Computer Science

Capstone Report - Fall 2023

Predicting Stock Price Using Candlestick Chart

Tianyu Liu,
Peiyang Wu

supervised by
Xianbin Gu & Promethee Spathis

Preface

Tianyu Liu earned a bachelor's degree in Computer Science and minored in Finance from NYU Shanghai. Peiyang Wu earned a bachelor's degree in Computer Science and Data Science from the same institution. Our research endeavors to ease the challenges associated with leveraging computer-based expertise for investment purposes, empower investors to make sound decisions and bolster overall profitability. Our target audience includes people of all ages who possess smart devices, such as cell phones, computers, and others, which can upload pictures and gain access to them easily. It could be investors, company decision-makers, policymakers, and the general public. Users will be capable of inputting the past 20 days of price fluctuations and receiving a prediction for the following 5 days of price movements. There are few studies examining the use of images as a data input for stock price forecasting, as most studies rely on raw data. However, people tend to prefer images in everyday life because they provide more information and are easier to use. If you're interested in using this as a useful investment tool, we recommend reading this report for further insights.

Acknowledgements

We would like to thank Professor Promethee Spathis who all have provided support for this capstone project. We would like to express our deepest gratitude to Professor Xianbin Gu for his invaluable guidance and support throughout this research. His expertise and insight have been pivotal in shaping both the direction and the substance of this study. We also extend our thanks to our parents and friends who have supported us throughout our research.

Abstract

This study explores the application of the Variational Encoder-Decoder (VED) for stock price forecasting, an innovative approach in the field of financial market analysis. Focusing on the SSE 50 Index, which consists of the largest stocks on the Shanghai Stock Exchange, the study examines the effectiveness of VED in predicting complex stock price fluctuations. By using images as input for prediction, the goal is to integrate advanced machine learning techniques with existing financial theories, thereby lowering the technical barriers to machine learning's auxiliary role in investment and contributing to the fields of financial analysis and investment management.

Keywords

Capstone; Computer science; NYU Shanghai; Stock Price Prediction, Candlestick Charts, Machine Learning, Convolutional Neural Networks, Variational Auto-Encoders(VAE), Transformers, SSE 50 Index, Financial Analysis, Market Prediction.

Contents

1 Introduction **5**

2 Related Work **6**

2.1 Stock Prediction 6

2.2 Grapgical Generation 6

2.3 CNN and GAN Based Model 7

2.4 VAE and Transformer Based Model 8

3 Solution **8**

3.1 Dataset 8

3.2 Task formulation 9

3.3 Model 10

3.4 Training configuration 11

4 Results and Discussion **13**

4.1 Experiments 13

4.2 Main results 16

4.3 Eval Loss Puzzle 17

5 Discussion **18**

5.1 Main Chanllenges 18

5.2 Limitation 20

6 Personal Contributions **22**

7 Conclusion **22**

1 Introduction

Stock price prediction is significant in finance and economics, as it enables investors and companies to assess and manage risks efficiently amid market fluctuations. This financial analysis aspect is crucial in navigating the volatile and unpredictable nature of stock markets. Robust predictions foster investor confidence, which is essential for maintaining a healthy level of participation in the stock market. Enhanced investor engagement not only enhances individual portfolios but also positively contributes to the overall health and stability of the economy. Additionally, the endeavor to achieve precise stock price predictions facilitates advancement in technology and methodology, especially in the areas of machine learning and data analytics. This collaboration between finance and technology not only improves financial analysis tools but also contributes to wider innovations that can reach beyond industries. Stock price prediction, therefore, has a significance that goes beyond immediate financial gains, impacting overall economic stability and technological advancement.

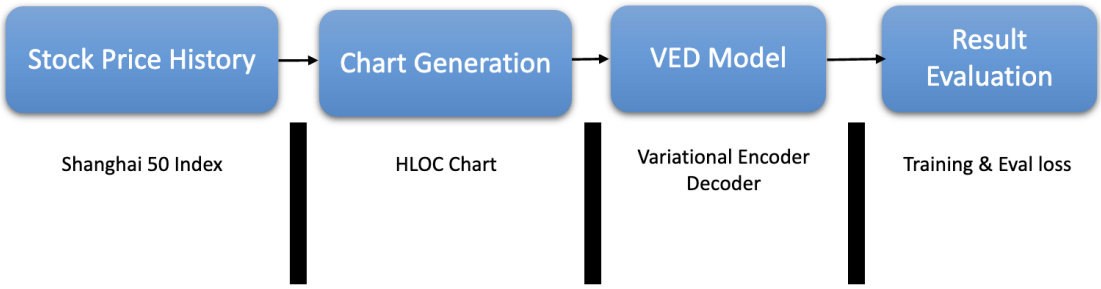

Figure 1: Overview of the training procedure

- **We proved the feasibility of utilizing the Variational Encoder Decoder to predict stock price with candlestick charts.** Especially, we employed the SSE 50 Index to train our model, which is a stock index of the Shanghai Stock Exchange that represents the top 50 companies by "float-adjusted" capitalization and other criteria.
- **We systematically tested the performance of VED across different hyper-parameter settings.** During the procedure, we compare the difference in outcome by training model with different structures under the same training set.
- **We analyze the result and provide possible explanations for the model performance.** Specifically, we analyze the difference when applying different loss functions and latent space.

Be think our contribution could inspire a novel way of price prediction using candlestick Charts, as normally researchers work with price singly. Our approach allows the implementation of more powerful modern models, such as Variational Auto Encoder(VAE), and Transformers.

2 Related Work

2.1 Stock Prediction

Initially, stock price prediction was governed by statistical models, particularly AutoRegressive (AR) models, adept at analyzing financial time series data. A prime example is the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model, renowned for its predictive accuracy in volatile markets [1]. The AutoRegressive Integrated Moving Averages (ARIMA) model also demonstrated effectiveness in short-term forecasting [2], but its reliance on stationary data limited its real-world applicability. Adebiyi et al. [3] found that Artificial Neural Network (ANN) models were more effective with non-stationary data. Furthermore, the Fuzzy Random Auto-Regression (FR-AR) model [4] incorporated fuzzy logic to better adapt to varying market data, though it faced limitations in predicting long-term trends.

2.2 Grapgical Generation

The article bears the idea of trend-based predictability using methods that flexibly learn price patterns that are most predictive of future returns, rather than testing hypothesized or pre-specified patterns. The article also provides raw predictor data are images—stock-level price charts—from which elicit the price patterns that best predict returns using machine learning image analysis methods. Through using different methodologies for data prediction, chart-image-generating is reasonably referred to as the chart generation method [5].

Previous research indicates that RGBStick representation has great potential to integrate human decision processes and deep learning for stock market analysis and forecasting [6]. Another article evaluates the performances of CNN and LSTM for recognizing common chart patterns in stock historical data. It presents two common patterns, the method used to build the training set, the neural network architectures, and the accuracies obtained. The article also contains graphing suggestions that transfer stock price data to an RGB graph. Though the prediction model we applied is different, we consider the graphing suggestion as an alternative chart-creating method[7].

2.3 CNN and GAN Based Model

CNN and GAN models are basic machine learning models, that were previously used for stock price prediction.

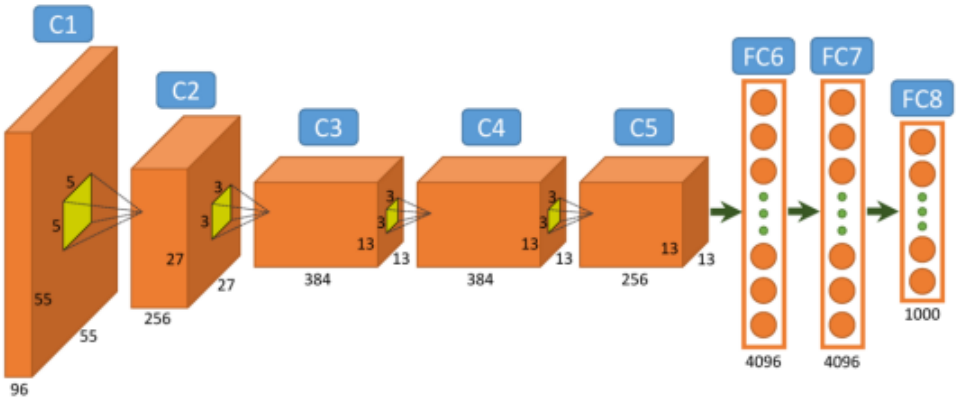

Figure 2: CNN Architecture

Previous study illustrates the possibility and function of candlesticks automatically recognizing function[8]. The passage highlights its capabilities and proposes a two-step approach to recognize candlestick patterns automatically. As the article mentions, the first step uses the Gramian Angular Field (GAF) to encode the time series as different types of images and then uses the Convolutional Neural Network (CNN) with the GAF images to learn eight critical kinds of candlestick patterns, whose result is promising. The recognizing method can be implemented to recognize and load data of the generated graph[9].

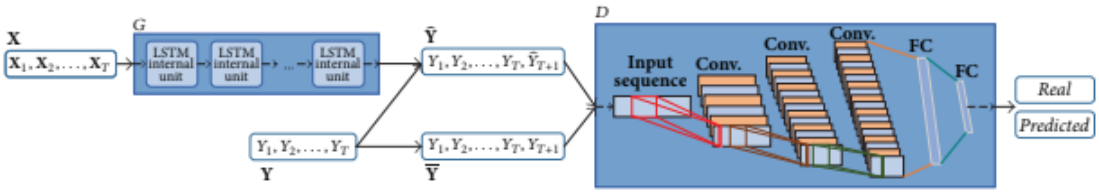

Figure 3: GNN Architecture

A generic framework employing Long Short-Term Memory (LSTM) and convolutional neural network (CNN) for adversarial training to forecast high-frequency stock markets has been analyzed, which takes the publicly available index provided by trading software as input to avoid complex financial theory research and difficult technical analysis, which provides convenience for the ordinary trader of nonfinancial specialty[10].

2.4 VAE and Transformer Based Model

Variational Auto-Encoder (VAE) and Transformer are two base models we utilize for Candlestick prediction.

VAE[11] introduces a novel architecture called TimeVAE for generating synthetic time-series data using Variational Auto-Encoders (VAEs). The results demonstrate that TimeVAE accurately represents temporal attributes, performs well on next-step prediction tasks, and incorporates domain-specific time patterns for interpretable outputs.

The Generative Adversarial Networks for synthetic data generation in the time-series domain also yield promising outcomes. Such synthetic architecture shows interpretability, the ability to encode domain knowledge, and reduced training times, and results on similarity tests show that the VAE approach is able to accurately represent the temporal of the original data attributes[12].

The utilization of Transformer-based models in time series forecasting has garnered attention and recognition in the academic community. A noteworthy essay from the Chinese University of Hong Kong, as referenced, underscores the effectiveness of Transformers in this domain. These models, originally designed for natural language processing tasks, have demonstrated their versatility in handling time series data for predictive purposes[13]. Additionally, research has shown that a transformer's reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks[14].

3 Solution

3.1 Dataset

We acquired historical data of all stocks currently trading in the A-share market of China through the Wind terminal. The duration of historical data extraction varies for each stock. We collected daily data for each stock starting from its first day of listing until one day prior to the data download. Includes the opening, closing, high, and low prices, as well as the average price and changes in price. It also includes information on turnover, total market capitalization, outstanding market capitalization for A-shares, total share capital, and outstanding share capital for A-shares.

Although we obtained a large dataset, we narrowed our dataset down to the 50 most representative stocks in the Shanghai stock market, which is the SSE 50 index. According to the Shanghai Stock Exchange's official website, the SSE 50 Index is composed of a sample of 50 scientifically and objectively chosen stocks that exhibit high liquidity and large scale in the Shanghai stock

market. The purpose of this selection is to comprehensively reflect the overall market situation of a group of leading companies with significant market influence in the Shanghai stock market.

3.2 Task formulation

When analyzing the data, we discovered that aside from halted trading on certain legal holidays in China and unusual circumstances resulting in consecutive days of stagnant stock prices or null values which require processing via the dropna() function in the pandas package, there are also instances where the stock price sharply declines. After conducting online research and consulting with the professor, it is probable that dividends or stock bonuses are the root cause of this situation. To remove the price distortion caused by ex-rights and ex-dividends and maintain stock price continuity, we have opted for the forward adjustment method to address this issue. Forward adjustment is determined by the current stock price while maintaining it unchanged. Then adjust the previous price and make adjustments to the K-line before the ex-dividend to align the charts. With this approach, we alleviate any anomalies in the data from affecting the stock price movements.

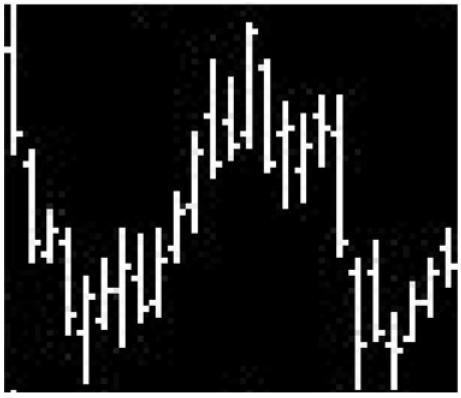

Figure 4: Chart Example

We utilized the HLOC chart-making method pioneered by [5]. This method involves representing each day's data as a rectangle, which is three pixels wide and has a black background. The center column of pixels represents the high price and low price, while white pixels connect the high price and low price. Furthermore, a white pixel square represents the opening price on the left column with a width of 1 pixel, while a white pixel square represents the closing price on the right column with a width of 1 pixel. This representation allows for the display of the four most crucial pieces of information regarding a stock throughout the day: the opening price,

closing price, high price, and low price. By adjusting the placement of these images based on the vertical price and linking them together, we can generate a visual representation of the stock's price fluctuations.

3.3 Model

We utilized the Variational Autoencoder Model (VAE), a type of neural network architecture utilized for unsupervised learning. VAE is capable of generating new data similar to the training data through its generative model. The architecture consists of two parts: an encoder and a decoder. The encoder accepts an input and maps it to a latent space, presenting a lower-dimensional representation of the input. The decoder takes this latent representation and maps it back to the original input space.

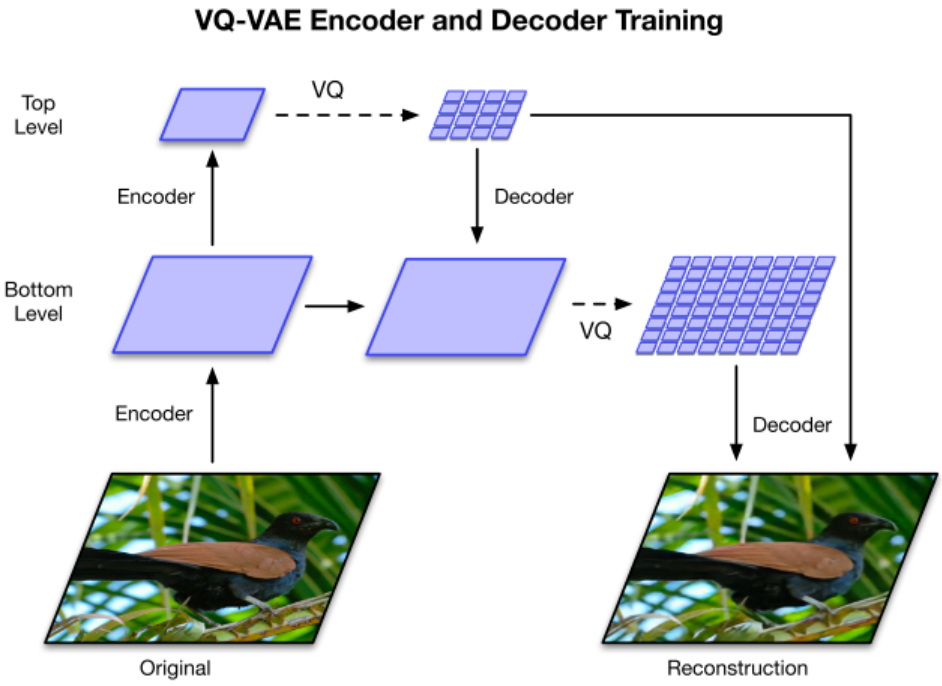

Figure 5: VQ-VAE model Architecture
[15]

However, predicting stock prices does not necessitate restoring the original image entirely. Instead, we maintain the original encoder component and add labels to transform the VAE into a supervised learning model, after mapping the latent space. This approach enables us to extract and predict crucial information. Given that we keep most of the VAE, we build a model called the Variational Encoder-Decoder (VED) model.

3.4 Training configuration

During our training process, we first implemented our chart generation function to obtain a candlestick chart. Then, the charts are transmitted to a three-dimensional matrix to train the neural network. Throughout the fine-tuning phase, we diligently save checkpoints at regular intervals, capturing the evolving state of the model.

Concurrently, the evaluation function evaluates the performance of every checkpoint, computing evaluation loss and saving corresponding generated images.

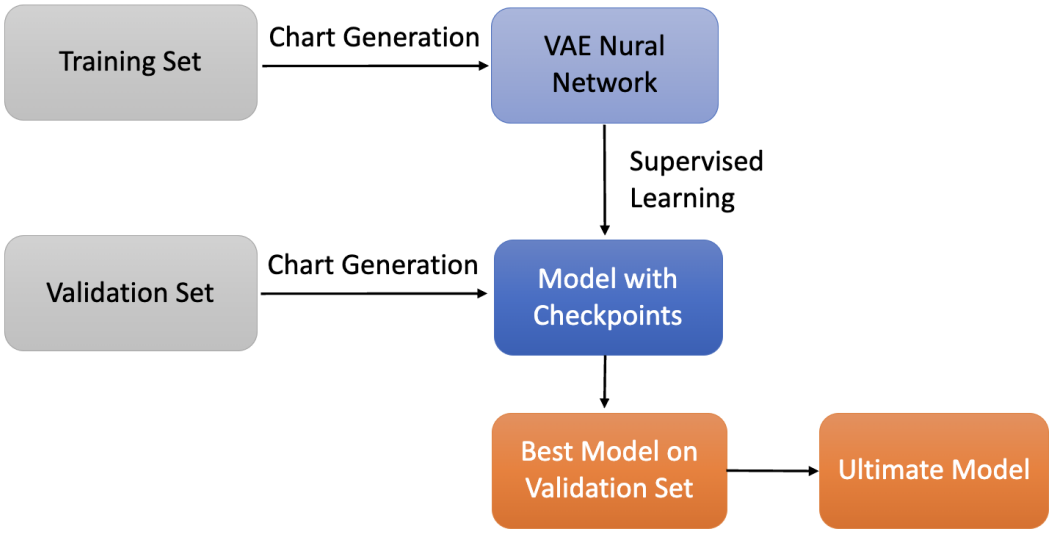

Figure 6: The process of model training. The candlestick charts are decoded to train the neural network.

In our setting, the model is fed with the stock price of 20 days, to generate the stock price for the next 5 days. The chart for the first 20 days is considered as inputs and the next 5 days are labels.

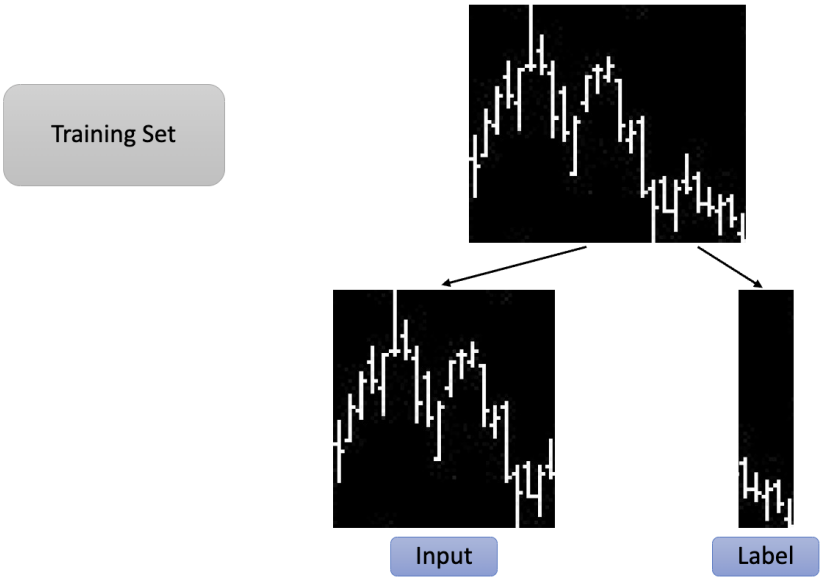

Figure 7: The process of generating Input Data and Labels. The first 20 days are considered as input and the next 50 days are considered as labels.

The result of the output will be trained based on the label.

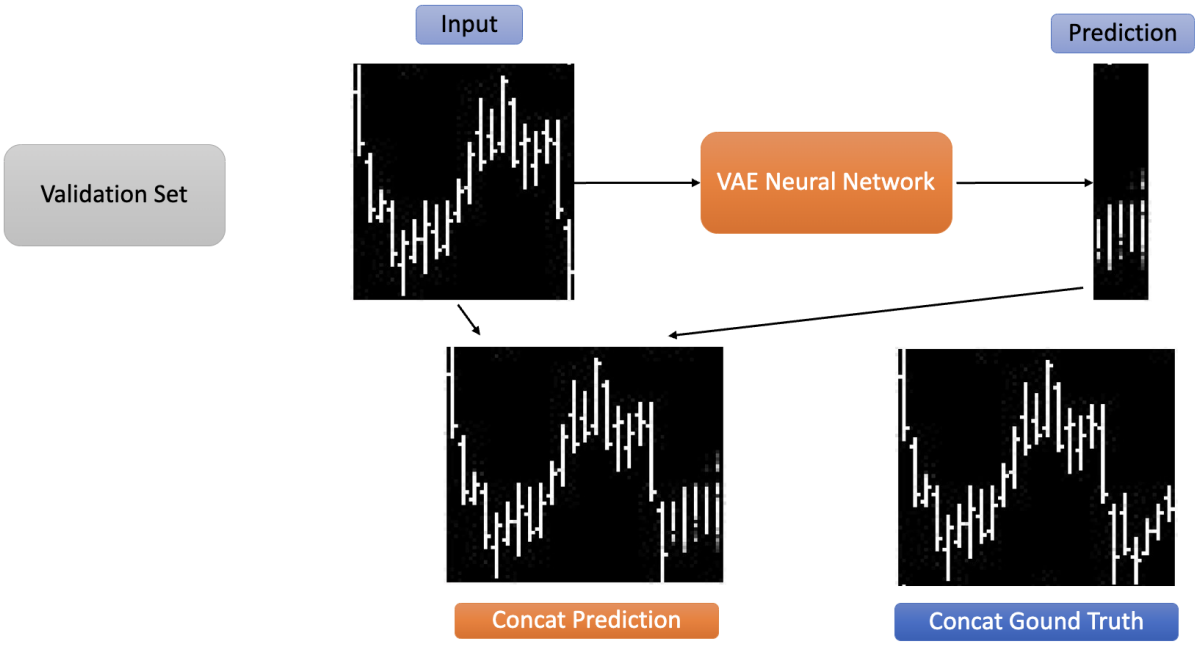

Figure 8: The process of generating output with the trained VAE neural networks. The result can be concatenated and compared with the input.

4 Results and Discussion

4.1 Experiments

In our experiments, we choose to build a convolutional VED. The original 3-dimensional matrices are encoded through 4 layers of Convolutional Neural Networks, then through 4 layers of linear (fully connected) neural, and decoded. It's important that we set the dimension of the fully connected layers after the chart is encoded. During our training process, this hyper-parameter is carefully changed to ensure the latent space has enough space to contain all the information while free from unexpected noise.

We evaluate our result through the default loss function build-in Pytorch. It's important to know that the default function does not necessarily present the exact need of our evaluation. For graph evaluation, the default function tends to score any differences between the predicted output to the ground truth, while predicting stock price we want the predicted candlestick line to overlap with the label. This calls necessity to choose the loss function.

4.1.1 Loss Function

We implement Binary Cross Entropy (BCE) loss. It is a criterion that measures the error of a classification task between the target and the input probabilities. The BCE loss is calculated as the negative average of the logarithm of the corrected predicted probabilities 2. By standard, BCE loss is especially effective when used for reconstruction in an auto-encoder.

$$\mathrm{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_{\mathrm{i}} log(\mathrm{p}_i) + (1 - y_{\mathrm{i}}) log(1 - \mathrm{p}_i)$$

Because in the candlestick a pixel is either black or non-black, symbolizing whether there is a meaningful output at that point, each pixel gets a binary outcome. BCE is just used to evaluate the performance of a machine learning model by measuring the difference between the predicted and actual binary labels 1, thus theoretically it's an ideal loss function for this target.

During the process, some team members came up with the point of implementing the Mean Squared Error (MSE) loss function. We realized the possible problem that the problem was not a regression problem that MSE is supposed to do, but we still trained a model based on the loss function. The result shows the model tends to turn all candlesticks in gray to minimize the Mean loss.

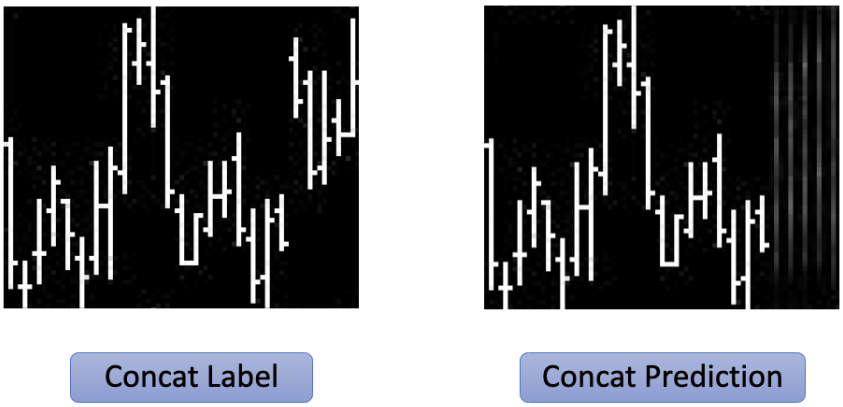

Figure 9: Utilizing MSE loss function will generate the image opposite to our expectation

Considering the MSE loss function

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - \text{y}_\text{i})$$

It assesses the average squared difference between the observed and predicted values and attempts to minimize it during the training process. It tends to generate the overall distribution of the likelihood that in one position a pixel could be black or non-black. It turns out to be a vague, gray candlestick that we don't prefer in this task.

4.1.2 Latent Space

The latent space in a VAE is a low-dimensional representation of the input data that is learned by the VAE. The latent space is a continuous vector space that can be used to generate new data points by sampling from it. The size of the latent space is a hyper-parameter that can be tuned to balance the trade-off between the model's capacity and its ability to generalize to new data. A smaller latent space may lead to better generalization, while a larger latent space may capture more details of the input data, which could be noise that barricades detailed generation.

Latent Space, in our model, points to the dimension of the fully connected neural network after the convolutional neural network Decoder. We notice a change in the dimension will result in a change in final outcome. A lower latent space will result in missing information, resulting in inconsistency. Excessive latent space will in contrast result in a hallucinated candlestick line.

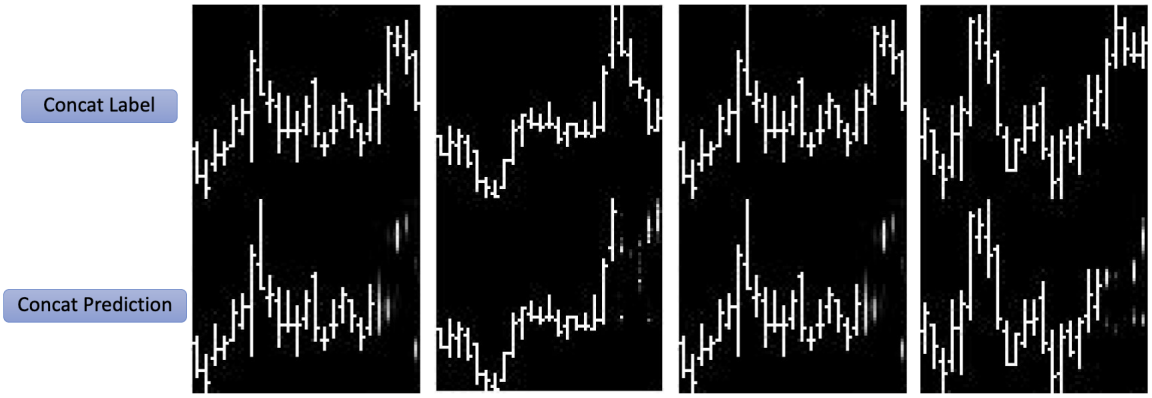

Figure 10: Setting latent space dimension = 16, the candle stick might be inconsistent

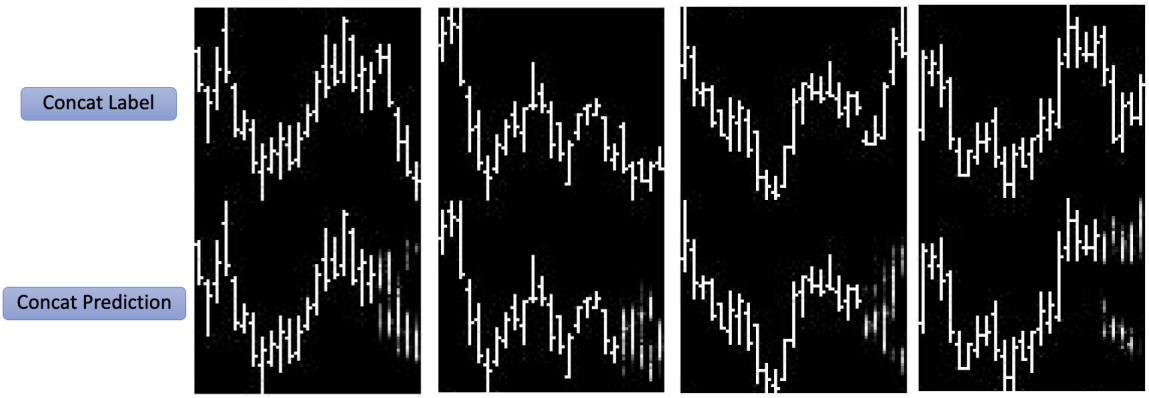

Figure 11: Setting latent space dimension = 128, the candle stick might hallucinate

4.2 Main results

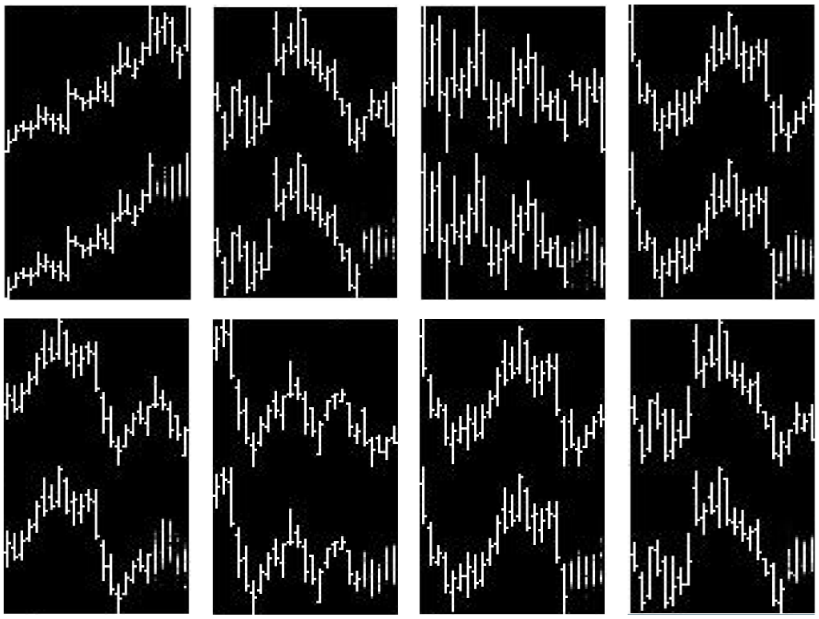

Figure 12: Our prediction results

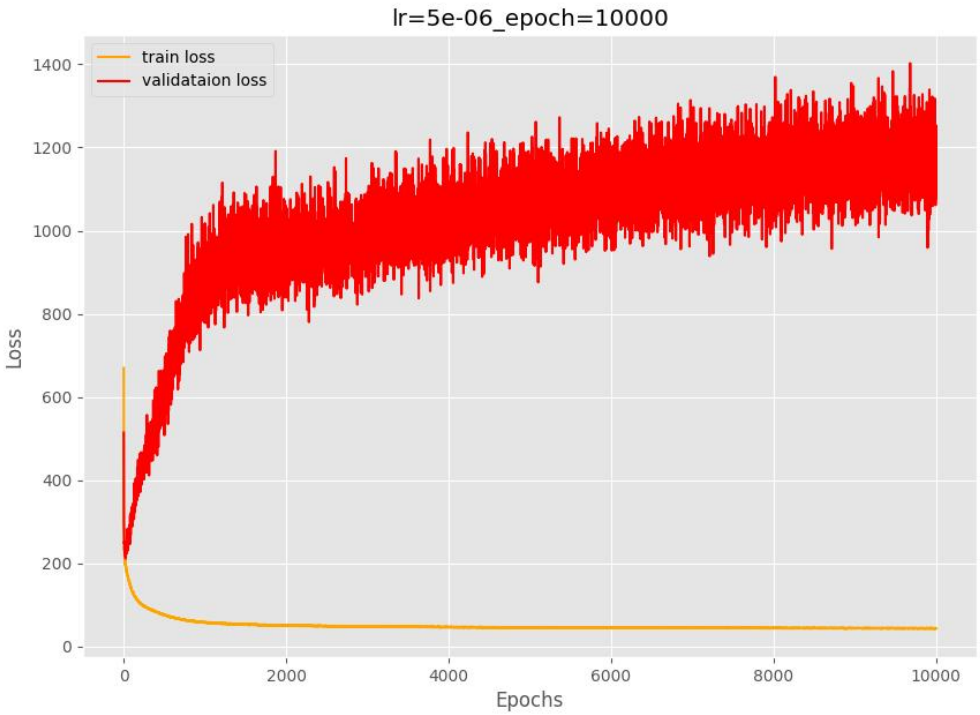

Figure 13: Our BCE Train Loss & Validation Loss

After completing the final parameterization, the parameters of our ultimate model are as follows:

1. lr = 0.000005
2. latent space = 64
3. epoch = 10000

Our model produces mostly clear predictions, as shown in Figure 12. However, there are still some unsatisfactory results, such as the unclear or absent depiction of the number of individual days. Nevertheless, these occurrences are a small minority and have been excluded from the results display.

Main discussion of the loss function please refer to section 4.1.1 4.3

4.3 Eval Loss Puzzle

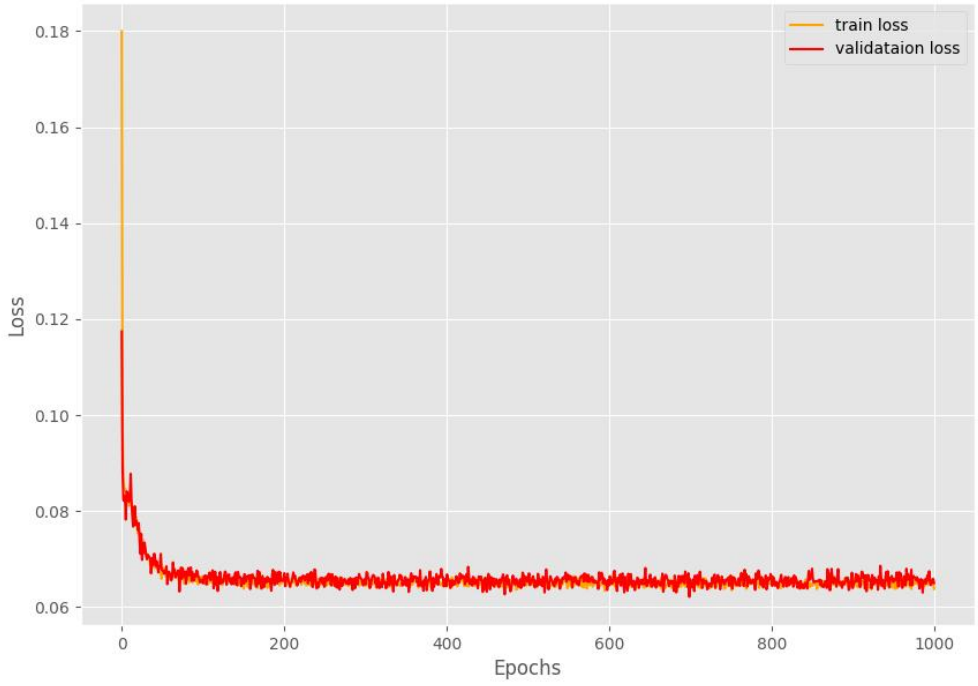

Figure 14: Our MSE Train Loss & Validation Loss

During the training process, we found that MSE and BCE loss functions produced different prediction results. When we chose the MSE loss function, the validation loss kept decreasing, but the generated prediction images were blurry and had long white vertical lines. On the other hand, when we chose the BCE loss function, the validation loss kept increasing, which was opposite to

the trend of the training loss. However, the generated prediction images met our expectations and gradually showed a trend. This phenomenon puzzled us because the train loss and validation loss of the machine learning models we have encountered so far have shown a relatively uniform trend when the results meet our expectations. A similar issue is also encountered by our supervisor, who is parallelly conducting the test with Transformers. We would consider of rewriting the eval loss function for future work.

5 Discussion

5.1 Main Chanllenges

5.1.1 Activation of CUDA Environment

Considering the amount of our training data and the complexity of the VED neural network, we note the importance of applying HPC to accelerate our training process. NYU HPC Greene provides clusters with high-performance GPU, so we tried to accelerate them with a CUDA environment.

The CUDA is a development environment that enables developers to create high-performance applications accelerated by GPUs. It offers a comprehensive collection of libraries, tools, and APIs for NVIDIA GPU-based application development. CUDA acceleration is supported on Greene, RTX8000, and A100.

Singularity is required for HPC, which is somewhat distinct from a personal desktop environment, as it is a container framework specifically developed for scientific applications on HPC resources. Docker containers can be run natively using Singularity as a replacement for Docker on HPC systems. On the Greene system, Singularity enables CUDA environment integration with HPC. We selected CUDA 12.1 to ensure compatibility with Pytorch 2.11 and the Transformers $flash\_attn\_2\_cuda$ package.

5.1.2 Storage Overflow on HPC

The training process for image generation in our model incurs considerable costs in terms of stock price, necessitating over 1600G of memory to generate and store all the images in a single instance. Such a requirement risks causing memory and storage overflows in the High-Performance Computing (HPC) system. To address this challenge, we adopted a strategy focusing on the SSH 50, a subset of stocks deemed most representative of the overall stock price trends. This selection

significantly reduces the memory and storage demands.

Additionally, we implemented a 'lazy generating' Dataloader. This approach involves a rewrite of the '__get__' method to ensure that it generates only a random chart on-demand, and then produces a new chart as required. This method is particularly effective in conserving memory because it avoids the need to preload and store all images at once. However, it's important to note that this technique might lead to the generation of repeated charts during the training process. Despite this, the overall reduction in memory usage is substantial, making this a viable solution for managing the high resource demands of our image generation training process.

5.1.3 Program Deployment

The NYU High-Performance Computing clusters run on the Slurm queue management system, an indispensable tool for distributed computing workloads. Slurm, short for Simple Linux Utility for Resource Management, allocates resources and manages job queues effectively. One distinguishing attribute of Slurm in the NYU HPC ecosystem is its ability to operate Docker containers via Singularity, rendering it a viable substitute for Docker in HPC settings. This compatibility addresses security and performance issues related to running Docker containers in high-performance computing settings.

At NYU, program deployment occurs on Greene, one of the university's advanced HPC clusters. The deployment process involves configuring the necessary settings through a Bash file, which is crucial as it contains all the environment arrangements and configurations required for efficient program operation on the cluster. Following this, the process of submitting a job is completed through a batch file, outlining the specific resource requirements, including CPU time and memory.

The ultimate stage of deployment employs a .sbatch file, critical to the operation of Slurm. This file comprises directives and commands for the HPC system to manage and execute the job while ensuring effective allocation and usage of the designated resources. This efficient approach, which covers everything from configuration to resource allocation, emphasizes the NYU HPC clusters' ability to effectively manage and execute various computational tasks and research projects.

5.2 Limitation

5.2.1 Dataset Selection

We only selected the SSE 50 dataset for both our training and testing models. However, the dataset lacks diversification as it solely focuses on stocks of similar sizes. Therefore, it does not encompass stocks from various industries, sizes, and markets across different countries. The SSE 50 consists of 50 highly liquid large-cap stocks, making it difficult for the model to reflect the trading patterns of small and medium-sized stocks. These smaller stocks are more prone to stock market manipulation or policies. Given that most of the companies listed on the SSE 50 are industry leaders, it's not feasible to include industry-specific trends. Additionally, our model may not yield accurate predictions for a particular industry due to varying trading patterns in each country's market. The SSE 50 dataset used to train the model may not suffice for forecasting international markets.

Given a diverse market, a single model cannot predict every stock effectively. To solve dataset selection issues for future training, datasets should be categorized based on the size of stocks, industries, and countries. Consequently, corresponding datasets should be chosen for training and testing the corresponding models.

5.2.2 Loss Function Selection

The objective of our project is to predict the direction of stock price movement, rather than its exact numerical value. This objective warrants a revision of our current approach, especially in terms of the loss function employed in our model. We have been using the Binary Cross-Entropy (BCE) loss function, which, although regularly used in binary classification tasks, may not be the most optimal for our specific aim of forecasting directional movement. The observed high evaluation loss with the BCE loss function indicates a poor alignment with the nuances of our prediction task. BCE primarily concentrates on class membership probability, which in our case refers to whether the stock price will increase or decrease. Nevertheless, it may not adequately capture the complexities of the dynamic and frequently unpredictable nature of stock price movements.

Improving our model's performance necessitates revising the loss function through future research. We should explore more customized loss functions catering to our prediction task that accurately measure the correctness of price movement direction, instead of price prediction pre-

cision. For example, loss functions preferring heavy penalties on incorrect directional predictions could be more apt. Incorporating elements of time series analysis and the inherent volatility of stock prices into the loss function may result in more accurate predictions.

5.2.3 Neural Network Reconstruction

The utilization of the Rectified Linear Unit (ReLU) activation function in our machine learning model has gained popularity due to its simplistic and efficient promotion of faster convergence. However, this approach possesses certain limitations that may require more intricate neural network architectures to explore. The "dying ReLU" problem is the foremost issue, wherein neurons can permanently become inactive and solely output zero, resulting in a significant reduction of the model's learning capacity. Moreover, the rectified linear unit (ReLU) function is not centered around zero, which can result in optimization difficulties during training. Its linearity may also make it less capable of capturing intricate patterns in data compared to non-linear functions. Additionally, ReLU struggles to handle negative inputs as it simply zeroes them out, which could lead to important information being lost in the negative range of the input space. To improve our model's performance, particularly with more complex datasets, we should consider advanced neural network architectures. These might incorporate various activation functions, such as Leaky ReLU, Parametric ReLU, or even more sophisticated ones like Swish, that aim to address the limitations of standard ReLU. Incorporating such variations can help in preserving and efficiently processing a broader scope of information, allowing the model to detect more subtle patterns and dependencies in the data.

5.2.4 Resource Limitations

The problem of resource scarcity within the context of large model training is continuously growing and multifaceted. One significant issue is the extended queue times for accessing computational resources, indicating a high demand that the current infrastructure can hardly meet. This problem is worsened by the inadequate availability of essential hardware components, such as graphics processing units (GPUs), which are critical for efficient model training. The shortage of GPUs poses a significant obstacle, given their indispensable role in handling the demanding computational tasks necessary for training large models. Moreover, a noteworthy lack of memory capacity exists. As models increase in complexity and size, the need for more extensive memory to process and store vast amounts of data becomes imperative. This growing model complexity corresponds

with an increasing demand for more advanced and robust resources. However, obtaining these resources entails significant financial and equipment investments, which can pose a challenge, particularly for smaller organizations or individual researchers. The monetary aspect of procuring cutting-edge hardware and sustaining the essential infrastructure for extensive model training can be restrictive. The scarcity of resources not only impedes progress in the field of AI research and development but also creates worries regarding the fairness and accessibility of technological advancements. This predicament could result in a scenario where solely well-endowed entities can finance the development and training of state-of-the-art models.

6 Personal Contributions

1. Dataset Download(China A shares all stocks' history data, through Wind Terminal at NYUSH Library);
2. Code Modification;
3. Model parameter adjustment(All of the training on Grenne HPC is through my HPC account as Peiyang's are running another capstone project);
4. Most Final Report Writing;
5. All Final Report modifications.

7 Conclusion

Our exploration into the use of Variational Encoder-Decoder(VED) for predicting stock prices in the Shanghai Stock Exchange, particularly focusing on the SSE 50 Index, has yielded insights that are both promising and indicative of the challenges inherent in financial market forecasting. The choice of VAEs was motivated by their unique ability to handle complex, high-dimensional data, and their proficiency in generating robust generative models. Throughout the research, we utilized these capabilities to analyze historical stock price data, aiming to uncover underlying patterns that could inform future price movements. Our methodology involved training the VED with stock market data(SSE 50) using candlestick charts.

After continuously adjusting the code, the parameters of the model that can ultimately present better results are learning rate=0.000005, latent space = 64, epoch = 10000, the loss function is Binary Cross-Entropy (BCE) loss function.

Despite these promising results, our research encountered several challenges. The complexity of financial markets, influenced by a multitude of factors beyond historical price data, means that even advanced models like VED cannot always predict market movements with complete accuracy.

Furthermore, the 'black box' nature of VED and deep learning models, in general, poses interpretability challenges. While these models can make accurate predictions, understanding the reasoning behind these predictions is often not straightforward, which can be a significant hurdle in financial contexts where decision-making processes need to be transparent and explainable.

Another limitation we observed was the computational intensity and resource requirements for training and running these models. The need for extensive computational resources can be a barrier to entry for smaller institutions or individual researchers, highlighting a divide in access to advanced machine learning technologies.

In terms of future work, there is immense potential for refining these models. Integrating more diverse datasets, such as different industries, different markets, different sizes of stocks, and different countries, could enhance the models' predictive power. Additionally, efforts to improve the interpretability of VED could make these models more accessible and trustworthy for financial analysts and investors. In addition to selecting the dataset, subsequent reconstruction of a reasonable Loss function and the use of new and more complex neural networks may lead to better prediction results.

In conclusion, our study contributes to the growing body of knowledge on the application of machine learning in financial markets. While VED show potential in stock price prediction, their efficacy is contingent on the complexity of the market environment and reasonable dataset selection. This research opens the door for further exploration into the integration of machine learning with traditional financial analysis, paving the way for more sophisticated and nuanced approaches to stock market forecasting.

References

[1] W. Jiang, "Using the garch model to analyse and predict the different stock markets," *Uppsala University*, pp. 2–24, 2012.

[2] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo, "Stock price prediction using the arima model," in *2014 UKSim-AMSS 16th international conference on computer modelling and simulation*. IEEE, 2014, pp. 106–112.

[3] A. A. Adebiyi, A. O. Adewumi, C. K. Ayo *et al.*, "Comparison of arima and artificial neural networks models for stock price prediction," *Journal of Applied Mathematics*, vol. 2014, 2014.

[4] R. Efendi, N. Arbaiy, and M. M. Deris, "A new procedure in stock market forecasting based on fuzzy random auto-regression time series model," *Information Sciences*, vol. 441, pp. 113–132, 2018.

[5] J. Jiang, B. T. Kelly, and D. Xiu, "(re-)imag(in)ing price trends," 2020.

[6] U. Eren, "Rgbsticks: A new deep learning based framework for stock market analysis and prediction," *Journal of Soft Computing and Artificial Intelligence*, vol. 1, no. 2, pp. 78–85, 2020.

[7] X. Li, Y. Kang, and F. Li, "Forecasting with time series imaging," *Expert Systems with Applications*, vol. 160, p. 113680, 2020.

[8] M. Velay and F. Daniel, "Stock chart pattern recognition with deep learning," *arXiv preprint arXiv:1808.00418*, 2018.

[9] J.-H. Chen and Y.-C. Tsai, "Encoding candlesticks as images for pattern classification using convolutional neural networks," *Financial Innovation*, vol. 6, no. 1, pp. 1–19, 2020.

[10] X. Zhou, Z. Pan, G. Hu, S. Tang, and C. Zhao, "Stock market prediction on high-frequency data using generative adversarial nets." *Mathematical Problems in Engineering*, 2018.

[11] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[12] A. Desai, C. Freeman, Z. Wang, and I. Beaver, "Timevae: A variational auto-encoder for multivariate time series generation," *arXiv preprint arXiv:2111.08095*, 2021.

[13] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Proceedings of the aaai conference on artificial intelligence," pp. 11 121 – 8, 2023.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[15] A. Razavi, A. Van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," *Advances in neural information processing systems*, vol. 32, 2019.