Title: Deep Learning for event-based stock price dependencies across multiple market sectors
Group Name: TensorBros
Team: Charishma Ravoori (CR2ST)
Saurav Sengupta (SSY4D)
Navin Kasa (NK4XF)

1. Introduction

All modern businesses have value chains so diverse and widespread that their performance is highly dependent on the performance of businesses in other sectors. These dependencies could be stemming from the complementary or supplementary nature of products and services or the supply chain relationship usually in form of client-vendor one. Stock prices are usually a good representation of perception of business in market. While not always a direct measure of true performance of the business they are sensitive to the cross-market events. Stock prices of businesses in a sector usually move together. Studying the cross-sector sensitivity helps build models to predict stock prices based on observed sensitivity. This can help add another dimension to the existing price prediction algorithms. The key challenge is to extract meaningful events from historical time series data of multiple companies and build models that can identify relationship between stock prices and events. The problem at hand can be broken into two major components, extraction of events from time series data and event-based predictions.

2. Literature Survey

One of the ways in which event detection in Time Series is "change-point detection" where in dynamic phenomenon which have significant behavioral changes over time are identified [2]. The focus of this study is to apply data mining techniques to identify change points in a raw time series data where it is difficult to define threshold of change. The problem can be defined as identifying pre-defined number of piecewise segmented models between change points. Maximum likelihood estimate of the changepoints are computed based on the number of timepoints in a segment by a likelihood criteria function. Overall this approach predicts a change point based on the change in underlying parameters of data in a segment using model selection techniques.

Change point detection methods can be classified into two categories, real time detection which focuses on instantaneous changes and retrospective detection that allow for longer reaction times[2][3]. We focus on retrospective detection given the nature of our problem and the dataset provided. One such method involves tracking changing data adaptively and gradually forgetting the effect of past data using autoregressive modeling and detecting outliers relative to this trained model[4].

Event based prediction in the stock market deals with anticipating the fluctuation of stock prices as a result of certain events taking place like earnings announcements, large scale litigations or the release of a new product.

There has been quite a bit of work done in event-based prediction. Xiao Ding et al. discuss a method of using events extracted from news text to make predictions [5]. These events are represented as dense vectors (represented using event embeddings) which are then sent to a deep convolutional neural network to model the short-term and long-term influences of events on the stock price movements. Event embeddings are like templates for similar types of events, for example: If Apple and Google release a product, these events would have the same embedding.

The input to the model is a chronologically ordered sequence of event embeddings, where the events in one day are averaged together. The model understands the events as long-term, mid-term and short-term based on the time-span of the event. The different durations of events are extracted from the data by using a sliding window that combines 'l' neighboring events for long-term and 'm' neighboring events for mid-term. The model eventually learns the effects of these durations. The model takes in this time series aspect as well as a number of important local features (determined by the pooling layer of the neural network) to produce a binary output (+1 for an increase in stock prices and -1 for a decrease). The results for this study showed that using deep convolutional networks and event embeddings increase predictions by 6% over the S&P 500 index prediction.

3. Proposal

We aim to provide a method to model price changes in stock prices of companies in one sector based on events detected in the stock price changes in another sector. Essentially, we want to see if the price of the stock of an organization in one sector is in any way dependent on the events happening in unrelated sectors.

We take 5 different sectors in the market. For each sector we pick a random set of 10 companies that we take as representatives of that sector. We define Output Sector as the sector whose prices we want to predict. We define the Input Sector as the one for which we track the events. We also propose to take a window of time T for which we capture event E in the Input Sector, where each event E is characterized by m features. Therefore, at any given time T we capture m features for each of the 10 companies, giving us 10m features at time T for the Input Sector. We also calculate the price Index of the output sector, which is the weighted average of the traded stock prices in the time T of all the companies. This gives us out target variable PI(S), defined as the price index of the Output Sector. In the end we will have the following table:

PI(Output Sector)	1	2	…	10m
P1	e_1	e_2		e_{10m}
..				
P(t)				

The next step is to build a simple regression model, that models the prices of the Output Sector on the event features from the Input Sector. Since there are 5 sectors, and we model the price of each sector with every other sector, we have 5 x 5 = 25 different models that try to predict the dependence of the Output Sector (eg. S1) with itself and the 4 other Input Sectors (S2 to S5). In the end we can model each sector on every other sector, where we measure the test MSE of each model as an indicator of the performance of the model thereby as measure of hypothesized dependence and potential predictability of that dependence.

	S1	S2	S3	S4	S5

S1	MSE_{S1-S1}	MSE_{S1-S2}	MSE_{S1-S3}	…	…
S4					
S5	…	…	…	…	MSE_{S5-S5}

4. Data

We are measuring events as changepoints in Trade prices for 10 companies in each sector. The procedure and processing steps are explained in succeeding section. So, the input features are various characteristics of "peaks" among changepoints in sequence of their occurrence. Below are descriptions of our input features.

Feature	**S1**
height	Vertical distance of a "peak" changepoint from its lowest contour point. Measures the magnitude of the event.
width	Horizontal distance between left and right interpolated points at its lowest contour level. Measures how long the event occurred.
distance	Horizontal distance between current peak and previous peak. Measures the frequency of events.
left_slope	Slope of left edge of the peak. Measures how fast the peak rose.
right_slope	Slope of right edge of the peak. Measures how fast the peak fell.
target_sector_average_price	Average Trade price around the time of event for a window of size = 5% total Trade duration for that sector

For Phase 2 we did this feature engineering only for three sectors and for only one company per sector. And also, peaks are filtered such that only those are above 0.5x standard deviation of the changepoints are selected. Hence our final dataset is only 1794 in length. But these 1794 event peaks are identified across 100000 Trades in original dataset. Models are built for each sector pairs, so we have 6 different datasets

Dataset Type	**Train Event Count**	**Test Event Count**
Health Care-Financials	124	42
Health Care-Information Technology	124	42
Financials-Health Care	132	44

Financials-Information Technology	132	44
Information Technology-Health Care	416	139
Information Technology-Financials	416	139

5. Preprocessing

All Trade transactions for one company across selected sectors are processed using "banpei" API based on Singular Spectrum Transformation anomaly detection technique. Changepoints are generated across timeline of transactions. Standard deviation is computed for all the changepoints. "Peaks" or changepoints with height above 0.5x of standard deviation are considered as significant events.

These event peaks are measures across multiple dimensions like height, width, distance from previous peak, right and left slopes. The idea is to measure the magnitude, duration, frequency and speed of rise and fall of each event. The target variable is the average price over a sliding window of Trade transactions for each sector. The window length is set as 5% of the total length of Trade transactions for each sector. Additionally, an offset is introduced with length of 25% of window size. The window size and offset together ensures that the average price is calculated around the time of the event instead of just after the event.

Both predictors (event peak measures) and target variable (average price over sliding window) are fit to a normalized scale between 0 and 1. This ensures that the arbitrary units of measures do not impact modeling performance.

Every event peak observed is tagged two additional columns specifying the sector in which its observed and the target sector for which the average price is calculated.

6. Baseline

OLS models are built on each dataset to predict the standardized average price point for the observed event peaks. Following are the MSE for each sector pair.

Event Sector	Financials	Health Care	Information Technology
Financials		0.100076664	0.046955161
Health Care	0.088191858		0.056826344
Information Technology	0.071093045	0.056958597	

7. Method/Analysis:

FFNN:

A Feedforward Neural Network (FFNN) is a general neural network structure where the connections between the nodes are not cyclic. The optimal FFNN for this problem has 3 hidden layers with 100, 100 and 156 nodes respectively. The optimizer used is the Stochastic Gradient Descent optimizer which tries to find the minimal cost (in this case mean squared error). The metric being calculated by the cost function in this network is the 'Mean-Squared-Error'. Mean-Squared-Error calculates the average of the squared difference between the true response values and the predicted response values. The smaller this metric gets, the better the model is at predicting the true response.

Formula: MSE = mean((y – predicted_y)2)

Also, there is no regularization technique used in the model, as adding the dropout regularization made the model perform worse than it did without any regularization.

Our MSE for FFNN:

3 Hidden Layers			
	Healthcare	Information Tech	Financial
Healthcare		0.04506028	0.10944522
Information Tech	0.07133754		0.07534434
Financial	0.08195476	0.05239298	

CNN Model:

Because of the temporal nature of the data, for our CNN model we went with a 1-dimensional convolutional neural network architecture. A 1D CNN has been shown to work on forecasting time series-based data and has been used in stock price prediction in market data and human activity classification based on readings placed on sensors. Therefore, since our project proposes to predict the future stock prices of companies in one sector when events happen in another, we chose to go with a 1D CNN.

Motivation:

A CNN can generally find patterns in the data which can then be used to make more complex patterns in the later layers. However, we normally do not know what those patterns actually are since they can be very complex transformations. That is why, we detect the events beforehand so that we know where and when the anomalies occur in the data.

In a real-world scenario, potentially, if we detect events happening in one sector during one portion of the day, we could then make forecasts about what the stock prices in a different sector could be going forward. A CNN in this scenario can potentially detect patterns within those events and then be used to make forecasts.

1D Convolution:

A 1D CNN differs from a standard 2D CNN because of the way the filter or feature detector slides over the data.

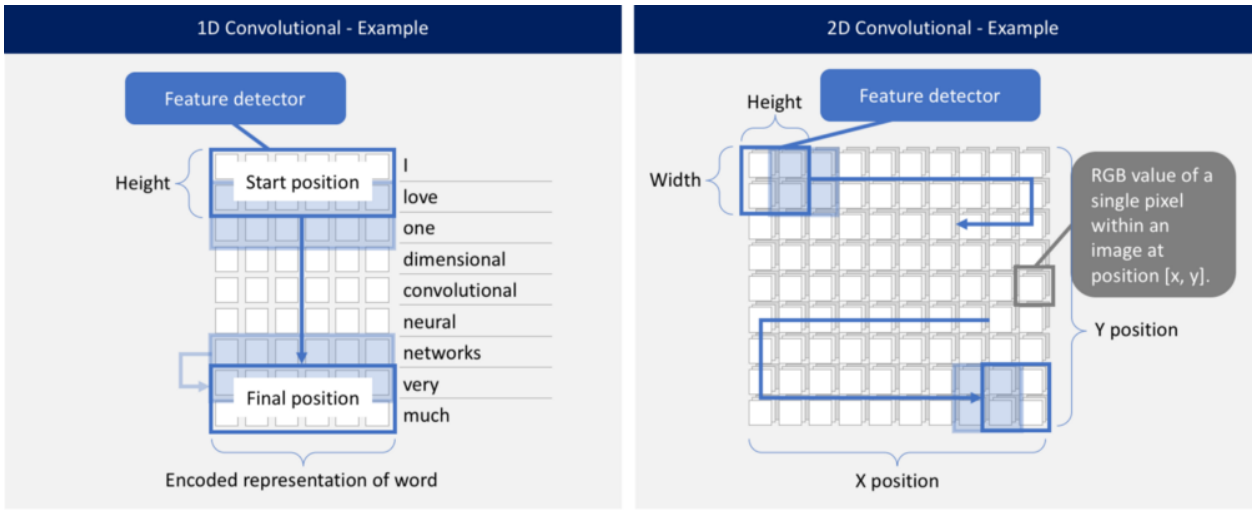

Figure x: 1D vs 2D CNN. © Nils Ackermann

Our dataset contains 5 predictor variables and 1 outcome variable which is the price of the stock some time steps after the event that has occurred. Since all the features for the event are detected at the same time, using a 1D convolution makes more sense.

We first split the dataset into *n* 5x5 matrices with *n* = math.floor(len(dataset)/5). We then apply our CNN over each of those *n* matrices.

8. Results

FFNN:

1 hidden Layer				Average MSE
	Healthcare	**Information Tech**	**Financial**	
Healthcare	-	0.045188446	0.111231424	
Information Tech	0.07301321	-	0.07551576	
Financial	0.08145144	0.052671816	-	
Average MSE				0.073178683
2 hidden Layers				
	Healthcare	**Information Tech**	**Financial**	
Healthcare	-	0.04452236	0.10946862	
Information Tech	0.07139059	-	0.07536054	
Financial	0.08247182	0.05251373	-	

Average MSE				0.072621277
3 Hidden Layers				
	Healthcare	**Information Tech**	**Financial**	
Healthcare	-	0.04506028	0.10944522	
Information Tech	0.07133754	-	0.07534434	
Financial	0.08195476	0.05239298	-	
Average MSE				0.072589187
4 Hidden Layers				
	Healthcare	**Information Tech**	**Financial**	
Healthcare	-	0.044688553	0.10964591	
Information Tech	0.07145021	-	0.07536031	
Financial	0.08200843	0.05250743	-	
Average MSE				0.072610141
5 hidden Layers				
	Healthcare	**Information Tech**	**Financial**	
Healthcare	-	0.044603225	0.10984366	
Information Tech	0.07191838	-	0.07587371	
Financial	0.08230654	0.052465796	-	
Average MSE				0.072835219

We observed that increasing number of epochs increased the MSE. 50 epochs has the optimal Average MSE.

		Healthcare	**Information Tech**	**Financial**	**Average MSE**
50 epochs	Healthcare		0.044533312	0.109489106	
	Information Tech	0.0716362		0.07531822	
	Financial	0.08185975	0.052437607		
Average MSE					0.072545699
100 epochs	Healthcare		0.04506028	0.10944522	
	Information Tech	0.07133754		0.07534434	
	Financial	0.08195476	0.05239298		

Average MSE					0.072589187
150 epochs	Healthcare		0.044779234	0.109271355	
	Information Tech	0.07141797		0.0757402	
	Financial	0.082236916	0.05248974		
Average MSE					0.072655903

CNN:

Architectures used and Training:

Base CNN Model Architecture (1 Convolution Layer):

Layer No.	**Layer Description**	**Parameters**
1.	1DConv	filters=16;kernel_size=2;activation=relu
2.	MaxPooling 1D	pool_size=2
3.	Flatten	
4.	Dense	size=10, activation=relu
5.	Output	y.shape[1] (normally =5)

Loss function= mean squared error or MSE.
Optimizer = Adam; n_epochs=20, batch_size= 4

1. The first layer is the convolution layer, with 16 filters and kernel size as 2.
   We slide through each 5x5 matrix 5-2+1 = 4 times to get 4 values out of each filter. As there are 16 filters we end with a 16x4 matrix after the convolution layer.
2. At the max pool layer with pool size=2 which means we end with 16x2 matrix since it gets the maximum of the 2 values over which it strides. Stops from overfitting the data.
3. Since the dataset is so small, we do not use dropout. We flatten the matrix to give us 16x2=32 values.
4. We then pass these values through 10 node fully connected layer.
5. We then pass these 10 values through the output layer of size 5 that outputs the 5 values for each matrix.

We use MSE as our loss function since we are using CNNs for a regression problem and we need the CNN to learn how to closely predict the prices. MSE is therefore a good statistic to minimize.

Base CNN	**Target Test MSE**		
Event Sectors	Healthcare	Information Tech	Financial
Healthcare		0.07165112	0.15533742
Information Tech	0.0751790		0.07534434
Financial	0.1361847	0.04337207	

2 Conv Layer CNN Model Architecture (1 Convolution Layer):

Layer No.	Layer Description	Parameters
1.	1DConv	filters=10;kernel_size=2;activation=relu
2.	1DConv	filters=20;kernel_size=2;activation=relu
3.	MaxPooling 1D	pool_size=1
4.	Flatten	
5.	Dense	size=10, activation=relu
6.	Output	y.shape[1] (normally =5)

Loss function= mean squared error or MSE.
Optimizer = Adam; n_epochs=20, batch_size= 4
We use MSE as our loss function since we are using CNNs for a regression problem and we need the CNN to learn how to closely predict the prices. MSE is therefore a good statistic to minimize.

Base CNN	**Target Test MSE**		
Event Sectors	Healthcare	Information Tech	Financial
Healthcare		0.061798430	0.0995535
Information Tech	0.0654841		0.0746830
Financial	0.0794793	0.049838350	

Final CNN:

Layer No.	Layer Description	Parameters
1.	1D Conv	filters=10; kernel_size=3; activation=relu
2.	1D Conv with dropout layer	filters=20;kernel_size=3;activation=relu;dropout=0.2
3.	1D Conv	filters=20;kernel_size=3;activation=relu
4.	Max Pool and Flatten	Pool size = 2
5.	Dense	size=128, activation=relu
6.	Output	y.shape[1] = 10

Each Matrix size changed from 5x5 to 10x5:
Loss function= mean squared error or MSE.
Optimizer = Adam; n_epochs=20, batch_size= 4

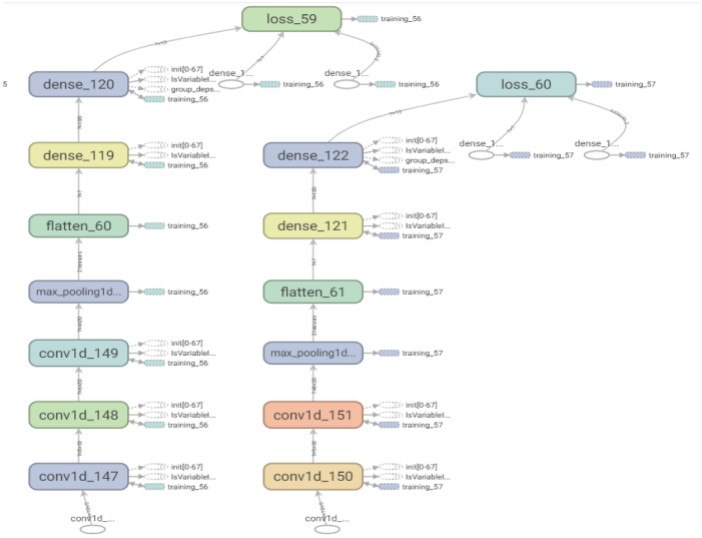

Figure 2: TensorBoard graph

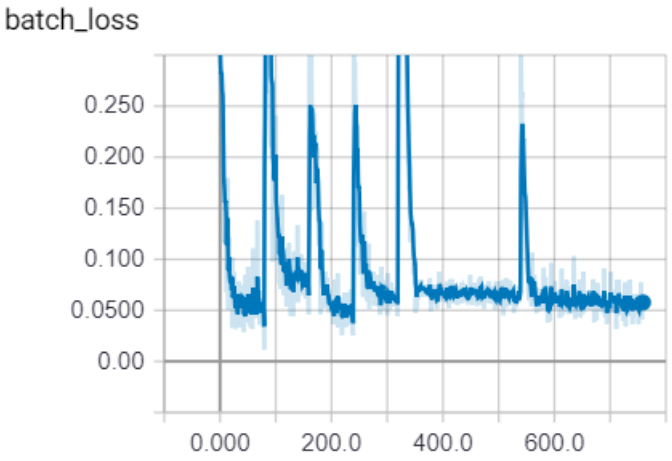

Figure 3:Batch Loss

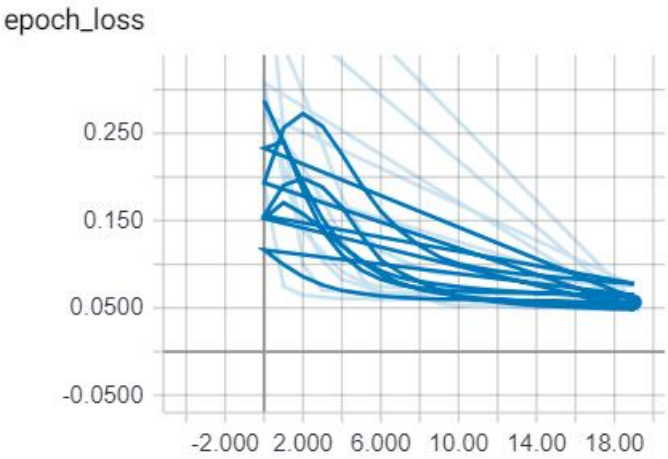

Figure 4:Epoch Loss

Final CNN	Target Test MSE		
Event Sectors	Healthcare	Information Tech	Financial
Healthcare		0.06446426	0.12121127
Information Tech	0.0758098		0.08117099
Financial	0.0660055	0.06218147	

CNN based on Fashion MNIST 2D CNN used in class assignment:

Layer No.	Layer Description	Parameters
1.	1D Conv	filters=32; kernel_size=3; activation=relu
2.	1D Conv	filters=64; kernel_size=3; activation=relu
3.	Max Pool and Flatten	Pool size = 2
4.	Dense with dropout	size=128, activation=relu, dropout=0.5
5.	Output	y.shape[1] (normally =5)

Test MSE based on Health Care Event and Financial Sector as Target = 0.05

9. Improvement

FFNN:

a. Regularization was not used for this network as adding the dropout regularization technique (tried with 2 different values for dropout rate) was resulting in higher MSE values. This could be because of very low dataset size. Hence regularization is not needed

	without dropout	with dropout = 0.3	with dropout=0.5
Information Technology - Health Care	0.07133754	0.075257294	0.073986016
Information Technology - Financials	0.07534434	0.08100667	0.08995011

b. Multiple variations of the size of layers were tested for the network. The combination that resulted in the lowest MSE was: Layer 1: 100 nodes, Layer 2: 100 nodes and Layer 3: 156 nodes. The number of layers with the optimal MSE value was 3 as can be seen in the table below (under optimization).

Layer 1	Layer 2	Layer 3	Average MSE
200	200	256	0.072589
100	100	156	0.072481
50	50	56	0.072569

c. The two optimization algorithms investigated were Stochastic Gradient Descent Optimizer and Adam Optimizer.

	Adam	SGD
1 Layer	0.07261	0.07317

2 Layers	0.07261	0.07262
3 Layers	0.07395	0.07258
4 Layers	0.07359	0.07261
5 Layers	0.07312	0.07283

A majority of the runs resulted in a lower average MSE value for SGD when compared to the Adam Optimizer.

CNN:

We use dropout regularization for training our final CNN. This improved the Test MSE for all CNN models except for the one for modelling Healthcare Events and Financial Targets. This means that the dropout stops the model from overfitting.

We tried different combinations based on number of convolution layers, input matrix size, number of filters and whether or not we used dropout.

S.No.	Model Description	Avg Test MSE
1.	1 Conv Layer Filters=10, Kernel Size=3, No Dropout Adam Optimizer	0.076427642
2.	**2 Conv Layers**; Kernel Zize=2; Filters=10,20; No dropout; Adam Optimizer	0.071806139
3.	Same model as 2. **10x5** matrix size	0.076972066
4.	**3 Conv Layers**; Kernel Size=2; Filters=10,20; **With dropout**; Adam Optimizer	0.0781630668
5.	Same model as 4; **SGD optimizer**	0.1053094289

Adding 1 more layer decreased the Test MSE since it is detecting more patterns in the events and hence will be able to better predict prices

Adding dropout increases the average CNN since we lose some data and as we are using less data than is standard for deep learning, it makes our prediction worse.

SGD Optimizer performs worse than Adam in this case, since again more data is needed for better optimization. Since Adam reaches optimum faster than SGD, training for more epochs would probably make the SGD optimizer based CNN work better. Since we train for 20 epochs with a batch size of 4 we are getting better performance out of Adam.

10. Future Work

The analysis presented in this report is done only on one company per sector. And only 3 sectors are selected. Changepoint detection is a computationally intensive operation especially

if the window size is set to higher values. Hence current window size is set to 50 and only limited companies and sectors are analyzed. We plan to extend this to entire dataset for final report. We expect the MSE values to improve a lot and represent the cross sector dependencies and their predictability more accurately once entire dataset is considered.

11. References

[1] Valery Guralnik, Jaideep Srivastava, Event Detection from Time Series Data", KDD '99 Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining - August 18, 1999

[2] M. Basseville, I.V. Nikiforov, Detection of abrupt changes: theory and application Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1993)

[3] Adams, R. P., & MacKay, D. J. C. (2007). Bayesian online changepoint detection. Technical Report. arXiv:0710.3742v1 [stat.ML]

[4] J. Takeuchi, K. YamanishiA unifying framework for detecting outliers and change points from non-stationary time series data, IEEE Transactions on Knowledge and Data Engineering, 18 (4) (2006), pp. 482-492

[5] Xiao Ding, Yue Zhang , Ting Liu , Junwen Duan, "Deep Learning for Event-Driven Stock Prediction", Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015), July 2015