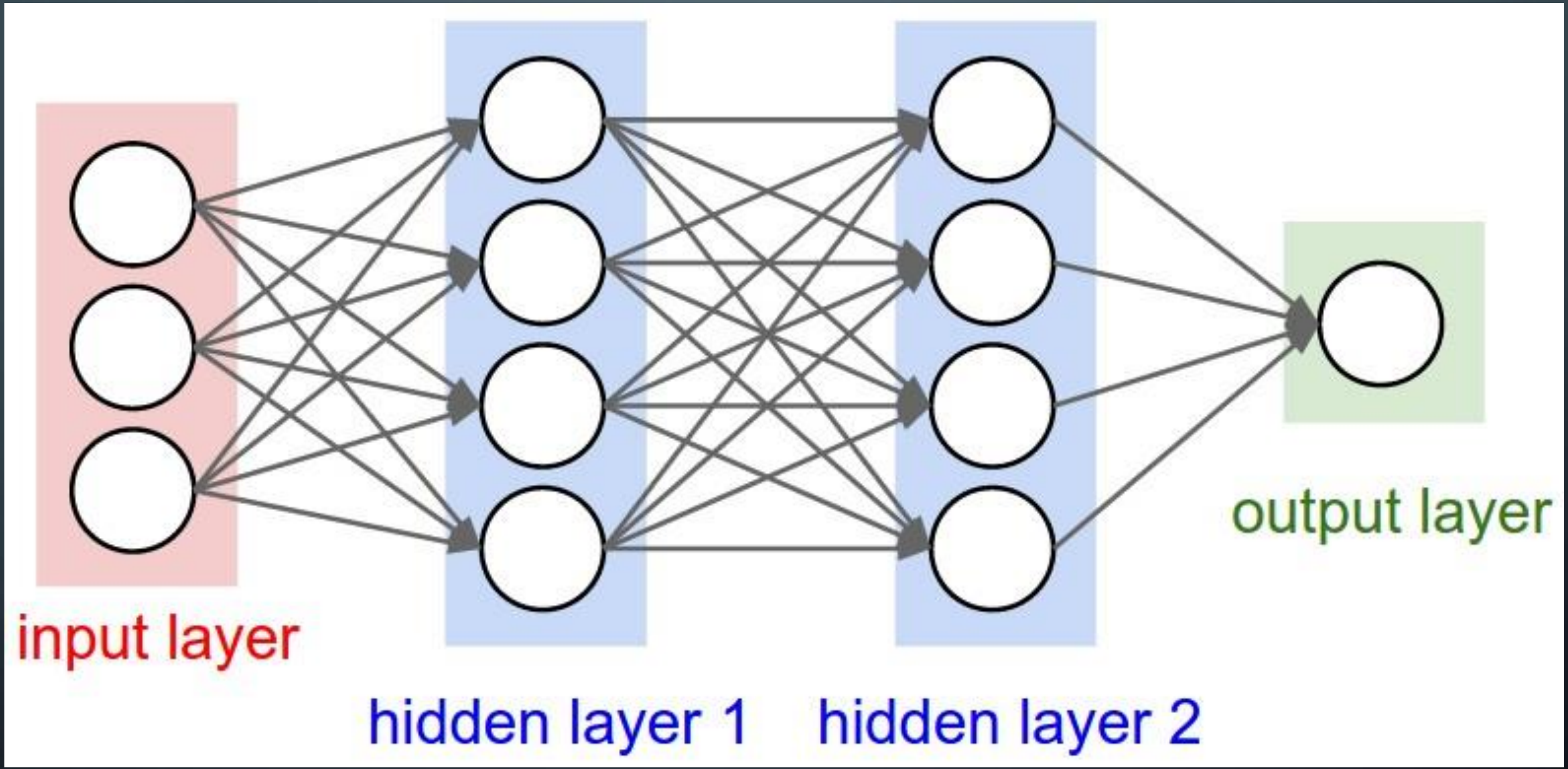
- “A computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs”
- “Artificial” Neural Networks are loosely modeled after the neuronal structure of our cerebral cortex

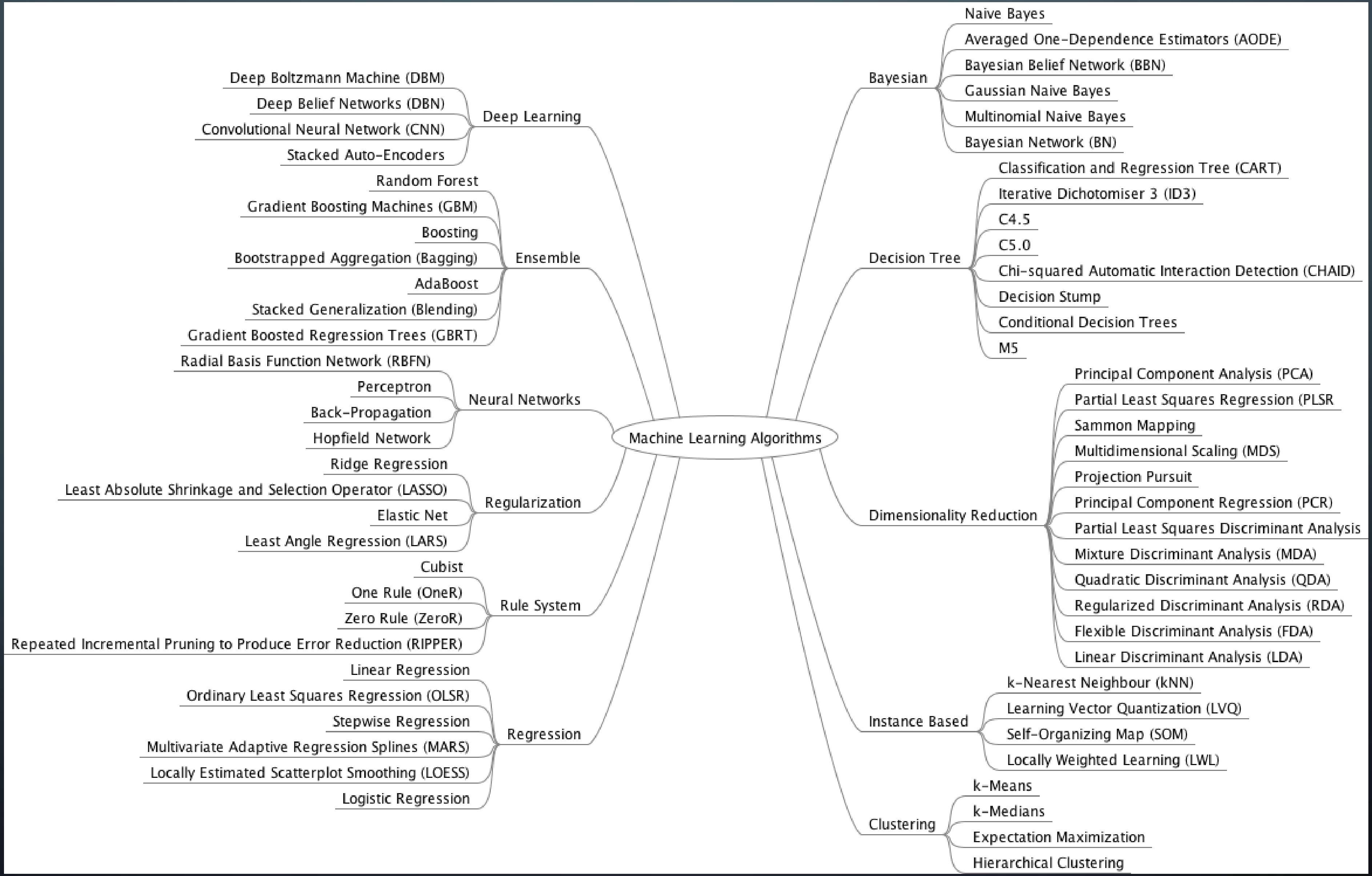
USES FOR NEURAL NETWORKS

- Data Mining
- Image Recognition
- Artificial Intelligence
- Grouping
- Language Processing
- Medical Purposes
- Energy Saving
- And much more

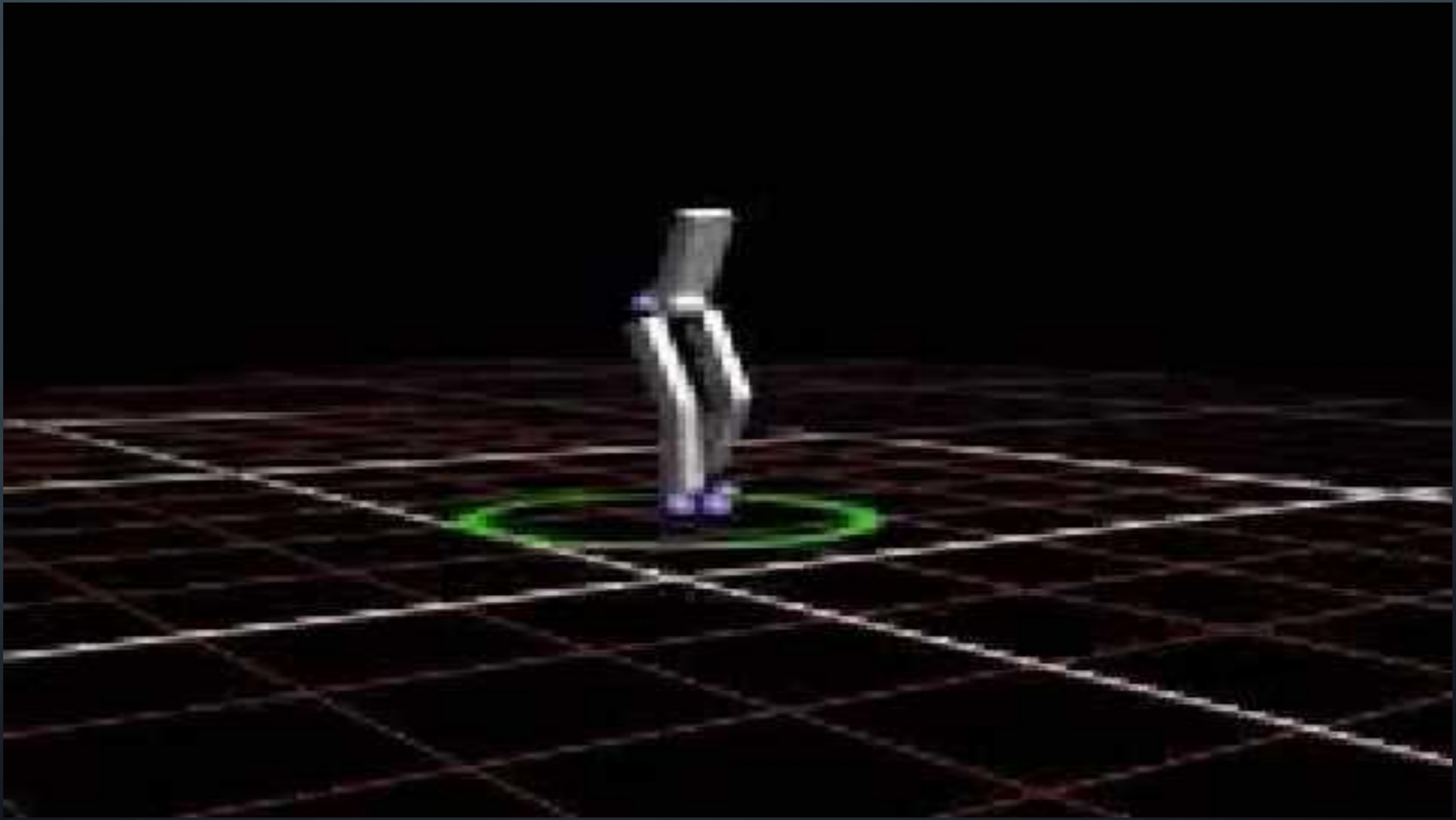
LAYERS

- There are (generally) 3 layers when building an ANN (Artificial Neural Network)
- Input Layer
  - Dynamic Variables that are needed to find the output
- Hidden Layers
  - Where most of the computation occurs
  - Misc. Algorithms applied, sometimes mix and matching multiple algorithms
- Output Layer
  - Predicted Answer


input layer
hidden layer 1
hidden layer 2
output layer



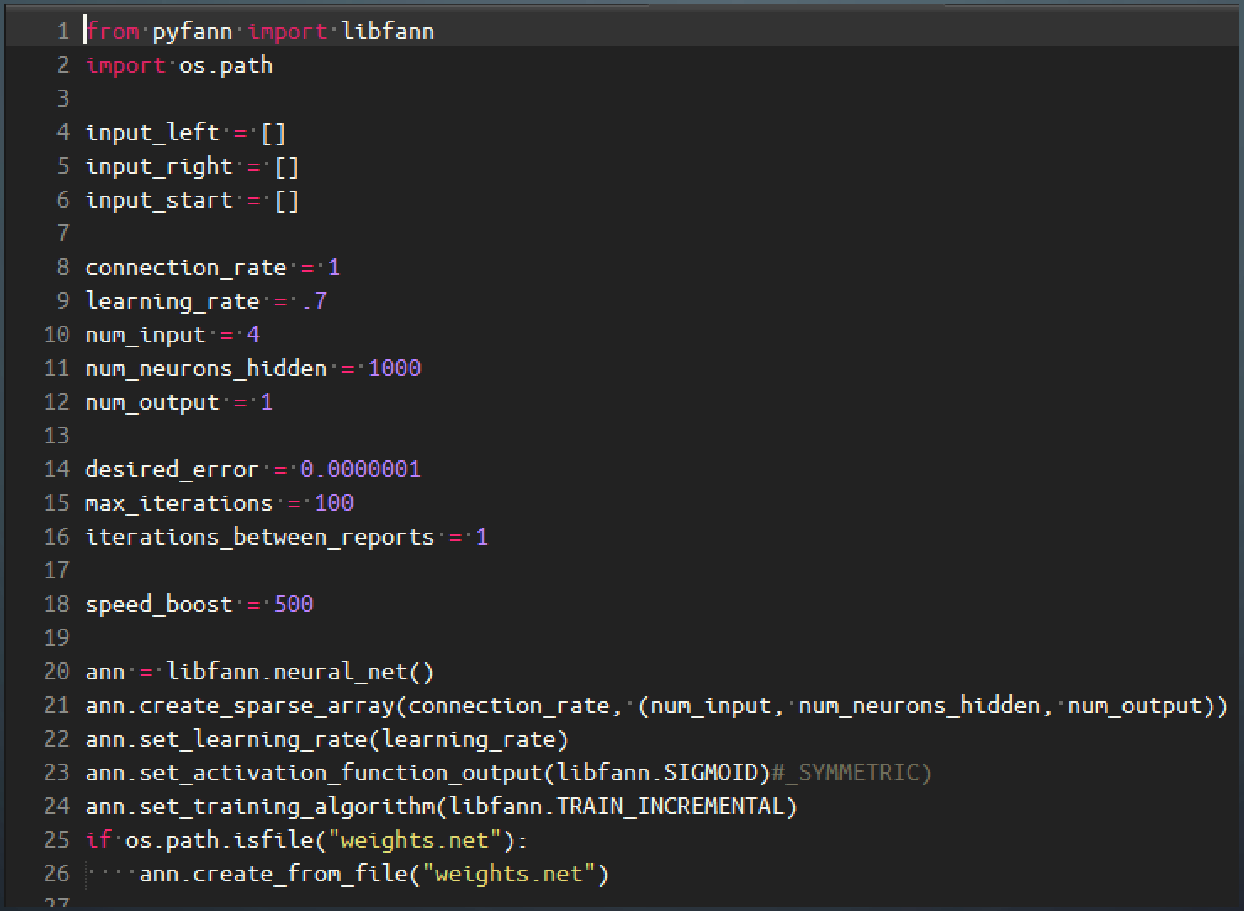
- Machine Learning Algorithms
  - Deep Learning
    - Deep Boltzmann Machine (DBM)
    - Deep Belief Networks (DBN)
    - Convolutional Neural Network (CNN)
    - Stacked Auto-Encoders
  - Ensemble
    - Random Forest
    - Gradient Boosting Machines (GBM)
    - Boosting
    - Bootstrapped Aggregation (Bagging)
    - AdaBoost
    - Stacked Generalization (Blending)
    - Gradient Boosted Regression Trees (GBRT)
  - Neural Networks
    - Radial Basis Function Network (RBFN)
    - Perceptron
    - Back-Propagation
    - Hopfield Network
  - Regularization
    - Ridge Regression
    - Least Absolute Shrinkage and Selection Operator (LASSO)
    - Elastic Net
    - Least Angle Regression (LARS)
  - Rule System
    - Cubist
    - One Rule (OneR)
    - Zero Rule (ZeroR)
    - Repeated Incremental Pruning to Produce Error Reduction (RIPPER)
  - Regression
    - Linear Regression
    - Ordinary Least Squares Regression (OLSR)
    - Stepwise Regression
    - Multivariate Adaptive Regression Splines (MARS)
    - Locally Estimated Scatterplot Smoothing (LOESS)
    - Logistic Regression
  - Bayesian
    - Naive Bayes
    - Averaged One-Dependence Estimators (AODE)
    - Bayesian Belief Network (BBN)
    - Gaussian Naive Bayes
    - Multinomial Naive Bayes
    - Bayesian Network (BN)
  - Decision Tree
    - Classification and Regression Tree (CART)
    - Iterative Dichotomiser 3 (ID3)
    - C4.5
    - C5.0
    - Chi-squared Automatic Interaction Detection (CHAID)
    - Decision Stump
    - Conditional Decision Trees
    - M5
  - Dimensionality Reduction
    - Principal Component Analysis (PCA)
    - Partial Least Squares Regression (PLSR
    - Sammon Mapping
    - Multidimensional Scaling (MDS)
    - Projection Pursuit
    - Principal Component Regression (PCR)
    - Partial Least Squares Discriminant Analysis
    - Mixture Discriminant Analysis (MDA)
    - Quadratic Discriminant Analysis (QDA)
    - Regularized Discriminant Analysis (RDA)
    - Flexible Discriminant Analysis (FDA)
    - Linear Discriminant Analysis (LDA)
  - Instance Based
    - k-Nearest Neighbour (kNN)
    - Learning Vector Quantization (LVQ)
    - Self-Organizing Map (SOM)
    - Locally Weighted Learning (LWL)
  - Clustering
    - k-Means
    - k-Medians
    - Expectation Maximization
    - Hierarchical Clustering



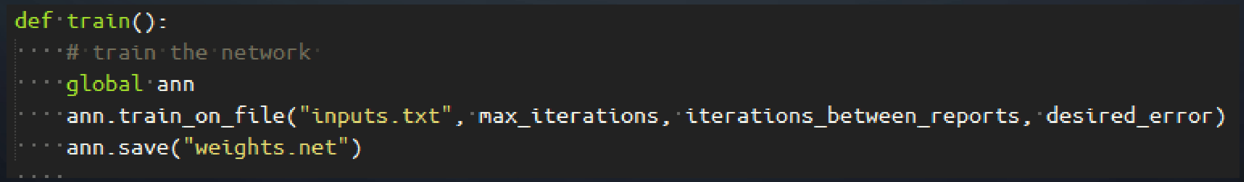
OPEN SOURCE SOLUTIONS

- OpenNN
  - C++ open source library
- FANN
  - Multi-Language open source library done by a graduate student (works very well tho)
  - Main code is in C, but has been exported to multiple languages as an API.
- PyBrain
  - Python based ANN. Open Sourced. Slightly older than others.
- TensorFlow
  - Python Based Computational System made by Google. Is used for much of their Machine Learning research. Open Sourced (sort of)

FANN EXAMPLE CODE



```python
from pyfann import libfann
import os.path

input_left = []
input_right = []
input_start = []

connection_rate = 1
learning_rate = .7
num_input = 4
num_neurons_hidden = 1000
num_output = 1

desired_error = 0.0000001
max_iterations = 100
iterations_between_reports = 1

speed_boost = 500

ann = libfann.neural_net()
ann.create_sparse_array(connection_rate, (num_input, num_neurons_hidden, num_output))
ann.set_learning_rate(learning_rate)
ann.set_activation_function_output(libfann.SIGMOID)#_SYMMETRIC)
ann.set_training_algorithm(libfann.TRAIN_INCREMENTAL)
if os.path.isfile("weights.net"):
    ann.create_from_file("weights.net")
```



```python
def train():
    # train the network
    global ann
    ann.train_on_file("inputs.txt", max_iterations, iterations_between_reports, desired_error)
    ann.save("weights.net")
```

QUESTIONS?