




# Using company-specific headlines and convolutional neural networks to predict stock fluctuations

Jonathan Readshaw<sup>1</sup> · Stefano Giani<sup>1</sup> 

Received: 16 August 2019 / Accepted: 8 July 2021 / Published online: 29 July 2021  
© The Author(s) 2021

## Abstract

This work presents a convolutional neural network for the prediction of next-day stock fluctuations using company-specific news headlines. Experiments to evaluate model performance using various configurations of word embeddings and convolutional filter widths are reported. The total number of convolutional filters used is far fewer than is common, reducing the dimensionality of the task without loss of accuracy. Furthermore, multiple hidden layers with decreasing dimensionality are employed. A classification accuracy of 61.7% is achieved using pre-learned embeddings, that are fine-tuned during training to represent the specific context of this task. Multiple filter widths are also implemented to detect different length phrases that are key for classification. Trading simulations are conducted using the presented classification results. Initial investments are more than tripled over an 838-day testing period using the optimal classification configuration and a simple trading strategy. Two novel methods are presented to reduce the risk of the trading simulations. Adjustment of the sigmoid class threshold and re-labelling headlines using multiple classes form the basis of these methods. A combination of these approaches is found to be more than double the Average Trade Profit achieved during baseline simulations.

**Keywords** CNN · Stock market · Headlines · Trading strategies

**Mathematics Subject Classification** 68T05 · 91B28 · 68T50

## 1 Introduction

Despite suggestions that the stock market is not predictable [1], many investors and researchers seek methods that can provide market fluctuation predictions to aid investment strategy. Advances in machine learning (ML) and natural language processing (NLP) have led to a shift in focus from technical to fundamental analysis. This new approach uses data such as news articles and historical stock prices and is based upon the efficient market hypothesis which states that an asset price reflects all available information [2]. Advances in predictive models have also led to more complex trading strategies. Most research regarding trading strategies and ML is focused on

technical analysis [3, 4]. From an ML perspective, financial markets are nonlinear and noisy systems, therefore, standard techniques to see through the noise can be applied, see [5] for a comparison of common techniques. Some researchers consider news and other fundamental data as part of their strategy [6], however, the development of trading strategies based solely on fundamental data is rare throughout the relevant literature.

Macroeconomic factors that drive wider market trends are believed to have a far greater influence on price trends for a given stock than specific headlines. Various studies have been conducted to analyse the relationship between macroeconomic variables and stock price trends. For example, [7] demonstrated a high correlation between gold price, exchange rate and stock prices for the Indian market. Similarly, [8] analyses the relationship between changes in Federal Reserve policy (e.g. Interest rates) and subsequent trends in the stock market, finding a strong trend across multiple sectors including Utilities and Financial.

Generalized autoregressive conditional heteroskedasticity (GARCH) has proven a popular statistical method for

---

✉ Stefano Giani  
stefano.giani@durham.ac.uk

Jonathan Readshaw  
readshawjonathan@gmail.com

<sup>1</sup> Department of Engineering, Durham University, Lower Mountjoy, South Road, Durham DH1 3LE, UK

evaluating market volatility [9] and is still widely used. Poon and Granger [10] summarise volatility modelling approaches that can be classified into two groups; historical volatility models and volatility implied from options. Works such as [11] and [12] find that GARCH and implied volatility models have individual strengths and tend to be influenced by separate factors, and ultimately a combination of both approaches can produce better results. This observation is in line with recent advances in volatility modelling that involve the use of hybrid models such as in [13] where various technical indicators are considered. However, all of the aforementioned approaches tend to focus on technical analysis without consideration of other factors such as news or Social Media.

Early research shows no relation between headlines and stock volatility [14], however, the development of more advanced predictive models and availability of larger datasets has led to more accurate market trend predictions based on headlines. Although complete news articles [15] or social media content [16, 17] are used in some works, the use of headlines has become the most common in this area of research due to the belief that they contain less noise than other sources of textual data [18]. This does not mean that other textual data sources are redundant; for instance, [19] shows how external sources such as Social Media can influence markets in multiple ways. Similarly, [20] shows how Social Media can have an indirect influence on the stock market by influencing consumer investor behaviour. Headlines are commonly sourced from major financial news outlets such as The Wall Street Journal [21]. A wide range of prediction targets are considered throughout the relevant literature, including major indices such as the S&P 500 [22] and collections of individual companies [17]. The time span of market fluctuations analysed is also varied. For example, Mittermayer [23] focuses on intra-day predictions, whereas long-term trends are briefly considered in the work of Ding [24]. Methods such as support vector machines [22] and complex decision trees [17] remain popular for predictive tasks of this nature. These commonly use a Bag of Words (BoW) feature representation approach, where words are represented independently without consideration of word order or context. Variations of this method include  $N$ -gram BoW, where phrases of length  $N$  are extracted as features as opposed to single words, and Term Frequency-Inverse Document Frequency (TF-IDF), which introduces the consideration of a word's frequency within a sentence and across the entire collection. However, these representations typically lead to sparsity issues when applied to a large corpus [25]. Probabilistic approaches such as the Naive Bayes method can also be applied to tasks of this nature [26].

The development of artificial neural networks (ANNs) has provided new classification and feature

representation methods for text-based tasks. An ANN is a collection of nodes known as neurons that are interconnected in layers. Originally proposed by Rosenblatt [27], the architecture is based on the transmission of signals and firing of biological neurons in a nervous system. Variations in the basic ANN architecture have been made to produce types of neural network with additional mathematical features suited to different tasks, see for example [28–30]. More recently, other aspects of ANNs gained popularity in the literature. Among them, there is the topic of self-organising ANNs [31, 32] which studies methods for ANNs to design themselves. A particularly popular type of ANN nowadays is convolutional neural networks (CNNs) commonly used for image recognition. CNNs utilise a convolutional layer to detect patterns in input data that can be used for accurate classification or prediction. For example, in image detection, these patterns may represent edges and shapes of a specific object depicted by its pixel values. More recently, CNNs have gained popularity in text-based tasks demonstrating state-of-the-art performance in multiple NLP tasks, including sentence classification [33] and sentence modelling [34]. Some applications of CNNs to market prediction exist in the literature, both for major indices [24] and discrete price prediction [35].

This work presents a CNN for predicting next-day stock price fluctuations of three major technology companies using headlines relating to each company. Next-day returns are used due to the inability to access the large amounts of historical intra-day stock price data required for intra-day fluctuation prediction. However, the effect of news headlines has been found to resonate during the next-day period [24]. Experiments are conducted to identify an optimal model configuration for trend classification in terms of the word embedding and convolutional layer states. Using class predictions from these experiments, trading simulations are presented based on day-averaged predictions for each asset. Finally, modifications to both the baseline trading strategy and labelling of headlines are made with the intention of reducing risk present in simulated trading.

The novelty and contribution of our work are primarily twofold: (1) We succeed in trained a shallow network with good overall performances, making our network easier and faster to train. This was also possible by employing multi-filters and an embedding layer able to extract more relevant information for the task. (2) More sophisticated strategies for trading based on multi-class to exceed the limitations of traditional binary-class strategy and threshold to incorporate in the strategies the confidence of the network in the class selection.

## 2 Model

As discussed, an ANN consists of interconnected layers of neurons. A neuron is a mathematical operator that receives one or more inputs and performs a weighted sum to generate its output. This output is often passed through an activation function that is chosen depending on both the positioning of the neuron in the network and the task that the network is applied to [36]. Activation functions are used to introduce nonlinearity to the network, allowing for more complex mappings between inputs and outputs in the network. The network *learns* by optimising each neuron's weightings to reduce the overall loss present in the system. In this work, a convolutional neural network (CNN) is implemented to carry out both binary and multi-class classification tasks. The overall structure of the network is comparable to those found in text-based CNN tasks throughout the relevant literature [37, 33] but with modifications to reduce dimensionality whilst retaining accurate classification. Figure 1 shows a general schematic of the network, outlining the constituent layers.

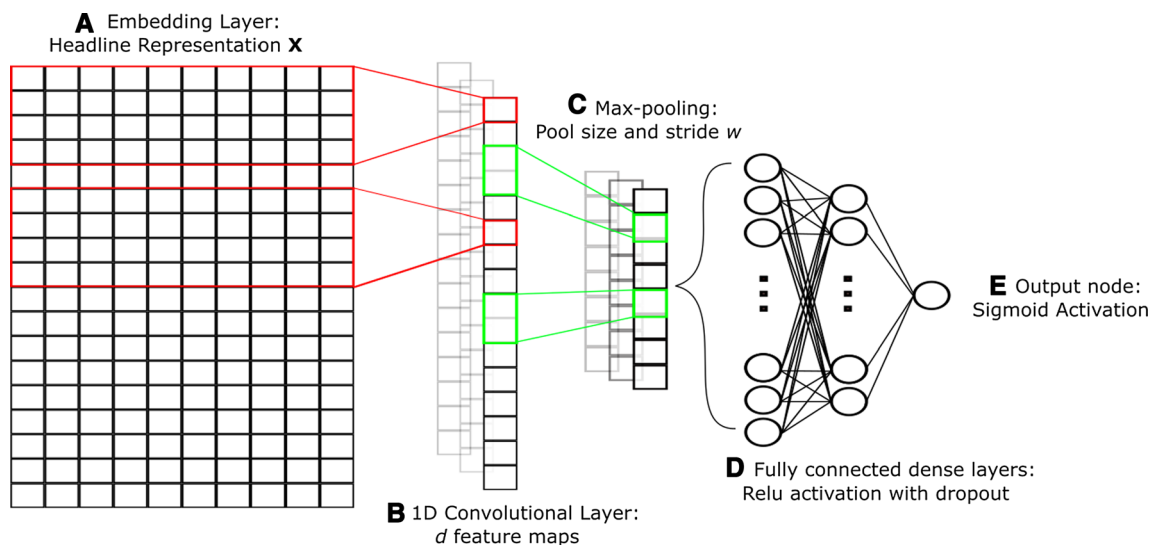
### 2.1 Preprocessing

Raw headlines are cleaned using a tokenization algorithm that converts text to lower case, separates each sentence into its constituent words whilst removing stop-words and punctuation ('and', 'or', ':', etc.) [38]. The remaining words in the collection of tokenized sentences are selected as features  $u$  and form a vocabulary  $V$ . The selection of uni-gram features (i.e. single words) is adequate here, as

phrases are evaluated using specific filter widths in the network's convolutional layer. Ordinal encoding is applied to the tokenized sentence to give vectors  $\{i_1, i_2, \dots, i_n\}$  where  $i_k$  is a unique integer index corresponding to feature  $u_k \in V$ . Here,  $n \in [1, m]$  is the length of each individual tokenized sentence. To simplify the implementation of the model, post-padding is applied to each vector to achieve uniform dimensionality across the set of encoded vectors. Dummy features, represented by a 0, are appended to each vector to achieve a set of  $m$ -dimensional vectors where  $m$  is the length of the longest tokenized sentence. For each vector  $\{i_1, i_2, \dots, i_n\}$ ,  $m - n$  zeroes are appended to the vector. Post-padding preserves word order by only adding dummy features to the end of each vector. The result of preprocessing is a set of encoded feature vectors  $\mathbf{i}_{\text{pad}} \in \mathbb{R}^m$  that are input to the network.

### 2.2 Embedding

Word embeddings are utilised to represent each feature  $u$  in a  $p$ -dimensional vector space, where  $p \ll \dim(V)$  to reduce the dimensionality of the problem. These representations aim to represent the semantic and syntactic context of features and the relation between similar features. Words that are interchangeable within a certain context or that often appear within close proximity of each other in a sentence are represented by similar vectors in the  $p$ -dimensional space and are therefore interpreted similarly by the rest of the network. Capturing context and semantics is not possible with representations such as Bag of Words or TF-IDF where features are represented with no relation to



**Fig. 1** General model schematic where letter labelling corresponds to subsections in Sect. 2. Preprocessing that precedes (A) is not included in the schematic. Note the embedding sentence representation  $\mathbf{X}$  is

portrayed as a matrix here with dimensions  $h \times p$  for ease of presentation, however, in the implemented model,  $\mathbf{X} \in \mathbb{R}^{hp}$  is a 1-dimensional vector

each other. In this work, both randomly initialised and pre-learned embeddings are tested. When using randomly initiated embeddings, each  $p$ -dimensional vector is initiated with random values and is tuned during the network's training epochs to form a vector representative of the feature's context in the training collection, and how it relates to other features. The random values that are used for initiation can be selected from a range of common statistical distributions. A selection of methods including random-normal and random-uniform is briefly tested during the implementation of the model. Pre-learned embeddings have been tuned on large collections of unlabelled textual data. Many well-defined collections of pre-learned embeddings formed from training on a variety of sources are available. The pre-learned embeddings used in this work are trained on a collection of 10 billion words in a Google News dataset [39]. The embeddings are formed using the *word2vec* method developed by Mikolov [40]. Two configurations of these pre-learned embeddings are tested in this work; *static* and *non-static* modes, as in the work of Kim [33]. In the *static* configuration, pre-learned embeddings are unaltered from their original state. However, in the *non-static* configuration, the network's embedding layer fine-tunes pre-learned vectors to better suit the use of a feature within the specific task. If a feature does not have a pre-learned vector representation in the *word2vec* collection, it is randomly initiated. Furthermore, the embedding representation of the dummy feature added during post-padding is initiated as a zero vector and remains constant throughout training for all of the configurations discussed. Each integer  $i_k$  maps to a unique embedding  $\mathbf{x}_k \in \mathbb{R}^p$  representing feature  $u_k \in V$ . Hence, for each encoded padded feature vector  $i_{\text{pad}}$ , the embedding layer returns the feature vector  $\mathbf{X} \in \mathbb{R}^{mp}$  formed by the concatenation of embeddings  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m$ .

## 2.3 Convolution

In the context of textual analysis, a convolutional layer contains a number of filters that are trained to detect similar or contrasting context and sentiment in groups of adjacent words. Feature maps representing the nature of these phrases comprise of dot-product results from each sliding filter. This method can detect semantically similar phrases due to feature relations expressed using word embeddings. The close proximity of vector representations for features in semantically similar phrases results in the calculation of similar dot-product results. Filter width  $h$  dictates the length of phrases that are to be evaluated. For example, a width  $h = 2$  produces feature maps based on bi-grams represented in  $\mathbf{X}$ . Consider a filter  $\mathbf{q} \in \mathbb{R}^{hp}$  sliding over a sentence represented by  $\mathbf{X}$ . The first feature map element

$c_1$ , which represents the first phrase of  $h$ -adjacent words in  $\mathbf{X}$ , is given by

$$c_1 = g(\mathbf{q} \cdot \mathbf{X}_{1:hp} + b_1) \quad (1)$$

where  $b_1$  is a trainable scalar bias term and  $g$  is an activation function. The notation  $\mathbf{X}_{1:hp}$  is used to represent the elements of  $\mathbf{X}$  from position 1 to  $hp$  inclusive. Rectified linear unit activation (relu) is used in this work and is defined as:

$$g(x) = x^+ = \max(0, x) \quad (2)$$

Each filter produces a feature map  $\mathbf{c} \in \mathbb{R}^{m-h-1}$  with elements representative of each phrase in  $\mathbf{X}$ . A single stride is implemented in this layer such that the filter slides by a single word to produce the next term in the corresponding feature map. This ensures every possible adjacent phrase of length  $h$  in the headline is considered. Using a single stride, a general expression for the  $k$ th element in a feature map  $c_k$  can be formed:

$$c_k = \max(0, \mathbf{q} \cdot \mathbf{X}_{((k-1)p+1):khp} + b_k) \quad (3)$$

Using  $d$  unique filters results in  $d$  corresponding feature maps. Fewer filters are used in this work than is typical to reduce the dimensionality of the task and hence improve efficiency despite the presence of additional hidden layers. This work explores the effect of varying the filter width  $h$  and the potential benefits of using sets of filters with different widths within a single layer. Using multiple filter widths in this layer aims to detect word patterns of different length in the original headline. Each filter is tuned during training to achieve feature maps that best represent the context of phrases and their relative importance for classification. The result of the convolutional layer is a collection of  $d$  feature maps that represent groups of  $h$ -adjacent features in  $\mathbf{X}$ .

## 2.4 Max-pooling

A max-pooling layer is used to down-sample each of the feature maps produced by the convolutional layer. Using a pool-size and stride given by  $w$ , each maximum value from groups of  $w$  adjacent elements in a feature map is sampled. During training, filters in the convolutional layer are tuned so that feature map elements corresponding to phrases that are highly relevant for the classification of  $\mathbf{X}$  are large. Therefore, sampling the maximum of a group of feature elements reduces the size of the problem whilst retaining representations of phrases that are vital for classification. Consider a pool size and stride  $w = 2$ , the pooled feature map  $\mathbf{c}^{\text{pool}} \in \mathbb{R}^{\frac{1}{w}(m-h-1)}$  corresponding to the original feature map  $\mathbf{c}$  is given by:

$$\mathbf{c}^{\text{pool}} = \{\max(c_1, c_2), \dots, \max(c_{m-h-2}, c_{m-h-1})\} \quad (4)$$

The remaining pooled feature maps are concatenated to form an input vector  $\mathbf{z} \in \mathbb{R}^{d(m-h-1)}$  for the hidden layers of the network. It is common for a max-over time method to be used in networks with a large number of filters  $d$ , where a single maximum element in each feature map is sampled [37].

## 2.5 Fully connected hidden layers

Two fully connected hidden layers are utilised in this work. The use of more than one fully connected layer is not common and aims to improve the transmission of relationships detected by convolution through the latter layers of the network. The activation of a given neuron  $z_k$  based on the previous layer of  $l$  neurons  $\mathbf{z}_{\text{prev}} \in \mathbb{R}^l$  is given by

$$z_k = \mathbf{z}_{\text{prev}} \cdot \mathbf{w}_k + b_k \quad (5)$$

where  $\mathbf{w}_k \in \mathbb{R}^l$  is a trainable weights vector and  $b_k$  is a trainable scalar bias term, both of which correspond to neuron  $z_k$ . Relu activation is applied to  $z_k$  as shown by Eq. 2. The weights  $\mathbf{w}_k$  are adjusted during training using the formula:

$$\Delta \mathbf{w}_k = -\eta \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_k}, \quad (6)$$

where  $\eta$  is the learning rate and  $J(\mathbf{w})$  is the discrepancy, loss or error between the predicted class and the ground truth, both terms are discussed in Sect. 2.6. The term  $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_k}$  can be understood as the sensitivity of the error on the change in value of the weight  $\mathbf{w}_k$ . If the value of  $J(\mathbf{w})$  is strongly dependent on the value of  $\mathbf{w}_k$ , the term  $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_k}$  is bigger in magnitude, as well as  $\Delta \mathbf{w}_k$ . Also, the bias terms  $b_k$  are updated with a similar formula:

$$\Delta b_k = -\eta \frac{\partial J(\mathbf{w})}{\partial b_k}. \quad (7)$$

In the case of the bias terms, the term  $\frac{\partial J(\mathbf{w})}{\partial b_k}$  is not dependent on the input coming from the previous layer. More details on the backpropagation process can be found in [41].

Dropout is utilised in each hidden layer to reduce overfitting and the time required to train the network. Overfitting occurs when a model fits the training set too closely and hence is unable to make accurate predictions based on new, unseen data. This can arise due to the high-dimensional nature of text-based tasks, often referred to as the Curse of Dimensionality [42]. Hence, a proportion of neurons at each layer is made inactive according to the specified dropout rate to reduce training dimensionality. Each trained weights vector  $\mathbf{w}$  is scaled during testing to

account for the probability of a neuron's exclusion due to drop-out.

## 2.6 Output node

For binary classification tasks undertaken in this work, a single output node is used with sigmoid activation. The sigmoid function returns values between 0 and 1 which can be interpreted as the probability  $\sigma(z)$  that an input headline belongs to class 1 and is defined as

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (8)$$

where  $z$  is the output neuron's activation as given by Eq. 5. In this work, class 1 corresponds to a next-day asset price increase, whereas class 0 represents a price decrease or continuation. Class 1 membership is allocated using a threshold  $\sigma(z) \geq 0.5$ . Development of a reduced risk trading strategy in this work alters this threshold to create a stricter margin. Multi-class classification is also undertaken in this work as part of trading strategy development. In these experiments, the output layer of the network contains three neurons where class probability is determined by softmax activation

$$S(z_k) = \frac{z_k}{\sum_{k=1}^3 e^{z_k}} \quad (9)$$

where  $S(z_k)$  is the probability of membership to the class corresponding to neuron  $z_k$ . The three classes used correspond to 'buy', 'inconsequential' and 'avoid' trading instructions and are discussed further in Sect. 6.

## 2.7 Training

For binary classification tasks, a binary cross-entropy loss function is used. Loss provides a measure of how accurately the network classifies the training data using the current set of weight functions. The loss (or cost)  $J(\mathbf{w})$  of the current weight functions  $\mathbf{w}$  (including bias terms) is expressed as

$$J(\mathbf{w}) = -\sum_k y^{(k)} \log(\sigma(z^{(k)})) + (1 - y^{(k)}) \log(1 - \sigma(z^{(k)})) \quad (10)$$

where  $y^{(k)}$  is the correct class label for each of the  $k$  training samples. A categorical cross-entropy loss function is used in multi-class tasks and is similar in nature to the binary cross-entropy function discussed. Labelling is based on the next-day stock price change of each asset and is described in depth in Sect. 3. Weights are updated based on batches of training data through gradient-descent and backpropagation. This method updates weights and bias terms to



converge towards a global cost minimum  $J_{min}(\mathbf{w})$ , see Sect. 2.4 for the formulas used to update the weights in the fully connected layer. Similar formulas are used also to update the values in the filters of the CNN layers. Each trainable word embedding and convolutional filter is also tuned to better represent features and phrases using gradient descent. The global minimum represents the state in which the model is fit to the training data with the least possible error. This implies that this minimum also represents a collection of embeddings that best represent context and semantics and a set of convolutional filters that are best at detecting word patterns vital for classification. The speed at which the model's weights approach this global minimum is determined by its learning rate  $\eta$ . If the learning rate is too large, the model risks overshooting the global minimum, however, it must be large enough to converge at a suitable rate. In all simulations, the learning rate  $\eta$  is set to 0.001. The model is trained using the Adaptive Moment Estimation method (ADAM) [43]. ADAM is a stochastic gradient descent method which differs from a "pure" gradient descent by the fact that the gradient is not computed considering all the available dataset but just a randomly chosen small sample called a batch. Using the entire dataset may be more accurate but computationally impracticable. A disadvantage of randomly choosing a sample is that the gradient is varying a lot depending on what data are in the sample making the overall computation of the gradient a very noisy procedure. To mitigate this, ADAM uses running averages of both the gradients and the second moments of the gradients. The two running averages are controlled by the exponential decay rates  $\beta_1$  and  $\beta_2$  with values 0.9 and 0.999, respectively, in all simulations. The number of epochs used during training varies depending on the set-up and they are all reported in Sect. 5.

## 2.8 Network architectures

In this work, we used three variations of a CNN classifier composed of one or more CNN layers and two dense layers. In Tables 1, 2 and 3, the structures of the three networks are described. All networks are structured as a stack of layers where the output from the embedding layer

**Table 1** Structure of the single-width network

Layer	Parameters
Dense	Size = 1, activation = sigmoid, dropout = 0.3
Dense	Size = 100, activation = relu, dropout = 0.3
Max-pooling	Pool size = 2, stride = 2
Convolution	Filters = 36, kernel size = <i>varying</i> , activation = relu

**Table 2** Structure of the multi-width network

Layer	Parameters
Dense	Size = 1, activation = sigmoid, dropout = 0.3
Dense	Size = 100, activation = relu, dropout=0.3
Max-pooling	Pool size = 2, stride = 2
Convolution	Filters = 12, kernel size = <i>varying</i> , activation = relu
Convolution	Filters = 12, kernel size = <i>varying</i> , activation = relu
Convolution	Filters = 12, kernel size = <i>varying</i> , activation = relu

**Table 3** Structure of the multi-class network

Layer	Parameters
Dense	Size = 3, activation = softmax, dropout = 0.3
Dense	Size = 100, activation = relu, dropout = 0.3
Max-pooling	Pool size = 2, stride=2
Convolution	Filters = 12, kernel size = <i>varying</i> , activation = relu
Convolution	Filters = 12, kernel size = <i>varying</i> , activation = relu
Convolution	Filters = 12, kernel size = <i>varying</i> , activation = relu

is going into at the bottom of the stack and the classification is coming out from the top of the stack. The values of parameters used in the networks are reported. For some of them, the reported value is *varying* meaning that the values of such parameters are not fixed and discussed in Sects. 5 and 6. In all networks, the overall number of CNN filters is 36, grouped in just one layer in the single-width network or three CNN layers as in the multi-width and the multi-class network. The optimisation of the single-width and the multi-width networks is discussed in Sect. 5, whilst the optimisation of the multi-class network is discussed in Sect. 6.2.

## 3 Dataset

This work uses two datasets from a single source to extract useful data. The first is a collection of dated headlines relating to various publicly traded from January 2007 to December 2016. Historical market data for these companies are provided in the second dataset. This market data are used to label headlines according to price change and does not contribute to any input variables for the model. It is only company-specific headlines that are used to predict market movements. From the original datasets, data relating to three technology companies, Amazon, Apple, and Microsoft, are used in this work [44]. This is due to the abundance of headlines relating to these companies, and hence, they form a mock portfolio used in later trading

strategy development. Along with the headline itself, the asset that the headline relates to and the date and time of its first creation are extracted from the first dataset. A relevance score is provided for each headline in the dataset where a relevance of 1 indicates the presence of the asset name in the headline. Only headlines containing the relevant asset name are used in this work; hence, headlines with a relevance of 1 are extracted.

For binary classification tasks, each headline is labelled depending on the change in the next-day stock price of the relevant asset. If the asset price at market close of the next trading day is greater than the price at market open, a positive return would be made on a single-day investment and hence the headline is labelled as class 1. If the asset price is unchanged or decreases over the next-day close-open period, the headline is labelled as class 0. Using next-day returns has two clear benefits for evaluating model performance and its practical use. Firstly, using next-day returns allows for equal treatment of intra-hours and out-of-hours news, whereas same-day returns would require substantial consideration of the time of release. Secondly, all headlines relating to an asset throughout the course of a single day can be evaluated by the model individually and subsequent next-day trading decisions can be made based on the collection of predictions for the entire day. This is explored further in the development of a trading strategy in this work.

It is not suitable to randomly split the data into training and testing sets in this context due to the fact that headlines from different sources describing the same event could appear in both sets, therefore, creating unwanted bias. Testing data are instead compiled of half-hourly unique headlines. The nature of the original dataset means headlines from different sources describing the same event appear within a very small time window. Hence, if there is only a single headline for an asset in a half-hour period, it can be concluded that the event it describes is unique to that headline. These time-unique headlines are therefore selected for use as testing data to avoid overlapping topics in both the training and testing sets. Only testing headlines on days where each asset has at least one half-hourly unique headline are selected. This ensures that the testing date range for each asset is identical and allows for a fair comparison of trading performance across the portfolio. Selecting testing data across the full range of dates in the collection aims to negate the effects of general market movements. For example, if testing data were restricted to 2016, returns would be affected by global events in that year and skew the results of the various experiments undertaken in this work. Furthermore, the high-frequency nature of trades simulated in this work ensures that long-term market trends are not important. Hence, the date range from which headlines are taken is not an important factor

when analysing the simulation results presented. There are many cases in the relevant literature of tasks similar to those undertaken in this work that simply use a random cross-validation testing split. Although these works demonstrate high accuracies, this is mainly due to a large number of overlapping topics in training and testing creating bias.

## 4 Experimental procedure

For experiments undertaken in this work, pre-learned *word2vec* embeddings of dimension  $p = 300$  are used, with  $d = 36$  total filters used in the convolutional layer. For configurations of the model that use filters with multiple widths, the total number of filters remains at 36. Specifically, three different widths are implemented with 12 filters each. A pool size and stride of  $w = 2$  are implemented in the max-pooling layer. The number of training samples is 43060, and the number of testing samples is 7395 spanning 838 unique days. Gradients are updated based on training batches of size 32. The number of epochs and dropout rate varies depending on the model configuration and is optimised using a grid search [45].

The first collection of experiments undertaken aim to identify an optimal configuration for the CNN presented. This involves identifying an optimal filter width  $h$  for the convolutional layer, comparing the effectiveness of a multiple filter width model with that of a single filter width model and studying the effect of using randomly initiated word embeddings and pre-learned embeddings. The result of this experiment will be a configuration of the discussed model and associated class probabilities which are used for the development of risk-minimising trading strategies. Two metrics are used to evaluate the classification performance of each model configuration; accuracy and F1-score. Model accuracy can be expressed generally as the ratio of the number of correct class predictions to the total number of predictions. F1-score is used to account for both the precision and recall of each configuration. Using a single metric to account for both of these measures allows for easier identification of an optimal configuration. Accuracy is quoted due to its common presence in the relevant literature for classification tasks. Precision and recall are defined as

$$\text{PRE} = \frac{\text{TP}}{\text{FP} + \text{TP}} \quad \text{REC} = \frac{\text{TP}}{\text{FN} + \text{TP}} \quad (11)$$

where TP is the number of true positive predictions, FP the number of false positive predictions and FN the number of false negative predictions. Further interpretation of precision is discussed in Sect. 6 where it forms the basis for a common metric used to evaluate day-averaged predictions

as opposed to those based on individual headlines. The corresponding F1-score is defined as:

$$F1 = 2 \times \frac{PRE \times REC}{PRE + REC} \quad (12)$$

## 5 Optimum model configuration

For results discussed in this section, the following terms are used to refer to different configurations of the CNN outlined in Sect. 2. Firstly, *single-width* is used to denote a configuration that uses a single filter width  $h$  in the convolutional layer, whereas *multi-width* denotes a model that uses three different values of  $h$ . Word embedding methods are described as *self-learned* for randomly initiated vectors, and *static* or *non-static* for non-trainable and trainable pre-learned embeddings, respectively. For example, a configuration labelled as *single-width self-learned* describes a model using randomly initiated word embeddings with a single filter width in its convolutional layer. Tables 4 and 5 show the classification results, optimal filter width(s)  $h$  and number of training epochs for *single-width* and *multi-width* implementations, respectively. Although the stated accuracies are lower than in other NLP tasks such as sentence classification [33], it is important to consider the nature of the task. Unlike other classification tasks, predicting stock price movements is heavily dependant on factors beyond the textual content of the headline such as general market movements and trader behaviour. It is common for a headline to have a positive sentiment towards a particular asset but for the price of the asset to decrease. Factors such as increased leverage and information possessed by traders that are not readily available can cause these results. However, the results obtained in this work are significantly better than random guessing (50%) and therefore it can be stated that the various implementations of the model are able to predict short-term market trends solely based on news headlines.

**Table 4** Classification metrics, optimal filter width and number of training epochs for single-width implementations

	Self	Static	Non-static
Accuracy [%]	59.6	57.4	<b>61.5</b>
F1-score [%]	57.6	57.0	<b>58.7</b>
Filter width $h$	3	4	4
Epochs	5	7	7

Best results are highlighted in bold

**Table 5** Classification metrics, optimal filter width and number of training epochs for multi-width implementations

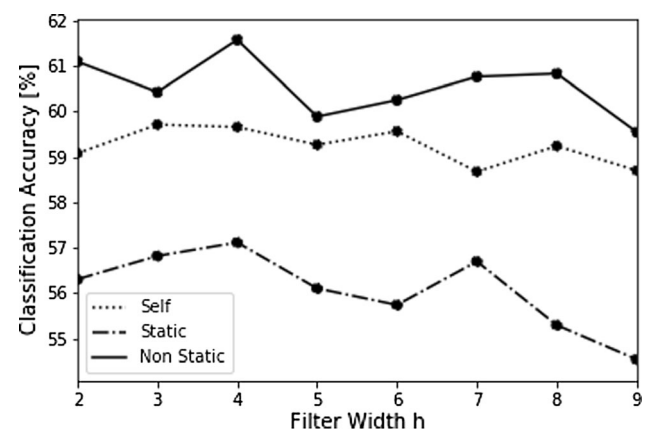
	Self	Static	Non-static
Accuracy [%]	59.2	56.5	<b>61.7</b>
F1-score [%]	58.3	56.5	<b>59.2</b>
Filter widths $h$	4, 6, 8	3, 4, 5	3, 4, 7
Epochs	5	7	9

Best results are highlighted in bold

### 5.1 Effect of Filter Width

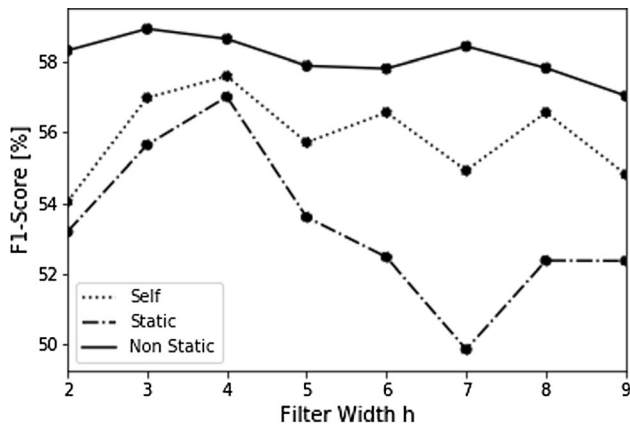
Table 4 provides the optimum filter width  $h$  for each *single-width* implementation of the model used in this work. Using a grid search over widths  $h \in [2, 9]$ , the effect of this parameter on classification accuracy and F1-score is established. Figures 2 and 3 show the variation in the relevant performance metrics with filter width  $h$  for *single-width* implementations of the model. As seen in Table 4, the optimum single width is similar for each word embedding state. The best classification results are produced by evaluating tri-gram or quad-gram phrases in the convolutional layer. These results can be explained by considering phrases within a headline that depict its sentiment. Phrases with a noun-verb-adverb structure tend to summarise the tone of a headline in its entirety. For example, the phrase “*shares fall sharply*” within a particular headline provides all the necessary information to make an accurate prediction regarding next-day stock price movement. Other information is commonly of little importance and can be considered noise. Hence, each model demonstrates an optimum filter width of 3 or 4 where these vital phrases can be represented in their entirety.

Each word embedding state also demonstrates good accuracy at some larger filter widths. This suggests that the



**Fig. 2** Variation in accuracy with filter width  $h$  for each *single-width* model configuration





**Fig. 3** Variation in F1-score with filter width  $h$  for each *single-width* model configuration

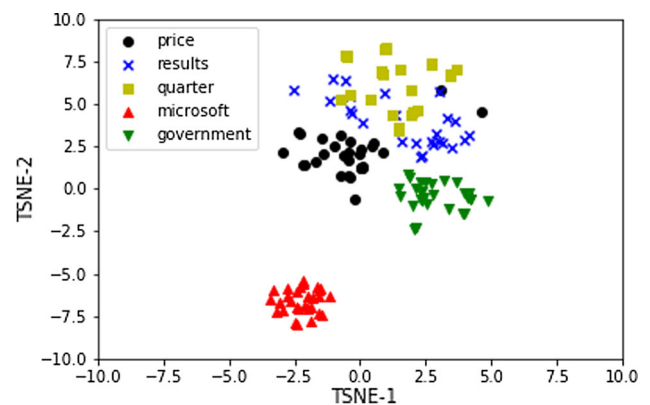
network is able to represent longer phrases that provide relevant information for predicting price fluctuations. It may be the case that combinations of the discussed three- and four-word phrases that carry sentiment can be represented as a single feature map element. Using the results obtained, each *multi-width* model implementation uses the three best feature widths in terms of F1-score for each word embedding state in the corresponding *single-width* model. Selection is based on the F1-score due to the relevance of precision and recall to latter trading simulations, where false positives and negatives are considered.

Table 5 shows the classification results for each *multi-width* implementation of the model. Although the *non-static multi-width* model configuration produces the best classification results for both metrics considered, the use of multiple filter widths for *static* and *self-learned* embeddings results in worse classification performance than in the equivalent *single-width* cases. In overall, the progression to multiple filter widths produced no consistent benefit to model performance across the word embedding states. This result can be explained by the reduced number of filters  $d$  used for each optimum filter width shown in Table 4. As outlined in Sect. 4, the number of total filters in the convolutional layer is equal for both *single-width* and *multi-width* implementations. For example, in the *single-width static* configuration, there are 36 filters with width  $h = 3$ , however, in the equivalent *multi-width* configuration, there are only 12 filters of this width. Therefore, despite the ability to represent phrases of different lengths using multiple filter widths, there are fewer filters available to represent the key tri-gram or quad-gram phrases discussed previously. Therefore, to observe significant improvements in classification using multiple filter widths, it is expected that the number of filters for each width would have to be greater than or equal to the total number of filters used in the *single-width* implementation. However, this increases

the dimensionality of the network and the time required for training. Furthermore, the ability to detect the presence of each phrase is dependant on the accurate tuning of each filter through gradient-descent.

## 5.2 Word embeddings

Figure 4 shows a projection of a sample of pre-learned *word2vec* word embeddings on a two-dimensional space. Each point corresponds to a word whose 300-dimensional vector representation is part of the 30 most similar vectors to the associated reference word shown in the figure legend. In this work, vector similarity is calculated using cosine similarity. The 300-dimensional embeddings are projected onto a two-dimensional space using t-distributed stochastic neighbour embedding (t-SNE) [46]. This method models each high-dimensional vector as a two-dimensional point such that similar high-dimensional vectors are modelled as similar points in the low-dimensional space. Hence, the similarity of high-dimensional word embeddings can be visualised. The clustering of points that represent vectors that are similar to a specific reference word demonstrates the principle of word embeddings, where terms used in a common context have similar positions in the 300-dimensional word embedding space. These terms are therefore interpreted similarly by convolutional filters. Figure 4 also shows the formation of larger clusters containing points corresponding to different reference words. This identifies larger groups of terms that have a similar context or that could appear in close proximity within a sentence. For example, the overlap between clusters corresponding to ‘quarter’ and ‘results’ could arise from frequent use of the phrase ‘third quarter results’. This overlapping, therefore, demonstrates how the use of word embeddings not only accounts for if terms are



**Fig. 4** Two-dimensional representation of the word embeddings of the 30 most similar terms to each reference term shown in the legend. Similarity between vectors is calculated by cosine similarity. Mapping of 300-dimensional word embeddings to two dimensions is achieved using t-SNE

interchangeable within a certain context but also how frequently terms appear in close proximity within a text. The isolation of terms similar to ‘microsoft’ (e.g. ‘photoshop’) is due to the specificity of these terms and the lack of any contextual relationships to the other reference words shown without fine-tuning.

For *single-width* and *multi-width* implementations of the model, word embeddings in the *non-static* state lead to the best classification accuracy and F1-score. Table 6 provides the most similar term to the terms shown in Fig. 4 for the *static* and *non-static* configurations following training. It can be seen that in the *non-static* state, embeddings corresponding to words that can be used in a range of contexts are tuned to better represent the context of investment news and the stock-market. For example, the similarity between ‘quarter’ and ‘half’ in the *static* state is a general relation that can be applied to many different contexts however the similarity between ‘quarter’ and ‘q2’ in the *non-static* state suggests a relationship based on a financial context. This refinement in context allows for the model to predict market trends based on headlines with greater accuracy. Table 6 also shows how words typically used in a consistent context (i.e. proper nouns) undergo little refinement. This embedding refinement is demonstrated across the literature for sentence classification tasks where the target variable is solely dependent on the textual data [33]. However, it is interesting to observe similar behaviour in this work where the model is still able to refine representations based on context despite the target variable depending on factors beyond the context of the headline (e.g. overall market behaviour).

Various initiation methods for *self-learnt* embeddings were tested. The best classification results were obtained using a random normal initiation, where vectors are initiated randomly with a normal distribution with mean and standard deviation equal to those of the pre-learned embeddings used in this work. The classification results using *self-learnt* configurations are better than those of *static* configurations for both *single-width* and *multi-width*

models. This result suggests that in the *self-learnt* configuration, the model is able to suitably learn a set of embeddings that represent the context of the collection. However, *self-learnt* embeddings often over-fit the context of the training data and fail to represent similarities between words that may be interchangeable in different contexts. This is because *self-learnt* embeddings are formed solely on the context of the training set and cannot account for words that do not appear in training. Additionally, if the context in which a word is used in a testing, headline is slightly different to that found in training, the self-learnt embeddings misrepresent this due to the limited word relationships that can be established from the small training set. This results in less accurate classification than *non-static* configurations where embeddings retain relations based on a variety of contexts from their initial states despite fine-tuning. For example, ‘half’ remains the fifth closest word to ‘quarter’ using *non-static* embeddings. A much larger training set would be required for general context relationships to be represented in *self-learnt* embeddings. These observations, therefore, suggest that *non-static* embeddings provide the best configuration not only because of their ability to be fine-tuned to the task in question but also because a more general context of words is retained in the embeddings allowing for better application to both unseen headlines and new tasks.

### 5.3 Overall optimal model

Based on the experimental results discussed in this section, it can be concluded that the best configuration of the model for predicting fluctuations in next-day stock prices is a *multi-width* implementation using *non-static* word embeddings. A *single-width* implementation using *non-static* embeddings would also be suitable for this task as the advantage of using multiple filter widths instead of a single width is not uniform across each word embedding state. *Self-learnt* embeddings are not suitable for implementation due to their tendency to over-fit the context of training headlines and inability to represent words not present in training. Hence, it is expected that the embeddings would not be suitable for testing headlines relating to companies from different sectors or textual data from a different source (e.g. Twitter). Conversely, *static* embeddings are unable to suitably represent the specific context of the task. The maximum classification accuracy achieved is comparable with those achieved by state-of-the-art methods [24], however, the comparison of performance across works that use different datasets is not conclusive. Results are heavily dependant on the number and quality of headlines in the dataset used, and since there is no standard dataset that is used by the majority of works in this field, it is hard to fairly compare performance. This is not the case in other

**Table 6** Most similar terms based on cosine similarity of word embeddings for both *STATIC* and *NON-STATIC* configuration of the network’s convolutional layer

Term	Most similar term	
	<i>Static</i>	<i>Non-static</i>
‘results’	‘findings’	‘earnings’
‘price’	‘premium’	‘stock’
‘quarter’	‘half’	‘q2’
‘microsoft’	‘adobe’	‘adobe’
‘government’	‘administration’	‘administration’

fields such as image recognition, where baseline datasets exist. The next section aims to develop a trading strategy based on the *multi-width non-static* model implementation to manage risk and prevent incorrect stock purchases.

## 6 Trading simulations

This section seeks to evaluate the performance of the discussed CNN model using a simple trading strategy. It is first necessary to assess whether the optimal model configuration in terms of individual headline classification results in the best trading results. The baseline strategy used in this section is as follows. For each unique day in the testing set, the mean of the sigmoid outputs  $\sigma(z)$  for headlines relating to each individual asset is calculated. This mean value  $\sigma(z)_{\text{mean}}$  aims to represent a prediction based on all the headlines relating to the associated asset throughout the day. If  $\sigma(z)_{\text{mean}} > 0.5$ , shares in the relevant asset are purchased on the next trading day at market opening and sold at market close. Simulated returns can therefore be calculated using the next-day close-open price change of the asset. Using this strategy, a metric that evaluates the trading performance of the model can be formed based on precision. In this context, a true positive (TP) indicates a prediction  $\sigma(z)_{\text{mean}} > 0.5$  where the next-day returns are positive, whereas a false positive (FP) describes such a prediction where the next-day returns are negative. Therefore, precision describes the proportion of next-day investments made based on the model's predictions that lead to a positive return. This metric is commonly referred to as the percent profitable (PP) metric in trading applications. Another common metric used is average trade profit (ATP) which evaluates the average return of executed trades. The total percentage return based on the described strategy across the entire date range present in the testing set is also used to evaluate the performance of the model.

Table 7 shows the trading performance of each of the model configurations discussed in Sect. 5. The *multi-width*

**Table 7** Trading performance for each model configuration using the baseline trading strategy

		Returns [%]	PP [%]	ATP [%]
Single-width	Self-learnt	238.8	57.4	0.195
	Static	221.1	55.3	0.185
	Non-static	272.8	58.5	0.226
Multi-width	Self-learnt	293.5	58.0	0.245
	Static	140.3	53.7	0.111
	Non-static	<b>317.8</b>	<b>60.9</b>	<b>0.284</b>

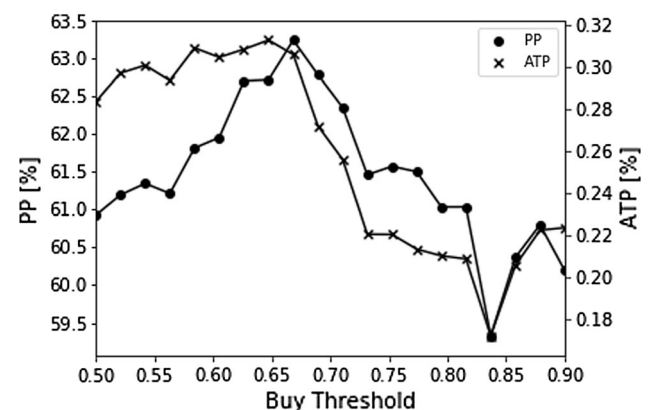
Best results are highlighted in bold

*non-static* implementation of the model demonstrates the best trading performance according to each of the metrics used. Hence, it can be concluded that the optimal configuration for the classification of individual headlines also provides the best trading performance for simulations considered in this work.

Using these baseline results, two methods to reduce the risk in the system are proposed. In the context of this work, the risk is associated with how frequently the model's predictions result in a next-day investment leading to losses, and the average loss incurred by these incorrect investments. Hence, a system with reduced risk would demonstrate higher PP and ATP metrics. It is expected that lower returns will be made using these methods due to the risk-return trade-off that exists in market trading [47]. The methods discussed aim to provide strategies that can be used during periods of greater market volatility, where reduced risk is of greater priority than large returns. The two methods proposed involve the adjustment of the buy threshold associated with  $\sigma(z)_{\text{mean}}$ , and modification of the original classification task from binary to multi-class.

### 6.1 Buy threshold

In the baseline trading strategy discussed, the mean prediction for headlines relating to a specific asset throughout a single day is used to make trading decisions. The constraint  $\sigma(z)_{\text{mean}} > 0.5$  is used as a baseline, where 0.5 can be considered as a buy threshold. For each individual input to the network, the closer the sigmoid output  $\sigma(z)$  is to unity, the greater the certainty in the headline's membership to class 1. Therefore, increasing the buy threshold aims to select days where  $\sigma(z)_{\text{mean}}$  represents greater certainty that the next-day returns will be positive. Simulations are conducted across the entire testing date range for  $\sigma(z)_{\text{mean}} > t$ , where  $t \in [0.5, 0.9]$  is the buy threshold used.



**Fig. 5** Variation in PP and ATP with buy threshold across the entire date testing date range using a strategy developed from binary label classification

Figure 5 shows the variation in PP and ATP with buy threshold. The highest percentage of profitable trades is achieved using a buy threshold  $t = 0.67$  with an improvement of 2.3% over the baseline strategy. Despite this increase in PP, the overall returns using this method are reduced. Although a higher ATP than baseline is demonstrated at some buy thresholds, the stricter margin results in fewer investments being made. For example, using  $t = 0.67$ , 38.1% fewer investments are made across the testing period than in the baseline case. This result demonstrates the risk-return trade-off discussed. It is expected that the strictest buy thresholds would yield the most improvement in PP and ATP, however, both metrics demonstrate a decrease in these metrics for high values of  $t$ . This result is due to the model making incorrect predictions despite high values of  $\sigma(z)_{\text{mean}}$ . For example, the largest single-day loss from an investment is 11.3%, however, the model predicts  $\sigma(z)_{\text{mean}} = 0.93$  based on the previous day's headlines. The effect of these incorrect predictions with high  $\sigma(z)_{\text{mean}}$ , coupled with a significantly reduced number of trades, leads to lower performance metrics than in the baseline case at values of  $t > 0.75$ . These results demonstrate the shortcomings of making predictions based solely on company headlines, as it is possible for the network to make a positive prediction with high certainty based on a collection of headlines but for a significant loss to be made. This is due to market activity and trends that cannot be depicted within a single headline. Therefore, to see continued risk reduction at higher buy thresholds, general economic headlines or historical price trends must also be considered by the model. Despite this, the reduced risk is demonstrated across a range of moderate buy thresholds using this method. The optimum value of  $t$  achieved here is similar to that shown in the work of Ding et al. [24] who consider a similar method. However, their method is only applied to individual headline classification and does not consider trading decisions based on headlines from an entire day as presented in this work.

## 6.2 Modification to multi-class labelling

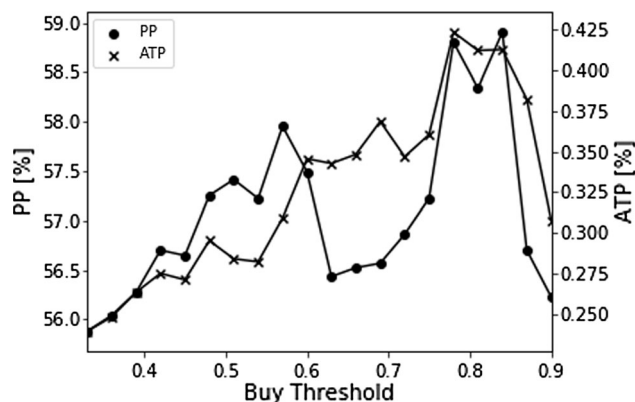
The second approach adopted to reduce risk involves the alteration of the original classification task from binary to multi-class. In the multi-class experiments conducted, individual headlines belong to one of three classes; 'avoid', 'inconsequential' or 'buy'. Headlines relating to an asset whose stock price falls by more than 0.5% during the next-day's trading are labelled as 'avoid'. If the next-day returns of an asset are greater than 0.5%, then corresponding headlines are labelled as 'buy'. Headlines that fall between these constraints are labelled as 'inconsequential'. This labelling system aims to encourage the model to identify

investment opportunities that are likely to provide a significant return. The 'inconsequential' class contains some investment opportunities with a positive return, however, the risk associated with them is larger than the potential reward. Whereas the mean of each sigmoid output on a single day per asset was used to make trading decisions in both the baseline and buy threshold experiments, a more complex decision process is required here. Using the softmax activation given in Eq. 9, three probabilities,  $S(z_1)$ ,  $S(z_2)$  and  $S(z_3)$ , are output by the network for each headline, corresponding to the probability of membership to each of the three classes. Here, classes 1, 2 and 3 correspond to 'avoid', 'inconsequential' and 'buy', respectively. The mean of each of these probabilities,  $S(z_1)_{\text{mean}}$ ,  $S(z_2)_{\text{mean}}$ ,  $S(z_3)_{\text{mean}}$  is calculated for each asset on each unique day in testing. The class corresponding to the maximum of these mean probabilities is assigned to the entire day for each asset. If the maximum mean probability corresponds to the 'buy' class, then the asset is bought at the next-day open and sold at close as before.

Using this initial multi-class strategy, performance metrics of PP = 55.9% and ATP = 0.240% are achieved. Hence, by altering the task to multi-class and using a buying strategy as outlined, a greater risk is observed than in the baseline *multi-width non-static* configuration. Hence, it is necessary to implement a buy threshold for the multi-class system. Instead of simply making a next-day investment if the maximum mean probability corresponds to the 'buy' class, implementation of a buy threshold also requires the mean probability  $S(z_3)_{\text{mean}}$  to be greater than some buy threshold  $t$ . Therefore, an investment is made if and only if the following requirements are met:

$$\max_{k \in [1,3]} (S(z_k)_{\text{mean}}) = S(z_3)_{\text{mean}} > t \quad (13)$$

Figure 6 shows the variation in PP and ATP for the multi-class system with the implementation of buy



**Fig. 6** Variation in PP and ATP with buy threshold across the entire date testing date range using a strategy developed from multi-class labelling



thresholds  $t \in [0.33, 0.9]$ . The addition of a strict buy threshold results in a significant improvement in both metrics considered. Minimum risk is achieved using a buy threshold  $t = 0.86$  where although PP remains less than in the baseline *multi-width non-static* configuration, ATP is more than doubled. This suggests that although more incorrect trades are being executed, the average loss of the incorrect trades is reduced. The maximum single-day loss incurred using this method is 7.6% compared to 11.3% in the baseline and binary buy threshold strategies. Furthermore, the average return of correct buys is greater than in the baseline case due to the restriction that returns must be greater than 0.5% for ‘buy’ allocation in multi-class labelling. The average return of correct buys using  $t = 0.86$  is 1.53% compared to 1.20% in the baseline case. In summary, the implementation of a strict buy threshold with multi-class labelling decreases trading risk by minimising the average loss of incorrect next-day predictions, whilst the restriction on percentage returns in the labelling of individual headlines as ‘buy’ leads to the average profit of correct investments being far greater.

## 7 Conclusion

In this work, a convolutional neural network is implemented to predict next-day stock fluctuations for three technology-based assets. Word embeddings in three states, *self-learned*, *static* and *non-static*, are considered, as well as *single-width* and *multi-width* convolutional layers. Experiments seeking to identify an optimal configuration in terms of accuracy and F1-score showed the presence of a filter width  $h = 3$  or  $h = 4$  as optimal. This result arises as key phrases depicting the headline’s overall sentiment can be evaluated in their entirety. Word embeddings in the *non-static* state were found to be able to adapt to the specific context of the task whilst retaining relationships based on general context and hence provide the best classification performance. A *multi-width non-static* implementation was found to be the optimal configuration of the CNN architecture, leading to a testing accuracy of 61.7%. However, the benefit of using multiple filter widths compared to a single width was not compelling in the conducted experiments.

A simple trading strategy using the mean sigmoid prediction for headlines relating to each asset on each testing day was implemented. The optimal model configuration for classification was found to produce the best simulated trading results in terms of returns and two common trading performance metrics; PP and ATP. This configuration was found to more than triple an initial investment over the 838-day testing period.

Two methods to reduce the perceived risk in investments made across the testing set were developed. The implementation of a moderately strict buy threshold led to some reduction in risk, however, further increase in this threshold resulted in increased risk compared to the baseline strategy. Alteration of the task to multi-class showed no reduction in risk on its own, but the combination of this with a strict buy threshold yielded an ATP more than double that achieved using the baseline strategy. Both of the discussed methods revealed downfalls of basing market predictions solely on company headlines. Therefore, combining methods presented in this work with predictions based on the technical analysis of stock trends should be considered in further research. Furthermore, headlines describing the general state of the economy could be considered in parallel to company-specific headlines. General indicators such as gold price, bond yields and moving averages of popular indices (S&P 500, Nasdaq) should also be considered as additional features for the model as these have been shown to strong correlation with market trends.

The training of the discussed model on separate collections of headlines grouped by business sector (oil and gas, finance etc.) should be undertaken in further work to form collections of embeddings and weights tuned to each sector. Subsequent testing headlines can then be evaluated using the set of weights and embeddings corresponding to the sector of the asset in question. This method has the ability to detect phrases based on the specific context of each sector has been demonstrated for the technology sector in this work. However, further work is needed to validate if this is beneficial compared to training a single collection of embeddings and weights for all sectors.

## Declarations

**Conflicts of interest** The authors declare that they have no conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.



## References

- Malkiel BG, McCue K (1985) *A random walk down Wall Street*. Norton New York
- Malkiel Burton G, Fama EF (1970) Efficient capital markets: a review of theory and empirical work. *J Finance* 25(2):383–417
- Dash R, Dash PK (2016) A hybrid stock trading framework integrating technical analysis with machine learning techniques. *J Finance Data Sci* 2(1):42–57
- Teixeira AL, De Oliveira ALI (2010) A method for automatic stock trading combining technical analysis and nearest neighbor classification. *Expert Syst Appl* 37(10):6885–6890
- Meda-Campana JA (2018) On the estimation and control of nonlinear systems with parametric uncertainties and noisy outputs. *IEEE Access* 6:31968–31973
- Rachlin G, Last M, Alberg D, Kandel A (2007) Admiral: A data mining based financial trading system. In: 2007 IEEE Symposium on Computational Intelligence and Data Mining, 720–725. IEEE
- Sharma G, Mahendru M (2010) Impact of macro-economic variables on stock prices in India. *Glob J Manag Bus Res* 10(7):08
- Al-Tamimi H (2006) Factors influencing individual investor behaviour: an empirical study of the uae financial markets. *Bus Rev* 5:225–232
- Hans Franses P, Van Dijk D (1996) Forecasting stock market volatility using (non-linear) Garch models. *J Forecast* 15(3):229–235
- Poon S-H, Granger CW (2003) Forecasting volatility in financial markets: a review. *J Econ Lit* 41(2):478–539
- Agnolucci P (2009) Volatility in crude oil futures: a comparison of the predictive ability of GARCH and implied volatility models. *Energy Econ* 31(2):316–321
- Kambouroudis Dimos S, McMillan David G, Tsakou K (2016) Forecasting stock return volatility: a comparison of GARCH, implied volatility, and realized volatility models. *J Futures Mark* 36(12):1127–1163
- Cheng C-H, Chen T-L, Wei L-Y (2010) A hybrid model based on rough sets theory and genetic algorithms for stock price forecasting. *Inf Sci* 180(9):1610–1629
- Bomfim Antulio N (2003) Pre-announcement effects, news effects, and volatility: monetary policy and the stock market. *J Bank Finance* 27(1):133–151
- Wuthrich B, Cho V, Leung S, Permuntilleke D, Sankaran K, Zhang J (1998) Daily stock market forecast from textual web data. In: SMC98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218)
- Bollen J, Mao H, Zeng X (2011) Twitter mood predicts the stock market. *J Comput Sci* 2(1):1–8
- Vu TT, Chang S, Ha QT, Collier N (2012) An experiment in integrating sentiment features for tech stock prediction in Twitter. In: Proceedings of the Workshop on Information Extraction and Entity Analytics on Social Media Data, pages 23–38, Mumbai, India, December. The COLING 2012 Organizing Committee
- Wong DPRK (2002) Currency exchange rate forecasting from news headlines. *Aust Comput Sci Commun* 24(2):131–139
- Gomez-Carrasco P, Michelon G (2017) The power of stakeholders' Voice: the effects of social media activism on stock markets. *Bus Strategy Environ* 26(6):855–872
- Siikanen M, Baltakys K, Kannianen J, Vatrappu R, Mukkamala R, Hussain A (2018) Facebook drives behavior of passive households in stock markets. *Finance Res Lett* 27:208–213
- Antweiler W, Frank MZ (2004) Is all that talk just noise? the information content of internet stock message boards. *J Finance* 59(3):1259–1294
- Schumaker Robert P, Hsinchun C (2009) Textual analysis of stock market prediction using breaking financial news. *ACM Trans Inf Syst* 27(2):1–19
- Mittermayer M (2004) Forecasting intraday stock price trends with text mining techniques. In: 37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the, pages 10 pp
- Ding X, Zhang Y, Liu T, Duan J (2015) Deep learning for event-driven stock prediction. In: Twenty-Fourth International Joint Conference on Artificial (Intelligence)
- Johnson R, Zhang T (2014) Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*
- Ritesh BR, Chethan R, Jani HS (2017) Stock movement prediction using machine learning on news articles. *Int J Comput Sci Eng* 4(3):153–155
- Rosenblatt F (1961) Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY
- Aquino G, Rubio JDJ, Pacheco J, Gutierrez GJ, Ochoa G, Balcazar R, Cruz RD, Garcia E, Novoa FJ, Zacarias A (2020) Novel nonlinear hypothesis for the delta parallel robot modeling. *IEEE Access* 8:46324–46334
- Chiang H-S, Chen M-Y, Huang Y-J (2019) Wavelet-based EEG processing for epilepsy detection using fuzzy entropy and associative petri net. *IEEE Access* 7:103255–103262
- Elias I, Rubio JDJ, Cruz DR, Ochoa G, Novoa JF, Martinez DI, Muñoz S, Balcazar R, Garcia E, Juarez CF (2020) Hessian with mini-batches for electrical demand prediction. *Appl Sci* 10(6):2036
- Ashfahani A, Pratama M, Lughofer E, Ong Y-S (2020) DEV-DAN: deep evolving denoising autoencoder. *Neurocomputing* 390:297–314
- de Jesus Rubio J (2009) SOFMLS: online self-organizing fuzzy modified least-squares network. *IEEE Trans Fuzzy Syst* 17(6):1296–1309
- Kim Y (2014) Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*
- Kalchbrenner N, Grefenstette E, Blunsom P (2014) A convolutional neural network for modelling sentences. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland. Association for Computational Linguistics
- Schumaker RP, Zhang Y, Huang CN, Chen H (2012) Evaluating sentiment in financial news articles. *Decis Support Syst* 53(3):458–464
- Raschka S (2015) Python machine learning. Packt Publishing Ltd, Birmingham
- Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P (2011) Natural language processing (almost) from scratch. *J Mach Learn Res* 12:2493–2537
- Vijayarani S, Ilamathi MJ, Nithya M (2015) Preprocessing techniques for text mining-an overview. *Int J Comput Sci Commun Netw* 5(1):7–16
- Miháltz M Word2vec google news model. <https://github.com/mmihaltz/word2vec-GoogleNews-vectors>. Accessed from 05 Nov 2018
- Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781
- Raschka S (2015) Python Machine Learning. Packt Publishing, Birmingham

42. Friedman Jerome H (1997) On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Min knowl discov* 1(1):55–77
43. Kingma Diederik P (2014) Ba Jimmy. Adam, A method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980)
44. Ofer D Financial news export. <https://www.kaggle.com/danofertwo-sigma-financial-news-export/output>. Accessed from 30 Oct 2018
45. Lerman PM (1980) Fitting segmented regression models by grid search. *J R Stat Soc Ser C (Appl Stat)* 29(1):77–84
46. van der Maaten L, Hinton G (2008) Visualizing data using t-sne. *J Mach Learn Res* 9:2579–2605
47. Ghysels E, Santa-Clara P, Valkanov R (2005) There is a risk-return trade-off after all. *J Financ Econ* 76(3):509–548

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.