**SoC Summer 2022 Final Documentation**

**CNN Based Stock Market Prediction**

**Shivprasad Kathane**

CNN, stockmarket, timeseries, NIFTY50, ensemble, machinelearningdeployment, trading

|  |  |  |
| --- | --- | --- |
| **Team Member Name** | **Roll Number** | **Email-Id** |
| Nilabja Mandal | 190020078 | mandalnilabja@iitb.ac.in |

**Brief Description**

|  |
| --- |
| Predicting Financial Instrument prices is as much difficult as it is rewarding. There is huge data generated in Capital Markets and leveraging these predictions can be made. Analyzing their behaviour and extracting useful insights can help traders. Traditional ML models have shown considerable performance in this task with SVMs, random forest ensembles, leading the scope. But with the improvement of our computational abilities deep learning have started to take the forefront. CNNs are highly useful in auto-extracting features which is in itself a difficult task.  The paper “***CNNpred: CNN-based stock market prediction using a diverse set of variables****”* by ***Ehsan Hoseinzade***and***Saman Haratizadeh***published *in Faculty of New Sciences and Technologies, University of Tehran, Tehran, Iran* on Mar 20, 2019 tries to predict next day’s stock prices using market data from the past as well as different markets as well using CNNs.  This project, **CNN Based Stock Market Prediction**, is an attempt to replicate their work and apply it for Indian Stock Indices. |

**Progress**

|  |
| --- |
| Weekly Targets were set with Phase 1 with majorly learning components including Timeseries as AR, MA, ARIMA, SARIMA, ARIMAX, SARIMAX, Deep learning Libraries as Tensorflow, Keras, and Object-Oriented Programming principles.  Phase 2 was understanding the Research paper “***CNNpred: CNN-based stock market prediction using a diverse set of variables****”* and Summarising the work done.  Phase 3 was majorly implementation of the CNNpred, Hyperparameter Tuning, Transfer Learning, Ensemble and Deployment.  Major challenge was not on the technical side but on logistics, like long training duration, limitations on Large File Uploading to Github, Deployment using Flask, Tensorflow compatibility with GPU. Few all-nighters with ample trial and error was the only solution.  Due to constraint in space, weekly progresses have been summarised within these following documents:  <https://github.com/mandalnilabja/soc2022/blob/main/Week6Assignment.docx>  <https://github.com/mandalnilabja/soc2022/blob/main/Week7Assignment.docx>  <https://github.com/mandalnilabja/soc2022/blob/main/Week8Assignment.docx>  <https://github.com/mandalnilabja/soc2022/blob/main/Week9Assignment.docx>  <https://github.com/mandalnilabja/soc2022/blob/main/Week10Assignment.docx>  Further avenues of development is incorporating online machine learning and creating a trading bot which takes leveraged long or short positions whenever a prediction is made with greater than 70% probability. |

**Results**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Based on the CNNpred paper, 2d CNN version with the original data (DJI, NASDAQ, NYSE, RUSSELL, S&P) was implemented. Data is from **Dec 31, 2009** to **Nov 15, 2017**. All the data is present in the github repo. The model is built using tensorflow and keras library.  Custom metric functions as recall, precision, f1, Macro f1 score were defined. The architecture of the model according to the CNNpred paper was built next. The model uses 82 features, last 60 days data, 3 Convolutional layers with filters (3\*1) and ReLU activation, Maxpooling layers (2\*1), output layer with sigmoid activation.  Then the data pipeline functions to feed the model training and testing was created as datagen and testgen. Last 5 months of data (after '2016-04-21') is used as test data. The rest is split into 75% training and 25% validation data.  Using loss function as Maximum Absolute Error and the Optimization algorithm as AdamOptimizer, the model was trained varying the hyperparameter batch size (128, 64, 32, 16).   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | 128 | 64 | 32 | 16 | | Accuracy | 0.48 | 0.54 | 0.50 | 0.55 | | MAE | 0.52 | 0.46 | 0.49 | 0.45 | | Macro F1 | 0.55 | 0.66 | 0.60 | 0.65 |   Training time was approximately 24m using GPU.  Thus, we can clearly see reducing the batch size to 64 improves the performance of the models significantly for evaluating metrics. But further exploration was required before any conclusion.  64 combinations of hyperparameters were shortlisted for modelling adopting a fractional factorial approach (45-2) for an efficient Response Surface for optimization search.  Optimization of performance metrics Accuracy, Mean Absolute Error, Macro Averaged F1 Score by hyperparameter tuning. A Grid-Search strategy for evaluating each hyperparameter combination out of all possible 64 models was adopted.  Following hyperparameters were varied to find the optimal model-   1. Loss Function (mae, binary\_focal\_crossentropy, binary\_crossentropy, hinge) 2. Optimizer (SGD, Adam, Adagrad, Adamax), 3. Epochs (20, 30), 4. Batch Size (64, 32), 5. Dropout Rate (0.05, 0.1, 0.15, 0.2),   The whole experiment took takes 20h+ of training time. And yielded the following results.  CSV containing performance parameters for 64 possible combinations of above hyperparameters-<https://github.com/mandalnilabja/soc2022/blob/main/CNNpred_WallStreet_performances.csv>  In the original CNNpred paper best Macro F1 score for 2d CNN pred was average of 0.54, 0.56, 0.55, 0.55 and 0.55 which is 0.55 but here Bagging ensemble on 2dCNNpred implementation using all 64 previous models outperformed any single predictor. A hard voting strategy was manually implemented since each model by itself is a weak classifier.  Accuracy: 53.75%  Mean Absolute Error: 46.24%  Macro Averaged F1 Score: 69.92%  This is statistically a significant improvement.  For the task of using CNNpred to predict Indian Stock Index, NIFTY50 Index was chosen. Open, High, Low, Close, Adjusted Close and Volume Data for the period between **2007-9-18** and **2022-7-27** was collected from NSE website.  Apart from 6 original features, 6 others were engineered based on popular technical indicators, namely ‘3 Days Moving Average’, ‘5 Days Moving Average’, ‘15 Days Moving Average’, ‘30 Days Moving Average’, Daily Trading Volume Difference and Weekly Difference in Closing Values.  Following hyperparameters were varied to find the optimal model-   1. Loss Function (mae, binary\_focal\_crossentropy, binary\_crossentropy, hinge) 2. Optimizer (SGD, Adam, Adagrad, Adamax), 3. Epochs (10, 20, 25, 30), 4. Batch Size (128, 64, 32, 16), 5. Dropout Rate (0.05, 0.1, 0.15, 0.2)   The whole experiment took takes 12h+ of training time despite using GPU. And yielded the following results.  CSV containing performance parameters for 64 possible combinations of above hyperparameters- <https://github.com/mandalnilabja/soc2022/blob/main/CNNpred_NIFTY50_performances.csv>  The models created was further bagged in an ensemble using a similar strategy as before. The ensemble was able to perform superior to classical models as predicted. Its evaluation metrics were:  Accuracy: 54.51%  Mean Absolute Error: 45.49%  Macro Averaged F1 Score: 70.50%  Compared to 2dCNNpred implementation on Wall Street Indices, even much fewer numbers of features were available and used, overall performance was evaluated to be much better on NIFTY50 CNNpred.  Github Repo: <https://github.com/mandalnilabja/soc2022>  (Repo contains all codes and documentation used including all models and data)  Demo Video: <https://github.com/mandalnilabja/soc2022/blob/main/CNNpred_demo.mp4>  Images:  Step1-    Step2-    Step3-    Step4- |

**Learning Value**

|  |
| --- |
| * Timseries Analysis- AR, MA, ARIMA, SARIMA, ARIMAX, SARIMAX etc. * Pandas, Numpy, Tensorflow, Keras, Sklearn, statmodels, fbprophet * Deep Learning- CNN and its architectures * Transfer Learning- Resnet50 and CIFAR10 * Ensembling- Bagging Models * Deploying Models on web using Flask API * HTML Forms * Use of Git and Github * Understanding of Feature Engineering and Selection * Hands on experience on working with real life financial data * Statistical techniques to optimize test accuracy- Response Surface Methodology |

**Software used**

|  |
| --- |
| * Python * HTML * Google Colab * Anaconda * Jupyter Notebook |

**Suggestions for others**

|  |
| --- |
| * Start Working with the ultimate goal in mind * Model Training takes a lot of time- Keep enough time for training * Ensemble works better than any single model |

**References and Citations**

|  |
| --- |
| Research Article:  Ehsan Hoseinzade, Saman Haratizadeh,  CNNpred: CNN-based stock market prediction using a diverse set of variables,  Expert Systems with Applications,  Volume 129,  2019,  Pages 273-285,  ISSN 0957-4174,  https://doi.org/10.1016/j.eswa.2019.03.029.  (<https://www.sciencedirect.com/science/article/pii/S0957417419301915>)  Abstract: Feature extraction from financial data is one of the most important problems in market prediction domain for which many approaches have been suggested. Among other modern tools, convolutional neural networks (CNN) have recently been applied for automatic feature selection and market prediction. However, in experiments reported so far, less attention has been paid to the correlation among different markets as a possible source of information for extracting features. In this paper, we suggest a CNN-based framework, that can be applied on a collection of data from a variety of sources, including different markets, in order to extract features for predicting the future of those markets. The suggested framework has been applied for predicting the next day’s direction of movement for the indices of S&P 500, NASDAQ, DJI, NYSE, and RUSSELL based on various sets of initial variables. The evaluations show a significant improvement in prediction’s performance compared to the state of the art baseline algorithms.  Keywords: Stock markets prediction; Deep learning; Convolutional neural networks; CNN; Feature extraction  Links:  <https://www.investopedia.com/terms/t/timeseries.asp>  <https://www.influxdata.com/what-is-time-series-data/>  <https://www.clarify.io/learn/time-series-data>  <https://www.altexsoft.com/blog/business/time-series-analysis-and-forecasting-novel-business-perspectives/>  <https://machinelearningmastery.com/challenging-machine-learning-time-series-forecasting-problems/>  <https://machinelearningmastery.com/taxonomy-of-time-series-forecasting-problems/>  <https://www.influxdata.com/time-series-forecasting-methods/>  <https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet/>  <https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775>  <https://wiki.wncc-iitb.org/index.php/Machine_Learning>  <https://wiki.wncc-iitb.org/index.php/Deep_Learning>  <http://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients>  <https://algotrading101.com/learn/machine-learning-for-finance-guide/>  <https://colab.research.google.com/github/lexfridman/mit-deep-learning/blob/master/tutorial_deep_learning_basics/deep_learning_basics.ipynb>  <https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article>  <https://wiki.wncc-iitb.org/index.php/TensorFlow>  <https://www.tensorflow.org/guide/basics>  <https://towardsdatascience.com/https-medium-com-piotr-skalski92-deep-dive-into-deep-networks-math-17660bc376ba>  <https://github.com/Kulbear/deep-learning-coursera/blob/master/Improving%20Deep%20Neural%20Networks%20Hyperparameter%20tuning%2C%20Regularization%20and%20Optimization/Tensorflow%20Tutorial.ipynb>  <https://www.tensorflow.org/tutorials/keras/classification>  <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/>  <https://www.tensorflow.org/tutorials/images/cnn>  <https://www.tensorflow.org/tutorials/structured_data/time_series>  <https://www.analyticsvidhya.com/blog/2021/08/hands-on-stock-price-time-series-forecasting-using-deep-convolutional-networks/>  <https://www.sciencedirect.com/science/article/abs/pii/S0957417419301915>  <https://data.mendeley.com/datasets/byjnr4kz5v/4>  <https://www.researchgate.net/profile/Janine-Zitianellis/post/Can_anyone_please_suggest_a_books_on_machine_learning_using_R_Programming/attachment/613a5b83647f3906fc975a71/AS%3A1066204907204608%401631214467436/download/Practical+Statistics+for+Data+Scientists+50%2B+Essential+Concepts+Using+R+and+Python+by+Peter+Bruce%2C+Andrew+Bruce%2C+Peter+Gedeck.pdf>  <file:///C:/Users/manda/Downloads/PracticalStatisticsforDataScientists50EssentialConceptsUsingRandPythonbyPeterBruceAndrewBrucePeterGedeck.pdf>  <https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/Index.ipynb#scrollTo=6K97if3hwgZ2>  <https://alphascientist.com/feature_engineering.html>  <https://alphascientist.com/feature_selection.html>  <https://machinelearningmastery.com/using-cnn-for-financial-time-series-prediction/>  <https://machinelearningmastery.com/using-cnn-for-financial-time-series-prediction/>  <https://www.deeplearningbook.org/>  <https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-i-hyper-parameter-8129009f131b>  <https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7>  <https://wiki.wncc-iitb.org/index.php/Deep_Learning#Practical_Aspects_of_Deep_Learning>  <https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c>  <https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/>  <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>  <https://medium.com/@dipti.rohan.pawar/improving-performance-of-convolutional-neural-network-2ecfe0207de7>  <https://machinelearningmastery.com/ensemble-methods-for-deep-learning-neural-networks/>  <https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7>  <https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d>  <https://machinelearningmastery.com/transfer-learning-for-deep-learning/>  <https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4>  <https://www.kaggle.com/code/aadhityaa/stock-market-prediction-using-cnn-lstm>  <https://github.com/siddiquiamir/ML-MODEL-DEPLOYMENT-USING-FLASK>  Repositories:  <https://github.com/hoseinzadeehsan/CNNpred-Keras>  <https://github.com/adrianswtam/legco2016>  <https://github.com/lexfridman/mit-deep-learning> |

**Disclaimer**

|  |
| --- |
| All the codes, text, data, images used in this project belong to their respective owners and I do not claim any right over them. Any such resource used in this project is for educational and research purposes only. All information provided are in good faith, however we make no representation or warranty of any kind, express or implied, regarding the accuracy, validity, reliability, availability or completeness of any information. UNDER NO CIRCUMSTANCES SHALL WE HAVE ANY LIABILITY TO YOU FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF THIS PROJECT OR RELIANCE OF ANY INFORMATION PROVIDED BY THIS PROJECT. YOUR USE AND RELIANCE ON ANY INFORMATION PROVIDED BY THIS PROJECT IS SOLELY AT YOUR OWN RISK.  Copyright Disclaimer under section 107 of the Copyright Act 1976, allowance is made for “fair use” for purposes such as criticism, comment, news reporting, teaching, scholarship, education and research. Fair use is a use permitted by copyright statute that might otherwise be infringing. |

**Licenses**

|  |
| --- |
| Licenses for softwares used has been included below:  Python: Python software and documentation are licensed under the [PSF License Agreement](https://docs.python.org/3/license.html#psf-license).  Google Colab: <https://colab.research.google.com/pro/terms/v1>  Anaconda: <https://www.anaconda.com/eula-anaconda-distribution>  Github: <https://docs.github.com/en/site-policy/github-terms/github-terms-of-service>  Tensorflow: <https://github.com/tensorflow/tensorflow/blob/master/LICENSE> |