Deep Learning and Neural Networks

Deep Learning

What society thinks I do

What my friends think I do

What other computer scientists think I do

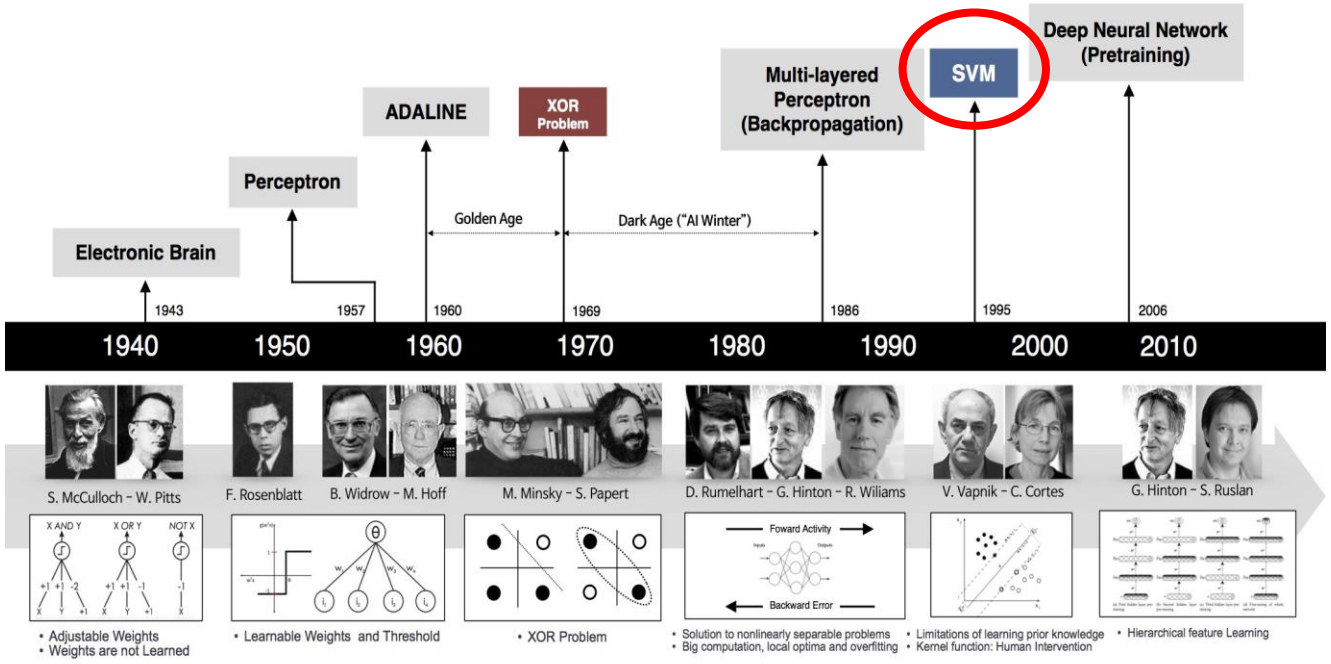
What mathematicians think I do

What I think I do

```
In [1]:
import keras
Using TensorFlow backend.
```

What I actually do

Historical Trends



beamandrew.github.io/

AlphaGo (2015)

- First program to beat a professional Go player

AlphaZero (2017)

DeepMind

AlphaZero AI beats champion chess program after teaching itself in four hours

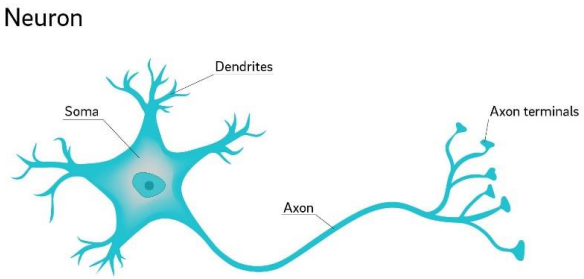
Google's artificial intelligence sibling DeepMind repurposes Go-playing AI to conquer chess and shogi without aid of human knowledge

Now, more stories about Deep Learning are on going ……

It is the time for us to learn Deep Learning ……

- A computational model that can learn.
- A model with parameters.
- Learns the parameters from the data.

Round Shape
Red color
Small Size



APPLE

Human

- Reading
- Past Experience

Information

Computer

So, 1. **what exactly is deep learning** ?

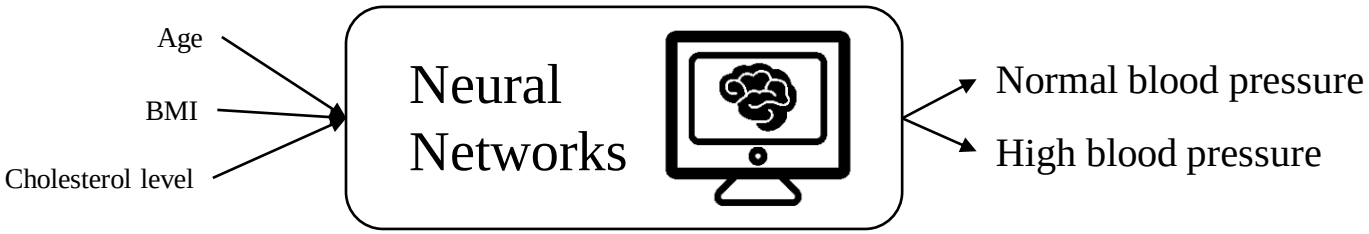
And, 2. **why is it generally better** than other methods on image, speech and certain other types of data?

The short answers

1. **'Deep Learning' means** using a **neural network** with **several layers of nodes** between input and output

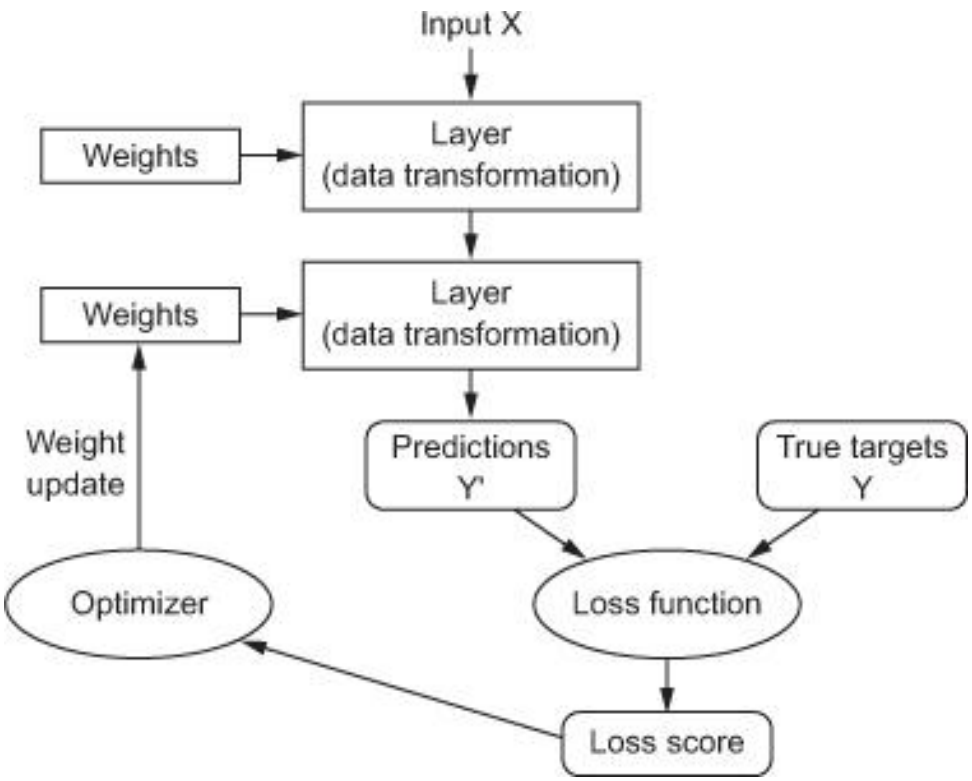
2. **the series of layers between input & output do feature identification and processing in a series of stages, just as our brains seem to.**

- Data:
  - Input (Age, BMI, Cholesterol level)
  - Output (Normal / High Blood Pressure)
- We teach the model using the data to make accurate prediction by optimizing parameters



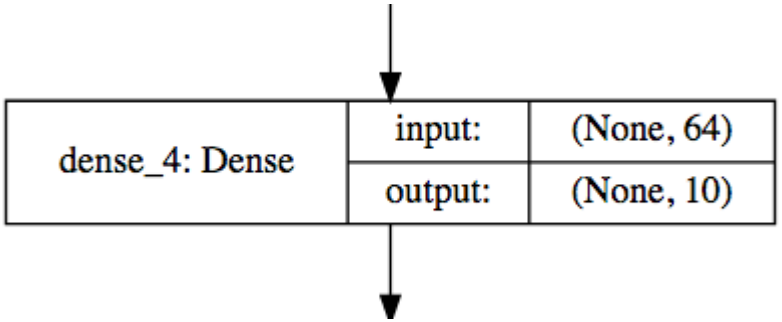
Anatomy of a deep neural network

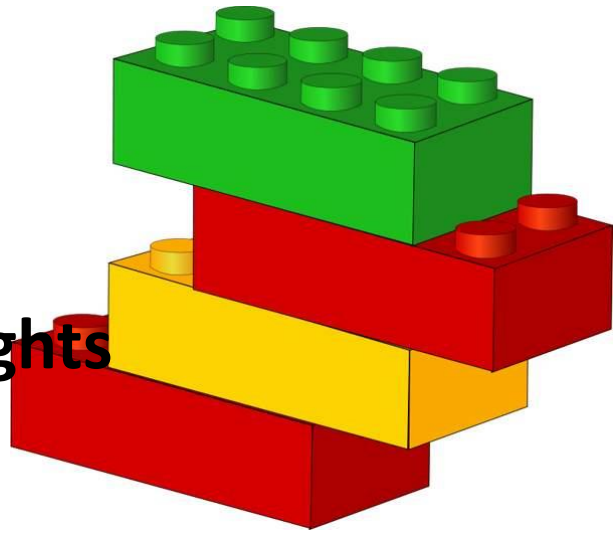
- Layers
- Input data and targets
- Loss function
- Optimizer



Layers

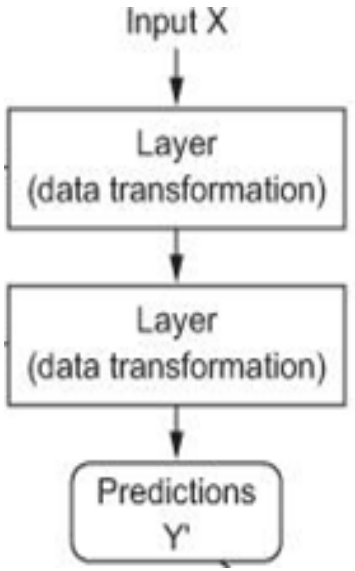
- Data processing modules
- Many different kinds exist
  - densely connected
  - convolutional
  - recurrent
  - pooling, flattening, merging, normalization, etc.
- Input: one or more tensors
  output: one or more tensors
- Usually have a state, encoded as **weights**
  - learned, initially random
- When combined, form a **network** or a **model**

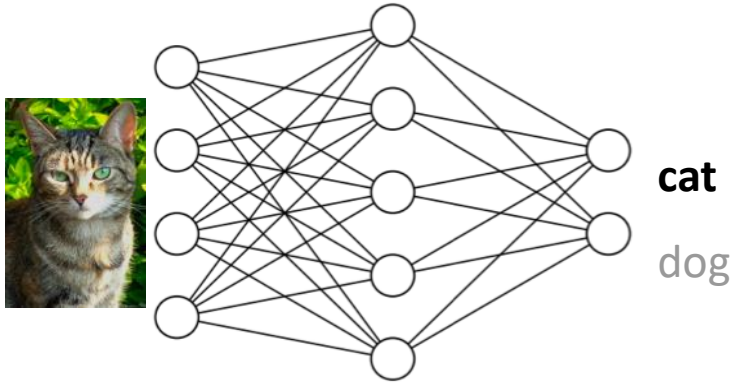




Input data and targets

- The network maps the input data X to predictions Y’
- During training, the predictions Y’ are compared to true targets Y using the loss function

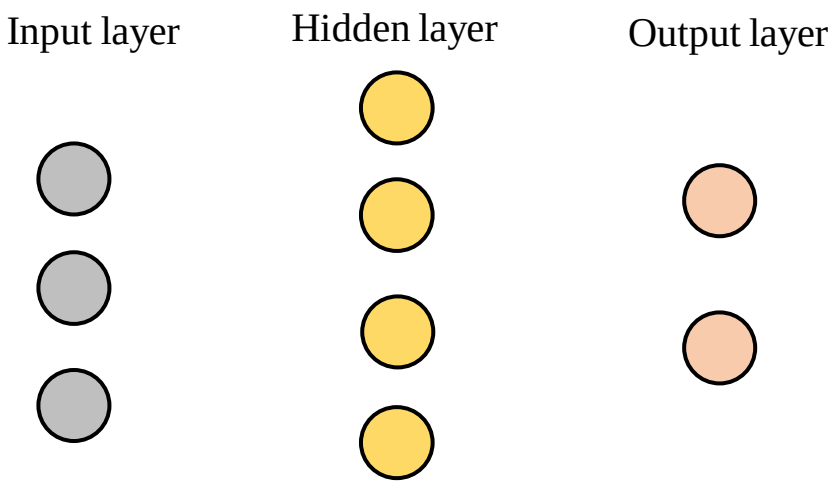


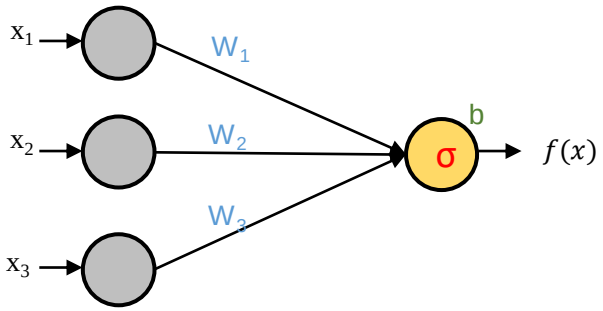


Loss function

- The quantity to be minimized (optimized) during training
  - the only thing **the network** cares about
  - there might also be other metrics **you** care about
- Common tasks have “standard” loss functions:
  - *mean squared error* for regression

- Made up of multiple layers of nodes



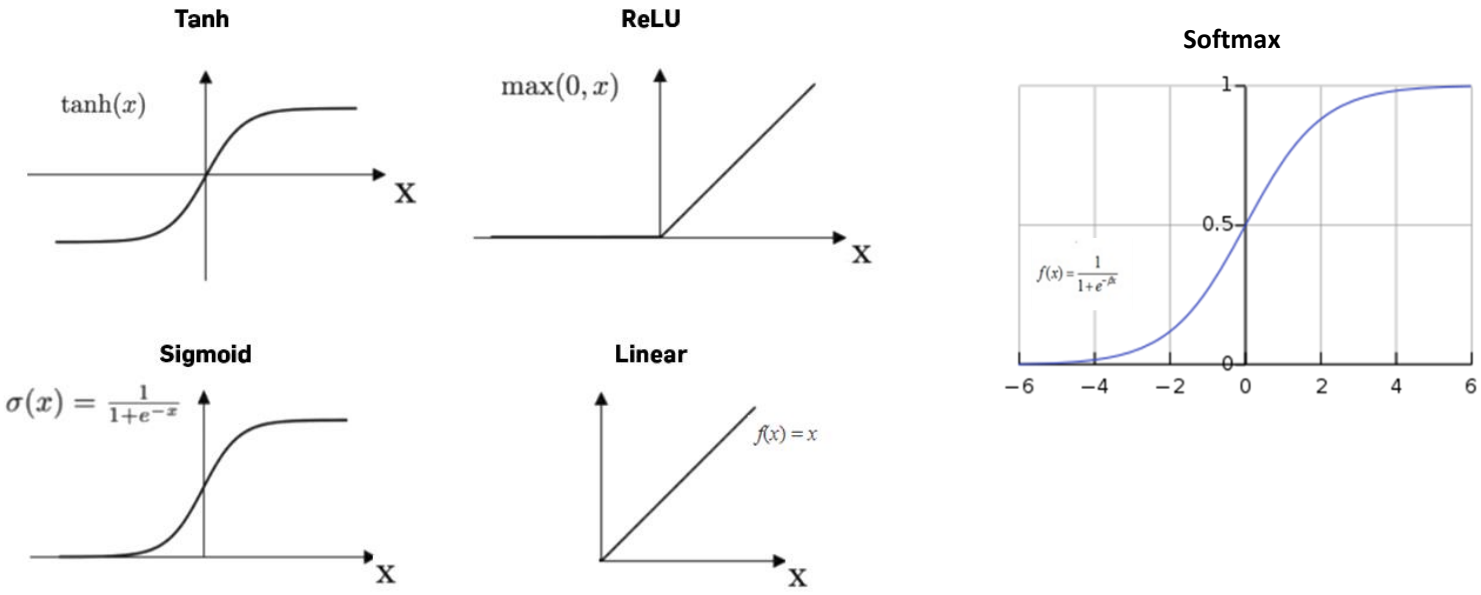


Input x Weight + Bias $(x_1 w_1 + x_2 w_2 + x_3 w_3 + \mathrm{b})$

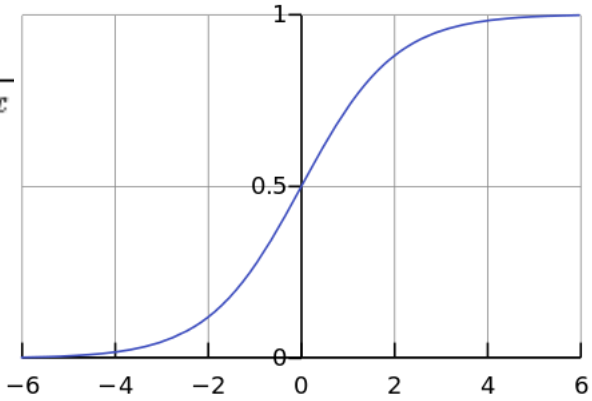
Apply Activation Function $\sigma(x_1 w_1 + x_2 w_2 + x_3 w_3 + \mathrm{b})$

Obtain output $f(x) = \sigma(x_1 w_1 + x_2 w_2 + x_3 w_3 + \mathrm{b})$

- Common activation functions:



$$f(x) = \frac{1}{1 + e^{-x}}$$



-0.06

2.7

-2.5

-8.6

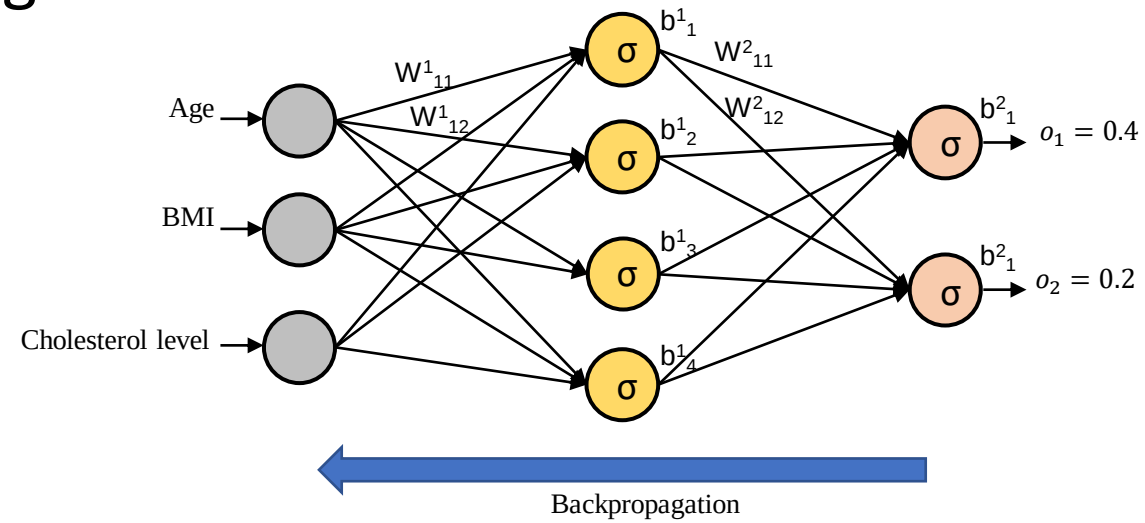
0.002

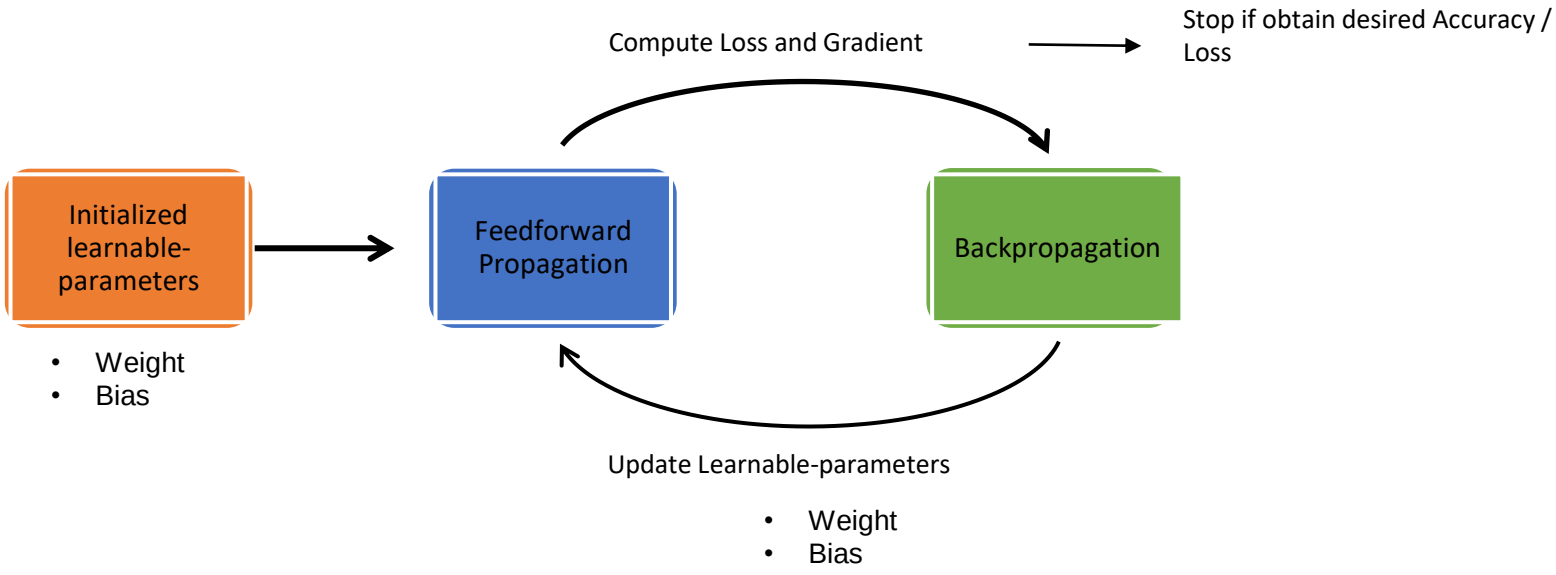
1.4

***f*(*x*)**

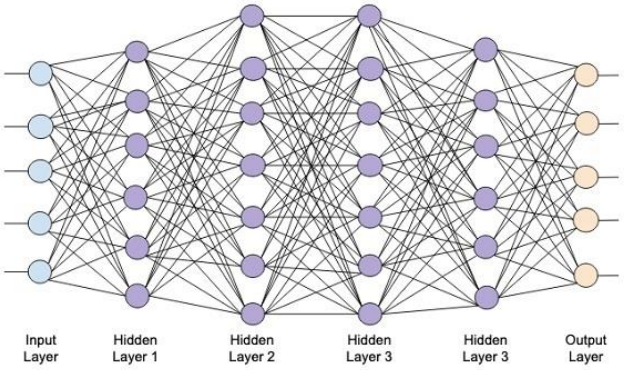
x = -0.06×2.7 + 2.5×8.6 + 1.4×0.002 = 21.34

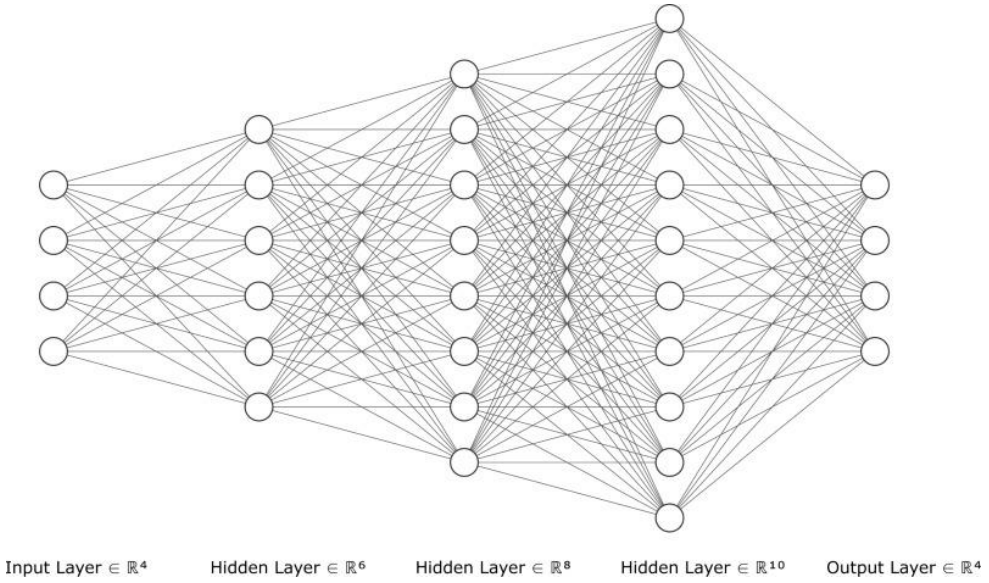
- Compute Gradients, i.e: $\frac{\partial L}{\partial w_{ij}^{n}}$, $\frac{\partial L}{\partial b_{i}^{n}}$
- Backpropagate the gradient to updates the weights




Compute Loss and Gradient
Stop if obtain desired Accuracy / Loss
Initialized learnable-parameters
Feedforward Propagation
Backpropagation
Weight
Bias
Update Learnable-parameters
Weight
Bias

- Increase number of node
- Increase number of hidden layer





- The larger the better???
- No → overfitting (layman term – memorize training data only, poor performance when testing data is used)
- What the reason of overfitting??? One of the reason is we have too many parameters

- Reason: Have too many parameters
- How to solve???
- Reduce the model size → number of node / hidden layer

- Another reason of overfitting is the weight value are too big.
- To solve this, we regularize the weight value – don’t let it grow too large

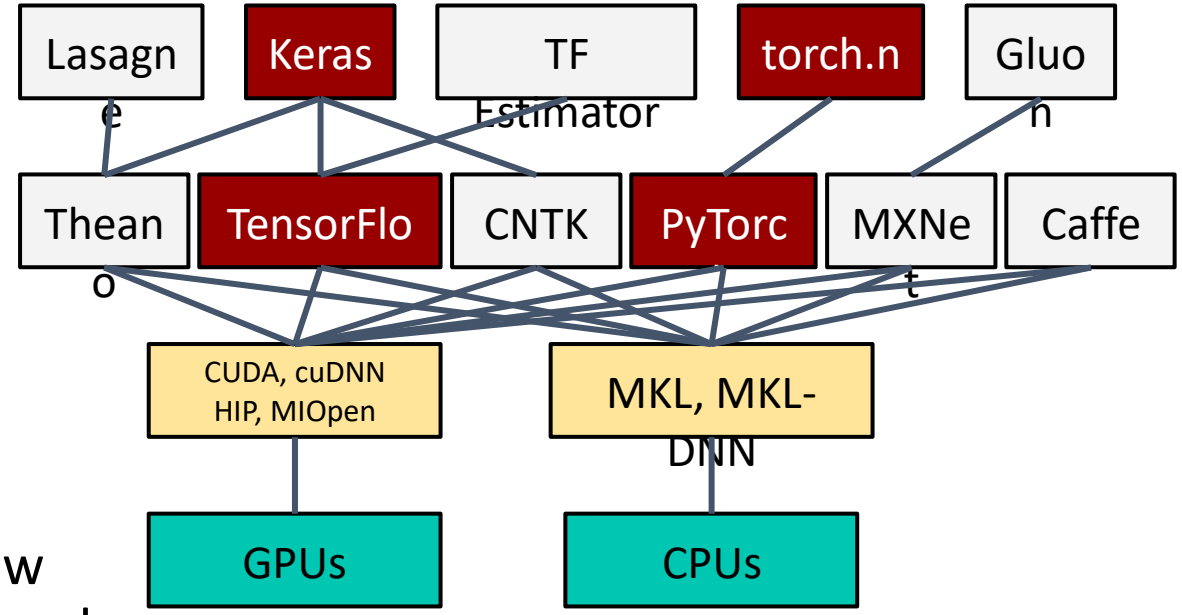
- Another way to solve: Use the idea of ensemble model
- Ensemble model → train multiple model then combine all of them.

High computational cost

Deep learning frameworks


Caffe
torch
Chainer
PyTorch
DEEPLEARNING4J
Caffe2
The Microsoft Cognitive Toolkit
TensorFlow
dmlc
mxnet
theano

Deep learning frameworks



- Keras is a high-level neural networks API
  - we will use TensorFlow as the compute backend
  - included in TensorFlow 2 as `tf.keras`
  - https://keras.io/ , https://www.tensorflow.org/guide/keras
- PyTorch is:
  - a GPU-based tensor library
  - an efficient library for dynamic neural networks
  - https://pytorch.org/