CNN on Capturing Asset Return Features and Forecasting Next-Day's Buy or Sell

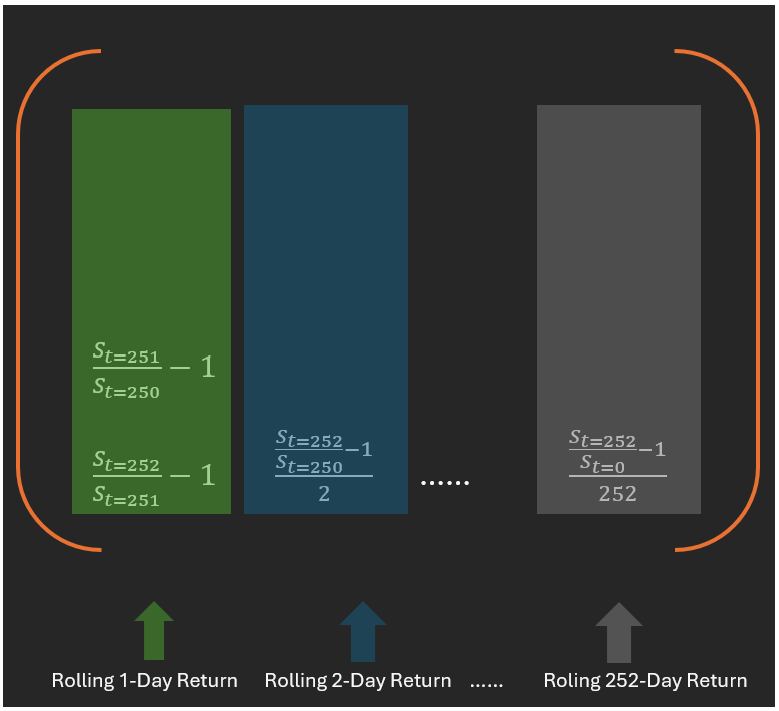
Convolutional Neural Network (CNN) has been proven to yield amazing results for pattern recognition; however, financial data is seemingly fussier than image data and can have a harder time being absorbed properly by deep learning frameworks, not to mention such limited uses exist in academia than industry by a great margin.

One of the greatest hurdles is trading tractability; to be more specific, image recognition is advantaged by large amounts of training and test data, given the latter can hold more significance than the former, and as a result, the model benefited by the law of large numbers; whereas in the financial world, even if we set a sufficiently large number of training data (e.g. 7000), having 30% of test data (e.g. 3000) is unrealistic due to trading them yields high transaction cost and difficulty. Therefore, my goal is to provide a trading strategy that's operationally easier and can be demonstrated with back-testing results using real-world data. In the following paragraphs, the methodologies, data, and results will be articulated in detail.

Methodology:

One differentiates CNN from regular NN is its ability to use kernels to scan the local features of matrices, which I hope will be taken advantage of by the nature of financial data in a sense small fussy seasonality can be captured by customized kernels. Therefore, the raw data I feed into the model are the rolling returns in the structure shown in Figure I. For example, if we want an N-by-N matrix of NFLX's rolling N-day returns up until 2024-03-31 with N = 252 (so in this case we have a year's worth of rolling returns), we would need to extract NFLX's adjusted closed from 2022-03-31 to get two years of data. In simple image recognition, people tend to use the square size of kernels to scan different convolutional layers. Nevertheless, it makes less sense to scan our rolling return matrices using squares; instead, I use 7-by-1 kernels to scan vertically the first layer to extract seven days of rolling N-day returns with N being fixed, and then use 1-by-7 kernels to scan horizontally the second layer to extract rolling k-day returns (k = 1…7) with the market dates being fixed. In this case, I'm telling the machine to look for patterns on 7-day momentum and collect useful information that could classify assets into "buy" or "sell".

Figure I.



Data

I use assets with substantial market caps from https://companiesmarketcap.com and ranked them by volatilities. Assuming I want to trade and unwind every day in March of 2024, then I need to calculate their volatilities using two-year horizons of daily return from 2022-03-01 to 2024-03-01. The top 30% of those assets with the greatest volatilities will be my test data, and the rest of the 70% is my training set. In this case, my X variables are a bunch of matrices with 2022-03-01 to 2024-03-01's rolling returns, and my Y variables are the returns from 2024-03-01 to 2024-03-02. A stochastic gradient descent (SGD) optimizer is used to minimize MSE loss when running CNN with different batches of the training dataset, then the out-of-sample test dataset is used to compute accuracies, namely, the probability of getting the buy-or-sell prediction correct for the next business day.

The reason for having the high-volatility assets be our test data, or in other words, the assets we want to trade, should have features that can be grabbed easily by the model and hence increase out-of-sample accuracy. Nevertheless, doing so bears a higher risk since a wrong guess in a high-vol asset generates more loss than in a low-vol asset; to reconcile I set the notional for each trading asset to be inversely proportional to its volatility, and as a result, our model has an easier time to classify assets into "buy" or "sell" and carry fewer downside risks.

Results

Table I shows the accuracy, daily return, and accumulative return using a sample size of 582 for March 2024. With the momentum being 0.5, the model can achieve a higher accumulative return than the momentum being 0.9. Nevertheless, a 0.9637% monthly return is slightly better

than a random guess, not to mention compensating the transaction cost of those 175 trading assets.

Table I.

		momentum=0.5			momentum=0.9		
Start	End	Accruracy	Daily Ret	Accum Ret	Accruracy	Daily Ret	Accum Ret
3/1/2022	3/1/2024	0.605714	0.002751	0.002751	0.6	0.002684	0.002684
3/4/2022	3/4/2024	0.451429	-0.00339	-0.00064	0.44	-0.00408	-0.00141
3/5/2022	3/5/2024	0.548571	0.00336	0.002715	0.56	0.003458	0.002043
3/6/2022	3/6/2024	0.56	0.003458	0.006182	0.56	0.003434	0.005484
3/7/2022	3/7/2024	0.565714	0.00364	0.009844	0.554286	0.003375	0.008878
3/8/2022	3/8/2024	0.588571	0.00125	0.011107	0.588571	0.00125	0.010139
3/11/2022	3/11/2024	0.451429	-0.00142	0.009671	0.451429	-0.00143	0.008698
3/12/2022	3/12/2024	0.502857	-0.00145	0.008212	0.502857	-0.00145	0.007241
3/13/2022	3/13/2024	0.502857	-0.00145	0.006755	0.508571	-0.0013	0.005934
3/14/2022	3/14/2024	0.508571	-0.0013	0.005449	0.508571	-0.0013	0.00463
3/15/2022	3/15/2024	0.428571	-0.00364	0.001789	0.422857	-0.0038	0.000816
3/18/2022	3/18/2024	0.511494	0.001399	0.00319	0.505747	0.001061	0.001878
3/19/2022	3/19/2024	0.557471	0.002469	0.005667	0.545977	0.002381	0.004263
3/20/2022	3/20/2024	0.54023	0.002245	0.007925	0.54023	0.002245	0.006518
3/21/2022	3/21/2024	0.545977	0.002363	0.010306	0.54023	0.002245	0.008778
3/22/2022	3/22/2024	0.522989	-0.00101	0.009289	0.522989	-0.00101	0.007762
3/25/2022	3/25/2024	0.505747	-0.00028	0.009004	0.505747	-0.00028	0.007478
3/26/2022	3/26/2024	0.45977	-0.00049	0.008509	0.471264	-0.00021	0.007265
3/27/2022	3/27/2024	0.471264	-0.00021	0.008296	0.465517	-0.00069	0.006567
3/28/2022	3/28/2024	0.45977	-0.00137	0.006916	0.45977	-0.00137	0.005189
3/29/2022	3/29/2024	0.505747	0.002703	0.009637	0.5	0.002528	0.00773

More results to come for the following months…

Helpful Resources:

https://www.youtube.com/playlist?list=PLQVvvaa0QuDdeMyHEYc0gxFpYwHY2Qfdh

https://towardsdatascience.com/conv2d-to-finally-understand-what-happens-in-the-forward-pass-1bbaafb0b148