

Machine Learning for Finance Project Report

Randy Shoemaker, Ghada Bakbouk, Anna Schmedding

May 2020

Abstract

We present and evaluate several machine learning based models for forecasting stock returns. Our models use Convolutional Neural Networks (CNN) implemented using Keras [2] with a Tensorflow backend [10] to learn temporal and cross-sectional patterns to aide in forecasting. This work highlights the importance of leveraging cross-sectional behavior when forecasting stock returns. We provide comparisons between the performance and usefulness of our models for forecasting stock returns.

1 Introduction

Forecasting stock returns is important in the financial industry. It has been studied by financial professionals [5] and computer scientists [11] alike. We present several convolutional neural network (CNN) based models for forecasting 5 day stock returns. Our simplest model, **simpleCNN**, makes forecasts by using a CNN to learn the temporal behavior of a single stock. **simpleCNN** underperforms the other models because it does not consider how the single stock's prices interact with those of other stocks. To take this kind of behavior into consideration, our other models learn the temporal and cross-sectional behavior of all stocks in a universe. Our model, **generalTemporal** learns the behavior of all stocks in a universe by performing a temporal convolution over the return data. **generalCross-Sectional** learns similar behavior by performing a cross-sectional based convolution over the return data. The previous two methods use one dimensional convolutions, **general2D**, by contrast learns the behavior of stocks in the universe by performing a two dimensional convolution. All of the previous models rely only on the 5 day return data for every stock in the universe. Our final model, **factorCNN**, also uses technical factors over the previous five days to aide in forecasting.

We now introduce the notation necessary to describe our models. Let $r_{i,t}$ refer to the 5-day return for stock i on day t . Specifically, $r_{i,t}$ is the return if we bough stock i on day $t - 4$ and sold it on day t . Then, for a universe of N stocks across M days we may represent the returns as

$$\begin{pmatrix} r_{1,1} & r_{1,2} & \dots & r_{1,M-1} & r_{1,M} \\ r_{2,1} & r_{2,2} & \dots & r_{2,M-1} & r_{2,M} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ r_{N,1} & r_{N,2} & \dots & r_{N,M-1} & r_{N,M} \end{pmatrix}$$

Given this notation, our **simpleCNN** built for stock i seeks to use $(r_{i,t-4}, r_{i,t-3}, r_{i,t-2}, r_{i,t-1}, r_{i,t})$ to

forecast $r_{i,t+5}$. The **generalTemporal**, **generalCross-Sectional**, and **general2D** seek to use

$$\begin{pmatrix} r_{1,t-4} & r_{1,t-3} & r_{1,t-2} & r_{1,t-1} & r_{1,t} \\ r_{2,t-4} & r_{2,t-3} & r_{2,t-2} & r_{2,t-1} & r_{2,t} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ r_{N,t-4} & r_{N,t-3} & r_{N,t-2} & r_{N,t-1} & r_{N,t} \end{pmatrix}$$

to forecast $(r_{1,t+5}, r_{2,t+5}, \dots, r_{N,t+5})$. Finally, **factorCNN**, seeks to use

$$\begin{pmatrix} r_{1,t-4} & r_{1,t-3} & r_{1,t-2} & r_{1,t-1} & r_{1,t} \\ r_{2,t-4} & r_{2,t-3} & r_{2,t-2} & r_{2,t-1} & r_{2,t} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ r_{N,t-4} & r_{N,t-3} & r_{N,t-2} & r_{N,t-1} & r_{N,t} \end{pmatrix}$$

as well as the same date, stock ranges for a set of $n = 5$ technical factors $\{\alpha_1, \dots, \alpha_n\}$

$$\begin{pmatrix} \alpha_{k,1,t-4} & \alpha_{k,1,t-3} & \alpha_{k,1,t-2} & \alpha_{k,1,t-1} & \alpha_{k,1,t} \\ \alpha_{k,2,t-4} & \alpha_{k,2,t-3} & \alpha_{k,2,t-2} & \alpha_{k,2,t-1} & \alpha_{k,2,t} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \alpha_{k,N,t-4} & \alpha_{k,N,t-3} & \alpha_{k,N,t-2} & \alpha_{k,N,t-1} & \alpha_{k,N,t} \end{pmatrix}$$

where $\alpha_{k,i,t}$ is the value of α_k on day t for stock i . These return and factor data are used to forecast $(r_{1,t+5}, r_{2,t+5}, \dots, r_{N,t+5})$. We discuss the details of our models in Section 3.

In Section 2 we discuss related work, in Section 3 we present our models, in Section 4 we discuss our datasets, in Section 5 we discuss training and hyperparameter tuning, in Section 6 we discuss how we evaluate our models, in Section 7 we discuss our results, and in Section 8 we discuss our conclusions and future work.

2 Related Work

Forecasting stock returns has been studied by financial professionals and computer scientists. As mentioned in [14] financial professionals tend to focus on utilizing factors and data cleaning with the goal of producing models with strong forecasting abilities [5]. These approaches overlook important aspects of machine learning such as hyperparameter tuning. Computer scientists, on the other hand, tend to focus on machine learning and neglect to utilize the technical factors hand crafted by professionals [3]. In our work we attempt to utilize the best of both approaches.

Previous work forecasting stock returns has utilized linear models [5] and other works have utilized neural networks and deep learning [14]. Works that utilize neural networks and deep learning tend to use recurrent neural networks (RNN), long short term memory (LSTM), and attention mechanisms [14] [6] [4]. Using RNNs makes sense because they are designed for temporal data. We instead use convolutional neural networks (CNN). While CNNs are typically used in

image classification and other computer vision tasks [9] [12], recently they have also been used for time series forecasting [13]. Our approach follows that of [1], with several modifications, for using CNNs for forecasting time series. Particularly, [1] partitions their training data similar to the way we do. They use five elements of a sequence to predict the next element while in our context this would cause severe look ahead since we are forecasting 5 day returns, we discuss this in more detail in Section 4. The neural networks which we used are generalizations of those of [1] where we allow for modifications for hyperparameter tuning. We discuss our networks in more detail in Section 3. In Section 4 we discuss our datasets, in Section 5 we discuss training and hyperparameter tuning, in Section 6 we discuss how we evaluate our models, in Section 7 we discuss our results before concluding in Section 8.

3 Models

In this section we present our CNN based models for forecasting 5 day returns. Our models include a simple model **simpleCNN** for learning the behavior of a single stock, **generalTemporal** for emphasis on learning the temporal behavior of the stocks in a universe, **generalCross-Sectional** for emphasis on learning the cross-sectional behavior of the stocks in a universe, **general2D** for learning the behavior of the stocks in a universe, and **factorCNN** for utilizing technical factors to learn the behavior of stocks in a universe. All of our models have a similar architecture as seen in Figure 1. The differences are the type of convolutions used and the dimensions of the dense layers. The factor model includes an additional dense layer to boost model capacity. Each model is programmed using the Keras Sequential class [2].

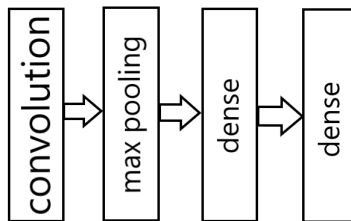


Figure 1: The general architecture of our models.

In the following sections we discuss each model in detail.

3.1 Single Stock Return Forecasting

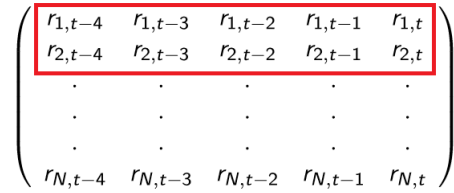
Using the notation in Section 1 the simple CNN model seeks to use $(r_{i,t-4}, r_{i,t-3}, r_{i,t-2}, r_{i,t-1}, r_{i,t})$ to forecast $r_{i,t+5}$. This model is trained on a single stock i to make forecasts of its returns. It has the architecture in Figure 1. The final dense layer outputs a single scalar, the forecast $\hat{r}_{i,t+5}$. The convolution goes over the input vector $(r_{i,t-4}, r_{i,t-3}, r_{i,t-2}, r_{i,t-1}, r_{i,t})$ with a window of 2 and a stride of 1. The max pooling layer has a window of 2. The width of the first dense layer is a hyperparameter and the learning rate of the model is a hyperparameter too.

3.2 One Dimensional Temporal Convolution

Using the notation in Section 1 **generalTemporal** seeks to use

$$\begin{pmatrix} r_{1,t-4} & r_{1,t-3} & r_{1,t-2} & r_{1,t-1} & r_{1,t} \\ r_{2,t-4} & r_{2,t-3} & r_{2,t-2} & r_{2,t-1} & r_{2,t} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ r_{N,t-4} & r_{N,t-3} & r_{N,t-2} & r_{N,t-1} & r_{N,t} \end{pmatrix}$$

to forecast $(r_{1,t+5}, r_{2,t+5}, \dots, r_{N,t+5})$. This model is trained on the universe of stocks across a training time period. It has the architecture in Figure 1, in this model the output of the final dense layer is a vector with N elements, one for each stock. The convolution goes over the entire temporal dimension of the input matrix and has a width of 2 in the cross-sectional dimension as seen in Figure 2 and has a stride of 1. The max pooling layer also has a window of 2. As with the simple stock model, for **generalTemporal** the width of the first dense layer is a hyperparameter as well as the learning rate.



$$\begin{pmatrix} r_{1,t-4} & r_{1,t-3} & r_{1,t-2} & r_{1,t-1} & r_{1,t} \\ r_{2,t-4} & r_{2,t-3} & r_{2,t-2} & r_{2,t-1} & r_{2,t} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ r_{N,t-4} & r_{N,t-3} & r_{N,t-2} & r_{N,t-1} & r_{N,t} \end{pmatrix}$$

Figure 2: The temporal convolution in **generalTemporal**.

3.3 One Dimensional Cross-sectional Convolution

As with the temporal model, **generalCross-Sectional** seeks to use

$$\begin{pmatrix} r_{1,t-4} & r_{1,t-3} & r_{1,t-2} & r_{1,t-1} & r_{1,t} \\ r_{2,t-4} & r_{2,t-3} & r_{2,t-2} & r_{2,t-1} & r_{2,t} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ r_{N,t-4} & r_{N,t-3} & r_{N,t-2} & r_{N,t-1} & r_{N,t} \end{pmatrix}$$

to forecast $(r_{1,t+5}, r_{2,t+5}, \dots, r_{N,t+5})$. Our **generalCross-Sectional** is trained on the entire universe of stocks across a training time period. It has the architecture depicted in Figure 1 and the output is a forecast vector of length N . The convolution goes over the entire cross-sectional dimension of the input matrix as seen in Figure 3 and has a window of 2 in the temporal direction. The max pooling layer has a window of 2. As with the previous models the hyperparameters for **generalCross-Sectional** are the width of the first dense layer and the learning rate.

$$\begin{pmatrix} r_{1,t-4} & r_{1,t-3} & r_{1,t-2} & r_{1,t-1} & r_{1,t} \\ r_{2,t-4} & r_{2,t-3} & r_{2,t-2} & r_{2,t-1} & r_{2,t} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ r_{N,t-4} & r_{N,t-3} & r_{N,t-2} & r_{N,t-1} & r_{N,t} \end{pmatrix}$$

Figure 3: The cross-sectional convolution in **generalCross-Sectional**

3.4 Two Dimensional Convolution

As with the previous two models **general2D** seeks to use

$$\begin{pmatrix} r_{1,t-4} & r_{1,t-3} & r_{1,t-2} & r_{1,t-1} & r_{1,t} \\ r_{2,t-4} & r_{2,t-3} & r_{2,t-2} & r_{2,t-1} & r_{2,t} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ r_{N,t-4} & r_{N,t-3} & r_{N,t-2} & r_{N,t-1} & r_{N,t} \end{pmatrix}$$

to forecast $(r_{1,t+5}, r_{2,t+5}, \dots, r_{N,t+5})$. It has the architecture shown in Figure 1 and similar to the previous two models its output is a vector of length N which is the forecast of the model. The convolution is 2×2 as shown in Figure 4 and has a stride of 1. The dimension of the max pool also has a window of 2×2 . The hyperparameters are the learning rate and the dimension of the first dense layer.

$$\begin{pmatrix} r_{1,t-4} & r_{1,t-3} & r_{1,t-2} & r_{1,t-1} & r_{1,t} \\ r_{2,t-4} & r_{2,t-3} & r_{2,t-2} & r_{2,t-1} & r_{2,t} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ r_{N,t-4} & r_{N,t-3} & r_{N,t-2} & r_{N,t-1} & r_{N,t} \end{pmatrix}$$

Figure 4: The two dimensional convolution in **general2D**

3.5 Factor Based Model

The most robust model is **factorCNN**, as mentioned in Section 1 it seeks to use

$$\begin{pmatrix} r_{1,t-4} & r_{1,t-3} & r_{1,t-2} & r_{1,t-1} & r_{1,t} \\ r_{2,t-4} & r_{2,t-3} & r_{2,t-2} & r_{2,t-1} & r_{2,t} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ r_{N,t-4} & r_{N,t-3} & r_{N,t-2} & r_{N,t-1} & r_{N,t} \end{pmatrix}$$

as well as the same date, stock ranges for a set of $n = 5$ technical factors $\{\alpha_1, \dots, \alpha_n\}$

$$\begin{pmatrix} \alpha_{k,1,t-4} & \alpha_{k,1,t-3} & \alpha_{k,1,t-2} & \alpha_{k,1,t-1} & \alpha_{k,1,t} \\ \alpha_{k,2,t-4} & \alpha_{k,2,t-3} & \alpha_{k,2,t-2} & \alpha_{k,2,t-1} & \alpha_{k,2,t} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \alpha_{k,N,t-4} & \alpha_{k,N,t-3} & \alpha_{k,N,t-2} & \alpha_{k,N,t-1} & \alpha_{k,N,t} \end{pmatrix}$$

We discuss our particular choice of factors in Section 4. The input data of the returns together with 5 factors makes a 3-tensor. Instead of an input vector as in **simpleCNN**, or an input matrix as in **generalTemporal**, **generalCross-Sectional**, and **general2D** the input of **factorCNN** is a 3-tensor $T_{i,t,k}$. The index i refers to the stock, the index t refers to the day, and the index k , analogous to the channel dimension in images, refers to the value of the particular factor α_k or 5-day return of stock i on day t . The convolution, as with convolutions over RGB images, goes over the entire channel dimension. The convolution is 2×2 in the temporal and cross-sectional dimensions respectively. The architecture of **factorCNN** differs from the other models in that it adds an additional dense layer. We add this layer to provide the model with more capacity. We felt this was necessary because of the additional information, in the form of technical factors, that this neural network had to learn from. As with the previous three models, this model outputs a vector of length N which is its forecast for each stock. The max pool has dimension 2×2 . This model has three hyperparameters, the learning rate, and the width of the first and second dense layers.

In the following section we discuss the rationale behind our models and further aspects of their architecture.

3.6 Discussion

Here we discuss the rationale of each model. The model **simpleCNN** is meant to see how effective a CNN can be at learning the temporal behavior of a single stock. It is also meant to be a baseline for our other models. We tested it on many randomly chosen stocks. This model was also intended to be a good starting point for which to build the more complicated models.

The model **generalTemporal** is meant to fully utilize the temporal behavior of each training example to forecast the 5 day return for each stock. At the outset it was our contention that this model would be the weakest of the non-simple models, and we will discuss this further in Section 7. This model takes the smallest amount of time to train due to the fact that the convolution only looks over two stocks over a period of five days.

The model **generalCross-Sectional** is meant capture the cross-sectional behavior of the stocks in our universe. It was designed to detect and utilize behavior such as stocks “stalking” other stocks. Initially, before delving into how we implemented our idea, we felt that this stock would underperform **general2D**, and we will discuss this further in Section 7. This model took significantly longer to train than **generalTemporal** because it looks over every stock for a period of two days.

The model **general2D** is meant to explore the importance of each dimension; temporal and cross-sectional. To fully explore this question, we believe that future work should be conducted where the dimensions of the kernel are varied. Initially it was our belief that this model would be second only to **factorCNN**, and we will discuss this further in Section 7. This model took slightly longer to train than the cross-sectional model.

The model **factorCNN** is meant to utilize the handcrafted features of financial experts, in addition to only using return data. We used 5 factors, which we discuss in Section 4. We decided to only use five for the sake of time, as this model takes significantly longer to train than the others. When we started we felt that this would be the strongest model due to its use of technical factors, and we will discuss this further in Section 7.

Initially we were going to let the kernel and pooling dimensions be additional hyperparameters of our models. Once we started hyperparameter tuning we realized that this was not realistic. After a discussion with Professor Liu during our class we decided to use his advice and make the learning rate and the width of the hidden layers hyperparameters. If we would have used all of the aforementioned values as hyperparameters it would have taken a very large amount of time to tune.

In addition to the previously discussed architectural choices we also choose to use the rectified linear unit, ReLu, as the activation function in each layer, including the convolutional layers. We made this choice to utilize non-linearity as discussed in [5].

In the following section we discuss our dataset before discussing model evaluation, results, and concluding.

4 Data

We use CRSP stock data at the daily level provided by WRDS. The range of our initial dataset is January 2009 to the end of December 2019 and it encompassed all available stocks during that time. In order to obtain the most liquid stocks we found the 3000 most liquid stocks in terms of share volume during the period from January 2009 to the end of December 2009. We did this to prevent look ahead. We then removed all stocks corresponding to PERMNOs that had a significant number of NaNs as entries, this left 1763 stocks. For training, validation, and testing we used data for these 1763 stocks from January 2010 to the end of December 2019. We call this dataset *3000MostLiquid*.

We used the closing price from *3000MostLiquid* to compute the five day returns for each stock, specifically the return for each day was computed assuming we bought the stock five days ago and sold it today. We call this dataset *general5DayReturn*. For **simpleCNN** randomly picked a few PERMNOs from *general5DayReturn* and built a forecasting model for each one. The data for each PERMNO was partitioned into a training set from January 2010 to the end of December 2017, a validation set from January 2018 to the end of December 2018, and a testing set from January 2019 to the end of December 2019. Each of these sets were made into input - ground truth pairs.

The dataset *general5DayReturn* was used for training, tuning, and testing **generalTemporal**, **generalCross-Sectional**, and **general2D**. The dataset was partitioned into training, tuning, and testing sets using the same date ranges used for the **simpleCNN** models.

The dataset used to train, tune, and test **factorCNN** is unique. As with the others the dataset used for **factorCNN** was derived from *3000MostLiquid*. We computed the five day returns in the same way as *general5DayReturn* except we couldn't use the same date ranges. This is because we had to make sure the date ranges for our computed factors exactly coincided with the date ranges for the returns. Before we discuss this issue we discuss the factors we used.

The code for the factors we used came from the factor code provided by Dr. Liu in Box. We used "newHigh", "alpha001", "alpha003", "alpha060". The later three are implementations from [7]. We also used a simple moving average (SMA) across a window of ten days, we scaled the SMA data down by a factor of 100 so it was in a similar range as the values of the other factors and the returns. We choose "newHigh" because it gives us information about a stock's high relative to

past high values. We choose “alpha001” because it tells us how a stock’s one day return relates to its volatility. We choose “alpha003” because it gives us information about how a stock’s opening price correlates with its volume. We choose “alpha060” because it gives us information about how a stock’s volatility with respect to other stocks relates to its prices. Finally, we choose to use a simple moving average because it gives us information about a stock’s price history. Once these factors were computed for each stock on each day in *3000MostLiquid* we had to make sure the date ranges of all of the factors coincided with the date ranges of the returns. The date range where all of these data sets overlapped was from February 12, 2009 to the end of December 2019. We refer to the dataset composed of the returns and factors in this specified date range as *factorData*. *factorData* was partitioned into a training set from dataset’s initial date to the end of May 2018, a tuning set from the beginning of June 2018 to the end of December 2019, and a test set from the beginning of 2019 to the end of December 2019. We lengthened the date range of training data for this model to aide its forecasting ability of the testing set.

All datasets described here can be found in our Box link. In the following sections we discuss Training, Evaluation, and Results, before concluding.

5 Training and Hyperparameter Tuning

During training we used the optimizer adam [8] to minimize the mean square error between the forecast of the network and the ground truth. A great deal of effort went into finding the optimal hyperparameters for our models. Below, we discuss hyperparameter tuning in detail.

5.1 Hyperparameter Tuning

In this section, we will include the methods and results of our hyperparameter tuning. Each subsection will include the parameters that we tested for each of our models, the values we tested, and the values that produced the best model for each model’s validation set. In general, the models are compared based on the correlation between the forecast and the ground truth. We calculate the correlation between the output vector of the model for a given day and the actual return of each stock for that day, this yields a correlation coefficient β_t for each day t . In order to evaluate the model, we take the average of the entries of this vector, and find the confidence interval of these entries with high confidence ($> 99.99\%$). The result is a single number that represents the correlation between the forecast of the model and the ground truth. This number, hereafter referred to as the correlation β for short, is the value used to evaluate the performance of the model.

The hyperparameters that we ran tests on are the *learning rate*, the number of *epochs*, and *dense_out*: the dimensionality of the output space for the first dense layer. There are other hyperparameters in the model beside the aforementioned ones, but we choose not to test them due to time constraints. These hyperparameters are:

- *Time step*: the number of returns we use in the input vector. Fixed at 5.
- *Kernel size*: the length of the convolution. Fixed at 2.
- *Pool size*: size of the max pooling windows. Fixed at 2.

We will include the fixed values of these parameters for each model. For tuning the hyperparameters of each model we wrote scripts to loop over different configurations of hyperparameters.

Each of these scripts produced csv files containing the configuration along with the performance of each configuration in terms of R^2 . These csv files along with the scripts contained in jupyter notebook files can be found in our box link.

5.1.1 Simple Stock Model

This model is trained on the data of one stock over the entire training period. This means that it is not possible to take the average of the correlations for multiple stocks, or find the confidence interval of the correlations, since there is only one stock. Hence, the performance of the model is evaluated using the correlation between the forecast vector and the returns vector.

We train this model on a single stock (PERMNO 10032). The hyperparameters we tested are:

| Parameter | learning rate | epochs | dense_out |
|-----------|---------------------|-----------------|------------------|
| Values | 0.01, 0.001, 0.0001 | 10, 25, 50, 100 | 1, 5, 10, 20, 50 |

To test these parameters, we ran a nested loop that iterates through these values, training each model 10 times for any given set of parameters we then evaluate each model on the validation set. This step helps mitigate some of the variation caused by the random weights assigned by Keras. We stored the output of these loops in a csv file, and examined it to find the top 15 models based on the correlation. Then, we found the best combination of parameters, and ran it 10 more times, saving the best model. The best combination that we found for this model is *learning rate* = 0.001, *d_out* = 25, and *epochs* = 1. The best model has a correlation of 0.044 on the validation set.

5.1.2 Temporal CNN Model

Since this model is trained on all stocks, we can use the evaluation method outlined in 5.1. The hyperparameters we tested are:

| Parameter | learning rate | epochs | dense_out |
|-----------|--------------------------|------------|-------------|
| Values | 0.00085, 0.0009, 0.00095 | 10, 20, 30 | 25, 50, 100 |

Similar to the previous model we ran loops to iterate over the aforementioned values, and then ran more experiments on the best combination. The best combination that we found for this model is *learning rate* = 0.00095, *d_out* = 25, and *epochs* = 10. The best model has a correlation of 0.290 on the validation set, with (0.233,0.348) for a confidence interval.

5.1.3 Cross-sectional CNN Model

The hyperparameters we tested for this model are:

| Parameter | learning rate | epochs | dense_out |
|-----------|-----------------------|------------------|-------------|
| Values | 0.0009, 0.001, 0.0011 | 1, 5, 10, 20, 30 | 25, 50, 100 |

The best combination that we found for this model is *learning rate* = 0.00095, *d_out* = 25, and *epochs* = 10. The best model has a correlation of 0.3 on the validation set, with confidence interval (0.240, 0.359)

5.1.4 Two Dimensional CNN Model

The hyperparameters we tested for this model are:

| Parameter | learning rate | epochs | dense_out |
|-----------|-----------------------|------------------|-------------|
| Values | 0.0009, 0.001, 0.0011 | 1, 5, 10, 20, 30 | 25, 50, 100 |

The best combination that we found for this model is *learning rate* = 0.0009, *d_out* = 100, and *epochs* = 30. The best model has a correlation of 0.302 on the validation set, with a confidence interval of (0.242, 0.362).

5.1.5 The Factor CNN

For this model, our experiments were less comprehensive due to the long time needed to train each iteration of this model. Moreover, there is an extra parameter that needs to be tuned in this model, which is the dimensionality of the output space for the second dense layer, *dense_out2*. We chose the values that we ran depending on the observations from the previous models. The hyperparameters we tested for this model are:

| Parameter | learning rate | epochs | dense_out | dense_out2 |
|-----------|---------------|--------|-----------|------------|
| Values | 0.0009 | 30 | 50, 100 | 50,100 |

The best combination that we found for this model is *d_out* = 100, and *d_out2* = 100. The best model has a correlation of 0.262 on the validation set, with a confidence interval of (0.178, 0.347).

6 Evaluation

In this section we present the performance of each model on their respective test sets. The correlation for the simple stock model is the Pearson correlation between the forecast vector and the ground truth vector of the five day returns. For the other models we computed the Pearson correlation between the forecast of each day and the ground truth for each day and averaged the β_i to find β , as explained in Section 5.1. We included the the confidence interval with 99.99% confidence. We used β to compute R^2 , which is the measure for the variability in the stock prices accounted for by the model. In the following section we give a discussion of our results.

| Model | Correlation | Confidence Interval | R^2 |
|---------------------|-------------|---------------------|---------|
| Simple Stock | -0.0109 | - | 0.00012 |
| Temporal CNN | 0.074 | (0.040, 0.109) | 0.0055 |
| Cross Sectional CNN | 0.087 | (0.124, 0.050) | 0.0075 |
| 2D CNN | 0.084 | (0.047, 0.121) | 0.0071 |
| Factor CNN | 0.084 | (0.048, 0.120) | 0.0070 |

7 Results

For the simple stock model, the model produced a forecast that negatively correlates with the stock price of the tested stock. We believe that the this model did so poorly on the test set because it did not take any cross-sectional information into account. The test set started a year after the end of the training set so we believe that it is reasonable to expect such poor performance. The

other models take the returns for other stocks into account, which explains why they outperform the simple model.

The temporal model performed the worst out of the multi-stock models. We believe that the reason for this poor performance is that it takes the least amount of cross-sectional behavior into account in the convolutional layer. This aligns with our initial assumption that this model would have the worst performance out of the multi-stock models. The cross-sectional model, however, takes more of this information into account, and thus has better forecasting ability.

The cross-sectional model had the best performance on the testing set. We assumed that it would be able to forecast returns better than the temporal model due to its ability to leverage cross-sectional behavior. Our assumption was correct regarding the temporal model, however we did not expect it to outperform the general 2D model and the factor model.

Our assumption that the 2-D CNN model would outperform the cross-sectional model was incorrect. We believe that this illustrates that cross-sectional behavior is highly important when forecasting stocks. This model was not able to leverage this behavior better than the cross-sectional model because the latter model took more of this information into account.

Finally, the results for the factor model were not as good as we expected them to be. There are multiple reasons that might have contributed to the poor performance of this model. First, the set of hyperparameters that we used for tuning is limited due to the long time required to fit and train different configurations of this model. Also, it may be the case that the model does not have enough capacity to capture the effect of the factors. Conversely, the additional dense layer may have added too much capacity thus making the model overfit. We think additional work should be done to decide which, if either, is the case. In the following section, we discuss some possibilities for future work to improve the performance of this model.

Overall, the comparison of these models provides some insight into the importance of temporal, cross-sectional, and technical factor information in forecasting stock returns. Although there are still many ways to expand on this work, it provides some intuition for why these aspects are needed in this area.

8 Conclusion and Future Work

This work contains implementations of five different CNN models forecasting the 5-day return of stocks in the U.S. market. The models perform differently depending on the dimensionality of the convolution, amount of cross-sectional relations accounted for, and hyperparameter tuning. We believe that our work highlights the importance of leveraging cross-sectional behavior when forecasting stock returns.

We believe that our models could be improved in several ways. First, we would like to experiment with different kernel dimensions. We believe that this could add to the forecasting ability of our models. We also think it would be beneficial to treat kernel dimension, pool size and the length of the time step as hyperparameters. While this would be quite time consuming we believe the models can be improved by tuning these. We would also like to improve **factorCNN** by adding more factors. We were forced to choose a limited set of factors due to time constraints. Each factor added increases the training time which in turn greatly increases the time need to tune our models. We believe that selecting factors which have low correlation with each other would be most effective, instead using many correlated factors. We also think it would be beneficial to carry out more hyperparameter tuning of **factorCNN**. We think that this model should be able to outperform the

others due to the fact that it takes additional information into account. Additional hyperparameter tuning should enable this model to outperform the others.

In order to mitigate the difficulties with training time, it may also be worthwhile to investigate a GPU implementation. We have not had time to investigate this here, but it is something that should be investigated going forward.

References

- [1] Jason Brownlee. *How to Develop Convolutional Neural Network Models for Time Series Forecasting*. URL: <https://machinelearningmastery.com/> (cit. on p. 3).
- [2] François Chollet et al. *Keras*. <https://keras.io>. 2015 (cit. on p. 1, 3).
- [3] Daizong Ding et al. “Modeling Extreme Events in Time Series Prediction”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*. KDD ’19. Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 1114–1122. ISBN: 9781450362016. DOI: 10.1145/3292500.3330896. URL: <https://doi.org/10.1145/3292500.3330896> (cit. on p. 2).
- [4] Fuli Feng et al. “Temporal Relational Ranking for Stock Prediction”. In: *ACM Transactions on Information Systems* 37.2 (Mar. 2019), pp. 1–30. ISSN: 1558-2868. DOI: 10.1145/3309547. URL: <http://dx.doi.org/10.1145/3309547> (cit. on p. 2).
- [5] Shihao Gu, Bryan Kelly, and Dacheng Xiu. “Empirical Asset Pricing via Machine Learning”. In: *The Review of Financial Studies* 33.5 (Feb. 2020), pp. 2223–2273. ISSN: 0893-9454. DOI: 10.1093/rfs/hhaa009. eprint: <https://academic.oup.com/rfs/article-pdf/33/5/2223/33098540/hhaa009.pdf>. URL: <https://doi.org/10.1093/rfs/hhaa009> (cit. on pp. 1, 2, 7).
- [6] Ziniu Hu et al. *Listening to Chaotic Whispers: A Deep Learning Framework for News-oriented Stock Trend Prediction*. 2017. arXiv: 1712.02136 [cs.SI] (cit. on p. 2).
- [7] Zura Kakushadze. “101 Formulaic Alphas (December 9, 2015)”. In: *Wilmott Magazine* 84 (2016). URL: <http://dx.doi.org/10.2139/ssrn.2701346> (cit. on p. 7).
- [8] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG] (cit. on p. 8).
- [9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf> (cit. on p. 3).
- [10] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: <https://www.tensorflow.org/> (cit. on p. 1).
- [11] F. Ming et al. “Stock Market Prediction from WSJ: Text Mining via Sparse Matrix Factorization”. In: *2014 IEEE International Conference on Data Mining*. 2014, pp. 430–439 (cit. on p. 1).

- [12] Julien Philip et al. “Multi-view Relighting Using a Geometry-Aware Network”. In: *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)* 38.4 (July 2019). URL: <http://www-sop.inria.fr/reves/Basilic/2019/PGZED19> (cit. on p. 3).
- [13] Lamyaa Sadouk. “CNN Approaches for Time Series Classification”. In: *Time Series Analysis*. Ed. by Chun-Kit Ngan. Rijeka: IntechOpen, 2019. Chap. 4. DOI: 10.5772/intechopen.81170. URL: <https://doi.org/10.5772/intechopen.81170> (cit. on p. 3).
- [14] Qiong Wu et al. *A Deep Learning Framework for Pricing Financial Instruments*. 2019. arXiv: 1909.04497 [cs.LG] (cit. on p. 2).