APPLICATIONS OF DEEP LEARNING: KEYWORD-CENTRIC NEURAL NETWORK

Archie Wiranata

Abstract

Arguably, the largest application of deep learning in Fintech is found in stock/index movement prediction through neural networks. This paper will explore a keyword-centric approach in utilizing neural networks to predict index movement with aims for more than 50% performance accuracy and computing time of less than 10 minutes. The data undergoes text preprocessing via NLTK and Tf-idf modules to featurize each news headlines to their feature words. Utilizing an RNN Encoder-Decoder and a 12-layer GRU, two models are built in the attempt to predict stock/index movement. This managed to achieve more than 50% performance accuracy in under 10 minutes computing time.

Keywords: Deep Learning, Fintech, Stock Market prediction, NLP, RNN, News analytics

1. Introduction

There has been a growing trend in deep learning financial applications in recent times. However, due to large computational costs and data scales, only sizeable financial institutions and corporations have had the ability to fully utilize these applications (1). This has disadvantaged home traders, smaller institutions and startups. This project aims to lessen that gap by building a low computational cost application for index movement predictions via online news article headlines.

The objective of this project is to build an application that can predict, up to an acceptable accuracy, the movement of stocks or indexes based on online news article headlines with an acceptable computing cost. Specifically, the model aims to achieve 60% predictive accuracy and a minimum accuracy of 50%. Computing cost should translate to a waiting period of less than 10 minutes in a common laptop/PC. This study will focus on testing back-end algorithms that is the preprocessing and deep learning algorithm on multiple datasets, and particularly concentrate on the results and benchmarking comparisons.

This report will first introduce the general background of the topic, its motivation, objective and scope. In the Data and Processing section, it will discuss data source and usage, and preprocessing. This is followed by Model Components and Model Architecture. Finally, the report will discuss the results and conclude with project status and secondary findings.

2. Data and Preprocessing

This section will state the data used, and, briefly, their sources, followed by how the data was classified to train the model. It will conclude with how the data is preprocessed.

2.1. X and Y Datasets

The model is trained using several data sources. X dataset, which includes news headlines and their respective dates, is derived from the News API (2) and stocknews dataset, derived from Reddit, from Aaron7sun in Kaggle_(3). Data from the News API is limited to a period of 1 month prior to the current date (15/01/19 - 15/02/19), with 23,356 data points. Meanwhile, data from stocknews has a wider data range of about 8 years (08/08/2008 – 07/01/2016), with 49,725 data points after preprocessing. In comparison to previous works (4), our training dataset is far larger and should be sufficient to produce similar, if not, better results.

The Y dataset is comprised of indexes downloaded online from Yahoo Finance. A multitude of asset classes were chosen (stocks, commodities, market indexes) to provide a clear spectrum of the performance for specific index natures. Market indexes like the Dow Jones Index (DJI), S&P500 (GSPC), and Wilshire 5000 (W5000) are indexes used to pre-train the model. Commodity and equity indexes, are used to both pre-train, train and test the model. These include Amazon (AMZN), Tesla (TSLA) and Google (GOOGL) for equity indexes, and oil price denoted by West Texas Intermediate index (WTI) and gold price denoted by gold/USD spot commodity price (XAUUSD).

Index movement is tracked by the difference between today and yesterday's index points; this is a binary data type. A positive or zero index movement is parameterized with 1 while a negative index movement is parameterized with 0. Equation 1 shows the relationship of index movement and index.

$$y = 1 \text{ when } x_t > x_{(t-1)}$$

$$y = 0 \text{ when } x_t < x_{(t-1)}$$

Equation 1. Relationship of Index Movement and Index

2.2. Global Data Source

Global Data Source (GDS) is comprised of the fixed database from stocknews and the returned topic headlines from NewsAPI when called with an empty keyword (""). It is then merged by date to a specific index. Unlike the work of Poulos (4), each data point represents a single news headlined, not a single date. This is done to allow the model to capture features of specific topics, not specific dates. Furthermore, this allowed the creation of more data points, which may help increase the learning performance of the model. GDS is used to pre-train the model. Aside from using their respective indexes, utilizing a Market index to pre-train may assist in transfer learning better, especially when predicting for a new index with very few historical data.

2.3. Index-Keyword-parameter-pairs

The Index-Keyword-parameter-pairs (IKPP) are news headlines and an index paired. News headlines are selected in relevance to the "keyword" and indexes are selected by the "index" parameter and correlate heavily with the "keyword" selected. They are used to re-train the model so that it may better predict for their respective indexes. IKPPs tests model performance against different asset classes. IKPP is later split to train/test ratio of 80/20.

2.4. Preprocessing

The first step in preprocessing the data was splitting the news headline columns from the stocknews data to a single news headline column. They were then paired with their designated dates. The NewsAPI data was also processed such that only the headline news articles and their respective dates were left. The news headline data was later paired with an index movement binary label according to their date. This produced the GDS and IKPP datasets.

The subsequent step was applying a term frequency-inverse document frequency (TF-IDF) vectorizer from scikit to extract important features from each news headline. This process removed repetitive stopwords (5), as specified from the NLTK module (6). It also featurized important words based on their term-frequency and their inverse document frequency (5). This decision benefited the training process, as only the words that might correlate with the index movements were chosen. The output of this vectorizer is a vector with length of vocabulary size denoting features found in a specific headline.

3. Model Components

This section will discuss the different components of the models.

3.1. TF-IDF Vectorizer

TF-IDF vectorizer (TF-IDF)is used to transform news headlines in feature-representing vectors. This requires data to be first fitted into the vectorizer so that it can learn to transform word inputs to vector outputs. This process can be represented through two parameters: "Fit", and "Transform". An issue of non-uniform vectors arises when multiple training sessions using different training datasets are used as the Fit parameter. With multiple training sessions, pre-training and re-training, and the requirement to maintain input shape for the pre-trained model, the models utilize two methods of TF-IDF vectorizer.

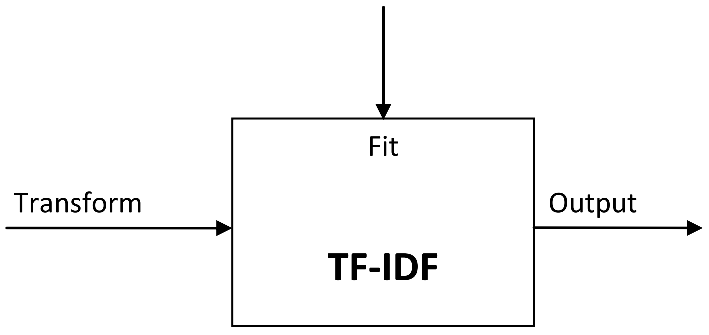

Figure 1. TF-IDF Vectorizer

Method 1: The most direct solution to this is to fit a unique dataset, and use the fitted vectorizer to transform all subsequent datasets. Advantages of this method is that it requires less memory and time. Disadvantages include missing out new features that did not appear in the fitted dataset. This method is used in Model 1.

Method 2: Another alternative to using TF-IDF is to fit and transform each dataset on their own. Unfortunately, this produced vectors of different shapes between the two training sessions. To combat this issue, a small three-layer network, an RNN Encoder-Decoder, is used. This method is used in Model 2.

3.2. RNN Encoder-Decoder

Based on the work of Cho and others (7), the RNN Encoder-Decoder (RNN-ED) is a network that is able to improve statistical machine translation systems. It encodes a sequence to a fixed-length vector in the first RNN layer. This vector is decoded into another set of sequence in the second RNN layer. The figure below shows RNN Encoder-Decoder network flow.

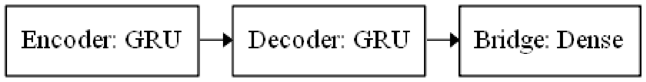

Figure 2. RNN Encoder-Decoder Network Flow

In this project, the RNN Encoder-Decoder is used to transpose specific feature sequences of a similar context to a feature sequence understood by the original network. An example would be “cold”, “freezing” and “chilly”. With the RNN Encoder-Decoder, these three words will be represented by the same feature before being inputted into the 12-layer GRU network. This mechanism allows the model to be pre-trained and re-trained with multiple working datasets.

In its original work (7), the number of GRU units in a single layer is 1000. Due to time and memory costs limitation, the chosen number of hidden units chosen for this model is 128. Further studies to deduce what network structure and sets of hyperparameters best optimizes the utility of this component can be done.

Additionally, a dense layer is used to connect both RNN-ED and the subsequent network. To fit the subsequent network, the number of hidden units in the dense layer is equivalent to the shape of the input length of the subsequent network. With a hyperbolic tan activation function and no biases applied, the dense layer acts as a bridge to connect both networks.

The RNN Encoder-Decoder is used along with TF-IDF (Method 2). The RNN Encoder-Decoder encodes feature vectors and decodes them to another set of feature vectors with a different shape. This translates different shapes of feature vectors from different TF-IDF processes to each other, hence allowing varying inputs shapes of feature vectors. RNN-ED is used in Model 2.

3.3. 12-layer GRU

The principal component of deep learning in this application is the 12-layer GRU (GRU12). Adopted from Poulos' work (4), the 12-layer GRU was found to be effective in classifying news headlines to stock market movements. His work, however, only classified the Dow Jones Index. Furthermore, source of his data was only limited to the stocknews dataset (3). Despite these limitations, it proved to achieve an acceptable accuracy rate at 54%. This showed promise in the network structure provided to predict market movements based on news headlines effectively, hence is used for this application. The figure below shows a layer of the 12-layer GRU. The full figure of the 12-layerGRU can be found in the appendices at section 7.3.

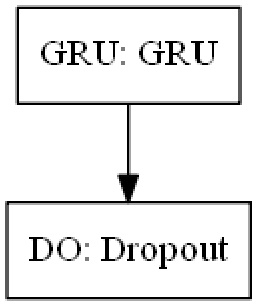

Figure 3. A Layer of the 12-layer GRU

The 12-layer GRU is used in both models. It takes its input from TF-IDF and RNN-ED in Model 1 and Model 2 respectively. Being the part of the model that is responsible for classifying headlines to their respective index movements, it is the unique part of the model that is pre-trained and re-trained. Pre-training is done with GDS and a specific index. Utilizing transfer learning, it is then re-trained with an IKPP dataset to predict for its specific index. No hyperparameter reconfiguration was done to the 12-layer GRU.

4. Model Architecture

This section will state the architecture of each model and briefly describe their training procedures.

4.1.Model 1

Model 1 is comprised of TF-IDF and GRU12. The figure below shows the network structure and process of Model 1.

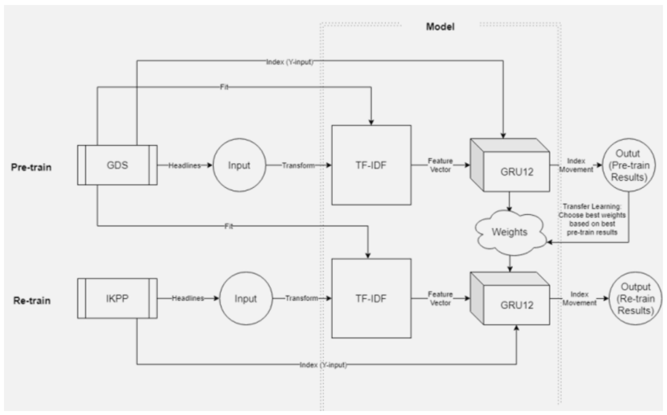

Figure 4. Network Structure and Process of Model 1

TF-IDF only takes GDS headlines for its "Fit" parameter. This allows the model to maintain a constant feature vector shape, which minimizes training costs. No changes or modifications is made for the GRU12 component. Masking of timesteps is performed before GRU12. A dense layer is placed in the output to produce a value representing index movement.

Pre-training: The objective of pre-training is to produce a pre-trained set of weights that could classify features and index movements. The best outputs are used to choose which specific set of weights are used in re-training. These pre-trained weights are saved and later re-trained to predict for a different index. Pre-training is executed by training GRU12 with GDS and a chosen index. Data from GDS and an index is split to before and after 10/02/19 for the train and validation set; The validation set is the last week available in GDS. This is done so that the neural network will choose weights that best predict the current macro-economic conditions.

Re-training: The objective of re-training is to adjust pre-trained weights so that they predict their respective indexes better. The pre-trained weights and model are loaded and is further re-trained by feature vectors from the transformation of the IKPP dataset based of the GDS fitting via TF-IDF. The outputs are used verify performance of the model. The network is reconfigured to only allow training in the last two layers (12th GRU and Dense layer). This configuration of locking pre-trained weights is essential for transfer learning. Data from IKPP is used to re-train the loaded model and the best set of weights. Due to the varying dataset size from the different IKPPs, 64 randomly chosen headlines and index pairs are used as a validation set; the rest are used for the training set. To fulfill the requirement for computational cost, a timer limits re-training up to 10 minutes. The output of the re-training process, un-trained accuracy, un-trained loss, re-trained accuracy, re-trained loss, total time and number of epoch training are used to judge model's performance.

4.2. Model 2

Model 2 is comprised of TF-IDF, RNN-ED, and GRU12. The figure below shows the network structure and process.

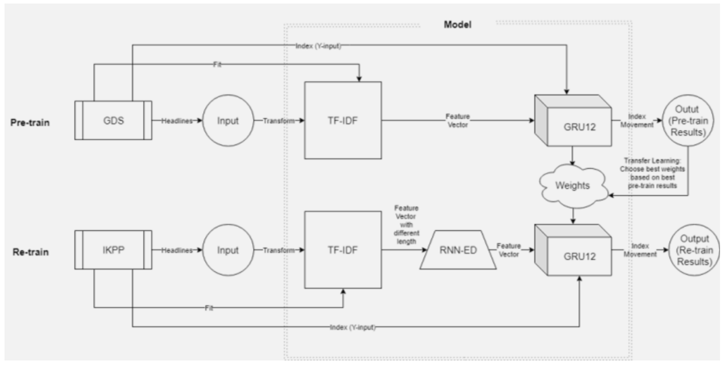

Figure 5. Network Structure and Process of Model 2

TF-IDF fits and transform every dataset on its own. No changes or modifications is made for the GRU12 component. Masking of timesteps is performed before RNN-ED. A dense layer is placed in the output to produce a value representing index movement.

Pre-training: Follows the same procedures as Model 1.

Re-training: The objective of re-training is to adjust pre-trained weights and train the RNN-ED so that they predict their respective indexes better. The pre-trained weights are loaded to GRU12 and is further trained by feature vectors from the transposition of RNN-ED after the transformation of the IKPP dataset via TF-IDF. This model is able to predict for IKPP datasets with varying lengths of feature vectors. The outputs are used verify performance of the model. The model is reconfigured to only allow training in the RNN-ED component and the last two layers (12th GRU and Dense layer). This configuration of locking pre-trained weights is essential for transfer learning. Similar to the retraining of Model 1, only 64 data points are used in the validation dataset. To fulfill the application's requirements of low computing cost, a timer is in place to limit the complete re-training process to 10 minutes. The output of the re-training process are un-trained accuracy, un-trained loss, re-trained accuracy, re-trained loss, total time and number of epoch training are used to judge the Model's performance.

5. Results and Discussion

This section will state results of each training sessions.

5.1. Pre-training session

GRU12 is pre-trained with eight GDS datasets, hence, providing eight results of performance accuracy. These results are then compared to the minimum benchmark (50%) and the maximum class imbalance of each GDS dataset; the maximum class imbalance is the larger class imbalance between the two classes of index movement (1 or 0). The results are compared to the minimum benchmark to signify how well GRU12 performs in comparison to a random walk directional forecast (8). This also provides an insight to how well GRU12 performs against our minimum expectations. The results are compared to the class imbalance to see if the accuracy attained by GRU12 can overcome class imbalance of the given dataset (9). This is done it to show that it could beat majority-class-only predictions within imbalanced datasets. These comparisons takes the difference between the accuracy of the model and the benchmarks given. These differences are then computed in a t-test and a hit-rate calculation. A t-test confirms the statistical significance of the difference between the performance accuracy of GRU12 and the selected benchmark. A hit-rate calculation simply finds the probability that, given any index as the validation metric, GRU12, using GDS as training data, will be able to predict with a performance accuracy of over the selected benchmark.

The table below shows the findings of these analyses.

Data tag	Index	Max. Class Imbalance (CIMB)	Min. Benchmark (BMK)	Performance Accuracy (ACC)	ACC - CIMB	ACC - BMK	loss
GDS-WTI	WTI	0.5277	0.5000	0.9818	0.4541	0.4818	0.6648
GDS-GSPC	GSPC	0.5957	0.5000	0.7791	0.1834	0.2791	0.5342
GDS-GOOGL	GOOGL	0.5355	0.5000	0.7664	0.2309	0.2664	0.9960
GDS-W5000	W5000	0.5765	0.5000	0.7289	0.1524	0.2289	0.5891
GDS-TSLA	TSLA	0.5163	0.5000	0.6407	0.1244	0.1407	0.6930
GDS-DJI	DJI	0.5572	0.5000	0.6168	0.0596	0.1168	0.6765
GDS-XAUUSD	XAUUSD	0.5207	0.5000	0.6081	0.0874	0.1081	0.6697
GDS-AMZN	AMZN	0.5405	0.5000	0.5766	0.0361	0.0766	0.6860
			average	**0.7123**	**0.1660**	**0.2123**	**0.6887**
			Standard deviation	**0.1330**	**0.1330**	**0.1330**	**0.1359**
			t-stat		**1.2488**	**1.5967**	
			p-value		**0.2519**	**0.1544**	
			Confidence level		**0.7481**	**0.8456**	
			Hit-rate		**1**	**1**	

Table 1. GRU12 pre-training results analysis

Overall, the performance of GRU12 exceeds expectations. With the lowest performance accuracy (GDS-AMZN) exceeding 50% and its highest performance (GDS-WTI) nearly at 100%. Furthermore, the average of these results are at 71.23% and that there is a considerable margin against their benchmark as shown by the average value of (ACC-CIMB) and (ACC-BMK). The confidence levels are not high enough to be statistically significant by scientific standards(95%). The hit-rates for GRU12 also signifies that with GDS as a train set, achieving a performance accuracy above the minimum benchmark and class imbalance is almost always.

Furthermore, in comparison to similar work by Poulos in the past (4), training GDS-DJI, a similar dataset, on GRU12 was able to overcome his performance accuracy by approximately 7%. This is likely due to the splitting up of data by each headline, not date, and the addition of better quality news headline data from commercial sources.

5.2. Re-training session

Each model are re-trained with 20 pairs of pre-trained weights and re-training datasets, hence, providing 20 results of performance accuracy.

Unlike processing pre-training results, re-training results are reorganized to two different groups, “Train” and “Test”. The Train group aggregates the results of each unique “Loaded Weights Dataset” (The GDS Dataset used to train the laded weights). This includes DJI, GSPC, W5000 and non-market indexes (AMZN, GOOGL, TSLA, WTI,XAUUSD).

The Test group aggregates the results of each unique "Re-training Dataset"(The IKPP dataset used to re-train the model); This includes AMZN, GOOGL, TSLA, WTI, and XAUUSD.

Each of these results-aggregations are then processed similarly, to pre-training results. They are compared to the minimum benchmark (ACC-BMK); this signifies how well Model 1 performs in comparison to a random walk directional forecast (8). This also provides an insight to how well the model performs against our minimum expectations. Each of these results-aggregations are also compared to the class imbalance (ACC-CIMB) to see if the accuracy attained by the model overcomes the class imbalance benchmark of the given dataset (9). An additional comparison is done between the untrained accuracy and the retrained accuracy (ACC-GAIN). Their difference signifies whether or not the re-training process proved to be beneficial to Model 1. This comparison takes the difference between the accuracy of the model and the benchmarks given. These differences are then computed in a t-test and a hit-rate calculation. A t-test confirms the statistical significance of the difference between the performance accuracy of Model 1 and the selected benchmark. A hit-rate calculation simply finds the probability that, given any pair of pre-trained weights and index used in re-training, Model 1 will be able to predict with a performance accuracy of over the selected benchmark.

ACC-BMK is the difference between re-trained accuracy and minimum benchmark (50%). It compares the performance accuracy against the minimum benchmark of the model and helps gauge the model's quality.

ACC-CIMB is the difference between the re-trained accuracy and the maximum class imbalance of the given index as listed in section 2.1.3. It shows that the model can overcome accuracy from single-class-only predictions.

ACC-GAIN is the difference between the re-trained accuracy and the un-trained accuracy. This shows how beneficial re-training is to the model and whether transfer learning is achieved successfully.

The following tables will focus on each group and their comparisons against benchmarks.

GDS Dataset	DJI	GSPC	W5000	non-market
Average	0.0250	**0.1031**	0.0219	0.0844
standard deviation	0.0770	0.0992	0.0140	0.1203
t-stat	0.3246	1.0397	1.5652	0.7012
p-value	0.7490	0.3115	0.1340	0.4917
Confidence level	0.2510	0.6885	0.8660	0.5083
Hit-rate	0.6000	0.8000	0.8000	0.6000

Table 2. Model 1 - train group (ACC-BMK)

The highest average value of ACC-BMK is from GDS-GSPC. This shows using GDS-GSPC as the training dataset for the pre-trained set of weight will achieve approximately 10% above the benchmark on the average.

GDS Dataset	DJI	GSPC	W5000	non-market
Average	-0.0421	0.0360	-0.0452	0.0173
Standard deviation	0.0779	0.0727	0.0313	0.0915
t-stat	0.5404	0.4956	1.4427	0.1891
p-value	0.5952	0.6258	0.1654	0.8520
Confidence level	0.4048	0.3742	0.8346	0.1480
Hit-rate	0.4000	0.6000	0.0000	0.4000

Table 3. Model 1 - train group (ACC-CIMB)

This table shows unfavorable results as most of the Average values are close to zeroes. This implies that no pre-training dataset in model 1 can produce pre-trained weights that significantly overcome the class imbalance benchmark. Hence, applying a majority-class-only prediction in these models may achieve similar results with very minimal computation rendering the model useless.

GDS Dataset	DJI	GSPC	W5000	non-market
Average	0.0469	0.0156	-0.0344	**0.1875**
Standard deviation	0.1138	0.0413	0.0648	0.2615
t-stat	0.4121	0.3780	0.5305	0.7171
p-value	0.6849	0.7096	0.6019	0.4820
Confidence level	0.3151	0.2904	0.3981	0.5180
Hit-rate	0.4000	0.4000	0.2000	0.6000

Table 4. Model 1 - train group (ACC-GAIN)

The highest value for ACC-GAIN is achieved by the non-market index GDS datasets. This implies that transfer learning from a larger dataset of text can be achieved through this model when paired up with its respective index.

IKPP Dataset	AMZN	GOOGL	TSLA	WTI	XAUUSD
Average	-0.0352	0.0625	0.0273	**0.1250**	**0.1133**
Standard deviation	0.0411	0.0423	0.0267	0.1295	0.0726
t-stat	0.8555	1.4771	1.0247	0.9654	1.5606
p-value	0.4029	0.1560	0.3184	0.3465	0.1351
Confidence level	0.5971	0.8440	0.6816	0.6535	0.8649
Hit-rate	0.0000	1.0000	0.7500	0.7500	1.0000

Table 5. Model 1 - test group (ACC-BMK)

The high values of ACC-BMK from IKPP-WTI and IKPP-XAUUSD shows that commodity indexes have superior performance in this model. One can argue that the larger market scale of commodities in comparison to equities make them more susceptible to be highlighted in news headlines. Hence, providing better data quantity and quality. This may be the cause for the superior performance.

IKPP Dataset	AMZN	GOOGL	TSLA	WTI	XAUUSD
Average	-0.0826	0.0042	-0.0062	0.0166	0.0255
Standard deviation	0.0411	0.0423	0.0267	0.1295	0.0726
t-stat	2.0090	0.0993	0.2307	0.1282	0.3510
p-value	0.0590	0.9220	0.8200	0.8993	0.7294
Confidence level	0.9410	0.0780	0.1800	0.1007	0.2706
Hit-rate	0.0000	0.2500	0.2500	0.5000	0.7500

Table 6. Model 1 - test group (ACC-CIMB)

These Average values are unfavorable as they are all very close to zero. This implies that none of the re-training dataset can produce a model that significantly overcome the class imbalance benchmark. Hence, applying a majority-class-only prediction in these models may achieve similar results with very minimal computation rendering the model useless.

IKPP Dataset	AMZN	GOOGL	TSLA	WTI	XAUUSD
Average	-0.0430	0.0703	0.0156	**0.1250**	**0.1016**
Standard deviation	0.0234	0.0605	0.0403	0.3156	0.1775
t-stat	1.8333	1.1619	0.3873	0.3961	0.5723
p-value	0.0825	0.2597	0.7028	0.6965	0.5738
Confidence level	0.9175	0.7403	0.2972	0.3035	0.4262
Hit-rate	0.0000	0.7500	0.5000	0.2500	0.5000

Table 7. Model 1 - test group (ACC-GAIN)

As shown in the table, both IKPP-WTI and IKPP-XAUUSD have high ACC-GAIN values. This tells us that on the average, models re-training for these IKPP datasets experience significantly more transfer learning than others. Using implications from section 4.2.1.3, pre-training weights using GDS with commodity indexes, and re-training them using IKPP datasets of the same commodity index could significant improve model performance through transfer learning.

One can argue that due to the market scale of commodities, they have higher correlation with other indexes, making them easier to learn despite pre-training using a different index.

GDS dataset	DJI	GSPC	W5000	non-market
Average	0.0281	**0.0656**	-0.0063	0.0313
Standard deviation	0.0959	0.0511	0.0948	0.0654
t-stat	0.2932	1.2840	0.0659	0.4781
p-value	0.7726	0.2146	0.9481	0.6380
Confidence level	0.2274	0.7854	0.0519	0.3620
Hit-rate	0.4000	1.0000	0.4000	0.8000

Table 8. Model 2 - train group (ACC-BMK)

The highest average value of ACC-BMK is from GDS-GSPC. This shows using GDS-GSPC as the training dataset for the pre-trained set of weight will achieve approximately 6% above the benchmark on the average. Compared to values from Model 1 as stated in section 4.2.1.1, it can be assumed that Model 2 is inferior as it has smaller Average values all-around.

GDS dataset	DJI	GSPC	W5000	non-market
Average	-0.0390	-0.0015	-0.0733	-0.0358
Standard deviation	0.0715	0.0235	0.0722	0.0661
t-stat	0.5451	0.0619	1.0153	0.5420
p-value	0.5920	0.9513	0.3227	0.5941
Confidence level	0.4080	0.0487	0.6773	0.4059
Hit-rate	0.4000	0.4000	0.2000	0.2000

Table 9. Model 2 - train group (ACC-CIMB)

Overall, these Average values of ACC-CIMB are lower than that of Model 1. They are also all below zero. This implies that no pre-training dataset can produce pre-trained weights that overcome the class imbalance benchmark. Hence, applying a majority-class-only prediction in this models may very likely achieve better results with very minimal computation rendering the model useless.

GDS dataset	DJI	GSPC	W5000	non-market
Average	**0.0969**	**0.0969**	-0.0375	0.0625
Standard deviation	0.1425	0.1425	0.0570	0.0741
t-stat	0.6797	0.6797	0.6581	0.8433
p-value	0.5049	0.5049	0.5184	0.4096
Confidence level	0.4951	0.4951	0.4816	0.5904
Hit-rate	0.6000	0.6000	0.2000	0.8000

Table 10. Model 2 - train group (ACC-GAIN)

Interestingly, unlike to Model 1, transfer learning occur at significant levels when weights are pre-train with GDS-DJI and GDS-GSPC. This is most likely caused by the addition of RNN-ED in Model 2. With its ability in feature transposition, it is possible that RNN-ED can draw connections from weights trained using DJI and GSPC to predict for non-market index movements.

One would hypothesize that W5000 would be better in this aspect as it has a more diversified portfolio, therefore would move more similarly to the sentiment of general news. As these result show otherwise, one can infer that DJI and GSPC are more well represented in the media than W5000.

IKPP Dataset	AMZN	GOOGL	TSLA	WTI	XAUUSD
Average	-0.0352	0.0000	0.0039	**0.1211**	0.0586
Standard deviation	0.0590	0.0494	0.0850	0.0391	0.0562
t-stat	0.5960	0.0000	0.0460	3.1000	1.0434
p-value	0.5582	1.0000	0.9638	0.0059	0.3099
Confidence level	0.4418	0.0000	0.0362	0.9941	0.6901
Hit-rate	0.5000	0.5000	0.5000	1.0000	0.7500

Table 11. Model 2 - test group (ACC-BMK)

In comparison to Model 1, Model 2 has lower Average values of ACC-BMK. This implies that Model 2, on the average, has a lower performance in predicting any random index movements. It did, however, retain a high value for IKPP-WTI.

IKPP Dataset	AMZN	GOOGL	TSLA	WTI	XAUUSD
Average	-0.0826	-0.0583	-0.0296	0.0127	-0.0292
Standard deviation	0.0590	0.0494	0.0850	0.0391	0.0562
t-stat	1.3997	1.1799	0.3482	0.3250	0.5201
p-value	0.1777	0.2526	0.7315	0.7488	0.6090
Confidence level	0.8223	0.7474	0.2685	0.2512	0.3910
Hit-rate	0.0000	0.0000	0.2500	0.7500	0.5000

Table 12. Model 2 - test group (ACC-CIMB)

In comparison to Model 1, Model 2 did not perform as well against the class imbalance benchmark. This implies that none of the re-training dataset can produce a model that significantly overcome the class imbalance benchmark. Hence, applying a majority-class-only prediction in these models may achieve similar results with very minimal computation rendering the model useless.

IKPP Dataset	AMZN	GOOGL	TSLA	WTI	XAUUSD
Average	0.0078	0.0078	0.0273	**0.0820**	**0.0586**
Standard deviation	0.0372	0.0469	0.0322	0.1718	0.1278
t-stat	0.2100	0.1667	0.8489	0.4774	0.4584
p-value	0.8359	0.8694	0.4065	0.6385	0.6519
Confidence level	0.1641	0.1306	0.5935	0.3615	0.3481
Hit-rate	0.7500	0.5000	0.5000	0.5000	0.5000

Table 13. Model 2 - test group (ACC-GAIN)

Similarly to Model 1, IKPP-WTI and IKPP-XAUUSD have a high ACC-Gain values. This tells us that on the average, models re-training for these IKPP datasets experience significantly more transfer learning than others. Using implications drawn from table 10, pre-training weights using GDS with DJI or GSPC, and re-training them using IKPP datasets of the same commodity index could significant improve model performance through transfer learning. The successful transfer learning from a different index and dataset could be accredited to RNN-ED as it transpose feature vectors to similar context hence allowing different dataset and index pairs to learn from each other.

One can argue that due to the market scale of commodities, they have higher correlation with other indexes, making them easier to learn despite pre-training using a different index.

5.3. Computational Cost

	Model 1	Model 2	Diff
Total time (seconds)	591.89	581.80	10.09
Epoch trained	209.95	68.30	141.65
Average time per epoch (seconds/epoch)	2.82	8.52	**-5.7**
Total data trained	97671.20	29493.60	68177.6
Average data point trained per epoch (data point/epoch)	465.21	431.8	**33.42**

Table 14. Average computational cost analysis

As shown in the table, has significantly more data trained. This may be a cause as to why Model 1 is superior on average. Furthermore, despite taking more time to train per epoch, each epoch in model 2 processes less data. This is very likely due to the addition of RNN-ED in Model 2. It adds in more parameters for the entire model to train during the re-training process. Hence, despite the ability to transpose feature vectors from one context to another, the RNN-ED adds more parameters to train and can affect the performance accuracy negatively.

6. Conclusion

This section will conclude the project status and secondary findings.

This project successfully manage to meet the minimum requirements. Despite having less overall performance accuracy, Model 2 met the minimum requirements of both performance accuracy and computational cost, where as Model 1 did not. Future studies to find a balance between these two models would be worth while pursuing.

These models were able to achieve high performance results using one unique network structure, however it does rely on multiple datasets and indexes to optimize results. Future studies to discover more pairings of pre-train and re-train datasets are required to predict for new assets.

The tables below show the bests combinations of GDS and IKPP datasets that produced the highest performance accuracy

Model 1:

IKPP	GDS	Asset class	Performance accuracy
AMZN*	W5000	Equity	0.5000
GOOGL**	**GSPC**	**Equity**	**0.6250**
TSLA	DJI	Equity	0.5625
WTI**	**WTI**	**Commodity**	**0.7656**
XAUUSD**	**XAUUSD**	**Commodity**	**0.6406**
		Average	**0.61874**

Table 15.Model 1 best performance

Model 2:

IKPP	GDS	Asset class	Performance accuracy
AMZN**	**W5000**	**Equity**	**0.5156**
GOOGL	GSPC	Equity	0.5469
TSLA**	**TSLA**	**Equity**	**0.5938**
WTI	W5000	Commodity	0.6406
XAUUSD	DJI	Commodity	0.6094
		Average	0.58126

Table 16. Model 2 best performance

* Did not surpass minimum performance accuracy
** Best between both models

Secondary findings:

- Model still cannot significantly overcome the class imbalance benchmark
- Model 1 has significant ability to perform transfer learning using larger datasets and commodity indexes to predict for the same commodity index
- Model 2 has significant ability to perform transfer learning using larger datasets and market indexes to predict for the a commodity index. Most likely due to RNN-ED and its transposing abilities
- GRU12, with the stated methodology, can outperform previous works (4)

6. Acknowledgement

I would like to express my sincere gratitude to my original supervisor, Dr. Yik-Chung Wu, for his valuable time, guidance and mentorship. His support and advice have motivated me greatly during the course of this project. My hopes and prayers for his recovery and perseverance for his family in this trying time.

I would also like to express my sincere gratitude to my current supervisor, Dr. Vincent Tam, for his valuable time, guidance and mentorship. His time and kindness in taking in and being responsible of more supervisees have helped me finish my project on time.

I would like to also thank Long for his technical assistance in my scripting process. His aid allowed me to execute this project efficiently.

I would like to thank my family for their endless support and love throughout my undergraduate studies.

7. Reference

1. **Ian Pollari, Anton Ruddenklau.** *The Pulse of Fintech.* s.l. : KPMG, 2018.

2. **News API.** Documentation - Everything. [Online] 2016. https://newsapi.org/docs/endpoints/everything.

3. **Aaron7sun.** Daily News for Stock Market Prediction. [Online] Kaggle, 2017. https://www.kaggle.com/aaron7sun/stocknews .

4. **Poulos, Jason.** *Predicting Stock Market Movement with Deep RNNs.* s.l. : UC Berkeley, 2017.

5. *Scikit-learn: Machine Learning in Python.* **F, Pedregosa, et al.** 2011, Journal of Machine Learning Research, Vol. 12, pp. 2825-2830.

6. **Bird, Steven, Edward Loper and Ewan Klein.** *Natural Language Processing with Python.* s.l. : O'Reilly Media Inc., 2009.

7. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.* **Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio.** 2014.

8. **Y. Hashimoto, T. Ito, T. Ohnishi, M. Takayasu, H. Takayasu, T. Watanabe.** Random walk or a run: Market microstructure analysis of foreign exchange rate movements based on conditional probability. *Quant. Finance.* 2012, Vol. 6, 12.

9. *Training Cost-Sensitive Neural Networks with.* **Zhi-Hua Zhou, Xu-Ying Liu.** 2006, IEEE Transactions on Knowledge and Data Engineering, Vol. 18, pp. 63-77.

10. **Bridge, Chris.** Error in Tensorflow backend combining stateful RNN into a larger model #9274. *Github.* [Online] Keras, 2018. https://github.com/keras-team/keras/issues/9274 .