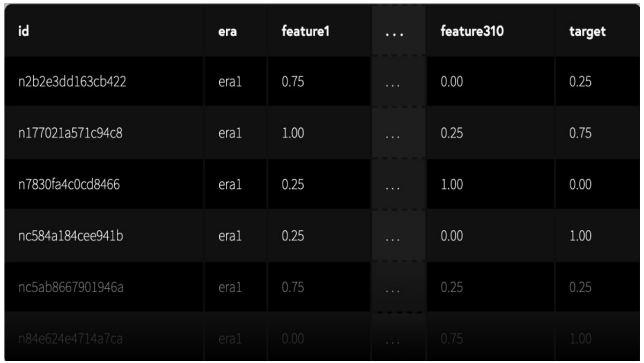
Predicting Stock Return using Neural Networks

Training Models on NumerAI Dataset

Team 6:
Ruiqi Jiang
Boyuan Chen
Rohit Devanaboina

What is NumerAI

- A “Crowd-Sourced” Hedge Fund that makes trading decisions based on predictive models submitted by community members

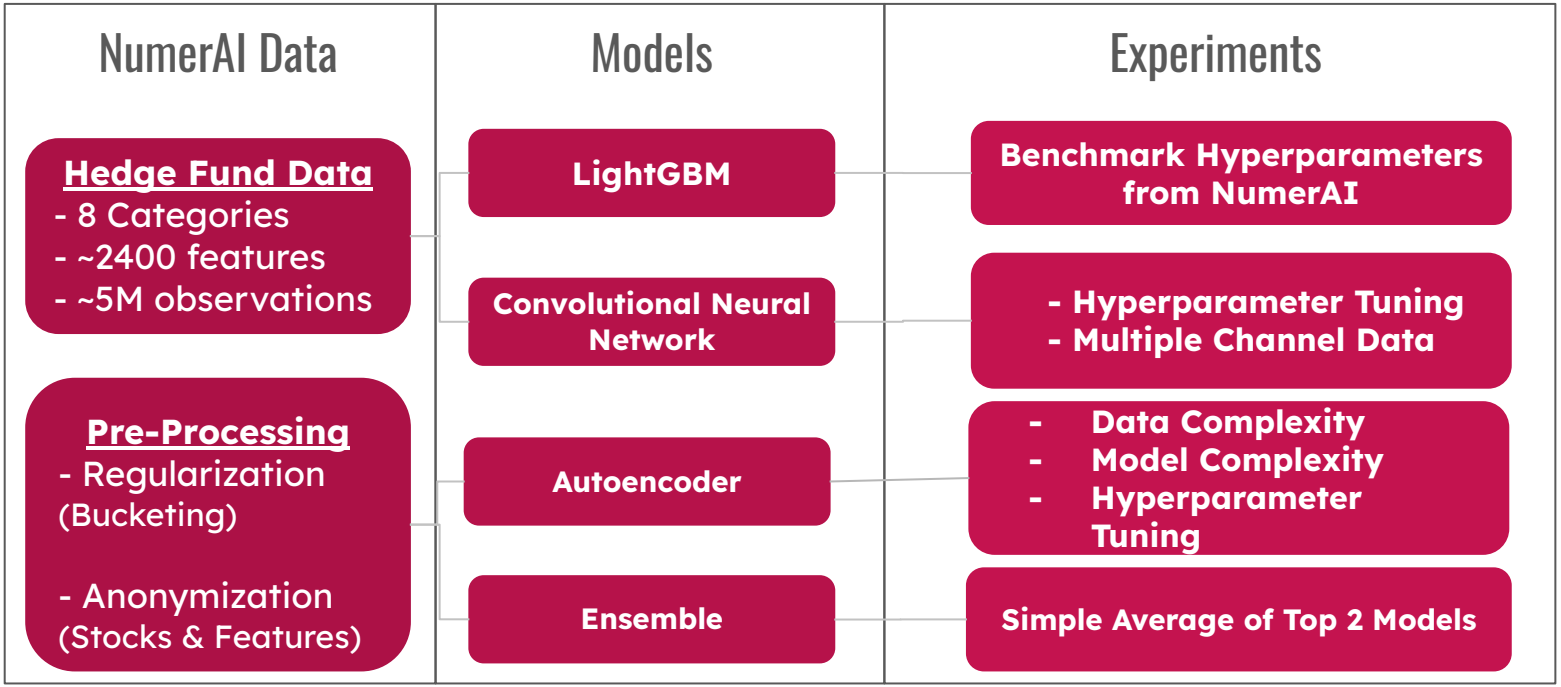


id	era	feature1	...	feature310	target
n2b2e3dd163cb422	era1	0.75	...	0.00	0.25
n177021a571c94c8	era1	1.00	...	0.25	0.75
n7830fa4c0cd8466	era1	0.25	...	1.00	0.00
nc584a184cee941b	era1	0.25	...	0.00	1.00
nc5ab8667901946a	era1	0.75	...	0.25	0.25
n84e624e4714a7ca	era1	0.00	...	0.75	1.00

- Key Terms
  - Era - A point in time
  - ID - Unique tag given to a stock in an era
  - Target - Return of the stock 20 days into the future
- Metrics of Performance
  - Correlation - Measures directionality and intensity of predictions
  - Meta Model Contribution - Model’s novel improvement to the ensemble model

Methodology

Objective: Predict return of a stock 20-days into the future given a snapshot of its current state



Benchmark - LightGBM

- **70%** of Models on NumerAI are LightGBM
- NumerAI has **benchmarks w/ hyperparameters** for easy training

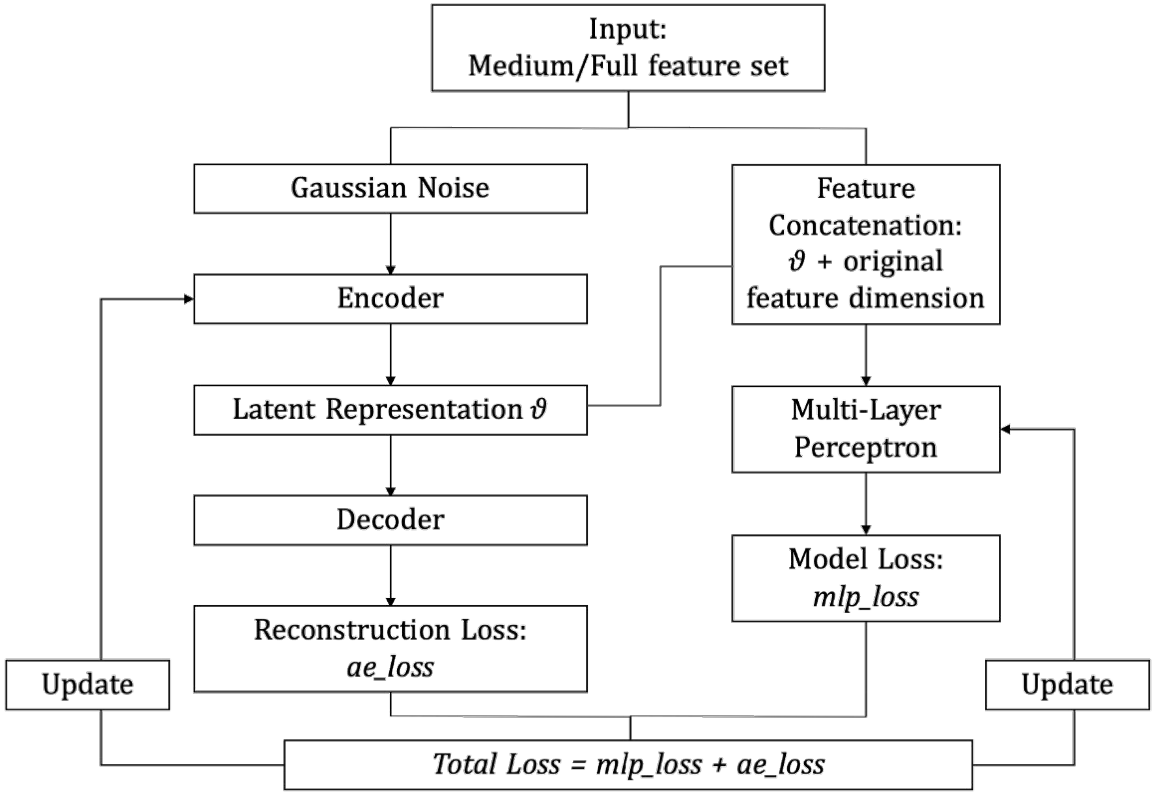
Hyperparameters:

- # of Trees : 2000,
- Learning Rate: 0.01,
- Max Depth: 5,
- Max Leaves: 2**5-1,
- "colsample_bytree": 0.1

Baseline Metrics	
Metric	**Value**
MAE	0.152
MSE	0.049
CORR	0.0078
BMC	-0.0005

Autoencoder + MLP

MLP + Autoencoder: Training Logic



Our strategy: train the MLP and Auto Encoder together.

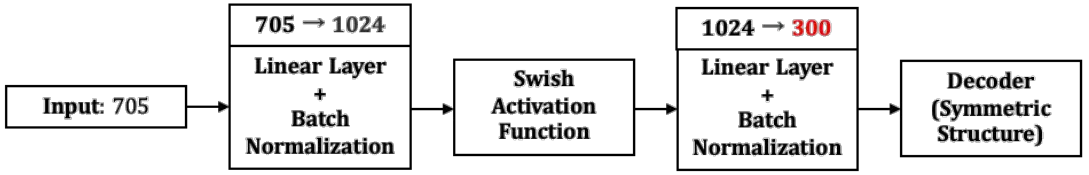
Default Settings:

- Optimizer: Adam
- Activation: Swish
- Loss Function: MSE
- Update Weights: total loss
- Learning Rate: 0.001
- Batch Size: 64

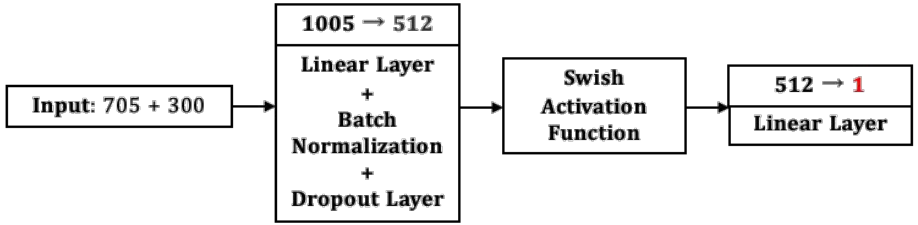
$$\text{swish}(x) = x\,\text{sigmoid}(\beta x) = \frac{x}{1+e^{-\beta x}}.$$

Experiment 1: Naive MLP & Autoencoder

- For Auto Encoder:



- For MLP:

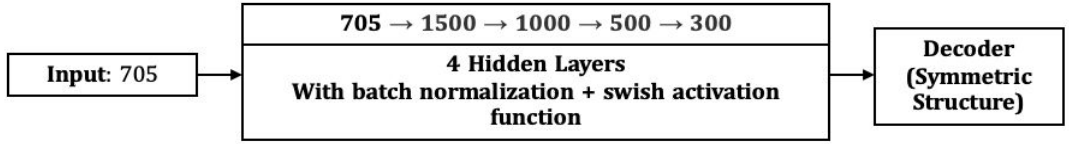


Epochs	Total Train Loss	ae_loss	mlp_loss	Validation loss
1	0.4246	0.3572	0.0674	0.0518
2	0.4354	0.3664	0.0689	0.0499
3	0.3798	0.3044	0.0754	0.0498
4	0.3960	0.3288	0.0672	0.0496
5	0.3880	0.3105	0.0775	0.0497
6	0.3465	0.2781	0.0684	0.0496
7	0.4607	0.4029	0.0577	0.0497
8	0.3472	0.2820	0.0652	0.0497
9	0.3376	0.2602	0.0774	0.0497
10	Early Stopping	Early Stopping	Early Stopping	Early Stopping

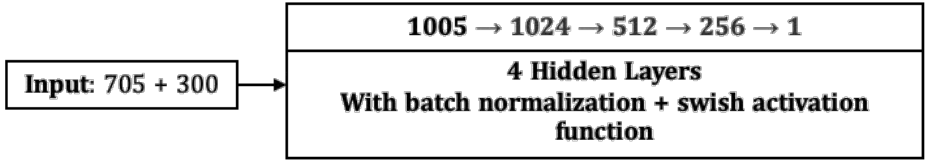
It indicates some level of underfitting

Experiment 2: Augmented MLP & Autoencoder + Hyperparameter Tuning

- Updated Auto Encoder:



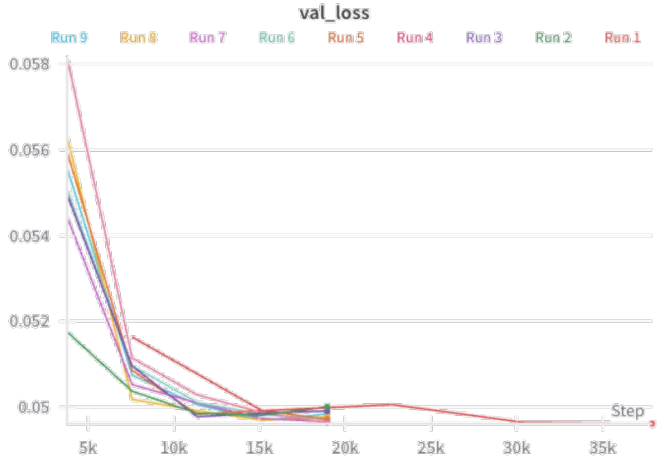
- Updated MLP:



Hyperparameter grid:

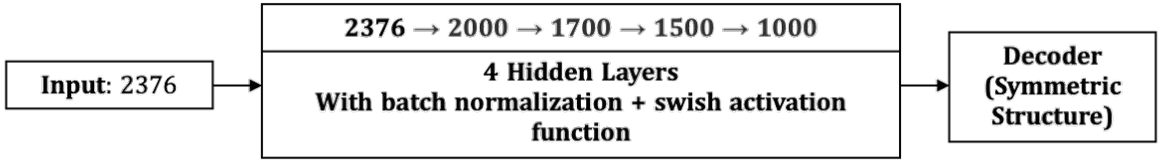
- Learning Rate: [0.01, 0.001, 0.0001]
- Batch Size: [32, 64, 128]
- Dropout Rate: [0.2, 0.3, 0.4]

Best Validation Loss: 0.04962, didn't improve a lot

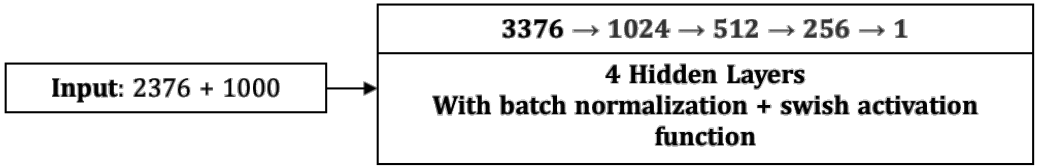


Experiment 3: Augmented MLP & Autoencoder + Hyperparameter Tuning

- **Updated Auto Encoder:**



- **Updated MLP:**



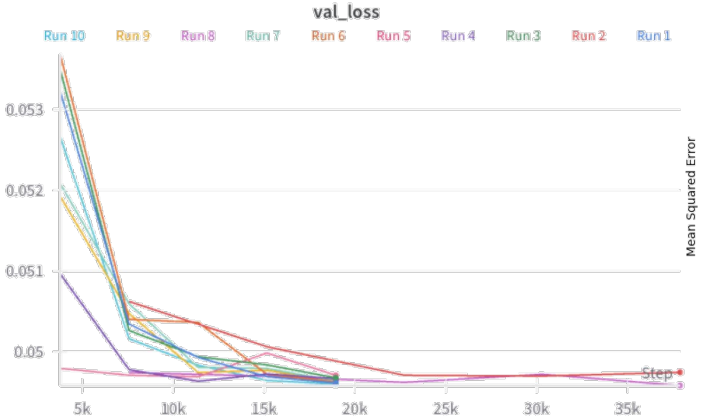
Hyperparameter grid:

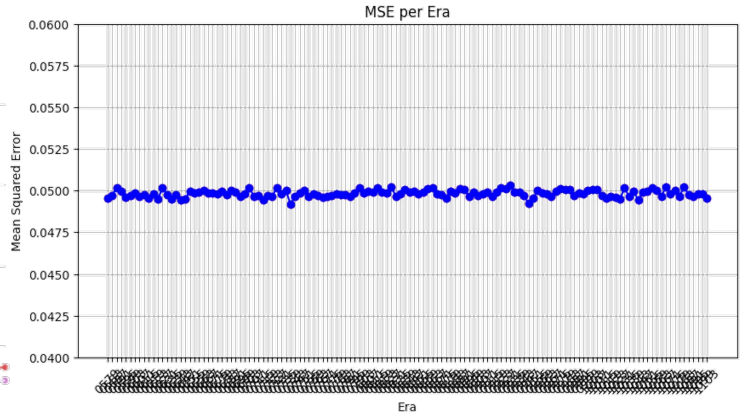
- Learning Rate: [0.001, 0.0001]
- Batch Size: [64, 128]
- Dropout Rate: [0.3, 0.4]

Best Validation Loss: 0.4958

Test Set MSE: 0.04983

Test Set MAE: 0.1543





Convolutional Neural Network

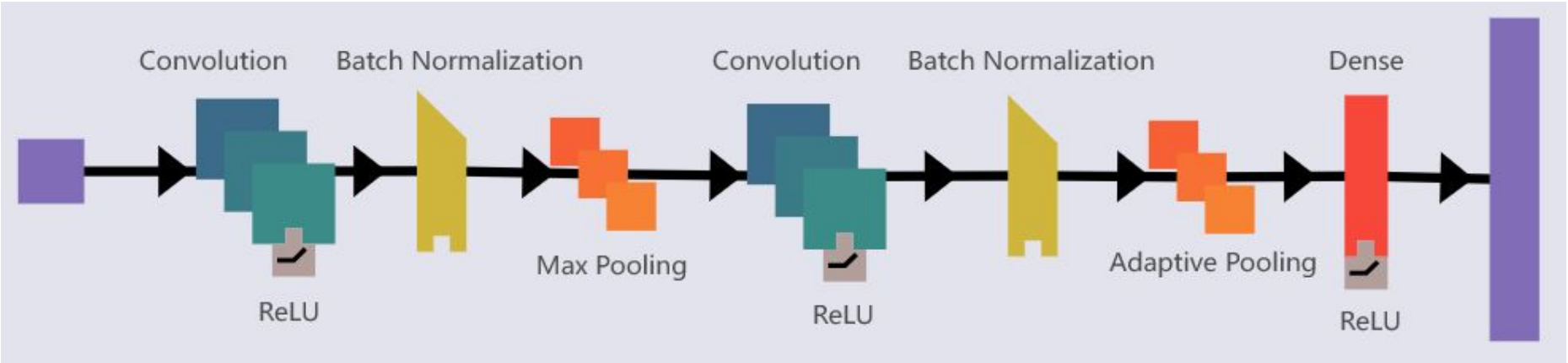
CNN: Training Logic

Our Strategies:

- Train the CNN with hyperparameter tuning.
- Train the CNN on reconstructed data which has more than one channel.

Default Settings:

- Optimizer: Adam (Adaptive Moment Estimation)
- Loss Function: MAE
- Learning Rate: 0.001
- Batch Size: 64



Experiment 1: CNN + Hyperparameter Tuning

Layer (type:depth-idx)	Output Shape	Param #
CNN	[64, 1]	—
├─Conv1d: 1-1	[64, 32, 1188]	128
├─BatchNorm1d: 1-2	[64, 32, 1188]	64
├─MaxPool1d: 1-3	[64, 32, 593]	—
├─Conv1d: 1-4	[64, 64, 297]	6,208
├─BatchNorm1d: 1-5	[64, 64, 297]	128
├─AdaptiveAvgPool1d: 1-6	[64, 64, 1]	—
├─Linear: 1-7	[64, 1]	65

Epochs	Train Loss (Before Tuning)	Train Loss (After Tuning)	Validation Loss (Before Tuning)	Validation Loss (After Tuning)
1	0.14993	0.15073	0.14982	0.15062
2	0.14977	0.15044	0.14970	0.15035
3	0.14971	0.14970	0.14962	0.14960
4	0.14974	0.15045	0.14967	0.15035
5	0.14972	0.14990	0.14965	0.14979
6	0.14979	0.14999	0.14971	0.14988
7	Early Stopping	Early Stopping	Early Stopping	Early Stopping

Hyperparameter Grid:

- Learning Rate: [0.001, 0.01]
- Batch Size: [64, 128]

Best Validation Loss: 0.14960
Test Set MAE: 0.1495

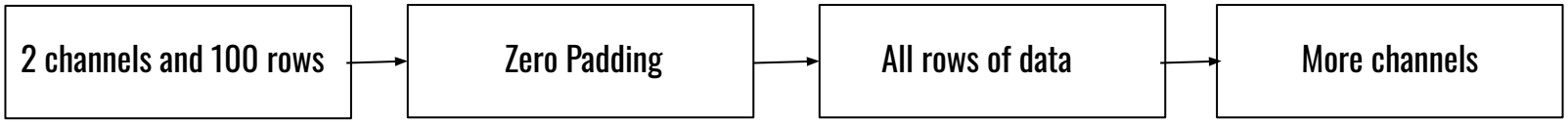
Reason:
A large batch size and learning rate can accelerate the training process, helping the model converge faster.

Experiment 2: CNN + Reconstructed Data

Original Situation:
1 channel with all 2376 features in it
Best Case Reconstruction:
8 channels with 335 features for each
Reason:
Enables the network to learn richer and more diverse features, enhancing its ability to represent input data.

Potential Channels	Number of Features Contained
constitution	335
charisma	290
agility	145
wisdom	140
strength	135
serenity	95
dexterity	51
intelligence	35

Process:

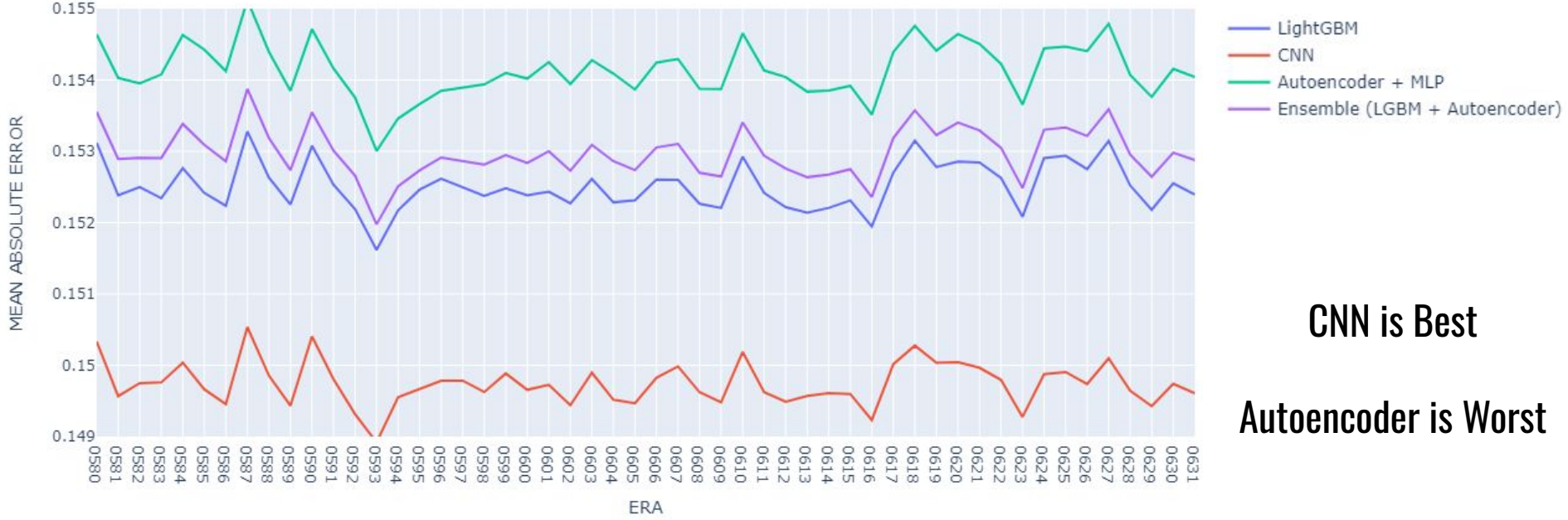


Stops here due to limited RAM

Future Explorations:
1. Truncation instead of padding; 2. Uses devices that support both GPU and higher RAM

Performance Metrics (1 Year Test Data)

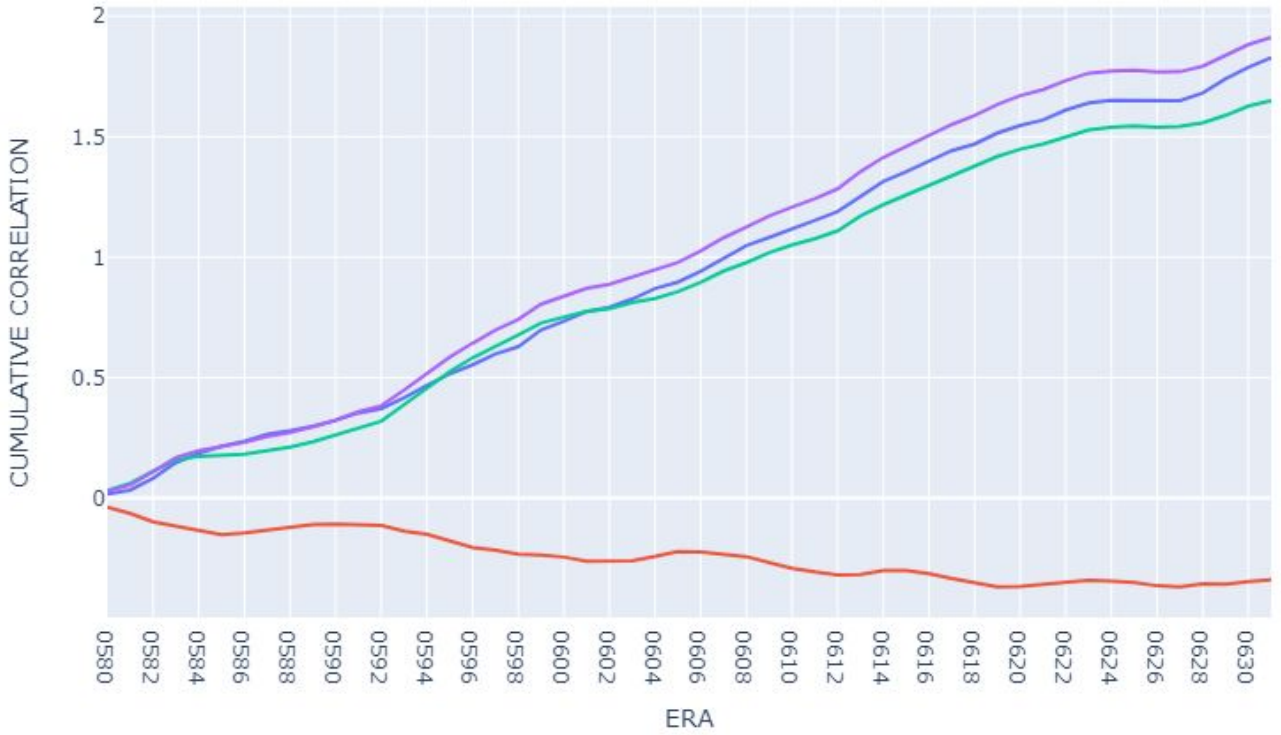
Training Performance - MAE by Model (1 Year Test Data)



CNN is Best

Autoencoder is Worst

True Performance - Cumulative Correlation over time

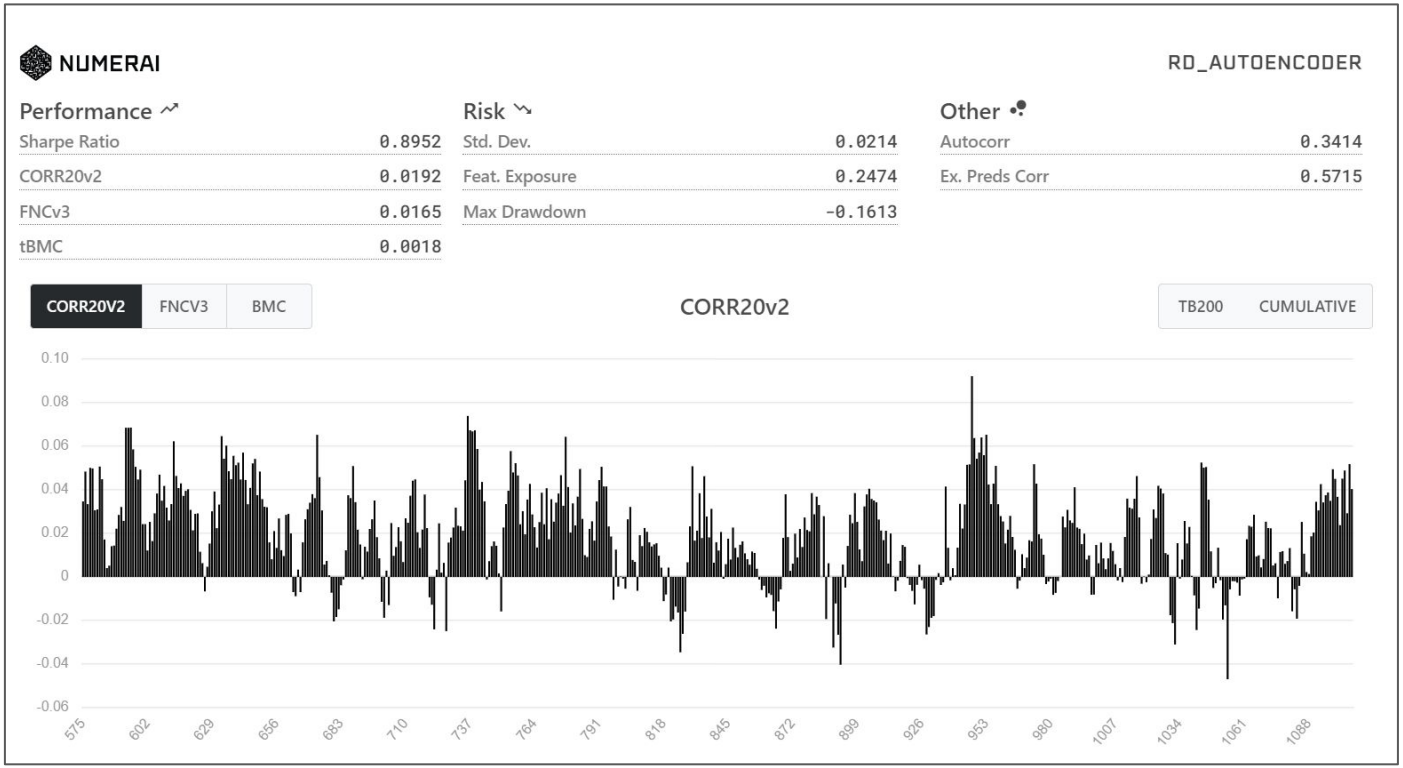


Trends Reverse

Ensemble w/ Autoencoder & LGBM is BEST

CNN has Negative correlation

True Performance - NumerAI Diagnostic for Autoencoder (BEST MODEL)



- Autoencoder Model has highest Correlation (~0.02 v.s. 0.08 Benchmark)

- Contribution is unique (BMC > 0)

- Model has consistently positive correlation over time

Challenges & Future Work

Challenges

- Accessibility: Lack of of pre-trained models
- Unconventional Data: Format was unique to this competition
- Costly Experiments: ~1 hour per hyperparameter combination

Future Work:

- Better Loss Function - A Correlation-based function may improve training
- Different Models - Lots of unique architectures to experiment with (CNN + LGBM)
- Longer Training - More epochs and deeper layers

Thank You!

NUMERAI