FDP INSTITUTE by CAIA

Webinar

July 20, 2023

Unleashing the Power of Neural Networks: A Personal Journey into Creating and Harnessing a Neural Network for Trading Stocks

Welcome

We will begin promptly at 10 AM ET.

If you are unable to hear the speakers, please let us know in the chat box. You may enter your questions in the Q&A, we will address them at the end of the presentation. You can find a copy of the slide deck and recording of this webinar: www.fdpinstitute.org/webinars

Financial Data Professional Institute

FDP Institute provides world class training and education to financial professionals to meet the accelerating needs of digital transformation in the industry.

Introductions

Hossein Kazemi, Ph.D., CFA
Senior Advisor,
CAIA Association &
FDP Institute

Tom Pickel, CAIA, FDP
Founder, Souppe

Today's Topic:

Unleashing the Power of Neural Networks:

A Personal Journey into Creating and Harnessing a Neural Network for Trading Stocks

Unleashing the Power of Neural Networks:

A Personal Journey into Creating and Harnessing a Neural Network for Trading Stocks

Tom Pickel, CAIA, FDP

July 2023

Agenda

- Introduction
- Neural network basics
- The training process
- Some more concepts
- **My network**

Why Deal with Artificial Intelligence?

- It is "the Future".
- Impact on our lives:

Natural Language Processing

- Email filters (spam, classification).
- Smart assistants.
- Better search results.
- Predictive text.
- Language translation.
- Text analytics.

Object Identification

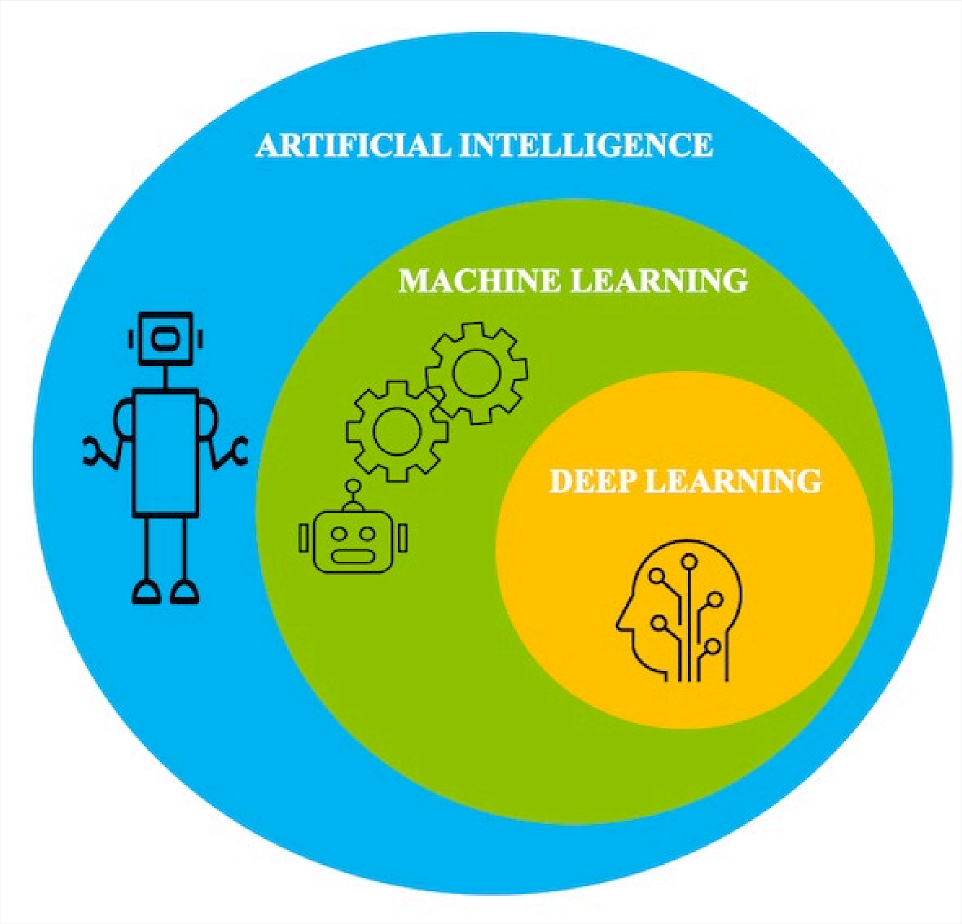
- Self driving vehicles.
- Facial recognition.
- Reverse image search.
- Medical diagnosis.
- Quality control.
- Voice-to-text translation.
- Immediate language translation, by voice or phone camera.

Personaliz-ation and Profiling

- Music/video/product recommendations.
- Personalized experience.
- Personalized services.

And much more..

What is Artificial Intelligence?



Source: IBM.

It’s (Mostly) About Numbers

- Computers work wth a binary system: only 1 and 0, “on” and “off”.
- Artificial intelligence models work with numbers:
    - Data.
    - Transformation.
    - Manipulation.
    - Error measurement.
- Algebra, calculus, statistics & probability and information theory.

The hardest part in AI: the ability to perceptualize and understand complex, abstract ideas.

Basics

- Learn Python: campus.gov.il Heb/Ar.

- While you learn, develop a project to practice.

- Learn the theory:

    - **The FDP curriculum**.
    - **Neural Networks From Scratch** by Harrison Kinsley and Danial Kukiela.
    - **Make Your Own Neural Network** by Tariq Rashid.

- Watch YouTube videos, such as Neural Networks from Scratch.

About Me

- Education: law and business with major in Finance and Risk Management.
- Alternative investments professional:
  - Financial modelling, asset valuation and business plans.
  - Alternative investment analysis.
- CAIA since 2017, FDP since 2021.
- Learned Python programming in 2019.
- 4 Main projects:
  - Real-estate classified ads.
  - Data mining tools.
  - Elite proxy servers.
  - **A feed-forward neural network for trying to guess the moves of publicly-traded equity.**
- Reading material: my blog at https://pickel.io.
- Founder and CEO of Souppe.

Introduction to Neural Networks

- A neural network is a **machine learning model**.

- It allows us to discover relationships-

Features : Target

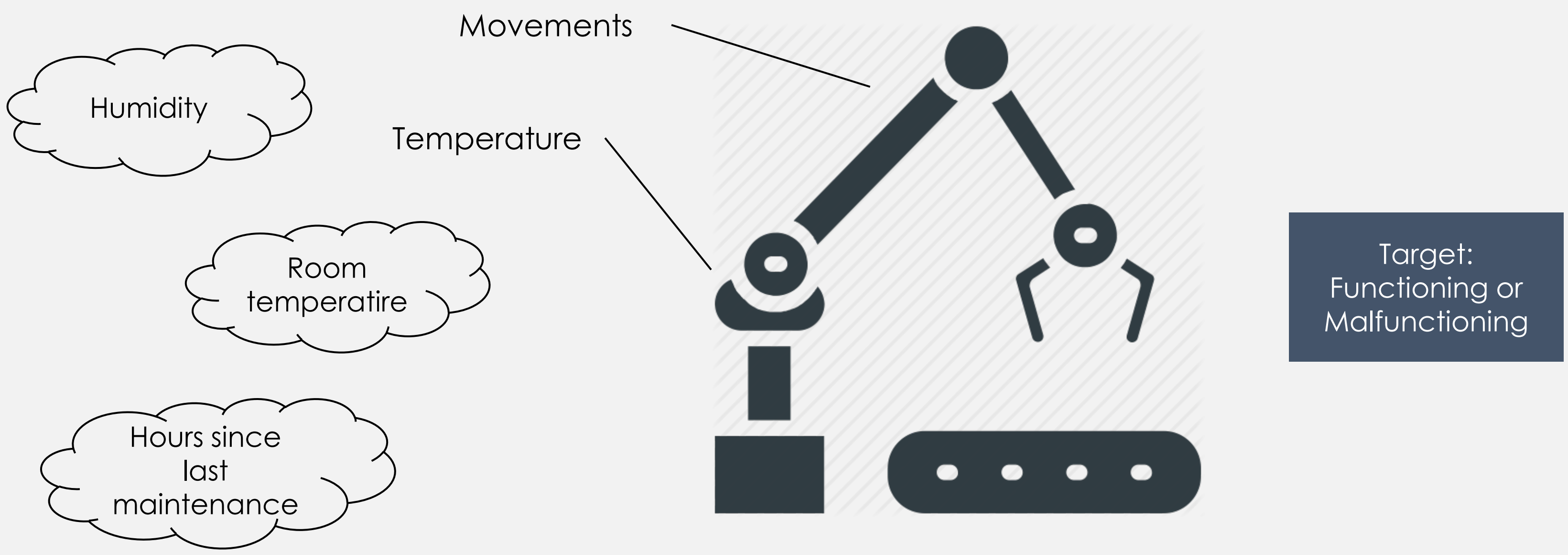
Charactaristics in the general population of subjects

The charasteristic or outcome we are trying to predict.

- **Supervised** vs. **Unsupervised** tasks.

Real Life Use of Machine Learning

- We have a robotic arm in a factory and we want to predict a possible malfunction in advance.



Real Life Use of Machine Learning

- Sampling our observations and filling-in the data:

#	Humidity	Room Temperature	Number of Movements	Hours Since Last Maintenance	Malfunction
1					
2					
3					
4					
5					
5					
6					
7					
...					
...					
...					

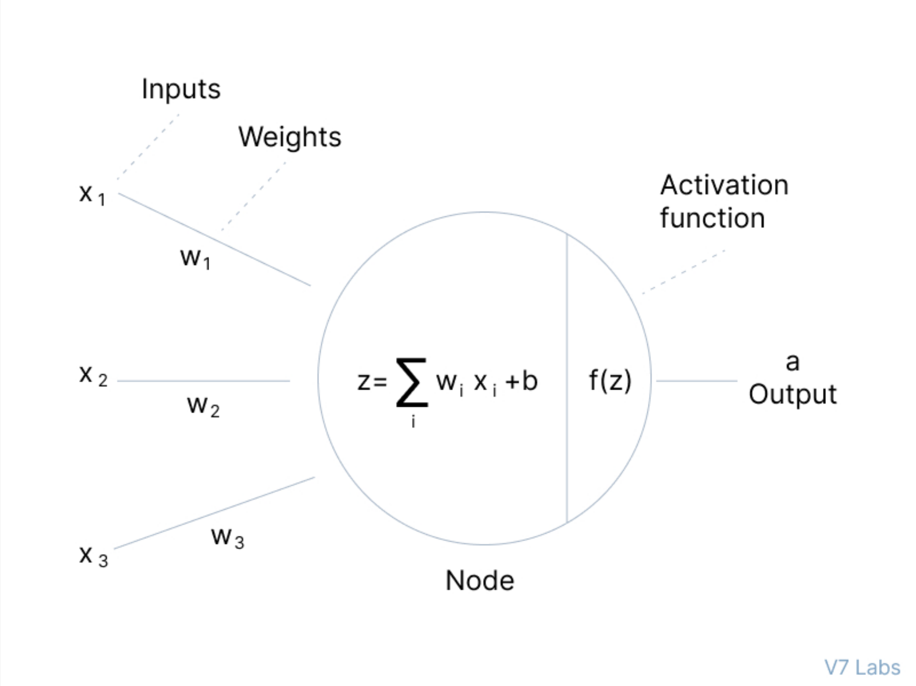
- Once full, this is our dataset.

Machine Learning is a powerful way to learn non-linear relationships between the various Features and the Target.

- The goal: **generalization**.

The Neuron

- The basic building block of a network.
- **A neuron** involves inputs (x_n), weights (w_n), a bias (b), and an activation function.
- Multiply inputs by weights, sum, add a bias and run through a non-linear function to form an output.



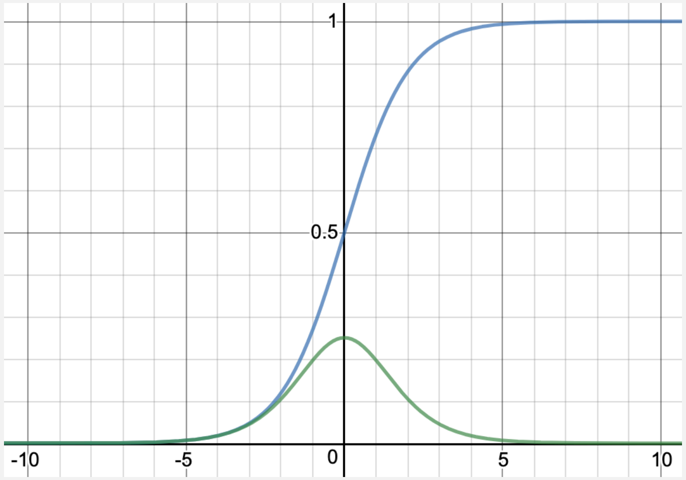
Source: [Activation Functions in Neural Networks [12 Types & Use Cases]].

Weights and Biases

- These are the neural network model' parameters.
- Each neuron has a weight for each input, and a bias.
- The model "learns" by tweaking these parameters.
- A **Weight** can be viewed as signal "strength".
- The **Bias** gives our model more flexibility and enables it to adjust a neuron's output independently of the inputs. It's like the intercept term in a linear equation.

The Activation Function (1/2)

- Activation functions add non-linearity.
- No non-linearity? No meaningful learning!
- There are many kinds of activation functions (see next slide).
- They all receive a neuron's initial output, perform a non-linear transformation on it and then prepare to output it to the next layer.
- Choose an activation function that suits the task and network architecture.



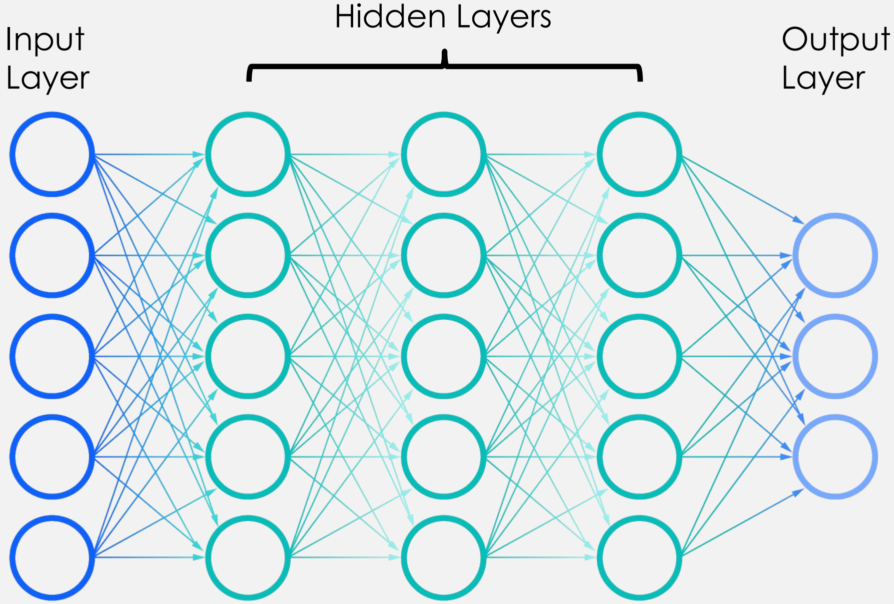
The Sigmoid activation function and its derivative, used in the backpropagation backwards step. Drawn in transum.org.

The Activation Function (2/2)

	Forward Step	**Derivative**	**Forward Step Value Range**	**Weight Initialization Method (rules of thumb)**
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = \sigma(x)(1-\sigma(x))$	[0, 1]	Glorot (Xavier)
Hyperbolic Tangents (tanh)	$f(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$	[-1, 1]	Glorot (Xavier)
Rectified Linear Unit (ReLU)	$f(x) = \max(0, x)$	$f'(x) = \begin{cases} 0 \text{ if } x \leq 0 \\ 1 \text{ if } x > 0 \end{cases}$	[0, ∞)	He
Exponential Linear Unit (ELU)	$f(x) = \begin{cases} x \text{ if } x > 0 \\ \beta(e^x - 1) \text{ if } x \leq 0 \end{cases}$ $(\beta > 0)$	$f'(x) = \begin{cases} 1 \text{ if } x > 0 \\ \beta e^x \text{ if } x < 0 \end{cases}$	[-1, ∞)	He
Scaled Exponential Linear Unit (SELU)	$f(x) = \begin{cases} \lambda x \text{ if } x \geq 0 \\ \lambda\alpha(e^x - 1) \text{ if } x < 0 \end{cases}$ $(\alpha \approx 1.6733, \lambda \approx 1.0507)$	$f'(x) = \begin{cases} \lambda \text{ if } x > 0 \\ \lambda \cdot \alpha \cdot e^x \text{ if } x \leq 0 \end{cases}$	[~(-1.78), λx]	LeCun
Inverse Square Root Unit (ISRU)	$f(x) = \frac{x}{\sqrt{1+ax^2}}$	$f'(x) = \left(\frac{1}{\sqrt{1+ax^2}}\right)^3$	$[-\frac{1}{\sqrt{a}}, \frac{1}{\sqrt{a}}]$	LeCun
Softplus	$f(x) = \log(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$	[0, ∞)	He
Mish	$f(x) = \tanh(\ln(1+e^x))$	$f'(x) = \frac{e^x \cdot \omega}{\delta^2}$ $(\omega = 4(x + 1) + 4e\ 2x + e\ 3x + e\ x\ (4x + 6),$ $\delta = 2e\ x + e\ 2x + 2)$	[0, 1]	LeCun
Swish	$f(x) = \frac{x}{1+e^{-\beta x}}$	$f'(x) = f(x) + sigmoid(x) * (1 - f(x)) = \frac{(e^{-x} \cdot (x+1)+1)}{(1+e^{-x})^2}$	[0, ∞)	Random

The (hidden) Layer

- **A layer is a set of neurons.**
- It has at least one neuron.
- A “hidden” layer: between the input and output layers of a neural network.
- A layer’s output is based on its input and the layer calculations.
- Each layer’s output is the next layer’s input.



Source: https://github.com/skawy.

What Happens Inside a Layer?

- A forward step: [input matrix] · [weight matrix] + [bias vector] in each hidden layer

$$
\begin{array}{c|c|c|c|c|c|}
 & F_1 & F_2 & & \dots & F_k \\
\hline
X_1 & x_1^1 & x_2^1 & & & x_k^1 \\
\hline
X_2 & & & & & \\
\hline
X_n & x_1^n & & & & x_k^n \\
\hline
\end{array}
\quad \cdot \quad
\begin{array}{c|c|c|c|c|c|}
 & N_1 & N_2 & & \dots & N_k \\
\hline
F_1 & w_1^1 & w_2^1 & & & w_k^1 \\
\hline
F_2 & & & & & \\
\hline
F_n & w_1^n & & & & w_k^n \\
\hline
\end{array}
$$

$$+$$

$$
\begin{array}{|c|c|c|c|c|}
N_1 & N_2 & & \dots & N_k \\
\hline
b_1^1 & b_2^1 & & & b_k^1 \\
\hline
\end{array}
$$

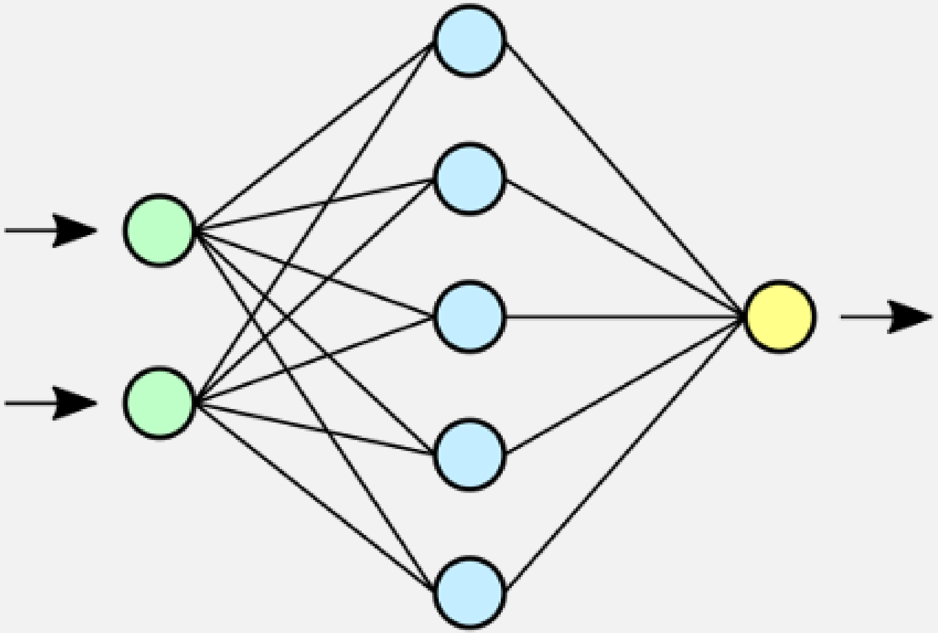
$$=$$

[output matrix]

```python
# Calculate output values
self.output = np.dot(inputs, self.weights) + self.biases
```

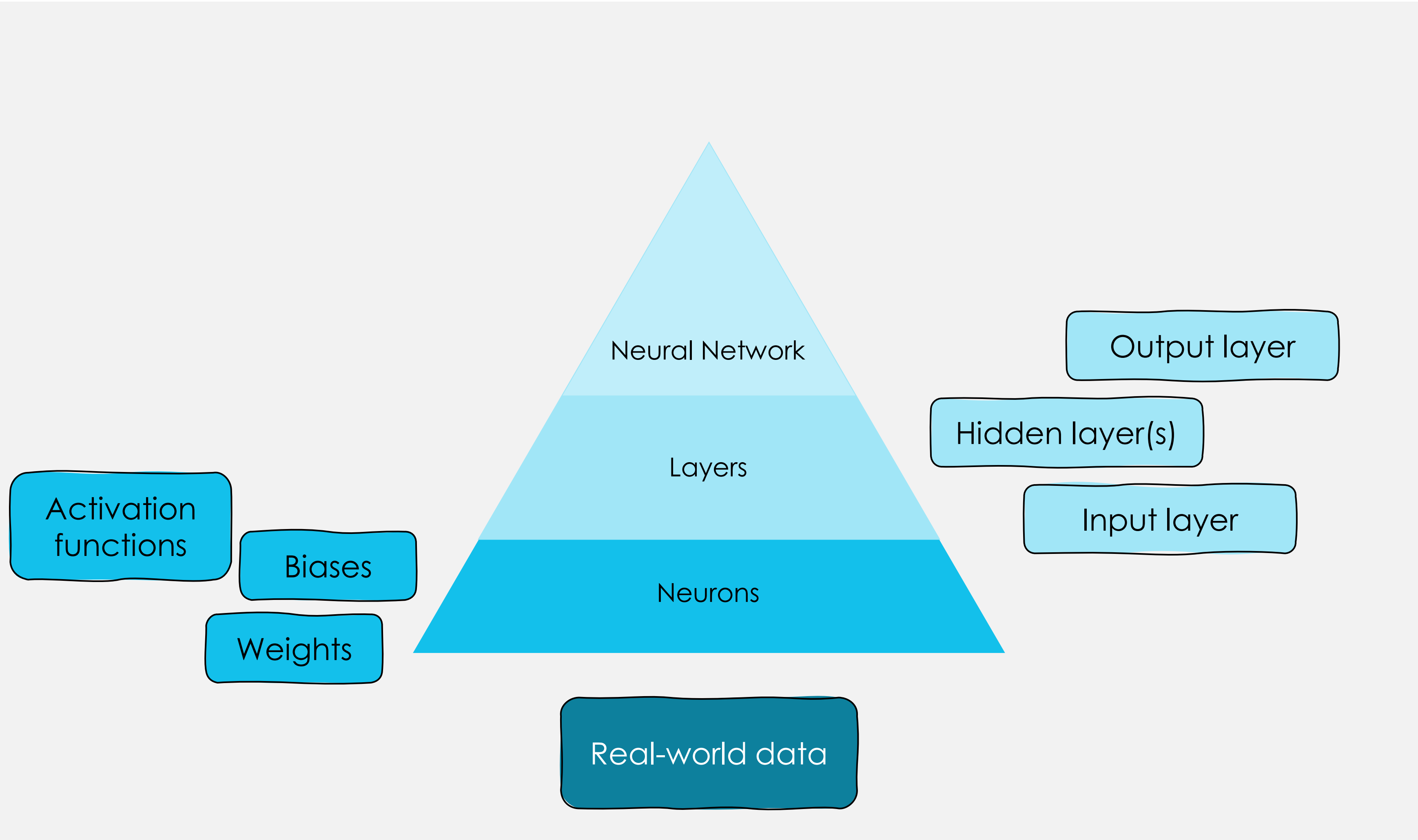
The Feed-Forward Neural Network

- A feed-forward neural network describes a network where information flows in one direction.
- We start at the input layer, move forward from through the hidden layers and produce an output:



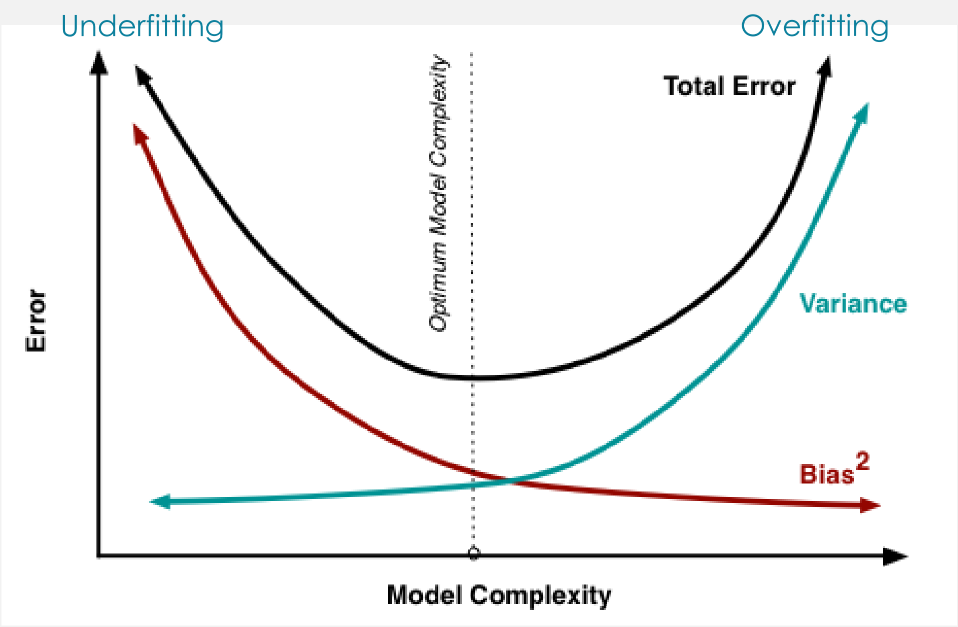
Source: [Wikipedia](#).

A Macro Look on Neural Networks



The Bias-Variance Tradeoff

- **The bias-variance tradeoff:**

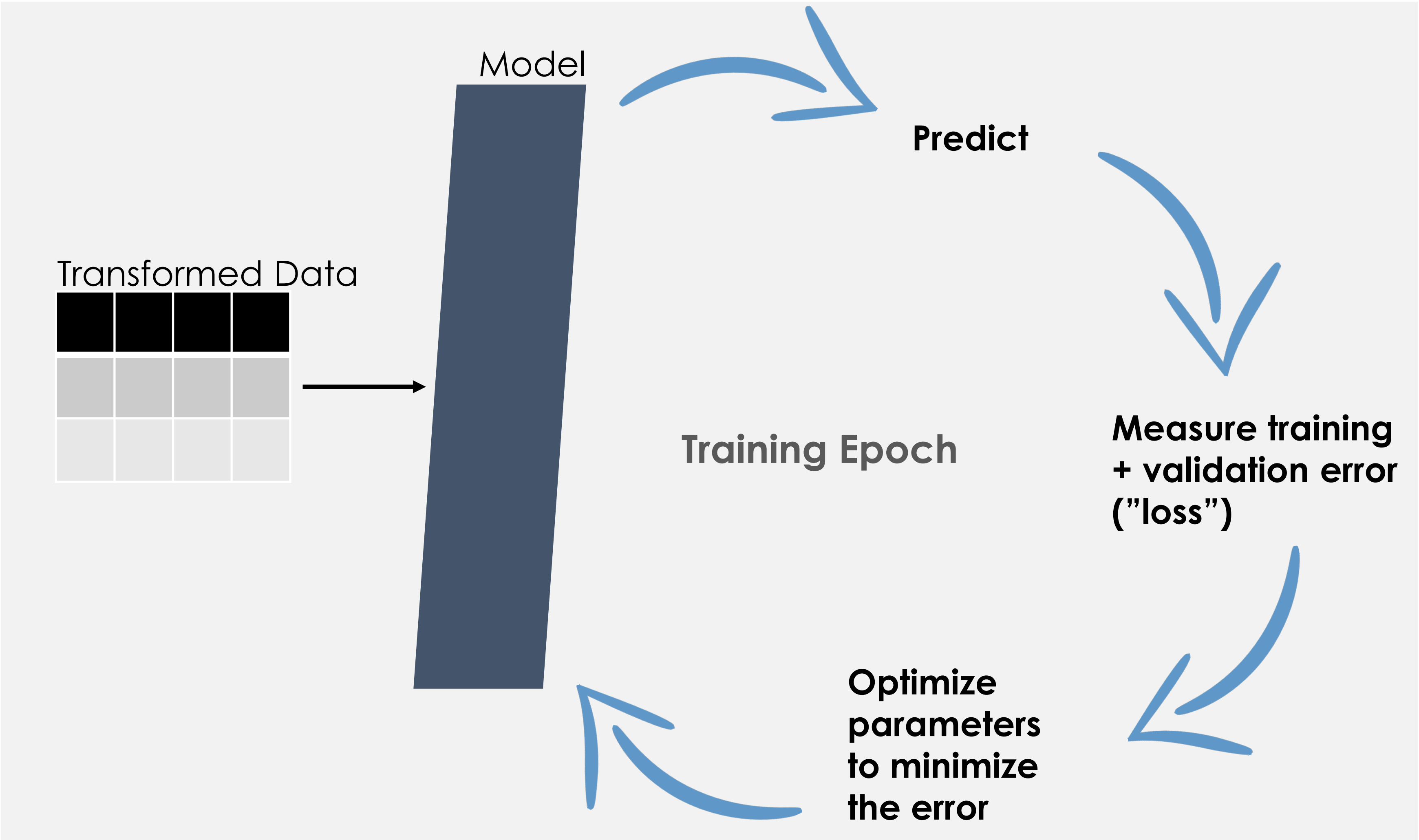


- An **underfit** model is not complex or flexible enough to catch the true relationships in the data.

- An **overfit** model it is too complex. It was able to spot the idiosyncratic relatioships in the training data set, and will therefore not be able to generalize well.

Source: Cornell.

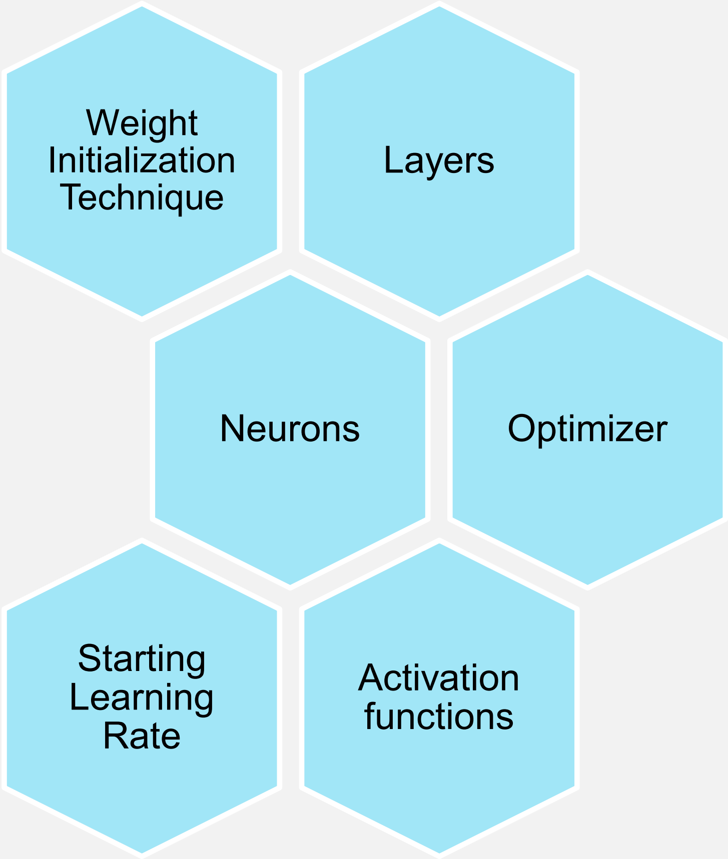
The General Training Process



Preparing the Dataset

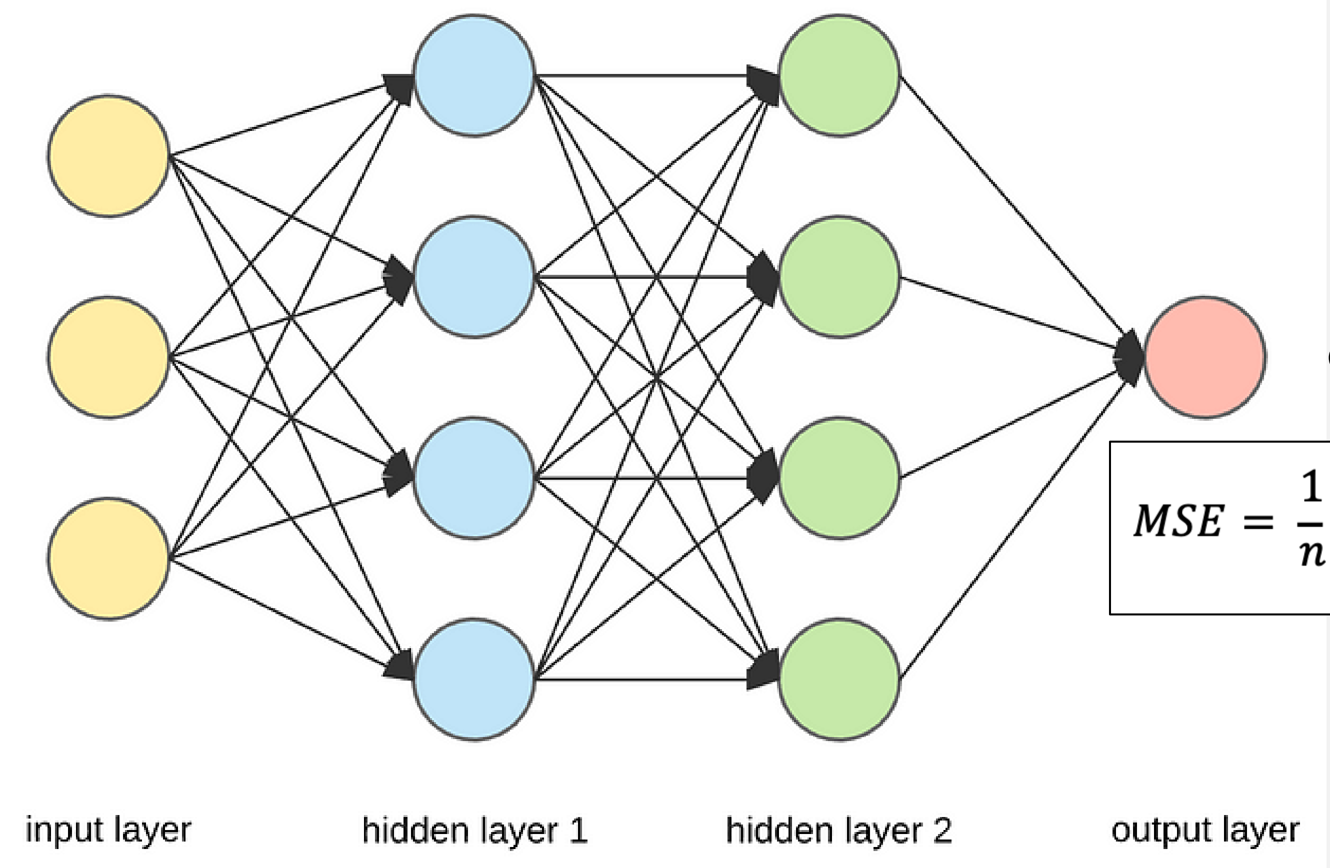
Create dataset	Collect observations.
Deal with missing data	Complete missing data or remove observation.
Transform dataset	i.e. standardize Features.
Randomize observation order	We want the model to learn about Feature-Target relastionships regardless of sampling order.
Split test dataset	Take 1%-20% of the initial dataset for **testing**.
Split validation dataset	Take 1%-20% of the initial dataset for **validation**.

Deciding on Hyperparameters and Network Architecture



Training: Forward Step

- Move forward, make calculations and measure the prediction error.



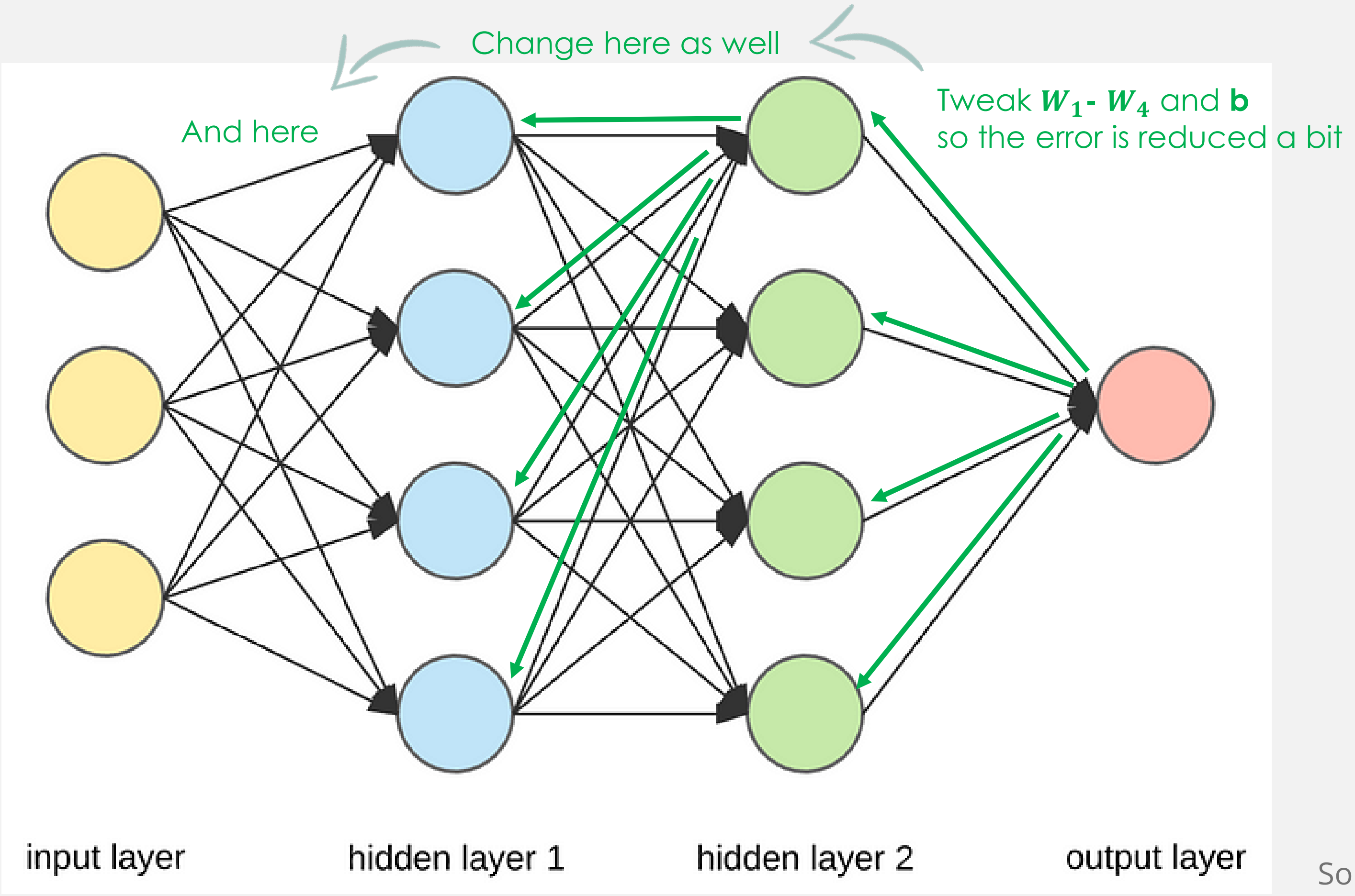
Cost = Error = Loss

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2$$

Source: [Arden Dertat.]()

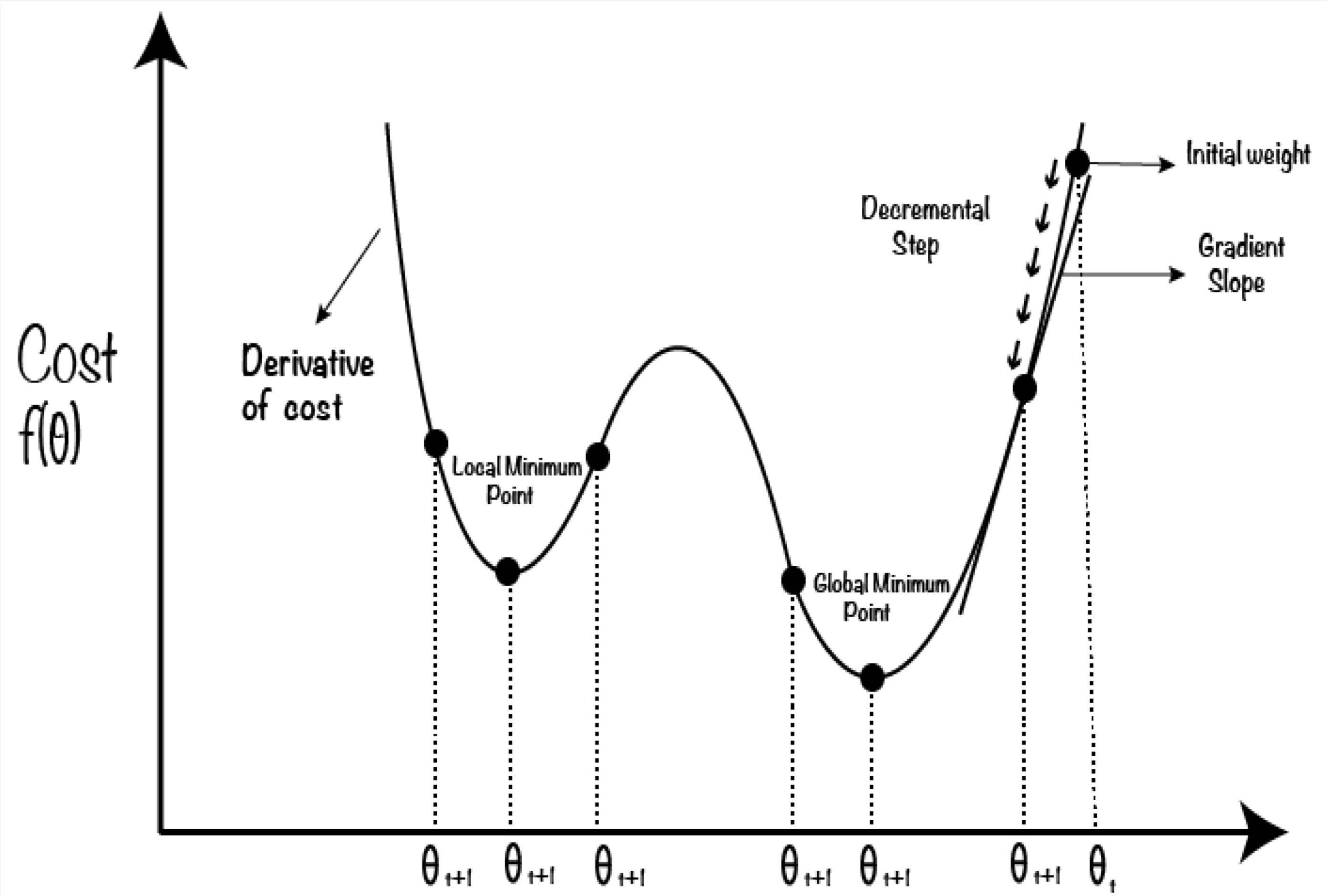
Training: Backward Step

- **Go back, fine-tune neuron weights and biases to minimize the prediction error.**



Source: [Arden Dertat.]()

Minimizing the Cost Function



Source: An Efficient Optimization Technique for Training Deep Neural Networks.

The Accuracy

- Another measure of a model's efficacy.
- Calculated as the percent of correct predictions out of total predictions made.
- Like loss, this is a measure for how well a model had learned.
- A quick score for our model's efficacy.
- We train on minimizing validation loss. Test accuracy is a secondary indicator for actual real-life success.
- For classification we have more types of accuracy.
- Highest test accuracy likely means a better model.

Putting the Model to Use

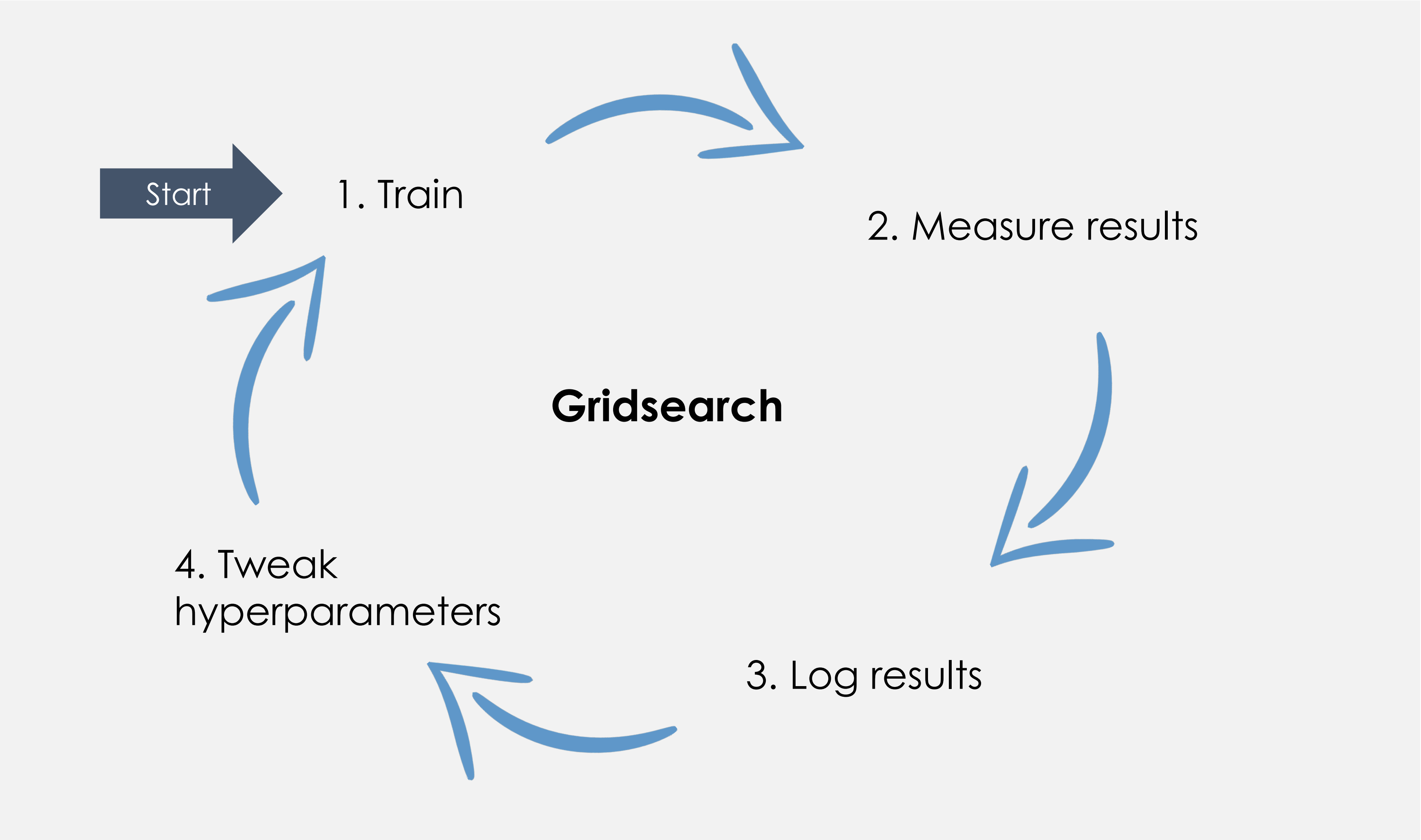
- We have a trained model, meaning we have the optimal setup that brings the test error to the minimum.

- After a training session we save the current build’s parameters to a file.

- When we need to use the network, we load the file and setup our network.

- We pass new Feature values as inputs and get an output we can use.

Gridsearch


Start
1. Train
2. Measure results
Gridsearch
3. Log results
4. Tweak hyperparameters

Regularization

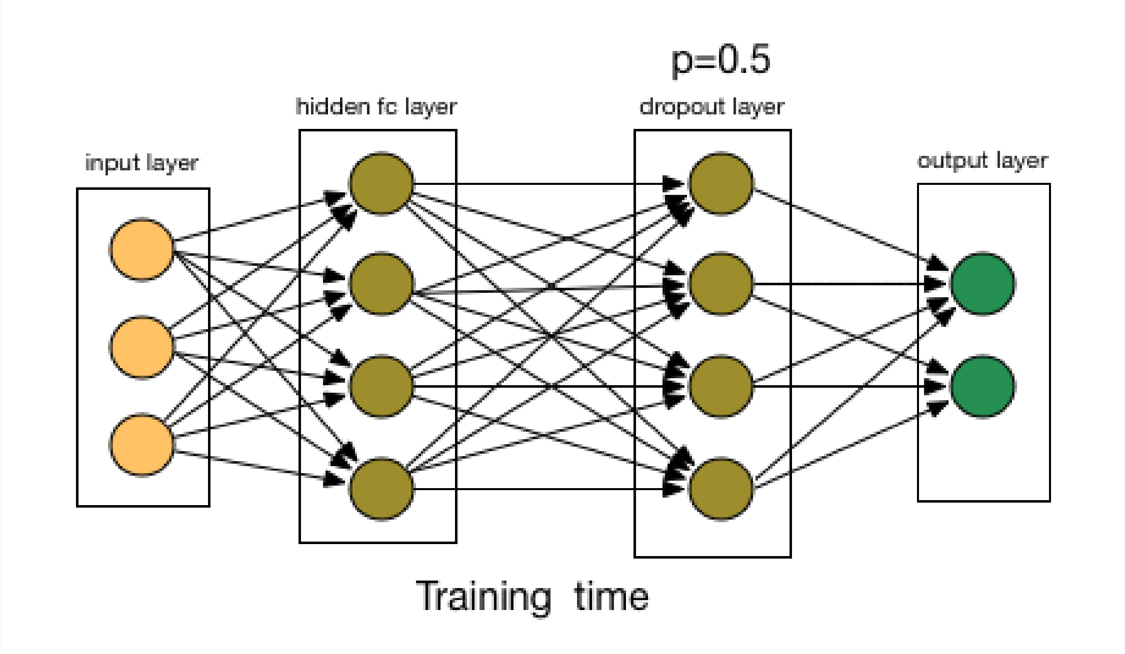
- **Regularization-** the actions we take to reduce model complexity.
- Controlling our model's complexity through feature importance:

Feature Shrinkage	Can **remove Features altogether** from our model.
Feature (dimension) Reduction	**Reduces** a Feature's importance in predicting our Target.
Elastic Net	Combines **both** Feature shrinkage and Feature reduction capabilities to our model.

- **Our goal-** to prevent overfitting and to increase generalization capabilities.
- Simple models are generally preferable over complex ones.
- Regularization decreases model complexity when needed.

Dropout

- **Dropout-** another form of regularization.
- Randomly ignore some neurons' inputs by multiplying them with 0.
- The goal: feature selection, decreased model complexity and increased generalization.

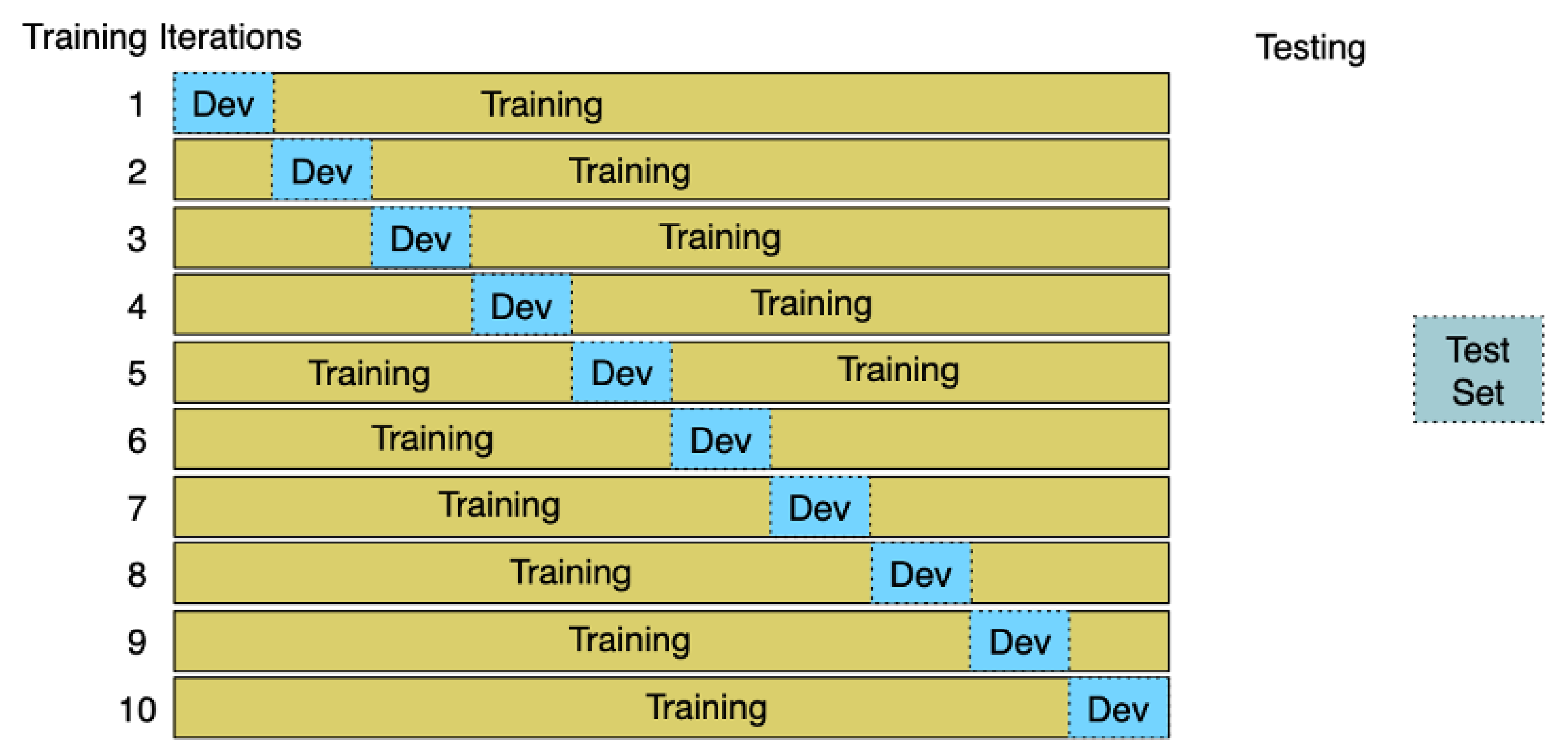


- More regularization techniques, caution when using several together.

Source: Analytics Vidhya.

K-Fold Cross Validation

- Train on different fold compositions k times, be more confident of our results.



Source: Jurafsky, D. and J. Martin. (2018). Chapter 4. Naïve Bayes and Sentiment Classification, In Speech and Language Processing.

Pseudorandomness

- Random sampling is crucial for training.
- The random distribution should be constant: **Pseudorandomness.**
- Our goal: replicable results.
- We instruct our program to use the same random distribution of numbers.
- In Python, we use:

```
# Set a specific random seed for our network
np.random.seed(1)
```

My Network

- Goal: to create something and try to see where we get.
- 275 Features: technical and fundamental analysis.
- Assumption: I can outsmart traders.
- Mission: predict the largest 125 companies’ tomorrow’s price movement.
- Reality: it doesn’t work well. There is more to do.
- An experiment, not a perfect mechanism.
- I only use Python’s basic machine learning packages: **Numpy** and **Pandas**. No Scikit-learn, TensorFlow and PyTorch.
- Object-oriented programming.

Predicting Stock Returns: Mission Impossible?

- There are many difficulties in trying to predict stock returns:
    - Difficulty in obtaining enough important data.
    - Noisy data, low signal-to-noise ratio.
    - High overfitting risk.
    - Non-stationarity, differing distribution statistics and perceptions.
    - Multicollinearity of Features.
    - Dynamic markets.
    - ”Black swan” events.

    And more.

Difficulties and Discoveries

- Getting into machine learning.
- Understanding neural networks.
- The first “eureka” moment: multi-dimentional thinking.

```
self.training_sample_x = self.training_sample.iloc[:, 0 : -1].to_numpy().reshape(-1, self.feature_dimentions)
self.training_sample_y = self.training_sample.iloc[:, -1].to_numpy().reshape(-1, 1)
```

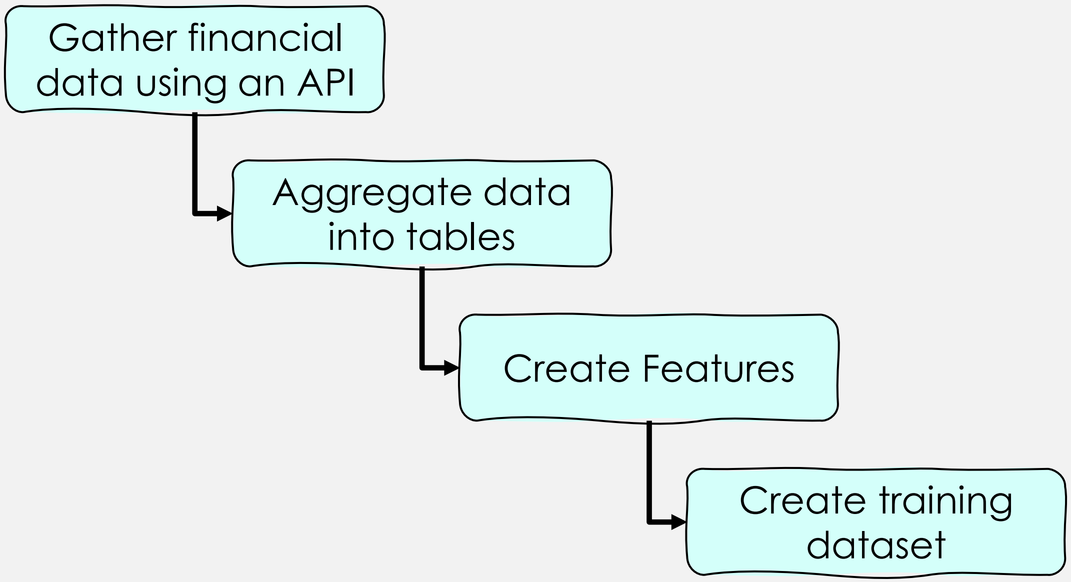
- More discoveries followed: hidden-layers of different size.

Network layers	**[256, 128, 128, 64, 32, 32, 1]**
Network activation funcs	**['Swish', 'Swish', 'Swish', 'Swish', 'Swish', 'Swish', 'Linear']**

- Trial and error: activation and loss functions.
- A truly fascinating experiment.

Data Preparation

- Stages of data preparation:



What My Experiment Does

Create datasets	Training from 1 January 2006 to 30 June 2022, Testing from 1 July to 31 December 2022.
Train, Validate, Test	10-fold cross validation create 10 sub-strategies that average to a super-strategy.
Log and Gridsearch	Let it run for days and weeks, log everything.
Backtesting	Backtest every super-strategy. Not used for strategy selection.
Super-strategy selection and use	Choose the best performing super-strategy (loss, accuracy) to use.
Get predictions	Get long and short position suggestions, trade.

The Result (1/5): Logging

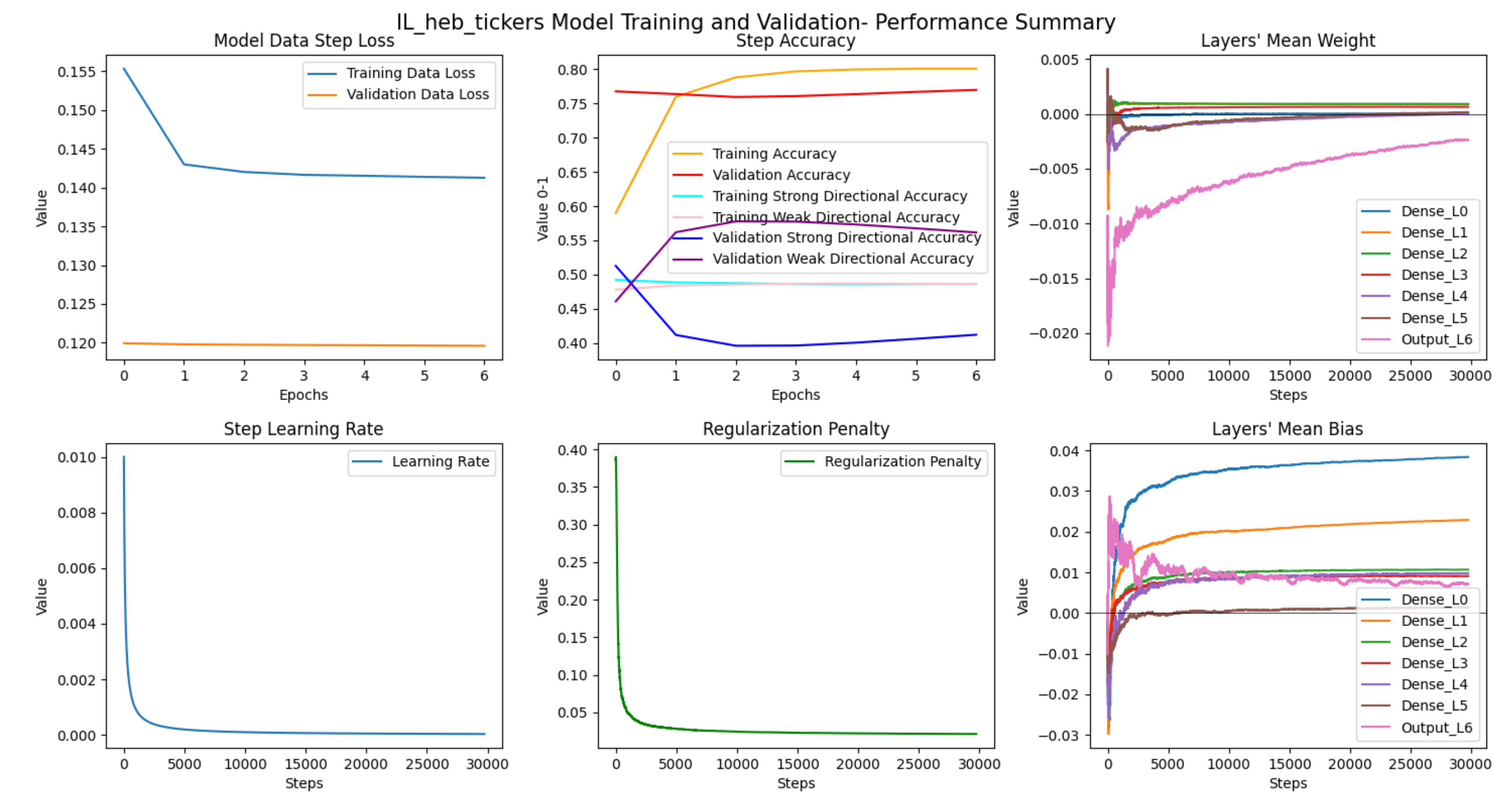
	169a
Feature dimentions	275
Training sample size	**543675**
Validation sample size	**135850**
Test sample size	**24726**
TVT folds	**{'Training': ['0', '1', '2', '3', '4', '5', '6', '7'], 'Validation': ['8', '9']}**
Training min Period Turnover threshold	150000
Clip target variable at	5
Data transformation	Standardize
Batch size	128
Max epochs	75
Training steps in each epoch	4248
Dropout rate	**0.1**
Dropout layers	**{0, 1}**
Network layers	**[256, 128, 128, 64, 32, 32, 1]**
Network activation funcs	**['Swish', 'Swish', 'Swish', 'Swish', 'Swish', 'Swish', 'Linear']**
Weight initialization method	LeCun
ISRU func. alpha	0.1
ELU, Swish funcs. beta	0.9
Optimizer	**Adam**
Optimizer starting learning rate	0.01
Optimizer decay	0.0001
Optimizer epsilon	1.00E-06
Optimizer beta_1	0.9
Optimizer beta_2	0.5
Optimizer momentum	0.8
Regularization lambda	5.00E-05

The Result (2/5): Logging

	169a
Loss function	**MSE**
Training sample proportion	0.8
Validation sample proportion	0.2
Training breaks at epoch	**6**
Minimum validation loss epoch	**6**
Test loss	**0.5796**
Test accuracy	**0.6911**
Test weak directional accuracy	0.5413
Test strong directional accuracy	0.4301
Test total directional accuracy	**0.9714**
Test observations quartile count	{'first_q': 10981, 'second_q': 78, 'third_q': 13038, 'fourth_q': 81}
Good results ratio (1q+3q/all)	0.9900
Final epoch loss- training	0.1413
Final epoch loss- validation	0.1196
Final epoch accuracy- training	0.8010
Final epoch accuracy- validation	0.7699
Final epoch strong directional accuracy- training	0.4863
Final epoch strong directional accuracy- validation	0.4122
Final epoch weak directional accuracy- training	0.4864
Final epoch weak directional accuracy- validation	0.5616
Final epoch total directional accuracy- training	0.9727
Final epoch total directional accuracy- validation	0.9737
Min epoch loss- training	0.1413
Min epoch loss- validation	0.1196

The Result (3/5): Training and Validation

- Graphic training and validation results:



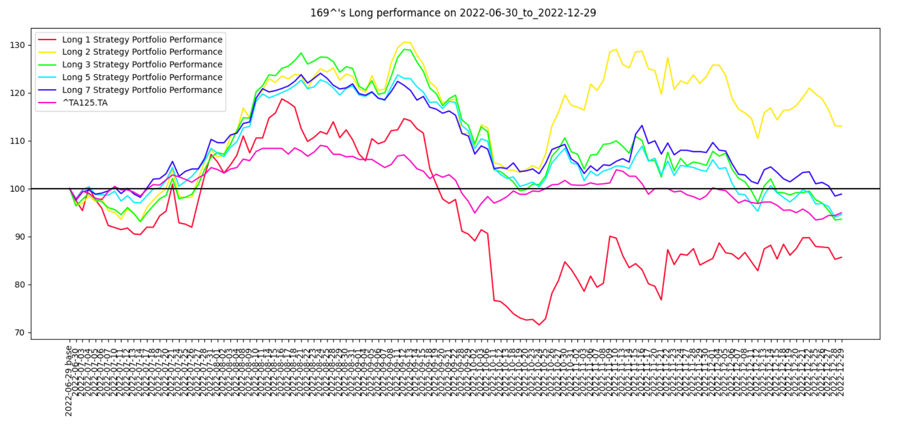
The Result (4/5): Testing

- Interesting test results:

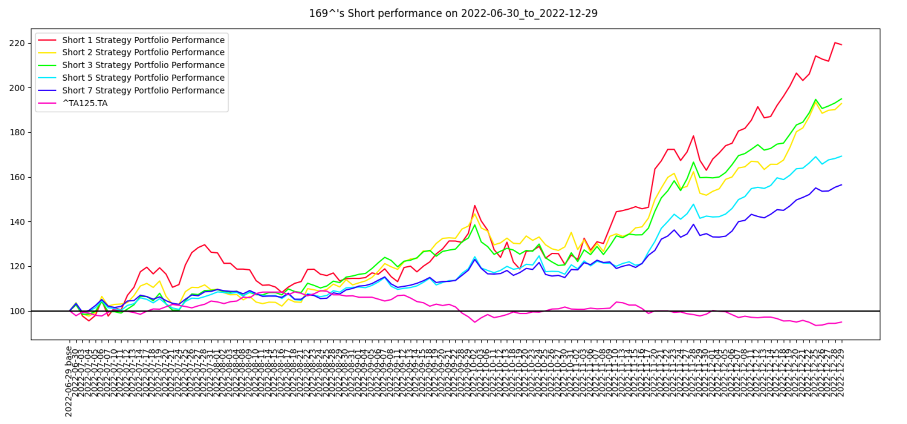
The Result (5/5): Backtesting

- Super-strategy **#169**'s backtesting results.

- **Long positions:**



- **Short positions:**



Thank you

tom@pickel.io

https://pickel.io

Progress bar courtesy of: howtogeek.com.

Q & A

Please join us for our upcoming webinar:

Register Here:
https://bit.ly/3Q7KWRP

Thank You

FDP INSTITUTE®
by CAIA

Contact Us:

fdpinstitute.org

info@fdpinstitute.org

@FDPbyCAIA

linkedin.com/company/FDP Institute