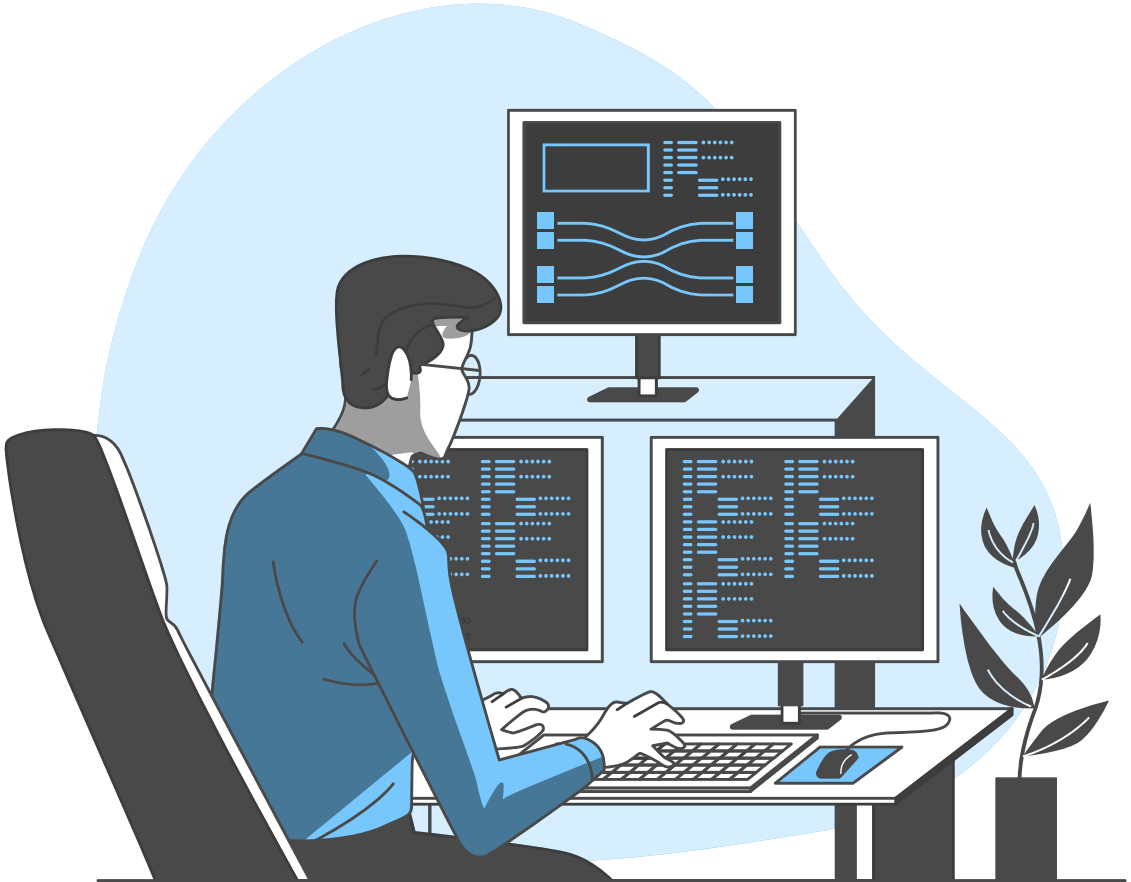


# Asset Pricing using Deep Neural Network

# Team Members

Nikhil Katiki

Naveen Kumaran Kumar

# Agenda

- Problem Statement
- Why Neural network for this problem statement
- DNN parameters considered
- Base model
- Findings of Tuning parameters exploration (D1 60 models and D2 25 models)
- Best vs Worst Model comparison
- Out of box trials
- Learnings

# Dataset 1

# Problem Statement

## Given

- A monthly stock price **returns for 690 months** (about 58 years) has been provided
- The dataset has **64 features** after pre-processing
- A total of **18719 stocks**

## Goal

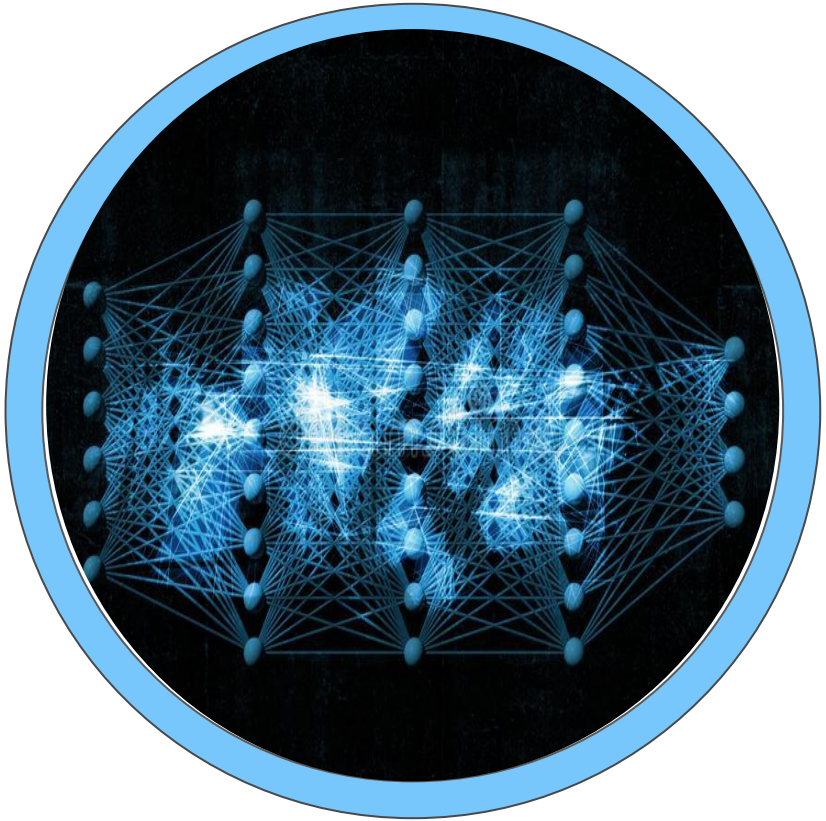
- To **predict the stock return accurately for 42 years** by creating the portfolio to sell bottom 10% stocks and buy top 10% stocks
- The model prediction is evaluated by Sharpe Ratio

## Performance Evaluation (Sharpe Ratio)

- Sharpe ratio is a **metric for the investors to evaluate the return** of an investment considering its associated risk.
- **Sharpe Ratio = (Return of portfolio – Risk free rate)/Standard deviation of the portfolio's excess return**
- **Higher the sharpe ratio better** the investment in terms of return considering their associated risk

- Reference - https://www.investopedia.com/terms/s/sharperatio.asp

# Why Neural network!!!



# Deep Neural Network Parameters

**OPTIMIZER**
Methods to find the objective function

**EPOCHS**
Number of times a dataset can be trained

**BATCH SIZE**
Batch processing for weights updating

**LOSS FUNCTION**
Loss functions are objective functions

**PATIENCE**
Keep a check on Validation accuracy

**#LAYERS**
Adding more layers

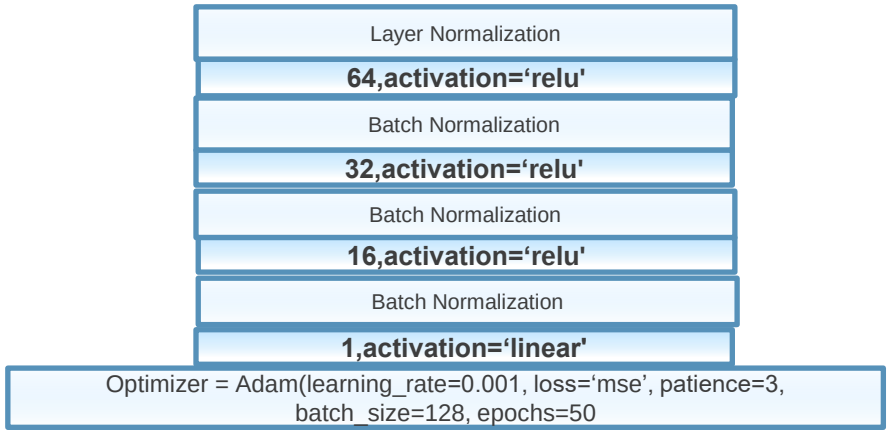
**ACTIVATION FUNCTION**
Adding Non-linearity to the model

**#NEURONS**
Number of neurons

# DNN paramters considered

| Sno. | Process | Rationale |
|---|---|---|
| 1 | Randomly | Since we wanted to take one direction |
| 2 | Layers | Increasing the number of layers increases prediction accuracy |
| 3 | Number of neurons | Increasing the number of neurons increases prediction accuracy |
| 4 | Activation functions | Functions to introduce the Non-linearity |
| 5 | Optimizer | Method to solve the objective function |
| 6 | Batch size | Updating weights using a batch |
| 7 | Learning Rate | Choosing an adaptive learning rate will help to “adapts” to the landscape |
| 8 | Epochs | Number of Epochs |
| 9 | Patience | Reduce the overfitting |

# Base Model – Sharpe ratio 5.12



# Number of layers and neurons

| Random tries from given base model | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Model No | Activation function | Batch size | Number of epochs | Optimizer | Number of layers | Network architecture | Patience | Sharpe Ratio |
| 1 | relu | 128 | 50 | adam (learning rate=0.001) | 4 | 64,32,16,1 | 3 | 5.12 |
| 2 | relu | 128 | 80 | adam (learning rate=0.001) | 7 | 64,64,32,32,16,16,1 | 30 | 5.03 |
| 3 | relu | 128 | 80 | adam (learning rate=0.001) | 5 | 64,32,32,16,1 | 30 | 5.07 |

- **Patience were increased significantly** from 3 to allow the model to learn
- **Learning** – Comparing models with a **change of multiple parameters is not an apple to apple comparison** to make inference

| No duplets with pyramid | | | |
|---|---|---|---|
| Model no. | Number of layers | Number of neurons | Sharpe Ratio |
| 4 | 5 | 64,32,16,8,1 | 5.29 |
| 5 | 6 | 64,32,16,8,4,1 | 5.37 |
| 6 | 7 | 64,32,16,8,4,2,1 | 5.22 |

- The number of layers were increased considering the **pyramid structure**.
- **Initially 64 were chosen as the cap** of the structure since there are 64 features
- **No duplets** were considered initially

# Number of layers and neurons

- Since it was a **regression problem what if we add two neuron layer before the last layer!**

| 1 Duplets with 2 neurons in last but one layer | | | |
|---|---|---|---|
| **Model no.** | **Number of layers** | **Number of neurons** | **Sharpe Ratio** |
| 7 | 8 | 64,32,16,8,4,4,2,1 | 4.86 |
| 8 | 8 | 64,32,16,8,8,4,2,1 | 5.08 |
| 9 | 8 | 64,32,16,16,8,4,2,1 | 4.50 |
| 10 | 8 | 64,32,32,16,8,4,2,1 | 5.30 |
| 11 | 8 | 64,64,32,16,8,4,2,1 | 5.61 |

| 1 Duplets without 2 neurons in last but one layer | | | |
|---|---|---|---|
| **Model no.** | **Number of layers** | **Number of neurons** | **Sharpe Ratio** |
| 12 | 7 | 64,32,16,8,4,4,1 | 4.68 |
| 13 | 7 | 64,32,16,8,8,4,1 | 5.29 |
| 14 | 7 | 64,32,16,16,8,4,1 | 5.44 |
| 15 | 7 | 64,32,32,16,8,4,1 | 5.18 |
| 16 | 7 | 64,64,32,16,8,4,1 | 5.25 |

- **Learning** – **Roughly there was an increasing trend of sharpe ratio when the duplet moved towards initial layers** with higher neurons for both cases

# Number of layers and neurons

| 2 Duplets without 2 neurons in last but one layer | | | |
|---|---|---|---|
| **Model no.** | **Number of layers** | **Number of neurons** | **Sharpe Ratio** |
| 17 | 8 | 64,64,32,32,16,8,4,1 | 5.12 |
| 18 | 8 | 64,64,32,16,16,8,4,1 | 4.99 |
| 19 | 8 | 64,64,32,16,8,8,4,1 | 5.29 |

| 3 Duplets without 2 neurons in last but one layer | | | |
|---|---|---|---|
| **Model no.** | **Number of layers** | **Number of neurons** | **Sharpe Ratio** |
| 20 | 9 | 64,64,32,32,16,16,8,4,1 | 5.15 |

- **Learning** – From all the **20 different model architectures**, we inferred that **adding more layers produced relatively better results**
- From running time, **we realized that it is not computationally feasible to add more layers** and tune other parameters.

# Optimizer

| Model no. | Optimizer | Sharpe Ratio |
|---|---|---|
| 21 | rmsprop(0.001) | 5.22 |
| 22 | SGD(learning_rate=0.001, momentum=0.0, nesterov=False) | 1.26 |
| 23 | SGD(learning_rate=0.001, momentum=0.0, nesterov=True) | 0.77 |
| 24 | SGD(learning_rate=0.001, momentum=0.9, nesterov=False) | 1.92 |
| 25 | SGD(learning_rate=0.001, momentum=0.9, nesterov=True) | 2.43 |
| 26 | Adagrad(learning_rate=0.001) | 1.89 |
| 27 | Adadelta(learning_rate=0.001) | 0.98 |
| 28 | Adam(learning_rate=0.001) | 5.61 |

- **Learning** – When **compared to SGD, momentum/Nesterov accelerated gradient, rms prop, Adagrad and Adadelta, Adam performed significantly better**.
- **Rms prop** was the **next high performing** optimizer

# Activation function

| Model no. | Activation function | Sharpe Ratio |
|---|---|---|
| 29 | tanh | 5.11 |
| 30 | prelu | 3.99 |
| 31 | leaky relu | 5.19 |
| 32 | relu | 5.61 |
| 33 | relu, tanh | 5.20 |
| 34 | relu, prelu | 5.18 |
| 35 | swish | 5.39 |
| 36 | elu | 4.97 |
| 37 | sigmoid | 5.16 |
| 38 | relu last but one linear | 5.07 |

## Learnings

- **Basic activation functions** like **tanh and sigmoid** were tried
- Apart from relu from various references, other proven better performing activation functions like **leaky relu, prelu, elu, swish** were tried – **prelu** performed significantly **bad** and **swish** perfomed significantly **better**, a **common disadvantage of computationally expensive is notable**

- **Alternate activation functions** were tried with 1)**relu, one better performing as per literature (prelu)** 2) **relu,another average performing activation function (tanh)** - **No significant differences** were observed.

- **Relu except linear activation function in last but one layer** – Did not perform better

- Reference - https://towardsdatascience.com/a-quick-guide-to-activation-functions-in-deep-learning-4042e7addd5b

# Batch size

| Model no. | Batchsize | Sharpe Ratio |
|---|---|---|
| 39 | 64 | 4.93 |
| 45 | 80 | 5.17 |
| 43 | 100 | 4.85 |
| 47 | 128 | 5.61 |
| 44 | 150 | 4.39 |
| 46 | 200 | 5.18 |
| 40 | 256 | 4.58 |
| 41 | 512 | 4.78 |
| 42 | 1024 | 3.76 |

- Various batch size ranging from 64 to 1024 were tried

**Learnings**

- **Increasing the batch size after a threshold decreased the sharpe ratio** as expected.
- **Sharpe ratio were fluctuating even with a batch size near to the high performing value (128)**. So might **need multiple trials to find optimal batch size**

# Learning rate

| Model no. | Learning rate | Sharpe Ratio |
|---|---|---|
| 48 | Adam(learning_rate=0.0005) | 3.968 |
| 51 | Adam(learning_rate=0.001) | 5.61 |
| 52 | Adam(learning_rate=0.007) | 5.272 |
| 53 | Adam(learning_rate=0.01) | 5.396 |
| 54 | Adam(learning_rate=0.015) | 5.281 |
| 55 | Adam(learning_rate=0.02) | 5.48 |

- Learning rate ranging from **0.0005 to 0.02** were tried.

- **Learnings**
- **Reducing the learning rate reduced sharpe ratio**, where **more** number of **epochs might be necessary**

- **Increasing the learning rate increased the sharpe ratio till 0.02 cutoff post which, it started decreasing**

# Ensemble model

| Exploring ensemble model | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Model No. | Activation function | Batch size | Number of epochs | Optimizer | Number of layers | Network architecture | Patience | Sharpe Ratio |
| 56 | relu in both network | 128 | 30 | Adam(learning_rate=0.001) | 8 and 6 | 64,64,32,16,8,4,2,1 and 64,32,16,8,4,1 | 10 | 5.35 |

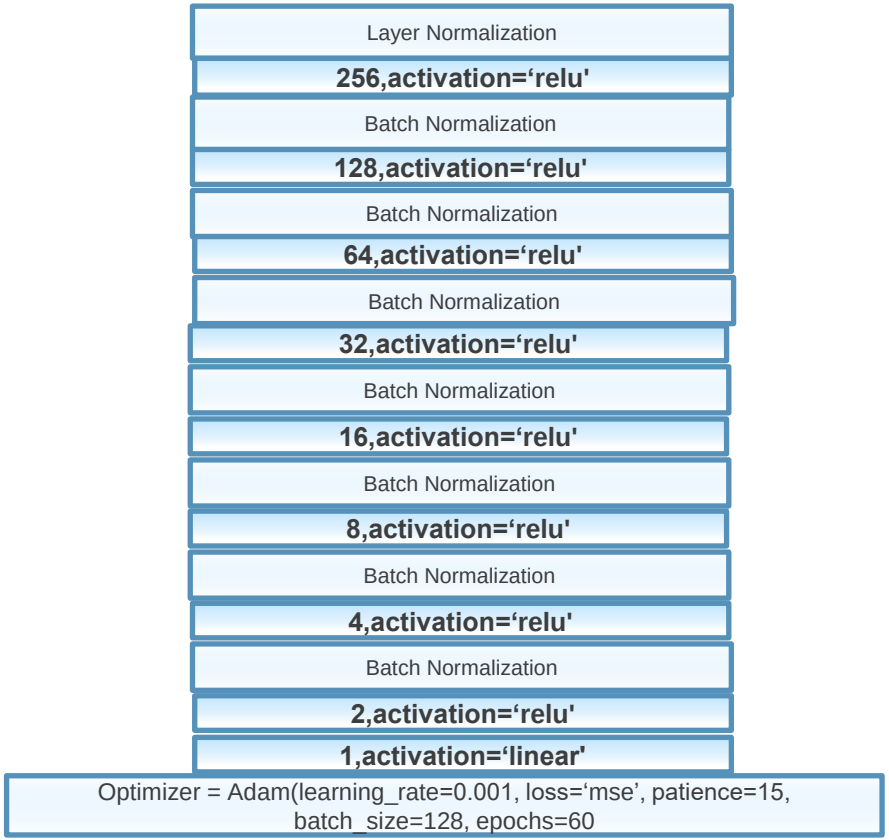
- Though ensemble model provided higher sharpe ratio, creating an **ensemble model of two DNN made sense when the validation dataset is unknown to get a stable model**.

# Time to increase number of layers, neurons, Epoch and Patience

| Exploring patience, number of epochs and increasing the number of layers and neurons | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Model No. | Activation function | Batch size | Number of epochs | Optimizer | Number of layers | Network architecture | Patience | Sharpe Ratio |
| 57 | relu | 128 | 60 | Adam(learning_rate=0.001) | 7 | 256,128,64,32,16,8,4,2,1 | 15 | 5.64 |
| 58 | relu | 128 | 70 | Adam(learning_rate=0.001) | 8 | 512,256,128,64,32,16,8,4,2,1 | 15 | 4.75 |

- Given other parameters fixed, increasing the number of layers, neurons and subsequently higher epoch and patience to train better increased the sharpe ratio significantly to **5.64**

# Final Model – Sharpe ratio 5.64



# Best vs Worst Model comparison

| Exploring patience, number of epochs and increasing the number of layers and neurons | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Model No. | Activation function | Batch size | Number of epochs | Optimizer | Number of layers | Network architecture | Patience | Sharpe Ratio |
| 57 | relu | 128 | 60 | Adam(learning_rate=0.001) | 7 | 256,128,64,32,16,8,4,2,1 | 15 | 5.64 |
| 58 | relu | 128 | 70 | Adam(learning_rate=0.001) | 8 | 512,256,128,64,32,16,8,4,2,1 | 15 | 4.75 |

| S.No | Activation function | Batch size | Number of epochs | Optimizer | Number of layers | Network architecture | Patience | Sharpe Ratio |
|---|---|---|---|---|---|---|---|---|
| 23 | relu | 128 | 30 | SGD(learning_rate=0.001, momentum=0.0, nesterov=True) | 8 | 64,64,32,16,8,4,2,1 | 10 | 0.77 |

- **Number of layers and number of neurons in each layer, Optimizer, number of epochs play a significant role** in model performance

# Out of box trials and takeaways

- Given the regression problem, **Activation function was removed in last but one layer,** but did not impact significantly in performance
- **Ensemble of two Neural Networks might be more suitable to produce a robust model** when validation set in unknown.

- **Alternate and random removal of batch normalization** did not impact the model performance significantly
- **Alternate activation function** were tried but did not produce significant improvement in performance when compared to relu

# Overall takeaways

- **Number of layers and neurons are the major parameter** which produces significant improvement in the model performance
- **Swish activation function performed significantly better apart from relu**. More epochs might improve the prediction
- After a threshold, it is **not necessary that if we increase the number of layers and epochs to very high value it will improve accuracy**

# Dataset 2

# Number of layers and Activation Functions

| Number of Layers | | | |
|---|---|---|---|
| **Model no.** | **Number of layers** | **Number of neurons** | **R2** |
| 1 | 6 | 2056,256,128,64,32,1 | -1.72 |
| 2 | 7 | 2056,1024,512,256,128,64,1 | -1.29 |
| 3 | 8 | 2056,1024,512,256,128,64,32,1 | -0.534073 |
| 4 | 9 | 4098,2056,1024,512,256,128,64,32,1 | -1.34 |

- **Learning** – With increase in layers, the R2 got better only till 8 layers. It started overfitting after that

| Activation Function | | | |
|---|---|---|---|
| **Model no.** | **Number of layers** | **Activation functions** | **R2** |
| 5 | 8 | tanh | -0.56 |
| 6 | 8 | Relu | -1.93 |
| 7 | 8 | Prelu | -1.27 |
| 8 | 8 | sigmoid | -0.37 |
| 9 | 8 | Sigmoid and tanh | -0.302594 |

- Individually, Sigmoid and tanh performed very well. One interesting finding is adding sigmoid and tanh alternately improved the R2

# Number of Dropuout layers and Loss Function

- Since it was a regression problem what if we add two neuron layer before the last layer!

| Dropout ratio and Number of Dropuout | | | |
|---|---|---|---|
| **Model no.** | **Number of layers** | **Number of neurons** | **R2** |
| 10 | 3 | 0.2,0.2,0.2 | -0.40 |
| 11 | 4 | 0.3,0.3,0.3,0.05 | -0.312615 |
| 12 | 5 | 0.3,0.3,0.30,0.1,0.05 | -0.40 |

- **Learning** – It is very evident about the importance of dropout layers as this could reduce the overfitting the data

| Loss Function | | | |
|---|---|---|---|
| **Model no.** | **Loss function** | **R2** | |
| 13 | Mean absolute error | -20.858745 | |
| 14 | an squared logarthmic e | -0.454134 | |
| 15 | mse | -0.250568 | |

- **Learning** – Solving the loss function with mse as hyperparameters gives the best model

# Number of Epochs, Batch size

| Epochs | | | |
|---|---|---|---|
| **Model no.** | **Epochs** | **Batch size** | **Sharpe Ratio** |
| 16 | 100 | 30 | -35.22 |
| 17 | 100 | 30 | -0.32 |
| 18 | 120 | 20 | -0.46 |
| 19 | 120 | 20 | -0.33 |
| 20 | 120 | 40 | -0.300459 |

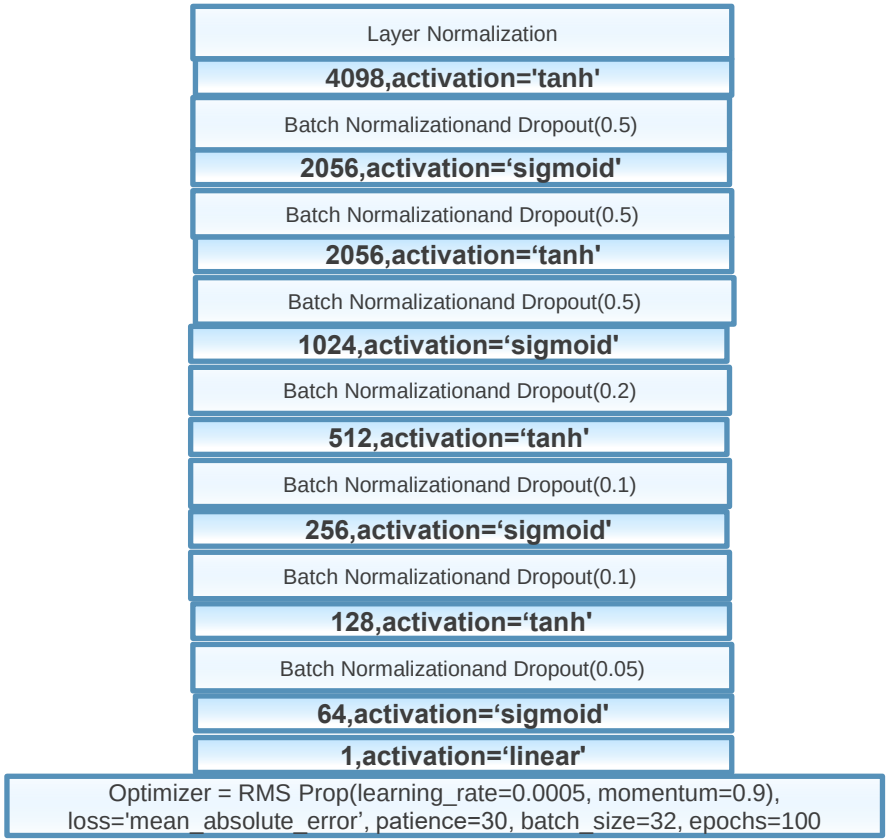
- **Learning** – With increase in Epochs, the R2 is improving but it is very computationally heavy

# Optimizer

| Model no. | Optimizer | Sharpe Ratio |
|---|---|---|
| 20 | SGD | -20.86 |
| 21 | Adam | -0.45 |
| 22 | RMSprop(learning_rate=0.005, momentum=0.9, nesterov=False) | -0.250568 |
| 23 | RMSprop(learning_rate=0.001, momentum=0.9, nesterov=False) | -0.28 |
| 24 | RMSprop(learning_rate=0.004, momentum=0.9, nesterov=True) | -0.63 |

- **Learning** – RMSprop with the momentum of 0.9 is the best among all the optimizers with the existing architecture

# Best Model


Layer Normalization
4098,activation='tanh'
Batch Normalizationand Dropout(0.5)
2056,activation='sigmoid'
Batch Normalizationand Dropout(0.5)
2056,activation='tanh'
Batch Normalizationand Dropout(0.5)
1024,activation='sigmoid'
Batch Normalizationand Dropout(0.2)
512,activation='tanh'
Batch Normalizationand Dropout(0.1)
256,activation='sigmoid'
Batch Normalizationand Dropout(0.1)
128,activation='tanh'
Batch Normalizationand Dropout(0.05)
64,activation='sigmoid'
1,activation='linear'
Optimizer = RMS Prop(learning_rate=0.0005, momentum=0.9),
loss='mean_absolute_error', patience=30, batch_size=32, epochs=100

# Exploding Gradient Descent

# Best Model vs Least Accurate

| | | Best Model | Least Accurate Model |
|---|---|---|---|
| S.No. | | 22 | 14 |
| R2 | | -0.250568 | -35.217213 |
| Optimizer | Optimizer | RMSprop | adam |
| | Rate | 0.0005 | Default |
| | Momentum | 0.9 | Default |
| Loss | | mean_absolute_error | mse |
| Epochs | | 100 | 100 |
| Batch size | | 32 | 32 |
| Patience | | 30 | 20 |
| verbose | | 1 | 1 |
| Layer1 | Neurons | Layer Normalization | Layer Normalization |
| | Activation Function | | |
| Layer2 | Neurons | 4098 | 4098 |
| | Activation Function | tanh | relu |
| Layer3 | Neurons | Batch Normalization | Batch Normalization |
| | Activation Function | | |
| Layer4 | Neurons | DropOut 0.5 | DropOut 0.5 |
| | Activation Function | | |
| Layer5 | Neurons | 2056 | 2056 |
| | Activation Function | sigmoid | relu |
| Layer6 | Neurons | Batch Normalization | Batch Normalization |
| | Activation Function | | |
| Layer7 | Neurons | 1024 | 1024 |
| | Activation Function | tanh | relu |
| Layer8 | Neurons | DropOut 0.2 | DropOut 0.2 |
| | Activation Function | | |
| Layer9 | Neurons | Batch Normalization | Batch Normalization |
| | Activation Function | | |
| Layer10 | Neurons | 512 | 512 |
| | Activation Function | sigmoid | relu |
| Layer11 | Neurons | Batch Normalization | Batch Normalization |
| | Activation Function | | |
| Layer12 | Neurons | DropOut 0.1 | DropOut 0.1 |
| | Activation Function | | |
| Layer13 | Neurons | 256 | 256 |
| | Activation Function | tanh | relu |
| Layer14 | Neurons | BatchNormalization | BatchNormalization |
| | Activation Function | | |
| Layer15 | Neurons | DropOut 0.1 | DropOut 0.1 |
| | Activation Function | | |
| Layer16 | Neurons | 128 | 128 |
| | Activation Function | sigmoid | relu |
| Layer17 | Neurons | BatchNormalization | BatchNormalization |
| | Activation Function | | |
| Layer18 | Neurons | Dropout 0.05 | Dropout 0.05 |
| | Activation Function | | |
| Layer19 | Neurons | 64 | 64 |
| | Activation Function | tanh | relu |
| Output Layer | | Output1 | Output1 |

# Learnings

**OPTIMIZER**
Methods to find the objective function

**EPOCHS**
Number of times a dataset can be trained

**BATCH SIZE**
Batch processing for weights updating

**LOSS FUNCTION**
Loss functions are objective functions

**PATIENCE**
Keep a check on Validation accuracy

**#LAYERS**
Adding more layers

**ACTIVATION FUNCTION**
Adding Non-linearity to the model

**#NEURONS**
Number of neurons

# Overall Learnings

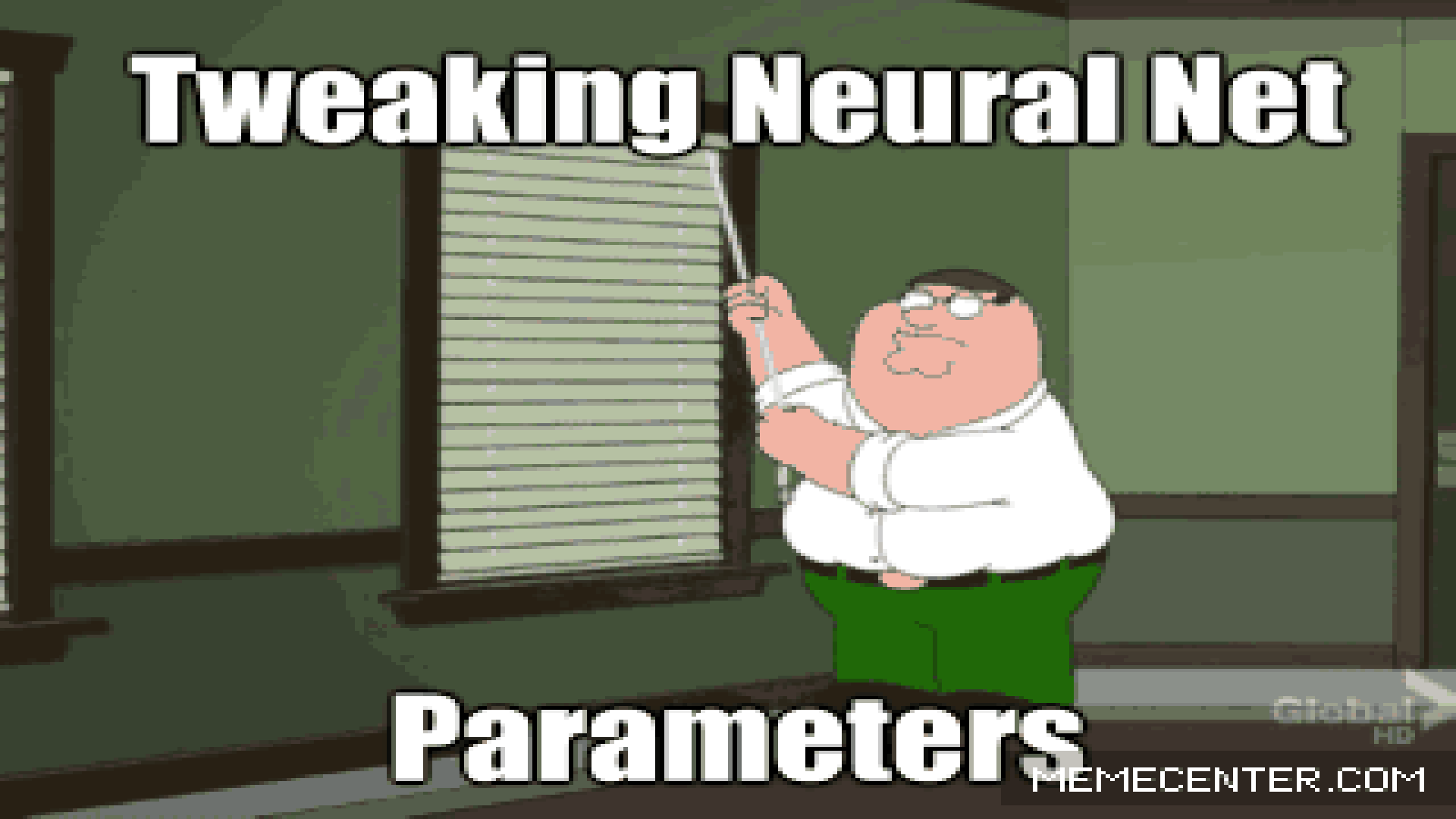
1. It is very important to use our intuition while training the deep neural networks as it is not practical to use all the possible combinations

2. There are new combinations that have tried:
    a. Kernel Initializer
    b. Encountered Vanishing Gradient descent
    c. Encountered Exploding Gradient descent
    d. Importance of Dropout
    e. Alternate activation functions

3. The Modelling exercise is all about trying different parameters and understanding the directions

"**We learned a lot about what not to do!!**"

*- Deep Learning thoughts*


Tweaking Neural Net
Parameters
MEMECENTER.COM