CS5098: Deeplearning for Financial Investments

Contents

List of Figures

List of Tables

1. *Preface*

I. Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. The main word count of this project report is XX XXX words long, this including the all the sections of report including the Appendix. I give permission for it to be made available for use in accordance with the regulations of the University Library. Permission is also given, for the report to be made available on the public school intranet, permission is also given for this work to be used in research within the University of St Andrews, and for any software to be released on an open source basis. I retain the copyright in this work, and ownership of any resulting intellectual property.

II. Ethics

This projects includes free available information from the internet, free existing datasets used for academic research. The result of this projects are not available publicly and will be kept in the department of Computer Science department at the University of St Andrews. Above all, this project does not raise any ethical considerations or risk.

III. Abstract

2. *Introduction*

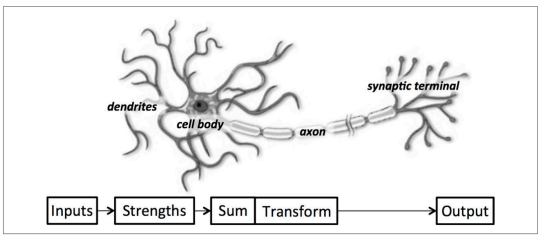
3. *Theory*

I. Neural Networks

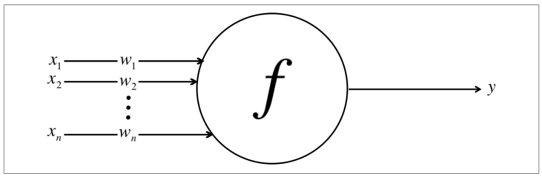
Neurons are the fundamentals units of the human brain. The brain consist of interconnected neurons though synapses, this structure is what we call a neural network. This phenomena is what enables us to experience everything around us. Biological fig. 3.1a neurons receives its inputs from dendrites, and the strength of each connection determines the contribution of the inputs to how stimulated the neuron would be to produce an output. Biological neurons are optimised to receive information from other neurons in the network. These neurons process information in their unique ways and then sends their results to other neurons in the network for further processing.

The artificial neural network (ANN) fig. 3.1b emerged from this phenomenom, where researchers goal was to create systems similar to the biological neural network to perform similar task. The first approach was pioneered by [McCulloch and Pitts, 1988] in 1943. Similar to the biological neurons the inputs of an ANN are represented as vectors $x_1, x_2, ..., x_n$ and this are connected to a neuron by a set of weights $w_1, w_2, ..., w_n$ and for each neuron the values connected to this neurons are summed up to form *logits* $z = \sum_{i=0}^{n} w_i x_i$ and this value in conjunction with the bias b is then passed through an activation function $y = f(z)$ and this gets sent to other neurons. This fundamental phenomena further extends to forming more complex networks such as convolutional and recurrent networks. This topics are discussed in depth further in this paper.

The formal way of representing an artificial neural network is $y = f(\sum w \cdot x + b)$



(a) *Biological Neural Network*

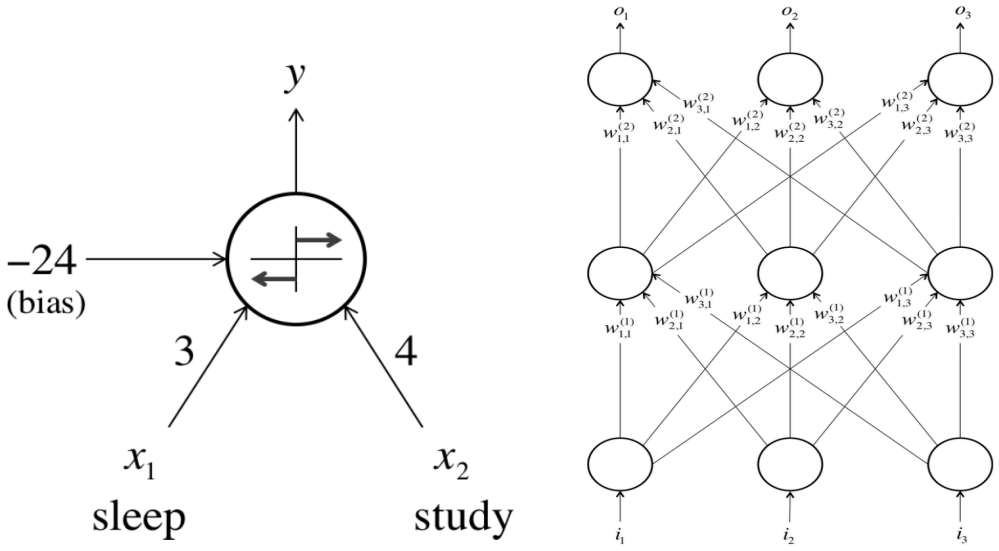


(b) *Artificial Neural Network*

I.I Feed forward networks & Deep Networks

The basic neural network is just a simple network with the input going straight to the output as illustrated here fig. 3.2a But in order to solve more complicated problems such as image classification, image recognition, hand written digit recognition, it is impossible for a single neural network to learn how to classify between different images. The human brain solves this problem because consists of a lot of neurons organised in layers [Mountcastle, 1957]. Information flows from the input layers I_i for example the human eyes into the internal layers called hidden units and this step propagates accross the network to the output layers denoted by O_i. Based on this concept, the idea of feed forward networks

fig. 3.2b were developed to learn complex representations of the input data set. Deep learning can be summarised as a feed forward network that has more then one hidden layer. This deeper layers go further to learning higher representations of the input information.



(a) *Simple Neural Network* (b) *Feed Forward Neural Network*

When neural networks were developed they had several limitations, this was because they could only perform linear predictions this means they just understand the data that has been inputed into the network, they cannot learn complex relationships between different items. Non linearity was introduced to solve this problem. This introduced non-linearity to the network, makes the network learn higher order representations and interaction of different input values.

I.II Activation Functions

Activations functions fig. 3.3 used in a neural network introduces nonlinearity. The 3 main activation functions used are sigmoid eq. (I.1), hyperbolic tangent eq. (I.2) and ReLU eq. (I.3). The sigmoid function is the most used. The sigmoid would output values close to 0 if the values of the logits are relatively small and when the logits are relatively large. The hyperbolic tangent values ranges between -1 and 1.

$$f(z) = \frac{1}{1 + e^{-z}} \tag{I.1}$$

$$f(z) = tanh(z) \tag{I.2}$$

$$f(z) = max(0, z) \tag{I.3}$$

Another common function used in neural machine learning is called the softmax function. This is used to represent a set output vectors as a probability distribution over a set of labels. In a classification problem, this is used to represent the probability of how likely our classes are given a set of input values. The higher the probability the more likely the class.

$$y_{softmax_i} = \frac{\exp^{z_i}}{\sum_j \exp^{z_j}} \tag{I.4}$$

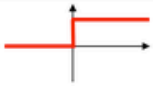
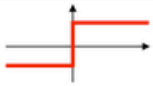
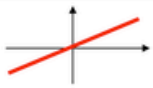
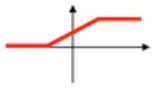
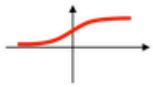
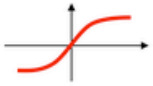
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer NN	

Figure 3.3: *Activation Functions*

I.III Backpropagation

Neural networks learns uses a process called backpropagation. This was pioneered by David Rummelhart and Geoffery Hinton in 1986 [Rumelhart et al., 1988]. The idea of back propagation is to learn how fast the errors changes with respect to the weights of an individual connection. Back propagation learns by trying to minimise the errors provided by the outputs with respect to the weights from the hidden layers and it uses the method of steepest gradient descent. This method can be imagined as a ball rolling down the hill. Back propagation uses the derivative of the errors with respect to the weights of each neuron to modify the weights connected to the output layers all the way back to the weights connected to the input layers. Back propagation uses the chain rule derivatives of the outputs weights and the hidden neurons to modify the weights of the network. The deeper the network, the more chain derivatives multiplications has to be done.

Other method of gradient descent includes stochastic gradient descent which is commonly used, this randomises which weights to modify during training time. The errors are determined by several loss functions which includes but not limited to the least mean squared method for regression problems, cross entropy for classification problems, max marginal and other variations. In this paper the cost function explored includes the mean squared error eq. (I.5) and cross entropy eq. (I.6).

$$E = \frac{1}{2}(y - \hat{y})^2 \tag{I.5}$$

$$H(p, q) = -\sum_{x} p(x) \log_2 q(x) \tag{I.6}$$

We can formally define the equation for back-propagation chain rule to obtain the partial derivative of the error E with respect to weight w_{ij} for a neural network with a single hidden layer. o denotes the output of the hidden layer and net_j denotes the logits of the hidden layer and w_{ij} represent the weights.

$$\frac{\delta E}{\delta W_{ij}} = \frac{\delta E}{\delta o_j} \frac{\delta o_j}{\delta net_j} \frac{\delta net_j}{\delta w_{ij}} \tag{I.7}$$

I.III.1 Problems with back propagation

Neural networks suffers from 2 major problems during back-propagation, this includes vanishing and the exploding gradient problem. The vanishing gradient problem occurs when applying the chain rule to deeper layers, the gradient become so small they do not make any change to the lower layers making them not learn anything. This topic is further explored in section V. The exploding gradient problem is the inverse of the vanishing gradient problem, this is where the gradients gets exponentially gets large. This is commonly caused by having un-normalised values in the network.

I.IV Convolutional Neural Networks

Convolutional neural networks have made a lot of advances in computer vision from image classification [Krizhevsky et al., 2012], to image recognition [He et al., 2015] and image captioning [Donahue et al., 2014] and much more. Convolutional neural networks evolved from studying the brain's visual cortex. In 1958 David Hubel and Torsen Wisel [Hubel, 1959] performed a series of experiments on cats and they uncovered the structure of the visual cortex. Their work shows that many neurons in the visual cortex only react to visual stimuli located in a limited region of the receptive field. The research also shows that the visual cortex is made up of hierarchical layers, where the lower layers learn low level representations of the image (lines, basic shapes) and the higher layers learn complex representations (style, texture) that comes from lower level representations.

The main idea behind convolution neural networks is that they are simply neural networks that uses convolution instead of general matrix multiplication in at least one of the layers [Goodfellow et al., 2016, p. 274]. Convolutional neural networks consists of a series of concepts, this includes Convolution as mentioned earlier, kernels which in image processing refers to filters E.g. Sobel edge detector, max pooling and last but not least fully connected layer. Convolutional neural network can be formalised as Convolution, Linear Recification and Max pooling.

In the case of this paper we will refer to a image data (2d image) as a 2d tensor. Convolution fig. 3.5 is just a weight average of a series of data. In the case of convolutional networks it is a dot multiplication of the section of the data (2d tensor) and the kernel, the area of the average section of the tensor is donated by the kernel size. The second stage fig. 3.6a consists of the activation which the convolution is passed through. The ReLU activation function is commonly used, the output from the ReLU function is then sent to the pooling stage. Pooling functions fig. 3.6b returns a summary of nearby outputs [Goodfellow et al., 2016, p. 282]. Max pooling returns the maximum of a rectangular neighborhood and average pool returns the average. The output of this is what we know as a convolutional layer. A diagram showing this process is shown fig. 3.7

I.V Recurrent Neural Networks

Recurrent neural network (RNN) like convolutional network has made major advances in artificial intelligence, this includes problems like Named entity recognition [Lample et al., 2016], Machine translation [Amodei et al., 2015]. Recurrent in comparision to deep feed forward networks contains feedback loops. The whole processes can be imagined as a series of time, where the hidden values from the previous time step is passed on to the next, and this process persists all the way the end of the time series fig. 3.8. This concept can then be further used for solving several problems this includes mapping a singular input to a series of outputs can be used for captioning images. Mapping an input sequence to a single output can be used for document classification. Mapping a sequence of inputs to a sequence of outputs can be used for machine translation. Vanilla RNN suffers from a problem called the vanishing gradient problem. This was briefly discussed in section III, where as the time steps increases in order for the network to learn anything, it has to apply a long chain derivative multiplication. This is a problem as

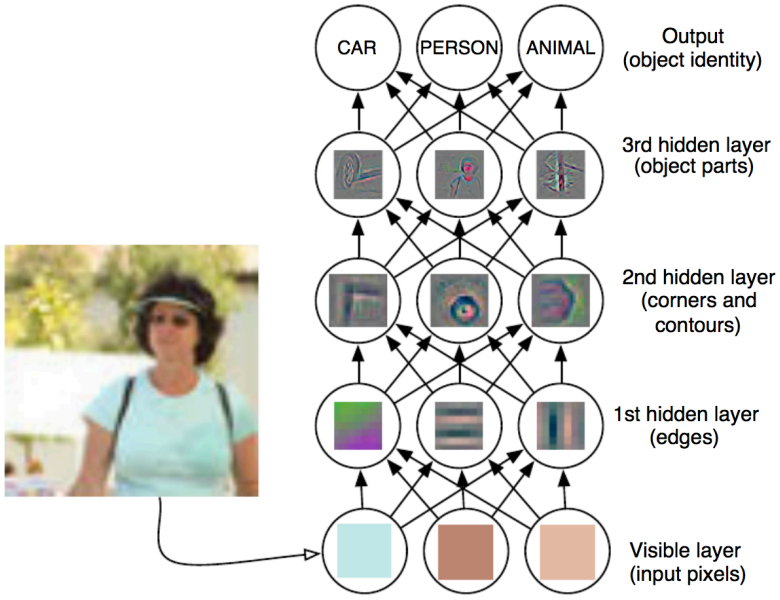


Figure 3.4: *Convolutional Neural Network*

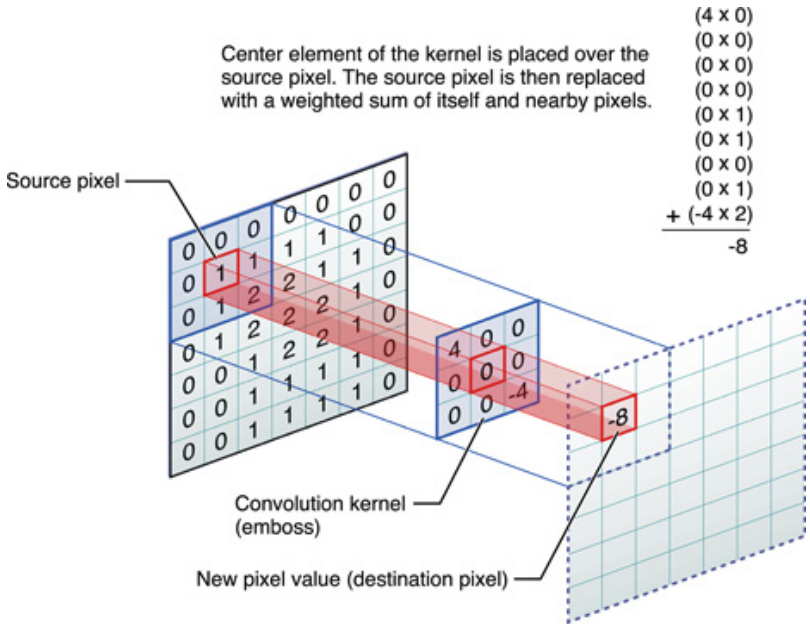
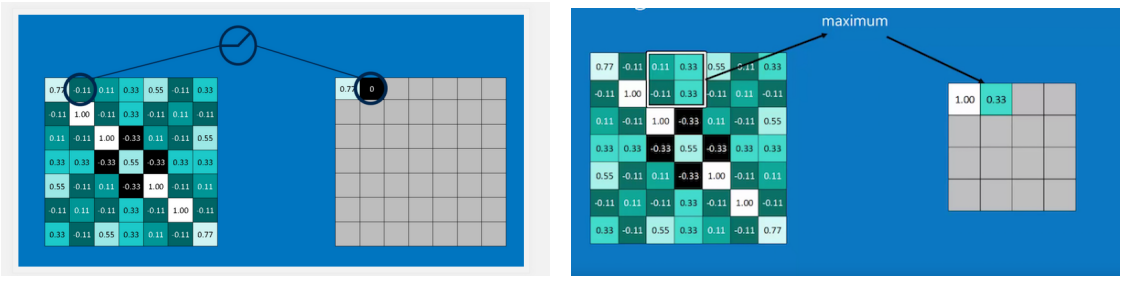


Figure 3.5: *Convolution*



(a) *Linear Rectifier* (b) *Max pooling*

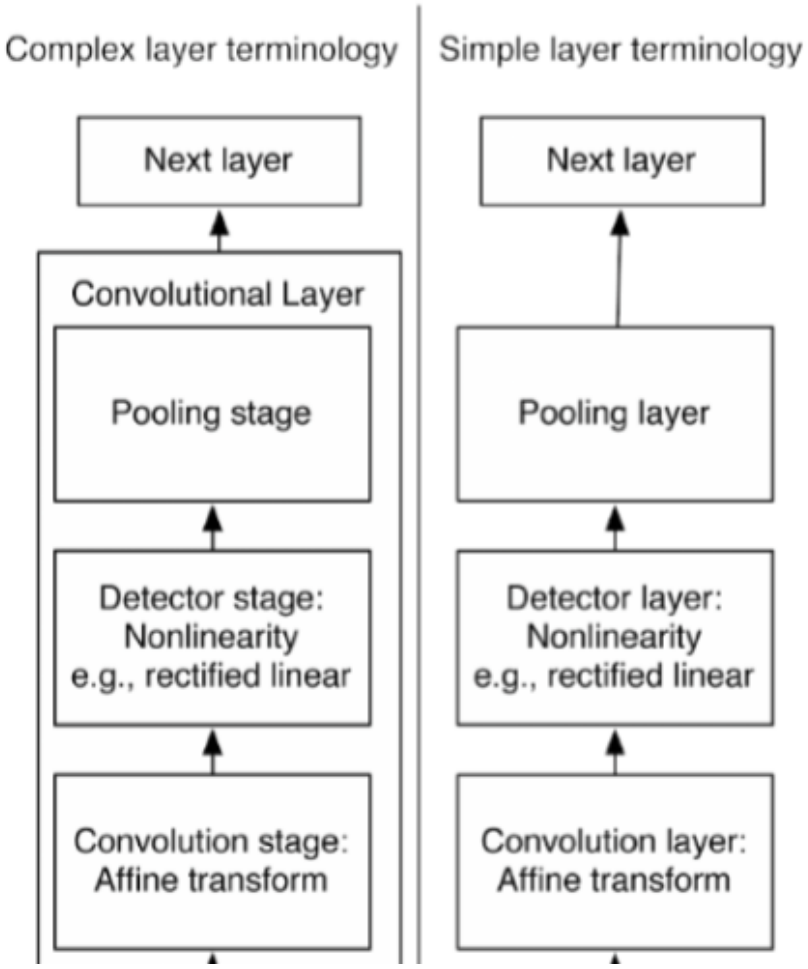


Figure 3.7: *Convolutional Layer*

multiplying small values would yield to the value approaching 0 at a rate proportional to the number of time steps. In order to solve this problem LSTM networks and GRUs were developed. In this paper we would only explore LSTM networks, information on GRU's can be found here [Chung et al., 2014].

Vanilla RNN's can be formalised by the following equations, where, x represents the inputs, W represents the weights and h representing the hidden layer layer, t represents the time step and σ representing the activation function in this case sigmoid.

$$y_t = \sigma(h_t) \tag{I.8}$$

$$h_t = \sigma(W^{hh} h_{t-1} + W^{hx} x_t) \tag{I.9}$$

$$x_t = \{x_{i_t}, ..., x_{N_t}\} \tag{I.10}$$

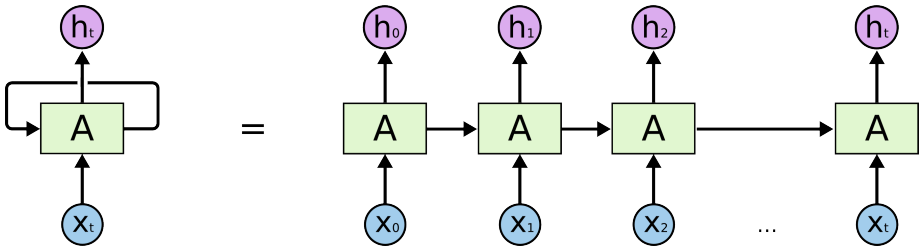


Figure 3.8: *Recurrent Neural Network*

I.V.1 LSTM Networks

Long short term memory or LSTM [Hochreiter and Schmidhuber, 1997] was a mechanism developed to solve the vanishing gradient problem[1]. The main idea behind LSTM mm's is to be able to transmit important information into the future. This process is achieved by the hidden layers having a series of gates that contributes how much effect the past and current contribute towards future units. The LSTM fig. 3.9 consists of several components or gates this includes

1. Forget gate: this gate decides how much information should be forgotten.

$$f_t = \sigma(W_f \cdot [h_t - 1, x_t] + b_f)$$

2. Input gate: This value is used to determine how important the current input is.

$$i = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\widetilde{C} = tanh(W_{\widetilde{C}} \cdot [h_{t-1}, x_t] + b_{\widetilde{c}})$$

3. Memory cell: This is one of the most important component as this hold important information that has been learnt over time.

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}$$

4. Output gate: this determines how much of the cell is exposed

$$o = \sigma(W_o[h_{t_1}, x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$

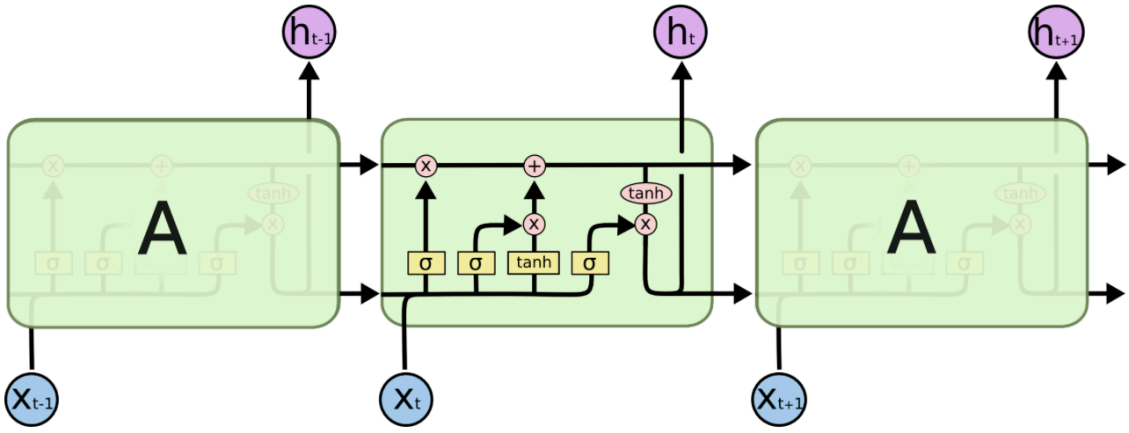


Figure 3.9: *Long Short Term Memory Network*

More complex variations of RNN's have proven to give good results in terms of machine translation [Yao and Huang, 2016] and other sequence to sequence task. This is the process of having two different RNN network, where one does a forward pass and the other performs a backward pass and the results are concatenated and then used for prediction.

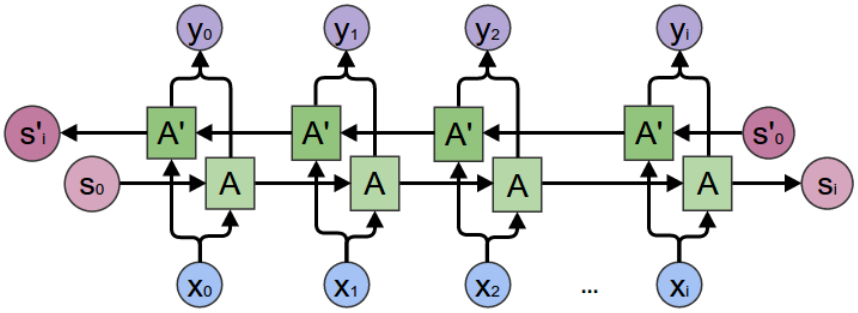


Figure 3.10: *Bidirectional Neural Network*

I.VI Optimisation

Optimisation of deep neural networks has been a big field machine learning, some of this techniques includes but not limited to creating variations of gradient descent optimisers such as Adam [Kingma and Ba, 2014], Adagrad [Duchi et al., 2010] which builds from stocastic gradient descent, RMSprop which improves up on Adagrad [Hinton et al., 2012] [Goodfellow et al., 2016, p. 284] and Momentum optimizers. Other optimisation methods are discussed below

I.VI.1 Batch Normalisation

Batch Normalisation [Ioffe and Szegedy, 2015] was introduced by S. Ioffe and C.Szegedy to address the vanishing and expoding gradient problem. They noticed the distribution of each layers inputs changes during training time as the values of the previous layers changes. They proposed a solution by adding an operation before the activation of each layer, which essentially scales and shifts the results using two parameters. The model will eventually learn the optimal values for these parameters during training time.

[1]This should not be confused with a layer of hidden units, this is not a layer of hidden units instead it is a mechanism of how each individual hidden neuron behaves.

I.VI.2 Dropout

Dropout [Srivastava et al., 2014] is a simple but effective technique used for preventing overfitting in deep learning. The algorithm essentially sets using a probability p the input values to 0. This gets fed into the neural network during training time, and this is removed during test time. This is illustrated by the diagram section VI.2

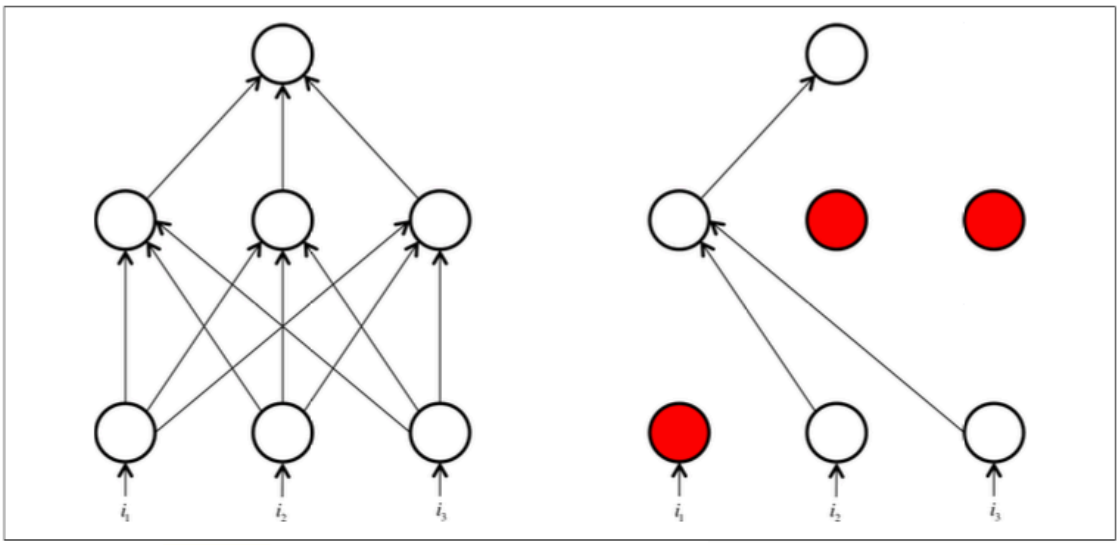


Figure 3.11: *Dropout*

I.VI.3 Regularisation

Another method for combatting overfitting is a method called regularisation. Regularisation modifies the cost function we try to minimise by appending additional terms that penalises large weights ϕ. Our new cost function then becomes eq. (I.11). As $f(\phi)it$ gets bigger as the values of ϕ gets larger and the regularization strength λ is another hyper parameter that can be modified. The most used form of regularisation is the $l2$ regularisation [Krogh and Hertz, 1991], this regularisation techniques specialises in penalising heavier weights vectors while preferring diffused weight vectors. Regularisation can be visualised fig. 3.12. Another common used regularisation technique is the $l1$ regularisation technique. This technique is used to increase the sparsity of the weight vectors. This is very useful in understanding what features are contributing to the networks decision. Other variations includes Max norm [Srivastava et al., 2014] which is another regularisation method that restricts the weights from getting too large.

$$loss = Error + \lambda f(\phi) \tag{I.11}$$

II. Representation Learning

This chapter aims to motivate the understanding for developing data representations and to explore a variety of techniques that are currently available. The main reason for data representation is to be able to represent and compress the high volume of data into meaningful representations that are well suited for the machines to learn. Representation learning is mostly used in cases where large amounts of unlabelled

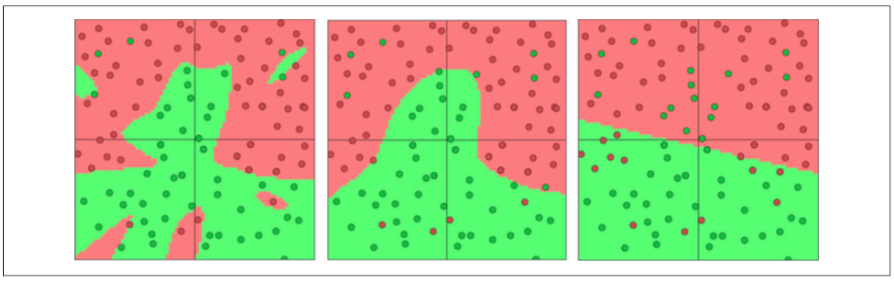


Figure 3.12: *Shows applications of different regularisation strengths (0.01, 0.1 and 1)* [Sutskever et al., 2014]

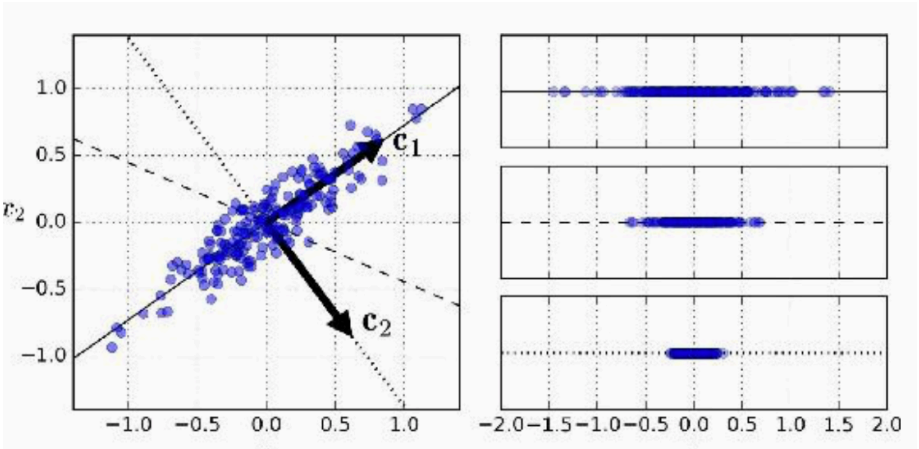


Figure 3.13: *PCA: Right images shows decending variance from top to bottom*

data. In terms of natural language processing, this processed is done by learning word embeddings or low dimensional word representations in terms of computer vision deconvolution [Noh et al., 2015] is a process used to learn image representations.

II.I Principal Component Analysis

Principal component analysis [Pearson, 1901] is a method for performing dimensional reduction of high vectors $n < d$ where n represents the lower dimensional vector, and d the dimension of the original data. It is a process used in finding a set of lower dimensional axes that conveys the most information about our current dataset. As shown in the image fig. 3.13 , the aim is to find an axis or dimension in the high dimensional space that preserves the maximum variance, this axis would be our first principal component. Afterwards it then proceeds to find the second axis orthogonal to the first one that accounts for the remaining variance, it then finds a third axis that is orthogonal to both axis and this process is repeated to the dimension length d of the dataset.

II.II Word Representations

II.II.1 Word2Vec

Word2Vec [Mikolov et al., 2013] is a neural learning frame work for learning word embeddings. It was developed by Tomas Mikolov in 2012. The process consists of representing words as vectors in some dimensional space and this vectors can be further used for performing several tasks that includes machine translation [Wolf et al., 2014] and sentiment analysis [Liu, 2017]. The paper proposes 2 dif-

ferent strategies for generating word embeddings this includes Continious bag of words (CBOW) and the Skip-gram model. Unlike other methods that uses concurrent counts. Word2Vec takes a different approach, the CBOW tries to predict a target word based on the context word while the skip-gram model performs the inverse of the CBOW method as it tries to predict a context word based on the target word. The main aim of Word2Vec is to learn similarities of words based on the context they are in. As illustrated in the diagram. We can see that the word "jumps" is similar to "leaps" as they appear in the same context.

From the word2vec paper after learning word representations, they emphasise this point by visualising a lower dimensional representations of words in the same context appears closer together. This includes countries and their capitals. The famous example is the "man is to woman as king is to queen".

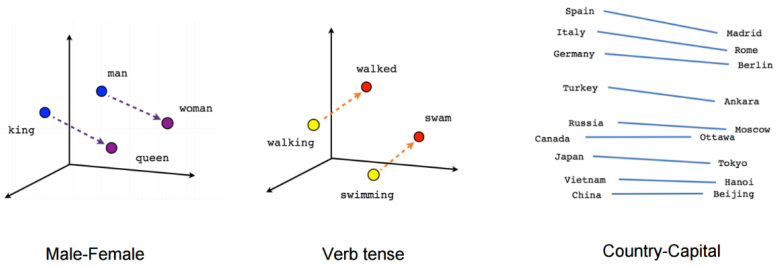


Figure 3.14: *Word2Vec Representation*

To formalise what Word2Vec is doing, each word is represented as a one hot encoding word of the vocabulary size $|V|$ fig. 3.16. This simply means that each word represented by a sparse matrix with a single index denoting where the word occurs. The aim is for the word to predict a context word in the skip-gram model which is another one hot encoding using a multilayer perceptron or feed forward network with one hidden layer. The aim is to learn weight d representations of each word in the vocab. The Word2Vec framework uses the NCE sampled loss function with cosine similarity to minimise its loss (section III.1). With this words that appears in the same context cluster together. The learned weights d are then used as vector representation for each word.

```
Brown fox jumps over the dog          Brown fox leaps over the dog
The boy jumps over the fence            The boy leaps over the fence
The man jumps over the pothole        The man leaps over the pothole
The rabbit jumps over the tortoise  The rabbit leaps over the tortoise
The company jumps over the hurdle  The company leaps over the hurdle
```

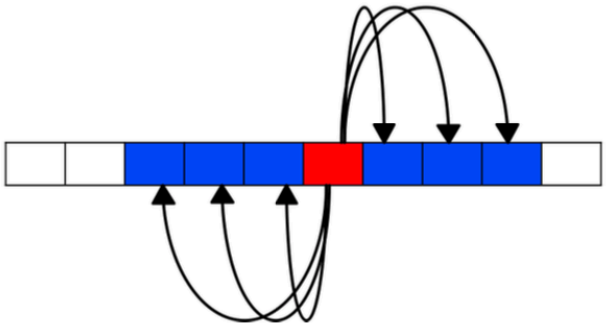


Figure 3.15: *Word2Vec showing the similarity between the word leaps and jump.*

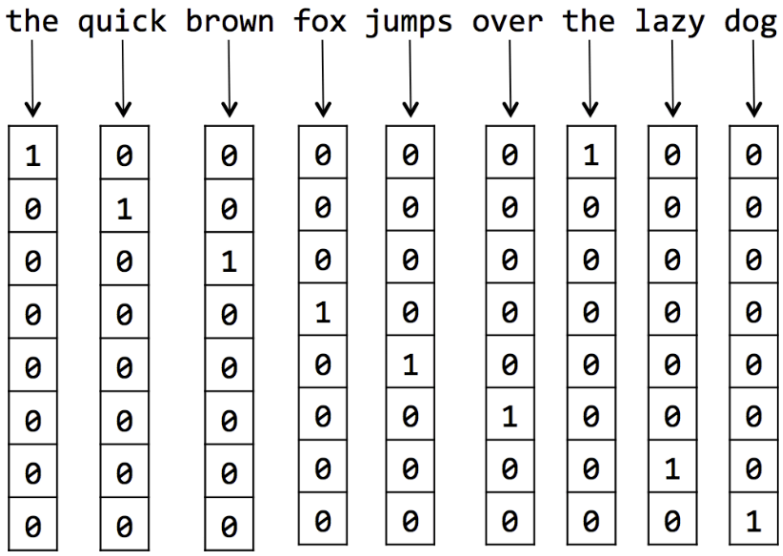


Figure 3.16: *One hot encoding representation*

II.II.2 GloVe

Gloval vectors or GloVe [Pennington et al., 2014] is a count based approach like LSA [Deerwester et al., 1990] which utilises concurrent statical information (counting what word appears before another word), and then perform dimensional reduction using matrix factorisation techniques like PCA section I. Unlike other methods like LSA, glove applies a log-smoothing transformation to the counts before performing dimensional reduction.

II.III Autoencoders

Autoencoders [Rumelhart et al., 1986] are deep learning strategies for performing unsupervised and transfer learning. Unlike PCA, auto encoders are simple circuits which aim to learn input representation with a minimal amount of distortion. Autoencoders are not only useful for learning feature representation of the input data, they can also be generalized to create new data from the trained data distribution [Kingma and Welling, 2014]. Autoencoders works by simply learning the same input data shown in fig. 3.17.

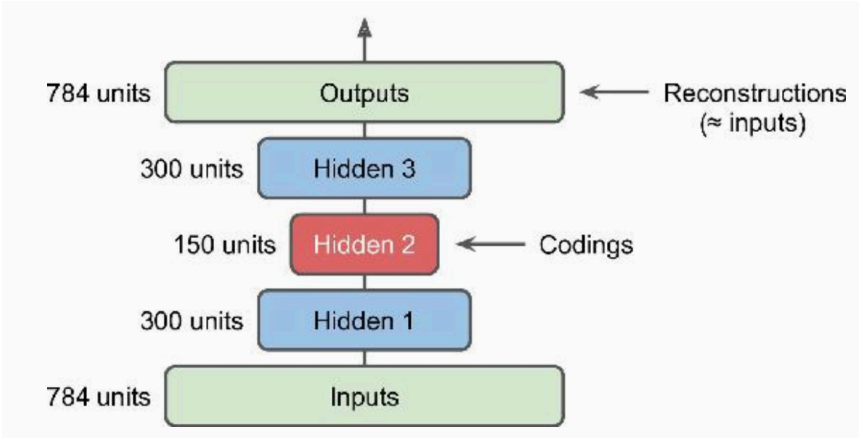


Figure 3.17: *Deep feed forward autoencoder*

II.III.1 Sequence to Sequence Encoding

Sequence to sequence learning [Sutskever et al., 2014] is a machine learning approach that is used to map a series of input sequence to a series of output sequence. This architecture is used a lot in Machine translation [Neubig, 2017] E.g. English to french. To elaborate this point further, shown in the picture fig. 3.18, this architecture can be used to model a chatbot that can provide responses based on the input sequence. It receives as inputs the sequence "How are you <EOL>" and the program will respond with "I am fine <EOL>" where <EOL> denotes the end of line.

This architecture consists of two parts, an encoder, and a decoder. The encoder is reponsible for taking the input sequence and encoding it to a lower-dimensional representation. The decoder is responsible for taking this encoded representation and prodicing a series of outputs This architecture uses word representation mentioned in section II this includes the word2vec, or the GloVe model as word embeddings. It takes this word representations and passes them into the a recurrent neural network as inputs, and then takes the encoded point which represents the last hidden unit of the encoded sequence

In our model, this architecture was used for encoding the headlines into a lower dimensional representation. This was motivated by [Kiros et al., 2015], where the same principle of encoding word vectors was used in several tasks such as semantic relatedness, paraphrase detection, image-sentence ranking. In our model the inputs and output was represented as the headline. The aim was to take a lower representation of the sentence sequence. This was done by taking the encoder hidden units vectors as the summarisation of the input sequence.

For the first stage of encoding the headline into a point, the [Dyer, 2014] Noise contrastive estimation (NCE) cost function was used, as the sequence to sequence model performs a classification operation over the vocabulary size, which is very large. Using the softmax function for computing the probability distribution over the outputs this will lead to exponential computational time.

The NCE loss function uses a binary logistic regression function to compare the embeddings of the target word with the embedding of the context word and random sampled non-context words. The probability of the non context words are summed up and subtracted from the probabilities of the context word comparison. The aim is to minimise the result attained from this operation.

For the second stage, this process was repeated into minimising the distance of the encoded headline from it self, hence performing another encoding operation. The loss function used in this case was the least mean squared method.

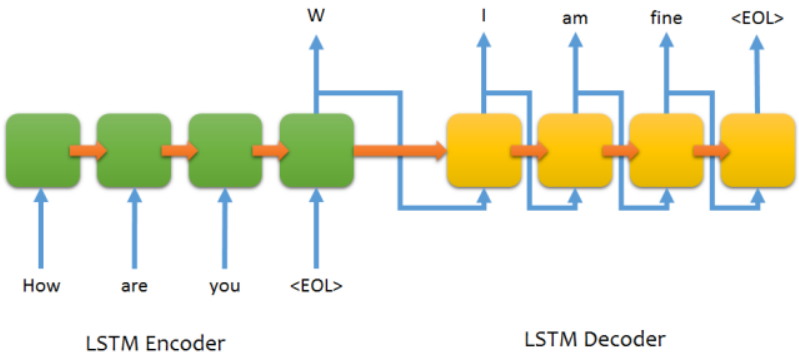


Figure 3.18: *Sequence to Sequence Autoencoder*

4. *Requirement Specification*

I. Aims and Objectives

Rather than try every neural architecture imaginable, we shall try and mimic best practices applied by investors daily. Hence, in the same way that convolutional neural networks (CNN) mimic our own visual cortexes, our neural architecture shall mimic the discovery process followed by investors. Since we are focusing on technicals, as well as fundamentals appearing in textual form, we must attempt to find an architecture that looks for price chart patterns similarly as a day traded, and fundamentals as both an arbitrager and a longer term investor. Finally we must identify a method to combine these in an efficient way. For example replicating architecture of [Ngiam et al., 2011].

Taking these in step, day traders (not considering event arbitragers yet) attempt to gauge the direction of the herd and profit from it. They are in affect comparable to cattle herders. In trying to fulfil there aim they primarily concentrate on identifying patters in price charts, looking for $patterns$ appearing in conjunction with the passage of $time$. Elaborating, a day trader may look for support and resistance lines (bands in which prices move), or heads and shoulders [Investopedia, 2017] (a famous technical patter that some investors swear by), and make a trade based on his belief of the likelihood of this pattern continuing tomorrow, in the next hour or etc. The difficulty in mimicking this kind of analysis is that neither CCN - which look for patterns and make decisions irrespective of where (time t or t-10) they occur - nor recurrent neural networks - which try and analyse the cumulative effects of the passage of time, or addition of variables - on their own replicate this process completely. Ideally we would like an architecture that allows us to capture both the presence of patterns and the passage of time. Consequently, we choose to input price and volume data into two parts of our network - one CNN network, in conjunction with multiday 'headlines vectors', and one long short term memory RNN (LSTM). (see architecture section for more details)

Other option: Consequently, we choose to implement a modern max pooling RNN [Lai et al., 2015] when considering price and volume data (see architecture section for more details).

5. *Design*

I. Technical Analysis and Informaiton

Technical analysis(TA) [Lo et al., 2000], is one of the oldest investment appraisal technique used to beat the markets. It allows traders to evaluate the "breadth and duration of price trends" providing evidence for imminent movements before fundamental analysis [Abarbanell and Bushee, 1997]. Whilst many papers have been published rebuffing the claims of technical analysts, perhaps most prominently a paper by Fama and French [Fama and French, 1993] [Fama and French, 2007] [Fama and French, 1997], numerous other papers have been published providing support for the technique. Regardless, we set out to try and use price and volume data to distinguish between buy, sell, and hold points in our n dimensional field.

Traditional technical analysis, as you have discovered, focuses on using price and volume data as inputs in signal identifying functions. Elaborating, individual price data can be passed through functions such as the Moving Average Convergence Divergence function (better known as MACD) or Relative Strength Index function (RSI). These try and extract important trading signals from the seemingly random noise that is a price chart, allowing investors' to identify immediate trading signals (eg: when RSI dips above/below 70/30). These functions have numerous hyper-parameters (eg. length of input data and etc), which are usually optimised over the training set. The problem with these techniques is that they tend to overfit on the training set, and rarely perform well over the test set. This is because no actual trading signal is identified in the process - the technique identifies esoteric relationships that would have worked in the past, however no real method for separating out noise is found.

A plethora of other techniques exists, but rather than waste time outlining these I move on to our goal. Similarly to technical analysts we are taking price and volume data as inputs. The more information we use (adjusted closing price, open and close spread, volume, and etc - data downloaded from yahoo finance) the better our network should be able to identify clusters using technical information. However there is a difference between using more useful information, and using simplified, and possibly completely irrelevant, information. Expanding upon this point, using RSI or any other technical indicator is pointless as: all such information is present in the price chart already, the process is not supported by theory (no theory other than pure data mining), and it relies enormously on blindly setting hyper-parameters (data mining of training set with no link to test set). Whilst the inherent amount of noise and lack of data may make our task of identifying clusters using raw price and volume data impossible using a neural network, methods (supported by theory) for efficiently extracting information from price and volume data exist. These are the methods we shall focus on.

Before continuing, below we describe two observations linked to a so called momentum effect [Freyd and Finke, 1985] that researchers have tried to explain using risk factor theory: the principle that risk should be compensated by reward. The medium to long term momentum effect, in simple terms, notes that securities that have moved up in value more than the market average are likely to continue moving up in value, and those that have declined in value are more likely to continue declining in value[1].

[1] Requires observing a medium to long term horizon of returns, and typically continues for a period longer than a month.

The short term momentum effect [2] on the contrary notes that companies that outperform the market today (in the short term) tend to underperform tomorrow (in the short term). The reason that we spent space explaining the above, is that many researchers attribute the success of technical analysis to these phenomena. Moreover, an understanding of the above allows us to determine the data preprocessing required to capture real persistent trading signals without having to worry too much about optimising hyper parameters [3]. We refer to [Ding et al.,] summarised in our outline that described the use of neural networks for effective 'technical' signal capturing. Per the paper mentioned, treating returns as panel data (both cross sectional and time series) allows us to capture an important trading signal. Elaborating, in every time period we can calculate the average and standard deviation daily/monthly return for all the companies being considered, and then use these to convert the returns to z-scores. Our model input would then consists of z-scores instead of returns. Similarly, to capture things such as the liquidity premium, we could calculate z-scores for volume as well. The above mentioned are two ways of 'actually extending' the data; unlike MA cross for instance.

To sum up, the data we will attain for the technicals includes: z-score for price/return and volume nominal volume (potentially allows for better individual security pattern spotting - heard behaviour) adjusted close and adjusted open (adjusted open calculation outlined previously or just open close difference) Additionally, if we focus on next day return we shall only have to focus on 20 days of data as we shall be capturing the short term momentum effect. Otherwise we shall focus on daily and monthly returns like [Ding et al.,].

Moving to another form of short term trading, event arbitragers traditionally focus on reacting to simple (requiring little data to understand) events that create momentary arbitrage opportunities [4]. An example of this may be someone shorting a stock after it announces its intent to acquire another company; on average companies tend to overpay. Whilst, due to the industry wide automation of event trading, and simple analysis required to spot such opportunities, the trading signal strength of these strategies is short (an optimistic estimate would be one to two days max), we nevertheless add tools to our architecture potentially capture such signals. We follow the approach set out by Ding et al. (2014, 2015) - see summary papers 1 and 2 - utilising gaited recurrent nets (GRN's) [5].

Finally, considering fundamental investors, traditionally these focus on identifying the long term fair price using non trivial historic data. Whilst prices may fluctuate in the short term due to excess buying or selling, in the long run they should, the theory goes, encapsulate the securities fundamental value. Taking an everyday example, whilst the price of sneakers may fluctuate due to seasonal or fashion linked reasons in the short term, in the long therm their price is linked to the companies manufacturing and marketing costs. Whilst as part of this dissertation we do not attempt to calculate a stock's fundamental value ourselves, due to the complexity of performing such analysis, we nevertheless try to capture the thinking involved in fundamental analysis, by analysing the interactions between everyday headlines. Specifically, we try and capture long term company strategic trends, thus improving the accuracy of our event based analysis. As such a task involves looking for patterns in data, we use CNNs for the task.

Overall, to summarise, we attempt to determine the (specify time frame here) short and medium direction - buy, sell hold (within threshold) - of stock price movements. Due to the complexities of this task we try and capture the trading signals arising from multiple analysis methods. As these signals are not mutually exclusive, we use a multistage neural network architecture involving RNNs, CNNs, and fully connected neural networks, to efficiently capture the inputs interactions, thus maximising our

[2] better linked to the liquidity premium

[3] Whilst cross validation would still help identifying the optimal return horizon and etc, at least we have theoretical backing for our choice

[4] Our use of the word arbitrage here is not entirely accurate, as events do not always move prices in the same way; in the kind of event analysis we are considering some risk is present

[5] Form of advanced RNN. Nikhil Buduma [Buduma and Locascio, 2017] notes that GRU's perform equally to LSTMs; however, they require fewer parameters to do so.

prediction accuracy over both time frames.

II. Whole Architecture

With roughly 200k headlines providing roughly 40k training instances we have to use the training data efficiently to accomplish our aim. In this endeavour we have developed a a modular architecture/approach that allows the reuse of trained weights, and furthermore the use of the unsupervised learning encoder decoder approach [6]. We start by describing our handling of an individual headline, before moving on to the analysis of multiple headlines in one day, and then multiple headlines in a span of 20 days. After thus describing our approaches to textual information we describe our handling of price and volume data, before concluding with a description of our overall architecture, which efficiently combines all inputs to reach a trading signal. Note, that whilst we use our training data repeatedly, we strictly avoid using test data at any point in our system's training.

To succeed in analysing textual information using neural networks, the information must first be transformed into a numeric format. Specifically, using an approach termed word2vec [Mikolov et al., 2013], we can convert a headline of strings into a sequence of vectors. Elaborating, every word can be represented as a point in an n dimensional space. This point conveys the context in which the word is used, a characteristic visible by examining the words appearing near it in the n dimensional space. Demonstrating through an example, the words cat and dog will form points closer together than the words cat and France [7]. We learn these representation by passing one hot encoded vectors through an encoder and storing the network's weights (skip-gram model). We use softmax negative sampling [Dyer, 2014] to train these weights.

One Headline: Our aim is to capture as much of the headline's/sentence's meaning and represent this as a point. Were we solely interested in performing arbitrage event investing, we could subsequently - using these headline point representations - look for clusters of points leading to either buy, hold, or sell decisions. Consequently we could than predict if a new test case headline is a buy or etc depending on its position in the n dimensional space.

To train a network to summarise a sentence as a point, we could use a similar procedure to word2vec called doc2vec [Le and Mikolov, 2014], which transforms sequences of text to paragraph vectors. Alternatively, we could use a summarisation tool outlined in [Ding et al., 2015], which first requires preprocessing of headlines into OPO format (summarised in previous document) before using a recursive neural network RNTN to calculate the equivalent of a paragraph vector. Finally, we could use GRN's encoder decoder, feeding in the sequence of text and storing the final output (midpoint of seq2seq). We choose to implement the final method, describing our training approach below. In training our headline encoder decoder we could either use a loss function linked to: company returns, or utilise a sequence to sequence autoencoder. As multiple headlines can appear in any given day, we choose the later of these [8]. Hence, we train the encoder decoder by passing in d - d is the maximum identified headline length in the training and test sample, with zero padding implemented to make all headlines of equal size thereafter 100 to 300 (link to choice made italic part of document) dimensional word vectors, and comparing the outputs with the original inputs. We use a seq2seq cross entropy loss function (outlined in: Sequence to Sequence example, or the cosine similarity loss function to train the encoder weights. Overall the encoder learns to represent the word vectors in 100 to 300 dimensions,

[6] Whilst advances in deep learning have allowed researchers to use techniques to perform end-to-end training, we follow the older method of training our network in steps to provide numerous base case performance figures

[7] Whilst cat and dog are two different animals, we could for instance substitute them both into the sentence 'i have a four legged X in my house' . Whilst the sentence itself would thus convey something different, we cannot for instance substitute the word France into it, characterising the contextual similarity between cat and dog, but not between cat and France.

[8] This should further lead to a better network, as the task is closer linked to our aim of obtaining a semantic representation of the headline

which due to the relatively small size of our training set, we can further reduce the dimension using pca whitening to 30 dimensions in our complete network (outlined below).

Note above I mention using pca whitening. As Nikhil Buduma [Buduma and Locascio, 2017, pp. 120] argues, pca struggles to capture non linear relationships, meaning that the use of a neural network encoder tends to better capture the important dimensions. Note due to the sparsity of training data we only consider a network with a maximum of two hidden layers Note, Deep Learning outlines the use of cosine distance as a loss function when comparing seq2seq. Further, Deep Learning provides a more advanced framework for padding (split sentences into buckets of certain lengths to reduce the overall quantity of padding required - this should tackle any vanishing gradient problem and speed up training). We may not have the necessary data for the second step which is why we implemented standard padding.

Multiple Headlines Appearing in One Day: We outlined in the introduction that our aim when analysing multiple headlines is to capture complex strategic trends. Whilst we proposed using cnn's to achieve this, we prefer to use a max pooling rnn whilst analysing multiple headlines appearing in a *single day*. Elaborating, we believe that capturing the overall importance of headlines appearing in one day will be more useful when analysing patterns appearing between days. We avoid taking averages of headlines as [Ding et al., 2015] per the suggestions of Mikolov 2014 [Le and Mikolov, 2014]. Further, our approach allows us to avoid discarding stories not classified as "top stories".

Describing the training approach in detail, as with an individual sentence we measure the maximum number of headlines appearing in a day (mh) in our training and test sample. We use padding to make all daily headline counts equal to this. As inputs we feed normal vectorised headlines through our pre trained individual headline encoder [9], reducing their dimensionality through pca whitening (already outlined our aim of avoiding pca). In this way we pass mh encoded headlines through our max pooling rnn giving a 30 dimensional (same as input after pca) output vector. We can either train by linking this output to returns using a softmax function, or rather by attempting to decode the output (seq2seq as outlined previously using an rnn decoder). The later of these may involve one hot encoding headlines and using the cross entropy loss function, or better still, as already outlined, cosine similarity.

if the training is linked to the returns, only the training data can be used

Multiple Days - incorporates price and volume: Using the pre trained network described in the previous section we analyse 20 training days (1 month - roughly following Ding et al.) of headlines. As in the previous section, we fix the weights in the pre trained network, allowing us to pass individual headlines through the network (when we mention input in this section from here on, we mean the output from our pre-trained networks). Padding is used in instances where data is unavailable.

To allow us to analyse headlines in conjunction with technical information, we concatenate two additional points to the network inputs. Elaborating to each day's headlines vector (input) we add the the days daily return and volume (could add first differences as well). This expands the input dimension to 32 (34 with first differences), spanning 20 day. Where padding is added for missing headlines, a vector of zeros with only price and volume data for that day is passed into the network. We follow the cnn encoder architecture outlined in [Ding et al., 2015].(number of layers - convolution and max pooling -, step sizes, and channels/depth).

Instead of concatenating simple/plain returns into the multi day network, z-scores can be used, thus incorporating cross sectional information. Theory relating to the momentum factor considers this to be the causal factor[10].

[9] If the maximum headline count is 5 this means reusing our individual headline encoder 5 times. Using tensor flow we fix these weights - again due to sparsity of data -, choosing to focus on training the multiple headline encoder given these input instead.

[10] The fact that something goes up in value is not important in predicting future trends. Rather the fact that something increases/decreases in value z times more than the mean is important in capturing price trends. This agrees with a paper summarised in the previous document.

Ideally we would like to be able to train the network using an unsupervised encoder decoder framework.Such an approach is outlined in [Ngiam et al., 2011] or [Zeiler and Fergus, 2013]. A visualisation of CNN decoder upsampling/deconvolution is available at: [Noh et al., 2015]. The approach functions as any encoder decoder framework thus far explained. The only complication arises when trying to understand the upsampling layer, which is why we provide a link showing the possible visualisations of this process.

Despite this possible framework, due to data sparsity and the complexity of the approach, we decide to follow a simpler method requiring the training of only a CNN encoder. We are thus forced to link the network to returns, once again using a cross entropy loss function (we identify ys into buckets using these as classes for supervised training) to train network weights.

II.I Price and Volume Data

Perhaps the best way of considering price and volume data is to use the analogy of considering speech. Whilst auditory signals may be analysed on their own, better performance is registered when both auditory and visual signals are considered in conjunction. An efficient means of considering these and training an unsupervised network using an encoder and decoder framework are outlined in: [Ngiam et al., 2011]. Price and volume data con also be viewed separately; however as they are inherently linked (like the shape our mouths is with sound when pronouncing a syllable) considering these together should lead to better performance. We use LSTMs in Ngiam et al. 2011 framework, as this allows the continuation and adjustment of signals, whilst considering time effects. Combined with our handling of price and volume data through the CNN outlined in the previous section, this allows us to satisfy our aim of capturing the essence of technical analysis.

Training follows the sequential and modular approach of Ngiam et al. We focus on a time frame of 20 training days (see note for expansion of this horizon) to coincide with data passed into our CNN. An important point to note is that by following the mentioned approach we learn a network that considers both inputs equally. This is achieved by training it to learn to replicate both price and volume data, when simply one form of input is provided. Once again we use a softmax loss function in this unsupervised training approach.

Note: may be able to incorporate momentum effect by adding 12 month historic returns before above mentioned 20 daily returns. This would be in accordance outlined in our previous summary.

II.II Bringing it all together

Freezing weight in all pre trained modules (may have to unfreeze wights in modules trained using returns), we bring everything together by concatenating module encoder outputs (follow fusion layer in: [Eitel et al., 2015]). To be more precise we concatenate outputs from the CNN as well as the price and volume joint encoder. Finally, to emphasise the current time period t, to this concatenation we also add the headlines vectors for today (output from one day headlines encoder using headlines from time t). This concatenated overall vector is then passed through a fully connected neural network with two layers (initially use best practice for setting number of network nodes - formula outlined in data mining coursework) using a ExC activation function (advanced ReLU with non zero gradient for negative values). As we freeze pre-trained weights we should be able to use cross validation to adjust the hyper parameters (enough data to do this considering we are only training a shallow layer).

II.III Further Methods (Testing and Reinforcement Learning)

Whilst we freeze the pre-trained weights in training our overall network, we consider the test sample as providing a small opportunity for adjusting these through reinforcement learning. Whilst any changes using a test sample of only roughly 2k headlines are going to be minimal, they offer a chance to improve our network nonetheless, without breaking any testing approaches. As a final note: Training, and

potential augmentation of our training size, can be achieved by randomly dropping input parameters; we can randomly zero out 50% of input parameters, forcing the network to better learn generalisable rules [Ngiam et al., 2011]. Nevertheless, such an approach is only applicable when batch normalisation is used. This is further the case when padding is applied.

Headlines and Natural Language Processing: Mention story with Anne Hathaway and Berkshire Hathaway Overall aim: Trying to take one headline or possibly multiple headlines, where each headline is represented as a point in an n dimensional space allowing us to identify clusters of headlines for which to buy, sell, hold provided we use a CNN on multiple points (and thus headlines) we attempt to identify the interactions (features) between them to once again identify clusters for buying, selling, and holding. The first step of this process involves representing a sentence of words as a single point from the work of [Ding et al., 2015]. as well as Richard Socher [Socher et al., 2013] we have been led to believe that Recursive Neural Tensor Networks offer the best opportunity to do this. The next step either involves using a fully connected neural network to analyse one headline/point, or using CNNs multiple points... (CNNs become fully connected Ns after numerous max pooling procedures).

II.IV Summaries of Main Papers

II.IV.1 Paper 1

Among the first to use non linear approaches (neural nets) on structured data, instead of just linear models on bags of words to evaluate headlines. Argues that 'unstructured terms cannot indicate the actor and object of the event' thus poorly evaluating an events impact on stock price movements. Moreover, proves that NN approaches outperform previous simpler linear (eg linear SVM) attempts to analyse events. Overall the paper's method achieves "accuracy of S&P 500 index prediction of 60%, and that of individual stock prediction of over 70%".

IV.1.1 **Methodology:**

Part 1a: Structured representation: "Each event is composed of an action P, an actor O1 that conducted the action, and an object O2 on which the action was performed. Formally, an event is represented as E = (O1,P,O2,T), where P is the action, O1 is the actor, O2 is the object and T is the timestamp (T is mainly used for aligning stock data with news data)" As quoted above the authors use a (O, P, O, T) format to capture the information in headlines. They start by extracting the predicate type P, "and then find the longest sequence of words Pv, such that Pv starts at P and satisfies the syntactic and lexical constraints proposed by Fader et al. (2011)" (paper: [Fader et al., 2011] - which describes open information extraction and dependency parsing). Next they identify O1, and O2; they find the nearest left and right noun phrases to P respectively (must contain subject and object otherwise headline is excluded).

Part 1b: Event Generalisation: To reduce the number of (O,P,O) types the authors use WordNet to "extract lemma forms of inflected words". This translates eg verbs like "changed" to "change" and nouns like "dogs" to "dog". After this VerbNet is used to tranche events; can be thought of as trying to put identified Ps into buckets representing events. "After generalisation, the event (Private sector, adds, 114,000 jobs) becomes (private sector, multiply class, 114,000 job)."

Part 2: Prediction Models: The paper mentions SVM's as well as bag of words representation; however below we focus on their non linear approach implementing (O,P,O). Before continuing we must note that to avoid sparseness the paper transforms (O,P,O) into (O1, P, O2, O1 + P, P + O2, O1 + P + O2) - aka back-off features. The paper does not specify what method is used to transform the words appearing

in (o,p,o) to vectors - We suspect they use the same (tfidf) score as with bag of words: "freq(tl) denote the number of occurrences of the lth word in the vocabulary in document d.

$$\text{tf}_l = \frac{l}{|d|} freq(t_l), \forall l \in [1, L] \tag{II.1}$$

where $|d|$ is the number of words in the document d (stop words are removed).

$$\text{tfidf}_l = \frac{l}{|d|} freq(t_l) \times \log\left(\frac{N}{|\{d : freq(t_l) > 0\}|}\right) \tag{II.2}$$

where N is the number of documents in the training set. The feature vector ϕ can be represented as $\phi = \phi_1, \phi_2, ...\phi_M = (\text{tfidf}_1, \text{tfidf}_2, ..., \text{tfidf}_m)$". Despite this, the authors mention using a two layer fully connected neural net (they try three layers but find no improvement in classification results) with two output classes (+1, -1) signifying buying and selling. Numerous Ys are used spanning three different horizons: one day, one week, and one month (with results showing diminishing results as the time horizon increases). The authors 'automatically align 1,782 instances of daily trading data with news titles and contents from the previous day' with headlines presumably being evaluated individually nonetheless. This implies that numerous headlines may share the same y in the authors paper.

II.IV.2 Paper 2

Improving upon section IV.1, Paper 2: [Ding et al., 2015]. At the moment the principal motivation to our own paper. The authors find an additional 6% classification accuracy on top of their previous paper section IV.1 when analysing S&P500 returns. They achieve this improvement principally by focusing on deep learning (cnn architecture) whilst implementing word vectors (event embeddings, which are dense vectors) to represent their previous (o, p, o) format. Additionally, they examine multiple headlines at once, instead of just using one headline per y. Note: "Embeddings are trained such that similar events, such as (Actor = Nvidia fourth quarter results, Action = miss, Object = views) and (Actor = Delta profit, Action = didn't reach, Object = estimates), have similar vectors, even if they do not share common words." Methodology: Part 1: Event Representation and Extraction: They follow the same outline as in their first aper to reach (o, p, o). Nevertheless, the authors better specify their approach, detailing their use of "ReVerb to extract the candidate tuples of the event (o1, p, o2), and then parse the sentence with ZPar [Zhang and Clark, 2011] to extract the subject, object and predicate". The same filtering is used as in section IV.1 - if o1 and o2 do not contain subject and object the are filtered out. For absolute clarity we paste the extract from the first paper describing this below: 'For each event phrase Pv identified in the step above, we find the nearest noun phrase o1 to the left of Pv in the sentence, and O1 should contain the subject of the sentence (if it does not contain the subject of Pv, we find the second nearest noun phrase). Analogously, we find the nearest noun phrase o2 to the right of Pv in the sentence, and o2 should contain the object of the sentence (if it does not contain the object of Pv, we find the second nearest noun phrase).'

Part 2: Event Embedding (This is different to paper 1): Side Note: Useful additional paper for understanding embedding: [Socher et al., 2013] - shows that "performance can be improved when entities are represented as an average of their constituting word vectors" video explaining above paper.

Essentially outlines the use of rntn's to combine word vectors to get sentence word vectors (simplification).... they use this to identify probable relationships between words (e1 and e2).... since they have three words (e,r, e) the network must use two layers between (e, r) (r, e2) and (er, re2) to get a final score. further links to explain above. Continuing with paper: We must note before describing the authors use of rntn's that all phrases both for Os and the P exceeding one word are averaged (average word vectors - allowing you to represent Os like "Nokia's mobile phone business and Nokia").

The authors use the skip gram model implemented on a large financial corpus to learn these 100 dimensional word vectors. Moving on to describing the training, the authors then implement the same RNTN described above (two layers and etc). This is trained using margin loss and l2 regularisation - a corrupted OPO is created by replacing one O with a randomly selected word from the dictionary, with the network being taught to identify the true OPO by minimising the marginal loss. (500 iterations are conducted for every OPO). Part 3: Prediction Model: Because with one headline a fully connected network is used, I detail the authors use of CNN's over the medium and long term. Whilst the author does not specify how he tackles multiple headlines in any given day, examining the pictures provided it appears that for a horizon of 30 days he uses 30 inputs.... perhaps implying that he uses simple averaging of headlines? Regardless the structure involves sliding a window of size 3 over the data (presumably for both med and long term) repeatedly, and then max pooling the resulting layers over the depth (number of repeated passes). Afterwards the resulting layer is passed through a fully connected network. Interestingly before passing the short medium and long term through the fully connected layer, the resulting vectors are concatenated creating "feature vectors $V_C = (V_l, V_m, V_s)$". Subsequently a sigmoid activation function is used to determine a score for two classes (would be better to use softmax): buy sell [+1, -1].

II.IV.3 Paper 3

Another paper [Feuerriegel and Fehrer, 2016] examining deep learning and its use in single headline analysis. It compares the performance of a recursive autoencoder against a benchmark random forest approach finding a (roughly) 6 boost in performance resulting from implementing the deep network. Methodology: the benchmark follows a similar approach to the linear model in the first paper. A headline is prepossessed by removing punctuations and etc, before being translated into a text document matrix (measures how frequently each word appears in the training set using the probability for each word) before being used as part of a random forest. Moving on to the autoencoder: Potentially outlines a method for converting a headline into a single vector representation without using section IV.1 and section IV.2's OPO structured representation. Instead the paper notes that one hot vectors can be recursively combined to create encoded vectors - after the entire sentence has been passed through we have a vector encoding the entire sentence.

II.IV.4 Paper 4

Outlined link, The user essentially converts headlines into word vector matrices (numerous words so matrix) which are then passed through a CNN that generates a buy sell hold signal for that headline. Justifies using CNNs by citing: [Kim, 2014] - this paper demonstrated the benefits of using CNNs in a variety of nlp tasks ranging from machine translation to sentiment analysis. Further the author of the supplementary paper shows that not much training is required of the CNN weight parameters to achieve the positive results.

III. Supplementary Information

This paper [Takeuchi, 2013] discusses exploiting the momentum trading strategy (buying things that have gone up more than the industry average over the past 3 months and selling the etc) using a neural network architecture with 5 layers including the input layer. In short the paper finds that analysing price charts (t-3 to t-13 of monthly returns and past 20 days of daily returns) can in conjunction with a enhanced trading strategy 'deliver an annualised return of 45.93% over the 1990-2009 test period versus 10.53% for basic momentum'. Methodology: the paper uses a training sample from January 1965 to December 1989 and a test sample up to 2009. The analyses is restricted 'to ordinary

shares trading on NYSE, AMEX, or Nasdaq's, with a closing monthly price higher than $5. Both monthly returns and daily returns are considered, with returns adjusted cross sectionally to z scores (like weighing a companies performance that month/day with respect to all other companies). This is done to capture the relevant information required for exploiting momentum trading. Regarding the neural network architecture, whilst the author notes that he uses stacked restricted Boltzmann machines for encoding the original input (33 input parameters - concatenated monthly returns and daily returns) before passing resulting outputted results through a fully connected neural network, in essence the entire procedure may be viewed as a neural network with an expanding then contracting and finally expanding node structure ("a five-layer network 33–40–4–50–2 consisting of an encoder that takes 33 inputs and reduces them to a 4 dimensional code and a classifier that takes these 4 inputs and outputs the probabilities for the two classes").

III.I Summary

To understand RNTN (Recurrent Neural Tensor Network) look at: link and link start of with simple concatenation of word a and word b multiplied by a Tensor to give a combined representation.... this is then passed through an activation function that spits out a score. To train such a simple model you could change a to non sensical word (call it z) and look for a-b getting a larger score than z-b. To better capture the interaction between a-b, to the simple model a second procedure is added (to understand this just do the matrix multiplication) where a is multiplied by a tensor and then b to give one number in a vector of size d (the dimensionality of the original word vector) Despite numerous papers denouncing technical analysis [FAMA and FRENCH, 2008], researchers agree that momentum can yield returns.

6. *Implementation*

Most of the work done on this project was done using Tensorflow. This involves both the construction and visualisation. Other python packages was also used like Scikit Learn and Pandas. The first step was to encode the headlines in to a lower representation. The second step was to encode multiple headlines for each company for each day to a lower representation (Daily Encoder). The networks mainly used involved sequence to sequence auto encoders and machine translation. During training time a lot of factors was considered which includes the size of the hidden layers, learning rate parameters and types of optimisers. The main optimiser used was the Adam optimiser. In addition, an early stop procedure was added to make sure if the validation error does not improve after 30 epoch, the network gets terminated. The best epoch with the lowest validation error is selected as the best model. It is to be noted that the main aim of this report is to maximise the probability of predicting a class (buy, hold, sell) by analysing headlines and technical data.

I. Headlines Encoding

The first auto-encoder consisted of auto-encoding the headlines using machine translation. Unlike deep auto encoders which uses least squared method as its loss function, machine translation uses cross entropy with logits, and the errors is measured with perplexity. How well the network can translate the value from one to the other. The inputs are represented as 3 dimensional tensors (Batch size, number of steps, frame dimension). Using an embedded matrix the frame dimension would be represented with a single integer value which would be used to index the embedding matrix for the corresponding vector. This value would then be fed into the network. This can be illustrated further by the image fig. 6.1. It is to be noted the lstm layers can be stacked up to form deeper layers. In the hyper parameters this would be denoted by the RNN depth parameter. To further explain what this step comprises of This will take a headline "Google ($300d$) buys ($300d$) youtube($300d$)" which would be 900 dimensional vector in total and compresses it to $300d$ vectors in the encoded state.

Due to the headline having variable length, the RNN network would need to find a way to deal with this type of input. The naive way would be to pad the vectors to the maximum sentence length of the corpus. This would lead to a problem of having very sparse matrix multiplication with a large number of zeros, which would lead in to weights becoming very small and not making a difference A different method would be to group the inputs in to reasonable groups of discretised sizes, and the incrementally train the inputs from lowest number to largest whist copying the weights from the small in to the large network.

I.I Loss and Evaluation

The graph shows a steady conversion rate after the first epoch. and the validation loss decreases with the training loss. The result of this process can be visualised fig. 6.2. Where red represents buy, blue represents sell and the others represents Hold and NA. The aim as mentioned earlier, is for news with

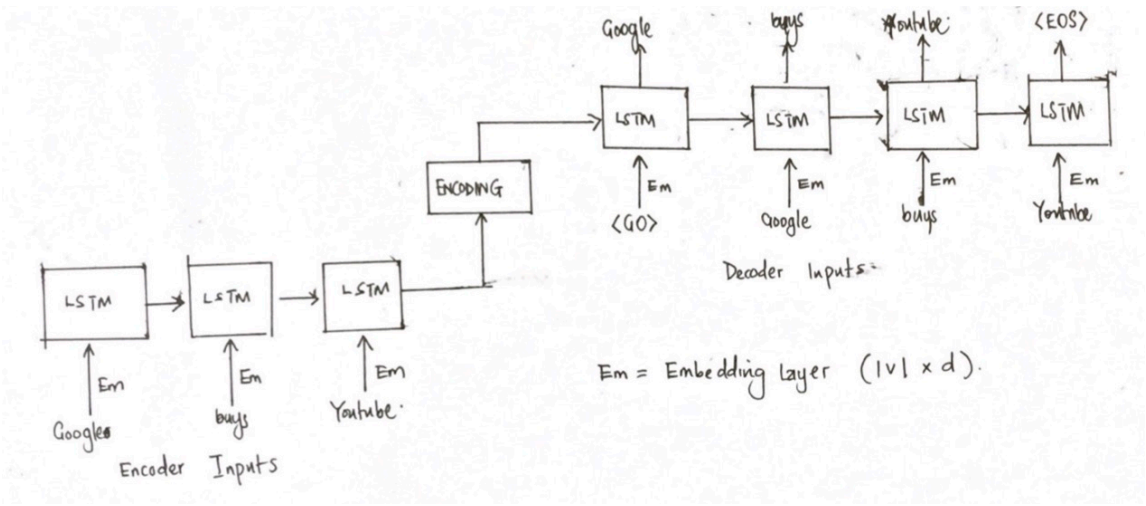


Figure 6.1: *Headline Autoencoder*

Hyperparameter	Value
Network type	LSTM
Batch size	64
Hidden neurons size	300 - 500
Regularisation parameter	0.3
Learning Rate	0.001
RNN Depth	1
Epoch	200

Table 6.1: *Headline Encoding Hyper parameters.*

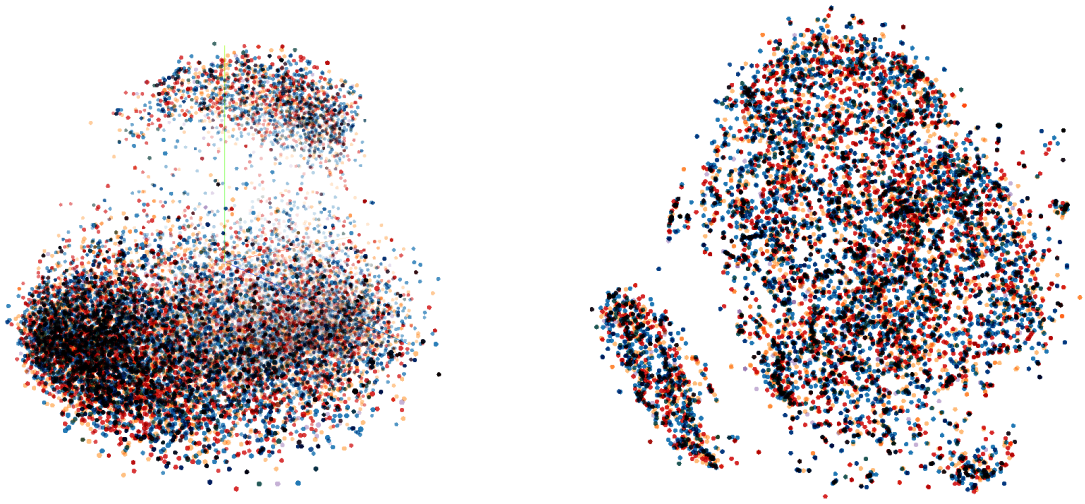
Epoch	Training Error	Validation Error
1	10.60524	10.62446024
2	4.2282748	62949
3	2.816638	6.019030
4	2.122330	6.1285
5	1.7702149	6.33384
6	1.54610	6.581824

Table 6.2: *Headline Encoding*

the same meaning should cluster together. As an example, we examined a random news and check the closest cosine distance to the current news and the results is shown as follows.

- Terraform announces election of two independent directors
- Terraform Power announces receipt of NASDAQ letter
- Terraform announces sale of UK portfolio

From this example, it shows the machine translation model was able to capture some meaning from the sentences. As you can see its able to "subjects" together, but with this, the model ignores the object in the sentence



(a) *Headline Encoding* (b) *TSNE Visualisation.*

Figure 6.2: *PCA and TSNE Visualisation of Headline Encoder*

II. Daily Encoding

For the daily encoding we used the learning through loss error propagation as mentioned in section I. But this time we changed a couple hyper parameters, this includes the dimension of the hidden neurons, learning rate, input frame (this comes from the previous encoded vectors). In this method we also included bi-directionality to see if this would improve convergence.

Hyperparameter	Value
Network type	LSTM
Batch size	32
Hidden Neuron size	300 - 500
Regularisation parameter	0.3
Learning Rate	0.0001
RNN Depth	1
Epoch	200

Table 6.3: *Day Encoding: Hyperparameters used in the day encoder model.*

We use PCA to visualise the data to see what information clusters together. The ideal case would be that similar information cluster together and others would represent a different point in space. What we figured out was that the information clustered together, but this was done based on the length of the headline for that day. This is due to the padding problem ??. But taking a closer look, we was able to analyse these mini clusters even further to see if it yields interesting results. Some of the example headline clustered together includes For the single headlines cluster this were some example of headlines clustered together.

- BRIEF-IAC InterActiveCorp posts qtrly adjusted earnings $0.73 shr
- BRIEF-Penns Woods Bancorp qtrly operating earnings $0.56 shr
- BRIEF-Perrigo sets qtrly dividend of $0.145 shr

For multiple headlines

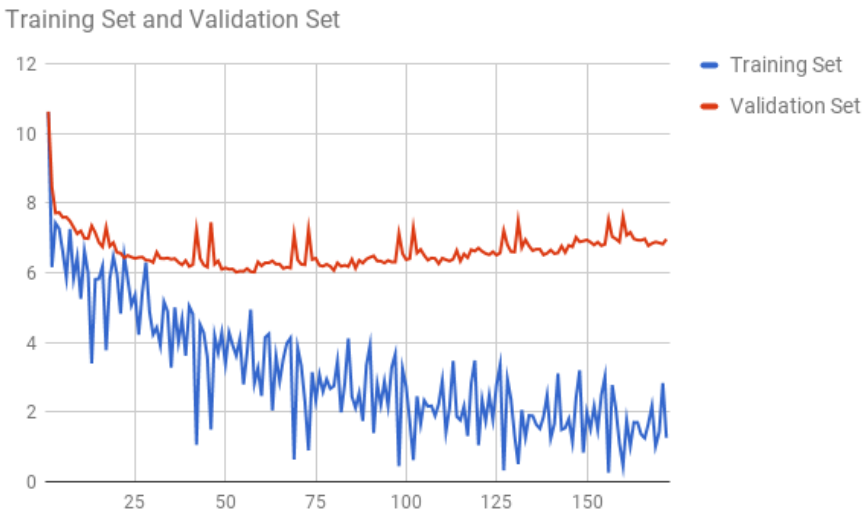
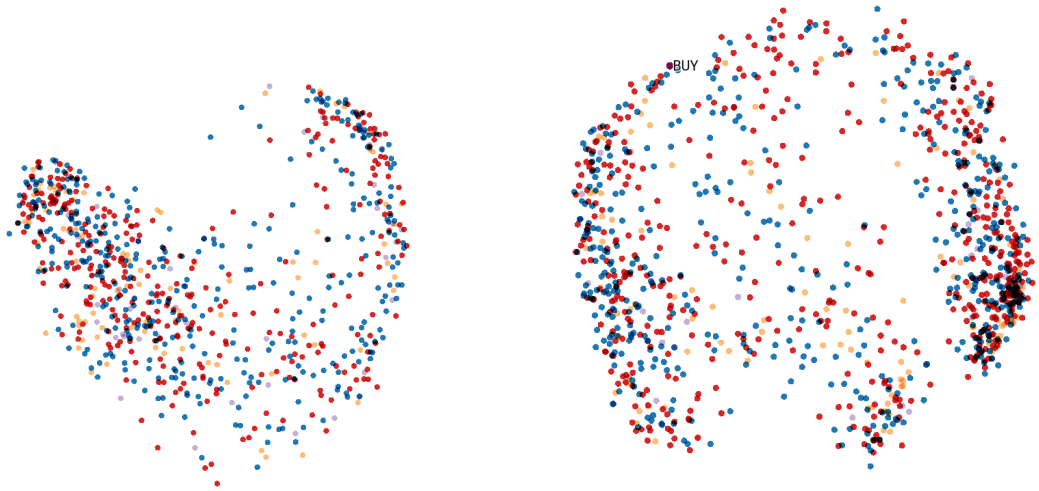


Figure 6.3: *Headline Encoder Loss: The validation error decreases with respect to the training error. The lowest validation occurs around 50 - 75*



(a) *Single Headline Cluster PCA* (b) *Multiple Headline Cluster PCA*

Figure 6.4: *Day Encoder: Visualising the clusters of information for single headline in a day and multiple headlines for multiple days.*

- BRIEF-Bellatrix appoints Brent Eshleman CEO , BRIEF-Bellatrix Exploration appoints Brent Eshleman as CEO
- BRIEF-Myriad RBM announces agreement with Sanofi , BRIEF-Planet.fr SA to acquire Addict Media

From this example, the cluster similar syntactic information together, This includes subjects like Bellatrix, RBM and objects like Brent, Sanofi together. In terms of of how they relate to decision, the model does not seem to have an effect on the decision. Instead it looks like random generated noise when it comes to making decision.

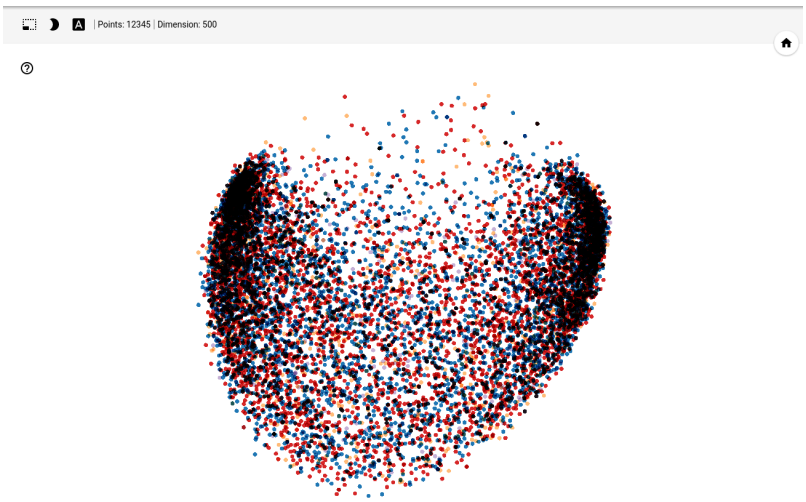


Figure 6.5: *Averaged day company headline*

II.I Average Model

In addition to performing sequence to sequence error propagation encoding on the first set of vectors, we also did an average vector encoding. This process consist of taking all the encoded vector for one day belonging to a single company and computing the average of that vector for that company for that day.

II.II Loss and Evaluation

The loss function used was least squared error. The values from some of the epoch are displayed as follows. For visualisation purposes the errors were log transformed.

Epoch	Training Error	Validation Error
1	129.559	240.44
2	118	99
3	106	45
4	91	23
5	53	12.14
10	14.07	12.521
20	2.105	12.68
50	0.044	12.417

Table 6.4: *Day Encoding Loss per epoch*

III. Technicals Encoding

While encoding the technicals information we assume that the volume and the price data is not random. We also make the assumption that some sequences of the technicals occurs more frequently than others, making us able to predict future returns. Using the same principle for the headlines, we should be able to cluster technical data and then link these information to the outcome which is one of three decision mentioned earlier (Buy, Hold, Sell). The hyper parameters for the technicals includes the following.

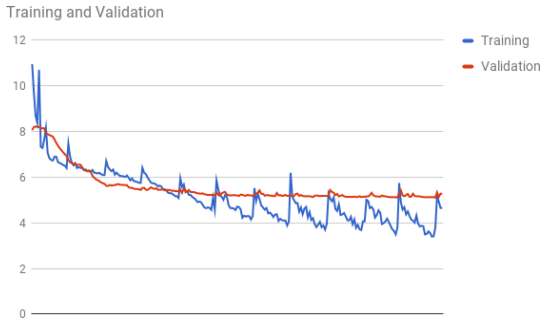


Figure 6.6: *Encoded Day Loss: The validation error decreases with respect to the training error.*

Due to the volatility of technical information in general, this network was trained for a longer epoch in order for the network to generalise better. As shown from the graph fig. 6.7, the training error have massive oscillation. This could be due to two main factors, having a high learning rate, or commonly caused by a high momentum when performing gradient descent. Visualisation the technical encoding

Hyperparameter	Value
Network type	LSTM
Batch size	128
Hidden Neuron size	20
Regularisation parameter	0.1
Learning Rate	0.01
RNN Depth	3
Epoch	1000

Table 6.5: *Technicals Encoding Hyperparameters*

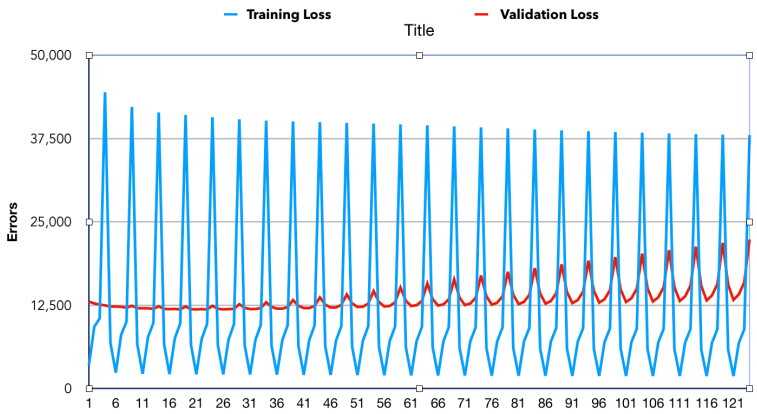
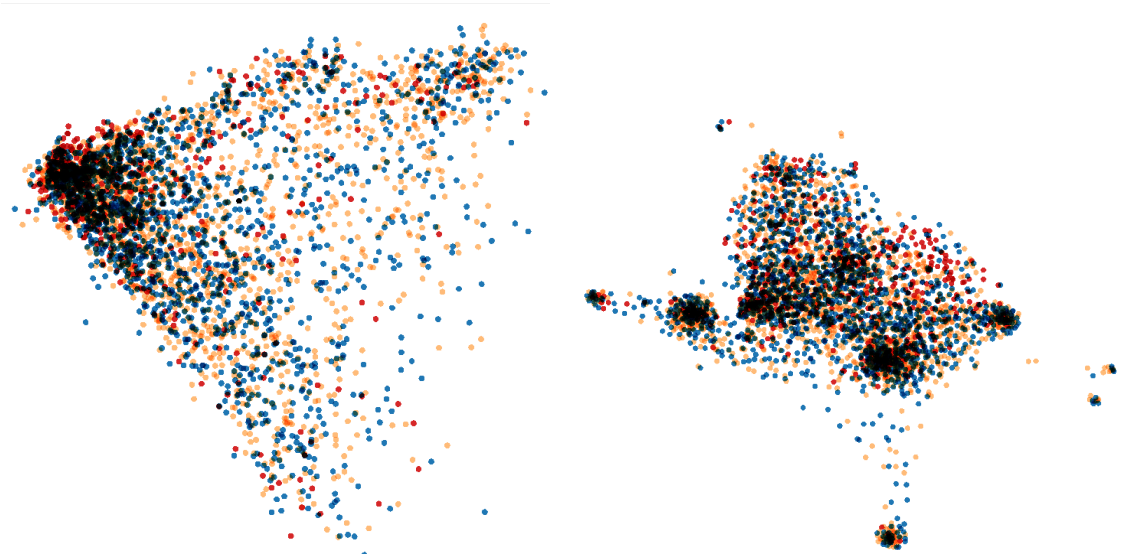


Figure 6.7: *Technicals Training: Massive Oscillation due to really high learning rate or high momentum.*

information can be viewed as follows



(a) *PCA Technicals Encoding* (b) *TSNE Technical Encoding*

Figure 6.8: *Images represents 20-dimensional encoded vectors.*

7. *Experiments*

Overall, our model was suboptimal on the validation set; the model had a tendency to predict a single class (HOLD), as this was the class with the highest frequency in the validation set. This results yield to us in search for a model that remained stable across increasing epochs. We developed several strategies that consisted of deep and standard Machine learning techniques to see if this would yield better results than the literature[1]. Models could be classified as good if they perform better than picking a random decision [2] The models we experimented with includes All of the results are listed as follows with the best model highlighted in bold.

- Varying CNN Window Size.
- CNN without Headline
- RNN encoded technicals
- CNN with RNN encoded technicals using averaged day headline vectors.
- Different Neural Network Architectures.
- Standard Machine learning techniques (Adaboost, Logistic Regression e.t.c)

	CNN + RNN No Headline M1	CNN + RNN No Headline M2	CNN + RNN No Headline M1 T+3	CNN + RNN No Headline M1 T+5	RNN M1
Training Accuracy (dcp)	0.38589	0.36375	0.37303	0.35729	**0.39126**
Min. Training Loss	**0.02127**	0.02140	0.02140	0.02147	0.02129
Min. Validation Loss	0.02152	0.02152	0.02148	0.02146	**0.02150**

Figure 7.1: *Table showcases just technical analysis results.*

	CNN + RNN MT 300	CNN + RNN MT 150	CNN + RNN MSE	CNN + RNN AVG	CNN x 3 + RNN
Training Accuracy (dcp)	0.36717	0.34975	0.36101	0.36020	**0.37080**
Min. Training Loss	0.02139	0.02147	0.02142	0.02145	**0.02133**
Min. Validation Loss	0.02144	0.02154	0.02150	0.02148	**0.02137**

Figure 7.2: *Table summarises the concatenated headlines with the technicals results.*

The table conveys adding headlines reduces the model accuracy, this implies that headline tends to add noise towards the accuracy of the model. In addition out model utilising a 3 layer deep fully connected layer, suggest that concatenating CNN encodings performs worse than just using technical information to perform inference. Furthermore the table supports [Ding et al., 2015] results, that any potential trading signals weakens as the time frame increases for computing Y's. This is seen by reducing accuracy and increasing training and validation loss.

For the other machine learning method, we used AdaBoost, Decision Tree, LSA, Logistic Regression, Naive Bayes, QDA Restricted Boltsman Machine, Random Forest, Support Vector Machine and K Nearest Neighbours. The inputs would is specified as the headers of the model, and the columns represents the models used, the highest accuracy is noted in bold.

[1] Best model is determined by the model with the lowest validation loss
[2] Random decision is when the model getting an accuracy of 33%

ACCURACY	CNN + RNN	CNN+RNN No Headline	RNN + Headline	RNN + MSE 400 dim Headline	Headline Only
AdaBoost	0.4854	0.4872	0.4812	0.4849	0.4557
Decision Tree	0.4891	0.4875	0.4817	0.4886	0.4594
LDA	**48.93%**	48.72%	48.93%	49.12%	45.30%
Logistic Regression (C=1)	48.78%	48.70%	48.80%	**49.14%**	45.54%
Logistic Regression (C=1000)	48.80%	48.72%	**48.96%**	49.12%	45.36%
Naive Bayes	45.86%	46.57%	40.86%	46.70%	36.65%
QDA	47.01%	47.09%	44.25%	48.22%	38.33%
RBM 100	46.12%	46.22%	46.12%	46.12%	**46.12%**
RBM 100, n_iter=20	46.12%	46.12%	46.12%	46.12%	**46.12%**
RBM 200, n_iter=40, LR=0.01, Reg: C=1	46.15%	46.12%	46.12%	46.12%	**46.12%**
RBM 200, n_iter=40, LR=0.01, Reg: C=100	46.15%	46.12%	46.12%	46.12%	**46.12%**
RBM 256	46.12%	46.15%	46.12%	46.12%	**46.12%**
RBM 512, n_iter=100	46.12%	46.12%	46.12%	46.12%	**46.12%**
Random Forest	48.17%	47.59%	47.01%	48.62%	42.59%
Random Forest 2	46.59%	46.94%	46.12%	48.20%	**46.12%**
SVM, adj.	48.49%	48.01%	48.28%	48.72%	46.09%
SVM, linear	47.70%	47.70%	47.83%	47.86%	**46.12%**
k NN	38.73%	40.44%	39.04%	40.46%	34.04%

*Unless otherwise specified results shown use 300 dim. encoded headlines using our MT model.

Figure 7.3: *Accuracy of the model based on the inputs.*

ACCURACY	RNN	RNN T+2	RNN T+3	RNN T+4	RNN T+5
AdaBoost	0.4849	0.4154	0.4404	0.4536	0.4517
Decision Tree	0.4886	0.4199	0.4344	0.4504	0.453
LDA	**49.12%**	42.09%	44.41%	46.28%	46.01%
Logistic Regression (C=1)	**49.12%**	41.73%	44.46%	46.12%	45.96%
Logistic Regression (C=1000)	**49.12%**	41.73%	44.46%	46.12%	45.96%
Naive Bayes	46.70%	38.88%	36.12%	34.70%	32.25%
QDA	47.83%	38.75%	34.94%	33.10%	30.36%
RBM 100	46.12%	40.33%	42.30%	44.73%	**46.78%**
RBM 100, n_iter=20	46.09%	40.33%	42.36%	44.78%	46.75%
RBM 200, n_iter=40, LR=0.01, Reg: C=1	46.09%	40.33%	42.28%	44.75%	46.75%
RBM 200, n_iter=40, LR=0.01, Reg: C=100	46.12%	40.28%	42.28%	44.78%	**46.78%**
RBM 256	46.12%	40.31%	42.30%	44.75%	**46.78%**
RBM 512, n_iter=100	46.12%	40.33%	42.28%	44.73%	46.75%
Random Forest	46.80%	38.02%	39.02%	40.70%	42.73%
Random Forest 2	47.96%	41.36%	**44.83%**	46.25%	46.36%
SVM, adj.	48.72%	**42.49%**	44.80%	**46.51%**	46.67%
SVM, linear	47.86%	42.28%	42.28%	44.73%	46.75%
k NN	40.46%	34.36%	35.99%	37.10%	38.36%

*Unless otherwise specified results shown are for Model1 (M1) which implements 20 dimensional seq2seq encoding using T+1 Ys

Figure 7.4: *Accuracy of the model based on the inputs.*

From the results provided above we can see that Logistic Regresssion provides the best results compared to other complex models.

ACCURACY	CNN + RNN	CNN+RNN No Headline	RNN + Headline	RNN + MSE 400 dim Headline	Headline Only
AdaBoost	0.4854	0.4872	0.4812	0.4849	0.4557
Decision Tree	0.4891	0.4875	0.4817	0.4886	0.4594
LDA	**48.93%**	48.72%	48.93%	49.12%	45.30%
Logistic Regression (C=1)	48.78%	48.70%	48.80%	**49.14%**	45.54%
Logistic Regression (C=1000)	48.80%	48.72%	**48.96%**	49.12%	45.36%
Naive Bayes	45.86%	46.57%	40.86%	46.70%	36.65%
QDA	47.01%	47.09%	44.25%	48.22%	38.33%
RBM 100	46.12%	46.22%	46.12%	46.12%	**46.12%**
RBM 100, n_iter=20	46.12%	46.12%	46.12%	46.12%	**46.12%**
RBM 200, n_iter=40, LR=0.01, Reg: C=1	46.15%	46.12%	46.12%	46.12%	**46.12%**
RBM 200, n_iter=40, LR=0.01, Reg: C=100	46.15%	46.12%	46.12%	46.12%	**46.12%**
RBM 256	46.12%	46.15%	46.12%	46.12%	**46.12%**
RBM 512, n_iter=100	46.12%	46.12%	46.12%	46.12%	**46.12%**
Random Forest	48.17%	47.59%	47.01%	48.62%	42.59%
Random Forest 2	46.59%	46.94%	46.12%	48.20%	**46.12%**
SVM, adj.	48.49%	48.01%	48.28%	48.72%	46.09%
SVM, linear	47.70%	47.70%	47.83%	47.86%	**46.12%**
k NN	38.73%	40.44%	39.04%	40.46%	34.04%

*Unless otherwise specified results shown use 300 dim. encoded headlines using our MT model.

Figure 7.5: *Accuracy of the model based on the inputs.*

ACCURACY	RNN	RNN T+2	RNN T+3	RNN T+4	RNN T+5
AdaBoost	0.4849	0.4154	0.4404	0.4536	0.4517
Decision Tree	0.4886	0.4199	0.4344	0.4504	0.453
LDA	**49.12%**	42.09%	44.41%	46.28%	46.01%
Logistic Regression (C=1)	**49.12%**	41.73%	44.46%	46.12%	45.96%
Logistic Regression (C=1000)	**49.12%**	41.73%	44.46%	46.12%	45.96%
Naive Bayes	46.70%	38.88%	36.12%	34.70%	32.25%
QDA	47.83%	38.75%	34.94%	33.10%	30.36%
RBM 100	46.12%	40.33%	42.30%	44.73%	**46.78%**
RBM 100, n_iter=20	46.09%	40.33%	42.36%	44.78%	46.75%
RBM 200, n_iter=40, LR=0.01, Reg: C=1	46.09%	40.33%	42.28%	44.75%	46.75%
RBM 200, n_iter=40, LR=0.01, Reg: C=100	46.12%	40.28%	42.28%	44.78%	**46.78%**
RBM 256	46.12%	40.31%	42.30%	44.75%	**46.78%**
RBM 512, n_iter=100	46.12%	40.33%	42.28%	44.73%	46.75%
Random Forest	46.80%	38.02%	39.02%	40.70%	42.73%
Random Forest 2	47.96%	41.36%	**44.83%**	46.25%	46.36%
SVM, adj.	48.72%	**42.49%**	44.80%	**46.51%**	46.67%
SVM, linear	47.86%	42.28%	42.28%	44.73%	46.75%
k NN	40.46%	34.36%	35.99%	37.10%	38.36%

*Unless otherwise specified results shown are for Model1 (M1) which implements 20 dimensional seq2seq encoding using T+1 Ys

Figure 7.6: *Accuracy of the model based on the inputs.*

8. *Evaluation*

9. *Conclusion*

10. *Appendix*

I. Experiments Visualisation

I.I CNN RNN FC

Convolutional Neural Network in combination with RNN and a fully connected layer at the end.

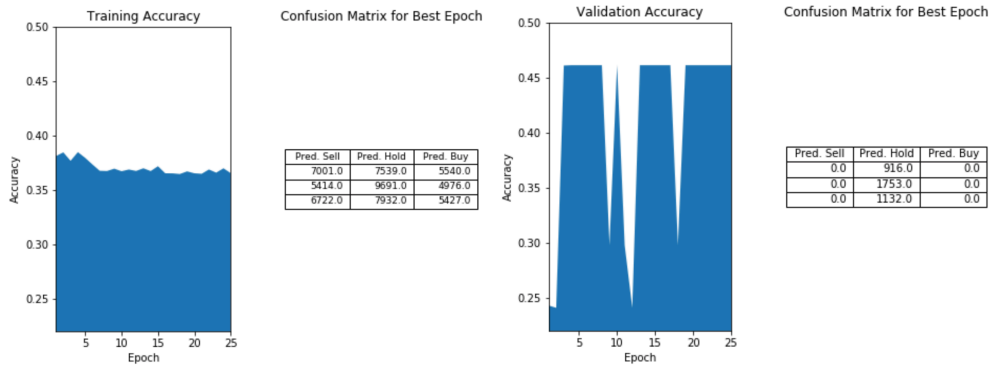


Figure 10.1: CNN RNN *Fully Connected: Training and validation accuracy across 25 epoch with tables showing the best model*

Bibliography

[Abarbanell and Bushee, 1997] Abarbanell, J. S. and Bushee, B. J. (1997). Fundamental analysis, future earnings, and stock prices. *Journal of Accounting Research*, 35(1):1–24.

[Amodei et al., 2015] Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., Elsen, E., Engel, J., Fan, L., Fougner, C., Han, T., Hannun, A. Y., Jun, B., LeGresley, P., Lin, L., Narang, S., Ng, A. Y., Ozair, S., Prenger, R., Raiman, J., Satheesh, S., Seetapun, D., Sengupta, S., Wang, Y., Wang, Z., Wang, C., Xiao, B., Yogatama, D., Zhan, J., and Zhu, Z. (2015). Deep speech 2: End-to-end speech recognition in english and mandarin. *CoRR*, abs/1512.02595.

[Buduma and Locascio, 2017] Buduma, N. and Locascio, N. (2017). *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*. O'Reilly Media.

[Chung et al., 2014] Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

[Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407.

[Ding et al.,] Ding, X., Zhang, Y., Liu, T., and Duan, J. Using structured events to predict stock price movement: An empirical investigation.

[Ding et al., 2015] Ding, X., Zhang, Y., Liu, T., and Duan, J. (2015). Deep learning for event-driven stock prediction. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 2327–2333. AAAI Press.

[Donahue et al., 2014] Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2014). Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389.

[Duchi et al., 2010] Duchi, J., Hazan, E., and Singer, Y. (2010). Adaptive subgradient methods for online learning and stochastic optimization. Technical Report UCB/EECS-2010-24, EECS Department, University of California, Berkeley.

[Dyer, 2014] Dyer, C. (2014). Notes on noise contrastive estimation and negative sampling. *CoRR*, abs/1410.8251.

[Eitel et al., 2015] Eitel, A., Springenberg, J. T., Spinello, L., Riedmiller, M. A., and Burgard, W. (2015). Multimodal deep learning for robust RGB-D object recognition. *CoRR*, abs/1507.06821.

[Fader et al., 2011] Fader, A., Soderland, S., and Etzioni, O. (2011). Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Fama and French, 1993] Fama, E. F. and French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of financial economics*, 33(1):3–56.

[Fama and French, 1997] Fama, E. F. and French, K. R. (1997). Industry costs of equity. *Journal of financial economics*, 43(2):153–193.

[Fama and French, 2007] Fama, E. F. and French, K. R. (2007). The anatomy of value and growth stock returns. *Financial Analysts Journal*, 63(6):44–54.

[FAMA and FRENCH, 2008] FAMA, E. F. and FRENCH, K. R. (2008). Dissecting anomalies. *The Journal of Finance*, 63(4):1653–1678.

[Feuerriegel and Fehrer, 2016] Feuerriegel, S. and Fehrer, R. (2016). Improving decision analytics with deep learning: the case of financial disclosures. In *ECIS*.

[Freyd and Finke, 1985] Freyd, J. J. and Finke, R. A. (1985). A velocity effect for representational momentum. *Bulletin of the Psychonomic Society*, 23(6):443–446.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

[Hinton et al., 2012] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

[Hubel, 1959] Hubel, D. H. (1959). Single unit activity in striate cortex of unrestrained cats. *The Journal of Physiology*, 147(2):226–238.

[Investopedia, 2017] Investopedia (2017). Heads and shoulders pattern.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

[Kim, 2014] Kim, Y. (2014). Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

[Kingma and Welling, 2014] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*.

[Kiros et al., 2015] Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R., and Fidler, S. (2015). Skip-thought vectors. *CoRR*, abs/1506.06726.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'12, pages 1097–1105, USA. Curran Associates Inc.

[Krogh and Hertz, 1991] Krogh, A. and Hertz, J. A. (1991). A simple weight decay can improve generalization. In *Proceedings of the 4th International Conference on Neural Information Processing Systems*, NIPS'91, pages 950–957, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Lai et al., 2015] Lai, S., Xu, L., Liu, K., and Zhao, J. (2015). Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2267–2273. AAAI Press.

[Lample et al., 2016] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. *CoRR*, abs/1603.01360.

[Le and Mikolov, 2014] Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

[Liu, 2017] Liu, C. (2017). Packing topological minors half-integrally. *CoRR*, abs/1707.07221.

[Lo et al., 2000] Lo, A. W., Mamaysky, H., and Wang, J. (2000). Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *The Journal of Finance*, 55(4):1705–1765.

[McCulloch and Pitts, 1988] McCulloch, W. S. and Pitts, W. (1988). Neurocomputing: Foundations of research. chapter A Logical Calculus of the Ideas Immanent in Nervous Activity, pages 15–27. MIT Press, Cambridge, MA, USA.

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

[Mountcastle, 1957] Mountcastle, V. B. (1957). Modality and topographic properties of single neurons of cat9s somatic sensory cortex. *Journal of neurophysiology*, 20(4):408–434.

[Neubig, 2017] Neubig, G. (2017). Neural machine translation and sequence-to-sequence models: A tutorial. *CoRR*, abs/1703.01619.

[Ngiam et al., 2011] Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. (2011). *Multimodal deep learning*, pages 689–696.

[Noh et al., 2015] Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366.

[Pearson, 1901] Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *In EMNLP*.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA.

[Rumelhart et al., 1988] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA.

[Socher et al., 2013] Socher, R., Chen, D., Manning, C. D., and Ng, A. Y. (2013). Reasoning With Neural Tensor Networks For Knowledge Base Completion. In *Advances in Neural Information Processing Systems 26*.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *ArXiv e-prints*.

[Takeuchi, 2013] Takeuchi, L. (2013). Applying deep learning to enhance momentum trading strategies in stocks.

[Wolf et al., 2014] Wolf, L., Hanani, Y., Bar, K., and Dershowitz, N. (2014). Joint word2vec networks for bilingual semantic representations. *Int. J. Comput. Linguistics Appl.*, 5:27–42.

[Yao and Huang, 2016] Yao, Y. and Huang, Z. (2016). Bi-directional LSTM recurrent neural network for chinese word segmentation. *CoRR*, abs/1602.04874.

[Zeiler and Fergus, 2013] Zeiler, M. D. and Fergus, R. (2013). Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901.

[Zhang and Clark, 2011] Zhang, Y. and Clark, S. (2011). Syntactic processing using the generalized perceptron and beam search. *Comput. Linguist.*, 37(1):105–151.