Deep Learning Project Milestone 2 Spring 2022: Stock Forecasting through the Numerai Challenge

Rohit Barve, Quay Dragon, Jackson Neal
Khoury College of Computer Sciences
Northeastern University, Boston, MA
`barve.r@northeastern.edu, dragon.r@northeastern.edu, neal.ja@northeastern.edu`

1 Overview

Stock price prediction poses a complex and potentially highly rewarding problem for data scientists. Equity prices are affected by a wide range of global factors. Relevant factors and analysis of their signals are the subject of intense scrutiny for players in the financial industry. Despite many proven best practices, the problem of reliable price prediction remains open. The promise of lucrative payout continues to prompt new approaches to the problem. In recent years, quant hedge funds have began to apply machine learning as a mechanism to guide their investment strategies.

Numerai is a quant hedge fund supported by thousands of data scientists that has shown promising performance over other hedge funds. In order to build a model to inform their trade decisions, Numerai hosts a continuous, open data science competition where data scientists can submit predictions. Participants can stake Numeraire (Numerai's cryptocurrency) on their predictions and receive payouts if their model performs well. Numerai then aggregates staked models into their own meta-model to power the hedge fund's performance. The competition has paid out over $\$62$ million to participants with competitors having an average return rate of 5.82% over three months [1].

The complexity of stock market prediction and the potential reward for success make participating in the Numerai challenge an attractive prospect. For this project, we develop models from the Numerai provided training and validation sets. Our models are then used to submit predictions on live tournament data and evaluated over a four week period. Submissions are measured on their target correlation, Meta Model Contributions, and Feature Neutral Correlation. Payouts and leaderboard positions are determined by an aggregate of the aforementioned metrics. Our aim is to produce a model that can successfully compete and perform as well as the top participants. In this report, we employ Deep Neural Network and Autoencoder with Multi-layer Perceptron architectures to compete in the Numerai data science competition.

2 Literature Survey

In recent years, innovative deep learning techniques have been successfully applied to sequential data forecasting. "A Survey of Forex and Stock Price Prediction Using Deep Learning" provides a summary of nearly 90 papers published since 2015 related to time series predictions of Forex price movements [5]. The papers surveyed span a wide range of models including Convolutional Neural Network (CNN), Long Short Term Memory (LSTM), Deep Neural Network (DNN), Recurrent Neural Network (RNN), and others. The paper also covers common performance metrics for these models, including Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), accuracy, and return rate among others.

RNN models have achieved improved time series predictions due to their ability to feed forward essential data. However, RNN architectures are often the victim of vanishing or exploding gradients and incur an arduous training process [8].

LSTM models are an extension of traditional RNN's and maintain longer term memory. LSTM's have consistently demonstrated best performance when predicting from time series data [5]. LSTM models resolve the gradient issues of RNN models but still experience a long training process [7].

An Autoencoder (AE) architecture has been demonstrated as a promising method of reducing dimensionality for stock market prediction models [3]. Stock market predictors often have large feature counts due to the extremely wide set of influences. An AE can be employed similarly to Principal Component Analysis (PCA) to reduce and engineer features [4]. AE models have been shown to improve accuracy on stock market predictions with temporal data [4].

A CNN can be applied to handle noisy stock market data and identify the most important features. Research has shown that coupling a CNN with an AE architecture results in better predictions than either model achieves alone [11] [12] [9]. [12] achieves promising results by employing a CNN with AE model paired with a Support Vector Machine (SVM) output layer.

"Attention is All you Need" introduces the transformer model as an efficient and efficacious alternative to recurrent and convolution approaches [10]. Transformer models have been applied to the problem of stock price prediction, receiving comparable results to RNN's through an attention mechanism that deciphers relations within a sequence [6].

3 Methods

All models are implentend in Python and can be found in our repository [1]. Due to the size of our dataset, creating a streamlined data processing and model training pipeline has been critical to our project success. All data processing, feature selection, PCA, and model hyperparameters are highly configurable. The training pipeline is accessible through a CLI with extensive logging abilities.

3.1 Feature Sampling and PCA

A common approach in the Numerai community shown to greatly improve accuracy is to reduce feature dimensionality through sampling. To choose feature subsets, we looked at two factors: target correlation and target volatility. Target correlation uses the Spearman correlation metric (explored below). We create feature sets of the top 250, 500, and 750 target correlated features. Target volatility is calculated as the standard deviation of feature correlation across eras. We create feature sets of the 100, 250, 500, and 750 most and least target volatile features. Any of these feature sets may be chosen as a hyperparameter when training a model.

We have also implemented the ability to perform PCA. PCA serves as an alternate method of feature dimensionality reduction, creating orthogonal features that account for maximum variance in the dataset. PCA may optionally be employed on the full features or any sampled set.

3.2 Base Model

RNN and LSTM architectures appear as an obvious choice for baseline models. However, these architectures cannot be applied to the Numerai dataset due to the non-sequential structure of the validation and test sets (illuminated in subsequent sections). Due to this, we chose to implement a simple DNN as our baseline model (BASE). The BASE model is composed of repeated sequences of fully connected, ReLU, batch normalization, and dropout layers. The model uses a Mean Squared Error (MSE) loss function. The baseline model does not need to be overly complex; many competitors see success with simple models and well chosen and engineered features.

3.3 Autoencoder with Multi-layer Perceptron

We implemented an Autoencoder with Multi-layer Perceptron (AE-MLP) architecture that has shown promise in an alternate stock market competition [13]. Figure 1 illustrates the AE-MLP architecture composed of a Gaussian noise layer, encoder and decoder layers, a fully connected output layer, and a multi-layer perceptron (MLP). The Gaussian noise layer adds noise sampled from a configured distribution to input values in order to help prevent overfitting and improve Feature Neutral Correlation (FNC), one of the main Numerai scoring metrics. The encoder is composed of fully connected, SiLU,

[1]`https://github.com/jacksonneal/mark_twain`

batch normalization, and dropout layers that reduce dimensionality. The decoder is composed of a single fully connected layer that then increases dimensionality to match that of the input. The decoder output is compared to the original input in $MSELoss_2$. The decoder output is passed to a single fully connected layer that attempts to predict target values, with loss calculated in $MSELoss_1$. The output of the encoder is also passed to an MLP module along with the original input. The MLP module has a similar architecture to the BASE model and its output is compared against target values in $MSELoss_3$. The total loss is an aggregate sum of the three loss calculations. Target prediction takes priority since two of the losses compare output to target values. In summary, the AE-MLP attempts to find a valuable intermediate representation of reduced dimensionality that can be mapped to the original input and enhance target prediction.

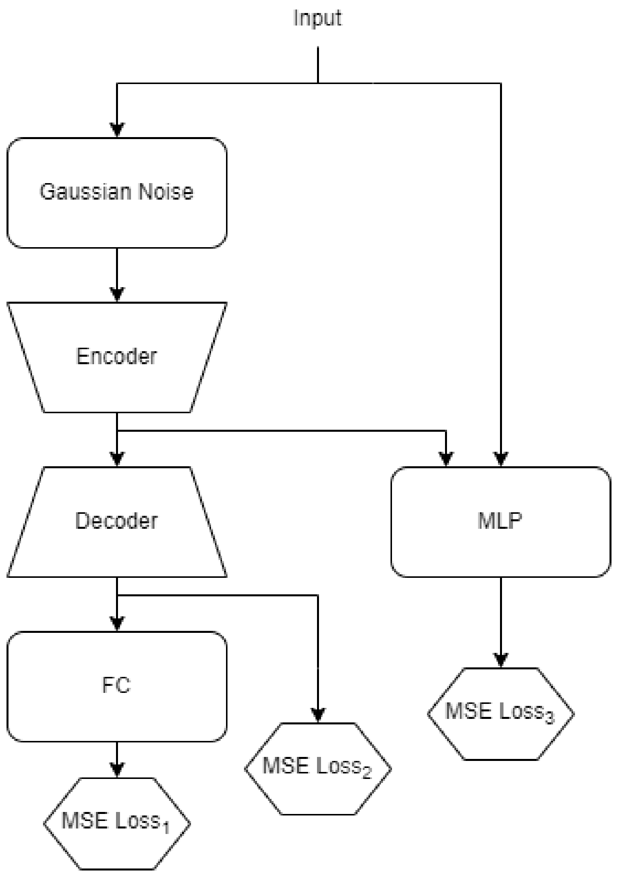

Figure 1: AE-MLP Architecture

3.4 Convolutional AE-MLP

Convolutional layers were most sucessful when coupled with the AE-MLP architecture. In replacement of linear layers, a convolutional layer, maximum pooling layer, batch normalization, and SiLU activation function were used in the encoder. A balance between kernel size and the dimensions of the linear decoding had the largest impact on the correlation score for the validation set. Relatively larger kernel sizes will perform better in this architecture due to the singular convolutional layer in the encoder. Compared to earlier CNN models, the increase in accuracy through incorporating the MLP structure and combination loss function substantiates that this structure greatly improves regular autoencoder models.

3.5 Transformer AE-MLP

A transformer architecture appeared as a potential model for our problem due to the temporal inconsistency in our data and the order-independent attention mechanism of the transformer. We

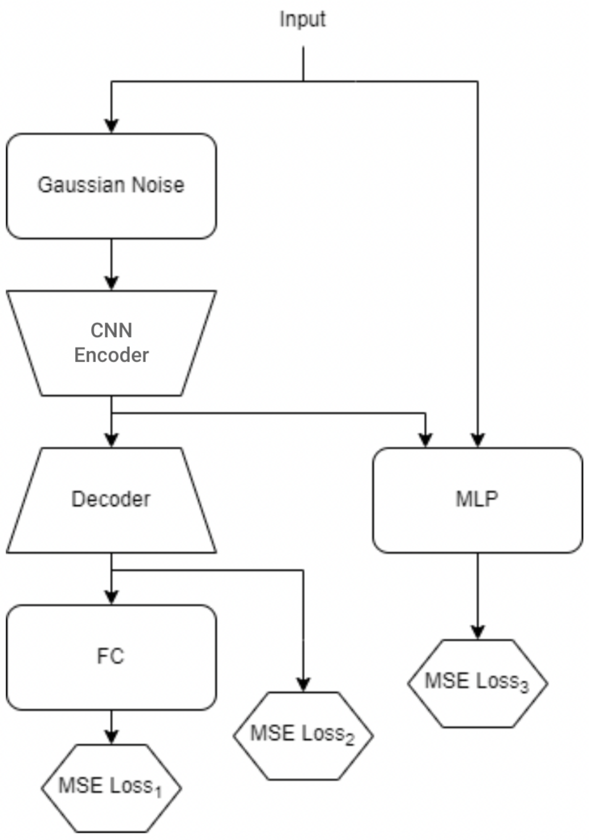

Figure 2: CNN-AE-MLP Architecture

explored a transformer variation of the AE-MLP model (T-AE-MLP) with a positional encoding layer and transformer encoder and decoder layers. The architecture is shown in Figure 3. To format our features, the input is encoded as a sequence where each column value becomes a token with an embedding size of one.

4 Experiments

4.1 Data and Metrics

4.1.1 Numerai Datasets

Training, validation, and test datasets are provided by Numerai. The training dataset contains $2,412,105$ rows. The validation dataset contains $539,658$ rows. The test dataset contains $1,412,932$ rows. Each record has $1,050$ features and 20 targets. All data and target values are obfuscated and normalized. The feature values are treated as abstract stock market data with no way to identify the true feature meaning. The 20 target values are paired into 10 abstract metric groups, corresponding to the metric readings 10 and 20 days into the future.

The training and validation sets are periodically updated. The test dataset is updated weekly. The test dataset is split into tournament and live data. Live data composes $5,300$ rows in the test dataset. Live data for the current week is moved to the tournament data group at the end of a week. The entire test dataset does not contain target values. Competitors submit predictions for target values of the test dataset. Predictions on the live records are used to evaluate models.

Competitors also have access to legacy datasets. These datasets may be useful for model evaluation during training.

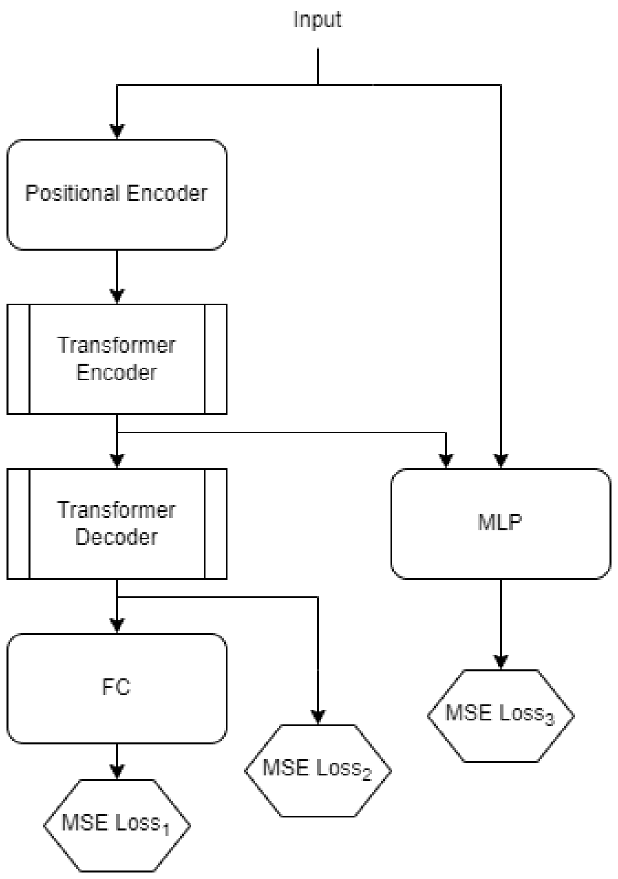

Figure 3: T-AE-MLP Architecture

4.1.2 Dataset Structure

Features in the training dataset yield target value correlation between 0.3 and 0.9. Features may be extremely highly or negatively correlated with each other.

Each record has an era column value. An era represents a one week period. The training dataset is composed of 574 sequential eras. Each era has $4-5$ thousand entries, with earlier eras generally being smaller. The era structure in the validation and test datasets is non-linear and not guaranteed. Due to this, the era value should not be used in predictive models. Numerai may change this in the future.

Feature values are normalized and have similar distributions, even across eras. Targets can take values $0.0, 0.25, 0.5, 0.75, 1.0$ and are roughly proportionally distributed $0.05, 0.2, 0.5, 0.2, 0.05$ respectively.

4.1.3 Accessing Datasets

Datasets are accessible through the `numerapi` Python package. The dataset files are available in CSV and IO optimized Parquet format. The datasets are also given in both `float32` and `int8` format. Due to the size of the dataset, we utilize the Parquet and `int8` versions.

4.2 Evaluation Metrics

As mentioned above, participants submit target values on the test dataset to receive scores for their models. One target is identified as the primary target and used for prediction evaluation. Primary target predictions for the live records in the test dataset are used to score submissions. After submission, predictions are evaluated over a four-week period against live results. Concretely, Numerai evaluates submissions through a combination of target correlation, Meta Model Contribution, and Feature Neutral Correlation measurements.

4.2.1 Target Correlation

Target correlation is computed by ranking output signals between $[0, 1]$, neutralizing them, and taking the Spearman correlation

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{1}$$

between the neutralized signal and the primary target, where d_i is the difference between the two ranks predictions and target values and n is the number of observations. Listing 1 shows how this can be computed in Python using `numpy` [1]. Target correlation is only calculated for the primary target, however, training models that can successfully predict the other auxiliary target values has been shown to facilitate success when predicting on live tournament data.

Listing 1: Spearman correlation using `numpy`

```
ranked_predictions = predictions.rank(pct=True, method="first")
correlation = np.corrcoef(labels, ranked_predictions)[0, 1]
```

4.2.2 Meta Model Contribution

Meta Model Contribution (MMC) is a custom metric designed by Numerai to promote original models by evaluating prediction correlation to a target "that is neutralized to all signals known to Numerai" [1]. The full algorithm can be found in the Numerai documentation [1]. At a high level, we should pursue novel models to score well on this metric. For example, we should avoid random forest models that, although perform decently, are popular in the Numerai community.

4.2.3 Feature Neutral Correlation

Feature Neutral Correlation (FNC) discourages models from heavy reliance on a small set of features. Algorithm 1 lays out the procedure for calculating FNC and the full details can be found in the Numerai documentation [1]. We should prevent our models from becoming reliant on few features to score high on this metric.

- Normalize submission predictions
- Neutralize submission predictions to Numerai features
- Calculate Spearman correlation of neutralized submission predictions to target

Algorithm 1: FNC Procedure

4.3 Experiment Results

We have ran several experiments using our BASE, AE-MLP, CNN-AE-MLP, and T-AE-MLP models with a variety of hyperparameter configurations. In order to efficiently explore the hyperparameter search space we use the Weights and Biases platform [2]. This platform allows us to provide hyperparameter values and ranges through a config file [2] and perform a "sweep" over the hyperparameter search space. Table 1 displays the hyperparameters and ranges we evaluated during our experimentation. Each model was also tested with the various feature sampling techniques we describe above in section 3.1. PCA values indicate the percentage of variance accounted for by engineered components. Since eras are overlapping, training on every 4th era is employed to keep records independent. As we trained more complex models, we narrowed our hyperparameter search based on observed results.

For the base model, we experimented with many different dimensions for the fully connected layers, settling on $[1920, 1920, 1920, 1024]$ as the optimal configuration and using this for the MLP

[2] `https://github.com/jacksonneal/mark_twain/blob/main/numerai/config/sweep.yaml`

Hyperparameter	BASE	AE-MLP	CNN-AE-MLP	T-AE-MLP
batch size	2-5k	5k	5k	500
dropout	0.0-0.3	0.15-0.25	0.18	0.15-0.25
initial batch norm.	true, false	true	true	true
epochs	10-25	8-10	8-10	8-10
learning rate	0.001-0.003	0.001-0.003	0.001-0.003	0.002-0.003
weight decay	0.01-0.1	0.05-0.1	0.05-0.1	0.05-1.0
pca	0.85-0.99	-	-	-
sample 4th era	true, false	true	true	true
encoded dim.	-	256, 512, 1024	1000	-
kernel	-	-	50-500	-
pool kernel	-		7-15	-
stride	-	-	1-5	-

Table 1: Experiment settings

dimensions in the other models. Full optimal run configurations are shown in Table 2. Training on the top 750 volatile feature set consistently performed best, confirming the Numerai community sentiment that feature selection is essential in this competition. Large batch sizes performed well, but a much smaller batch size was required to train the T-AE-MLP due to the size of the model and GPU memory constraints. PCA consistently performed poorly and was not used in optimal runs. The AE-MLP functioned best with an encoder output dimension of 512. We also experimented with the use of auxiliary targets, finding that models which accurately predict both the primary and a single auxiliary target perform best. Initial implementations of convolutional autoencoders proved that complex models performed poorly on the Numerai dataset. The robust CNN performed the most optimally when kernel sizes did not reduce the dimensionality of the feature set, thus mimicking a linear layers which use linear transformations for this reduction. This autoencoder was able to perform with accuracy near that of the BASE due to its similarity of its linear layers. As stated in the CNN-AE-MLP architecture in section 3.4, CNNs reached optimal performance when integrated into the AE-MLP structure with minor changes in the encoding layer.

Metrics for the optimal run configurations in Table 2 are shown in Table 3. We see the AE-MLP model perform best on the primary metric: validation Spearman correlation. Table 3 indicates a level of overfitting in the CNN-AE-MLP and T-AE-MLP models as they perform best on the training metrics but perform worse than even the simple BASE model on the validation metrics. Figures 4 and 5 demonstrate the progression of training and validation Spearman correlation scores during training. Although not immediately the highest scorer, we see the AE-MLP model able to continue to increase correlation while the other models stagnate, ultimately performing the best. The T-AE-MLP model is not shown as its step size would not be to scale due to a much different batch size

Hyperparameter	BASE	AE-MLP	CNN-AE-MLP	T-AE-MLP
feat. set	top 750 vol.	top 750 vol.	top 750 vol.	top 750 vol.
batch size	5k	5k	5k	500
dropout	0.1968	0.1955	0.1845	0.1874
initial batch norm.	true	true	true	true
epochs	10	9	8	10
learning rate	0.002644	0.002604	0.002125	0.002181
weight decay	0.09582	0.07866	0.07172	0.06573
pca	-	-	-	-
sample 4th era	true	true	true	true
encoded dim.	-	512	1000	-
kernel	-	-	415	-
pool kernel	-	-	15	-
stride	-	-	1	-

Table 2: Optimal settings

Metric	BASE	AE-MLP	CNN-AE-MLP	T-AE-MLP
Train Corr.	0.08634	0.0764	0.04898	**0.08978**
Train Sharpe	2.041	2.605	**2.881**	1.46
Train Spearman	0.06023	0.07101	0.06336	**0.07743**
Val. Corr.	0.026	**0.02783**	0.02204	0.01778
Val. Sharpe	0.8117	0.877	**0.9546**	0.3308
Val. Spearman	0.026	**0.02784**	0.02204	0.01779

Table 3: Optimal run results

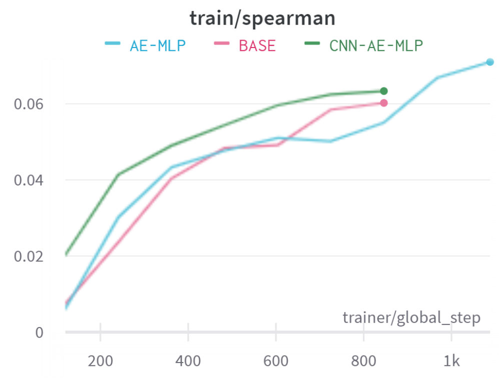

Figure 4: Training Spearman Corr.

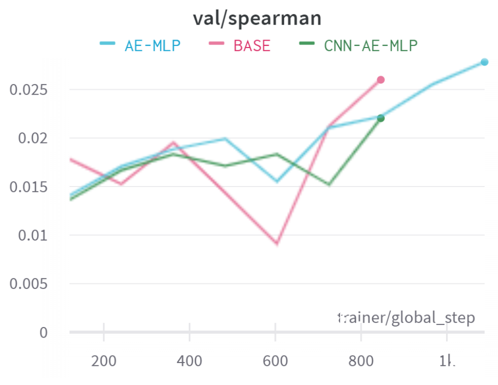

Figure 5: Validation Spearman Corr.

4.3.1 Numerai Submissions

We have submitted predictions for all models to the Numerai competition. We have only received scores for the BASE and AE-MLP models at this time. The BASE model has a live correlation score of 0.027, placing in the 28th percentile. The AE-MLP model has a live correlation score of 0.0362, placing in the 39th percentile. These are only our initial scores, and may differ greatly from our final ranking at the end of the round in four weeks.

5 Conclusion

In this project, we have successfully implemented several models to compete in the Numerai data science competition. We have implemented a configurable data processing and model training pipeline to facilitate rapid model development and experimentation. We successfully implemented an Autoencoder MLP architecture that has proven success in stock price prediction [13]. We also implemented relevant techniques from literature, employing convolution and transformers in our encoder strategy. We constructed novel feature sampling strategies to generate the most volatile features and achieve improved results. We were unable to achieve improved results from our more complex models. Future work should continue to tune and refine these more advanced models, while also considering the boosted ensemble methods that are popular in the Numerai community.

References

[1] Numerai tournament docs - numerai tournament. URL `https://docs.numer.ai/`.

[2] Weights and biases - documentation. URL `https://docs.wandb.ai/`.

[3] Shihao Gu, Bryan T Kelly, and Dacheng Xiu. Autoencoder asset pricing models. 2019.

[4] Hakan Gunduz. An efficient stock market prediction model using hybrid feature reduction method based on variational autoencoders and recursive feature elimination. *Financial Innovation*, 7(1):1–24, 2021.

[5] Zexin Hu, Yiqi Zhao, and Matloob Khushi. A survey of forex and stock price prediction using deep learning. *Applied System Innovation*, 4(1):9, 2021.

[6] Bryan Lim, Sercan O Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *arXiv preprint arXiv:1912.09363*, 2019.

[7] Qihang Ma. Comparison of arima, ann and lstm for stock price prediction. In *E3S Web of Conferences*, volume 218, page 01026. EDP Sciences, 2020.

[8] Abdel-Nasser Sharkawy. Principle of neural network and its main types. *Journal of Advances in Applied & Computational Mathematics*, 7:8–19, 2020.

[9] Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. Why self-attention? a targeted evaluation of neural machine translation architectures. *arXiv preprint arXiv:1808.08946*, 2018.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[11] Li Xie and Sheng Yu. Unsupervised feature extraction with convolutional autoencoder with application to daily stock market prediction. *Concurrency and Computation: Practice and Experience*, 33(16):e6282, 2021.

[12] Li Xie, Lin Liu, and Sheng Yu. A novel convolutional auto-encoder networks for daily stock market prediction. In *2020 7th International Conference on Information Science and Control Engineering (ICISCE)*, pages 96–100. IEEE, 2020.

[13] Yirun Zhang. Jane street: Supervised autoencoder mlp, Sep 2021. URL `https://www.kaggle.com/code/gogo827jz/jane-street-supervised-autoencoder-mlp/notebook`.