FinDL: A Deep Learning Library for Finance Applications

Gao Mo, Nathan Ng, Richard Tang, Zhiting Hu

University of California San Diego, San Diego
Emails: g1mo@ucsd.edu,n3ng@ucsd.edu,ritang@ucsd.edu,zhh019@ucsd.edu

Abstract

Time series data has gained much attention in numerous fields of scientific research and industry. With time series data available in almost any field, researchers and practitioners are able to extract different features and information from time series data to utilize in a variety of real-world applications. Deep learning models have also achieved great success and proved their validity and impact in time series forecasting tasks, finding great success in tasks such as predicting future stock trends. However, developing effective deep learning models for time series forecasting tasks requires an extensive amount of machine learning knowledge, which may be a barrier for financial specialists who lack this expertise. To bridge this gap, we introduce FinDL, a library designed for both financial specialists and machine learning engineers. FinDL provides an end-to-end machine learning pipeline for time series data, with out-of-the-box models that can be configured and fine-tuned according to the users' requirements. With this library, users can easily create and deploy machine learning models for finance-related tasks.

Introduction

Time series data, which are data points collected over time, can be found or generated in any domain. These types of data are extremely valuable due to its nature of holding trend sequence information. Such data are generally used in a wide variety of applications such as predicting future stock market prices, gold prices, weather forecast, and energy consumption forecasting. In the field of finance, financial specialists can use their domain expertise and knowledge to make informed decisions for finance related tasks. On the other hand, machine learning engineers are able to build deep learning models that can extract sequential information from time series trends and use these patterns to make automated predictions. Utilizing the domain expertise of financial specialists to guide machine learning systems built by machine learning engineers, model performance has the potential to significantly improve. However, this can be challenging due to resources, poor team communication, or time sensitive business needs.

To solve this issue, FinDL aims to provide users with modularized machine learning components to build their own pipelines that can be used off-the-shelf. Our library includes models that can be fine-tuned for the target task and lets users easily deploy machine learning systems quickly for financial time-series related tasks. The library also includes a data loader component and data preprocessing functions for users to load, clean, and transform data into machine learning ready state. We also provide various machine learning models for users to experiment and choose from to optimize their machine learning system's performance. Lastly, our library includes components to train, evaluate, and visualize the model's performance to help users create useful models and observe the model's performance during its training. All of these components were designed to provide users a simple and straightforward way to create an end-to-end machine learning pipeline. In regard to the prediction models, numerous deep learning models have received promising results in time series analysis. Recurrent neural networks are known for their capability of learning, remembering, and extracting information from sequential data. Commonly, RNN, along with its variants such as the Long Short-term Memory (LSTM), are used for time series forecasting applications. Hybrid deep learning models that combine multiple model structures together often yield promising and possibly

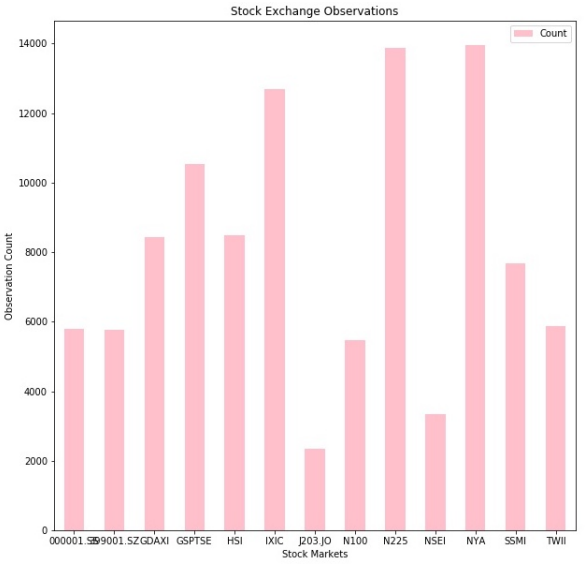

Fig 1. Distribution of stock exchange observations

better results as well. TreNet, for example, is a hybrid model that combines the results generated from CNN and LSTM together to make predictions through a feature fusion layer.

However, building the above-mentioned deep learning models using Pytorch or Tensorflow from scratch requires a significant amount of knowledge in using these machine learning frameworks as well as how to create deep learning networks. In many cases, users have to preprocess their data to meet the requirements of the data loaders provided by these APIs. In addition, users have to implement the training loop and loss visualization functions on their own. This process can be extremely time consuming and difficult for people who lack the experience using deep learning libraries or even programming in general. In comparison, FinDL allows users to easily load and preprocess data with a few lines of code, significantly simplifying the process to prepare data for machine learning. Model selection and model training also take only a few lines to implement, allowing users to focus more on defining their machine learning system instead of implementing the entire pipeline. FinDL is designed to be a modularized time series forecasting library where users have the ability to pick and choose different combinations of deep learning system components that they want to use. This allows user to experiment more with the performance of different combinations of deep learning models instead of dealing with the implementation of complex deep learning systems.

To demonstrate the functionality of our library, we use a dataset of different statistics from stock markets around the world, including the opening and closing prices from stock markets in the United States, China, Canada, and Japan. The data distribution is shown in Fig 1. We used the stock market exchange data from the New York Stock Exchange, which contains 13,194 observations starting from December 31, 1965 to May 28, 2021. We create and run a TreNet model to predict future trend durations and slopes using a pipeline that we created from FinDL.

Related Work

Lots of research has gone into developing high performing deep learning models. Recurrent Neural Network (RNN)[3] is known for its effectiveness in learning the historical dependencies of time series and sequence data. One well-known application of RNN in time series forecasting is the multi-factor RNN work frame proposed by Zhang et al. [4], which uses numerous features, such as the high, low,

and closing prices of every period, to make predictions. Multi-factor LSTM outperforms ARIMA and Gated Recurrent Unit (GRU) by including information in addition to the high price data for time T, which is the general input data for LSTM models. RNN's alternative structure, Long-short Term Memory (LSTM) [5], further improved the model's performance by resolving the vanishing gradient issue. Sagheer was able to successfully use LSTM to forecast future petroleum production using LSTM [6].

Convolutional neural networks (CNN) are generally used for image recognition, feature detection, and extraction [7]. Its feature extraction functionality makes it valuable in time series forecasting tasks. While it does not outperform RNN and LSTM models in time series forecasting, CNN's ability to extract local data features is crucial in improving a model's performance. Gated Recurrent Unit, proposed by Cho et al. [new], is a modified version of LSTM. Compared to LSTM, GRU is known for its simpler implementation and computation. Unlike LSTM, GRU does not pass memory information during its computation, therefore occupies less memory space and has a faster computing time. Due to its perk, GRU generally performs better on smaller datasets, while LSTM achieves higher accuracy on bigger datasets.

Hybrid models designed for time series analysis have also gained much popularity lately. For example, Livieris et al. [8] successfully analyzed and forecasted gold price time series data by using the raw data as input for both CNN and LSTM. The aim was for CNN to learn and extract the feature information and for LSTM to learn the trend sequence dependencies. TreNet, on the other hand, combines the results from CNN, which extracts the information from the local features of time series data, with the results from LSTM, which extracts information from the duration and slope of the current trend in the time series data. TreNet has been proven to outperform numerous traditional and hybrid models used for time series analysis, including LSTM, ARIMA, Hidden Markov Model, and Support Vector Regression.

However, the introduction of these deep learning models do not cover much information about the implementations. People who do not have much programming experience and background knowledge in deep learning may have trouble correctly implementing models, leading to wasted time and resources. With FinDL, these models come pre-implemented, allowing users to call our deep learning functions and specify which model they would like to use for their task by utilizing the different modules in the FinDL library.

Methods

Library Design of FinDL

FinDL was created with the goal of being useful in finance related tasks for both users with little experience in programming deep learning models and others who can implement deep learning networks. With these goals, we designed FinDL to be simple to use without any additional configurations, yet flexible enough to allow for in-depth fine tuning and customization. To accomplish this, we created independent, customizable module components for each step in the machine learning pipeline. We identified three general steps in machine learning pipelines, which include data selection, model selection, and model training. Within these steps, we defined independent components that can be used out-of-the-box which can be swapped easily for other components and fine-tuned according to one's need. Our library design and the modularized components are shown in figure 2.

For model selection, we implemented various time series model architectures that share a uniform training process that can be used without additional configurations. Since we created our models using the PyTorch machine learning framework, we chose to allow users to use PyTorch loss functions and optimizers with our models as they followed the design principles of our library. Under data selection, we identified two components that users generally needed in a machine learning pipeline, data loading and data pre-processing. For data loading, we provide users with a simple class that loads data and formats dates or time data in preparation for time series modeling. In the

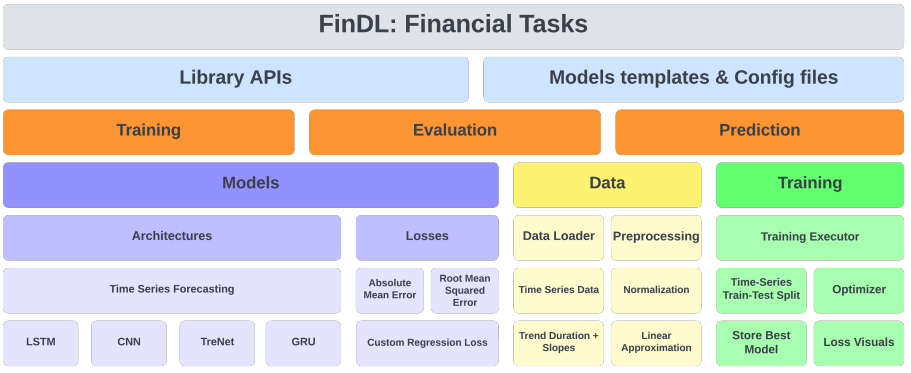

Fig 2. FinDL Module Stack

data pre-processing module, we created components that processed and restructured the data into a machine learning ready state. Many of these components are optional and users can select different combinations of data pre-processing components based on their time series task. Lastly, the model training component consists of a general training executor that works with all our models and can be configured to use different optimizers and loss functions. All of these components were designed to be independent, such that users can easily put components from each module together to create their own machine learning pipeline.

FinDL Machine Learning Workflow

To begin with, users will be able to load their raw time series data into FinDL's data loader, which filters and formats the data to be suitable for further processing. The output of the data loader will be fed into FinDL's preprocessor, which utilizes normalization and linear approximation techniques to extract significant information such as the trend information, slope and duration of each trend, as well as the local feature information. The users will then input this extracted information into the deep learning models, including TreNet, LSTM, and CNN, depending on their situation. With one line of API call, FinDL's training executor allows the users to effortlessly initiate the training process and saves the best performing parameters of the model for the users. In addition, the training and validation loss data will be saved for users to utilize FinDL's visualization functions to generate graphs. FinDL provides an end-to-end solution that enables the users to preprocess, analyze, and visualize their time series data in a deep learning pipeline without much programming experience required. The figure 3 captures the workflow of FinDL.

Data Loader

Our data loader component simplifies the process of loading and formatting time series data. It's capable of handling multiple file formats, including CSV and JSON files, and users only need to specify the file path and file format to load data in. The data loader also includes datetime formatting for strings and can filter for specific stock indices or groups. We found that these were common operations in finance related time-series predictions and included them in our library pipeline to make data processing much easier.

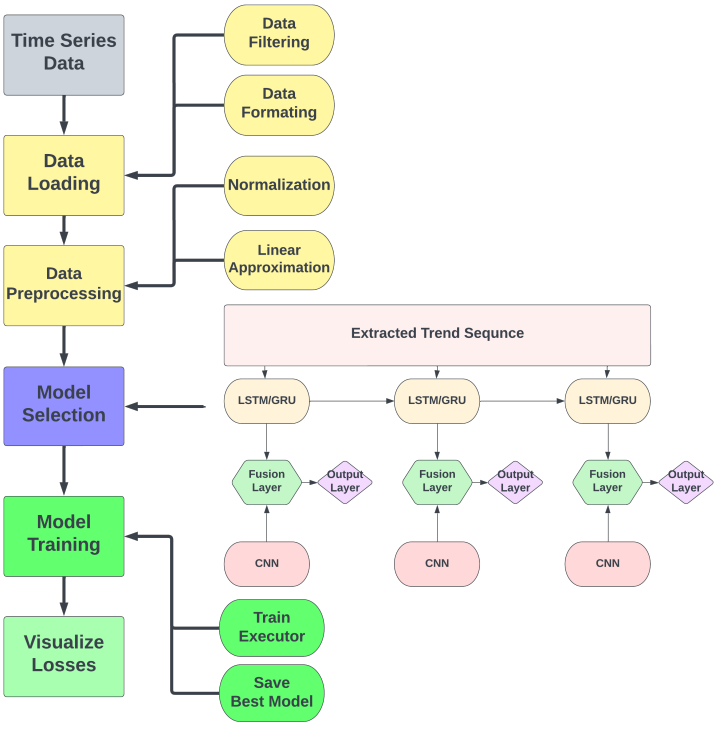

Fig 3. FinDL workflow to create and train TreNet

Data Preprocessing

Once data is loaded in using FinDL's data loader, users can select from a variety of different data preprocessing functions available in FinDL. All of these functions were designed to seamlessly integrate with our data loader, so that users can easily call on components in sequence without dealing with formatting the data. These data preprocessing functions include normalization, which is a common machine learning technique and has been found to make our models train more effectively. We include easy to use classes to help users scale and undo that scale with a single line of code. Furthermore, FinDL also includes advanced functions to extract trend durations and slopes from time series data, which provides valuable insight into the underlying patterns that raw time series data might not provide. Since most stock datasets include only the closing prices of stocks for specific stocks and dates and not trend data, we use a piecewise linear approximation where the length and slope of each segment corresponds to the trend duration and slope respectively. The algorithm we used to accomplish this is the SWAB segmentation algorithm introduced by Keogh et al. This technique combines the well-known sliding windows and bottom-up algorithms for piecewise linear approximation. This also allows users to experiment with different types of prediction tasks instead of only predicting subsequent raw time series data. Another critical function that FinDL library provides is a train, validation, and test split for time series data. We provide users with components to extract subsequences from their time series data to create training and testing sets for model evaluation.

Model

The deep learning models supported by FinDL include TreNet, LSTM, CNN, and GRU. TreNet consists of two layers of 1D convolution, a relu nonlinearity layer, a maximum pooling layer, and a dropout layer. The CNN stack takes raw stock price data points as inputs and returns the local feature information as output. The long short-term memory structure consists of an LSTM layer and a fully connected layer that returns historic trend features. The GRU structure functions similarly as the LSTM structure. It is more time and memory efficient but generally performs better on smaller datasets. Using both the outputs from the CNN stack model and the LSTM or GRU model, the outputs are mapped to the same feature space and added element-wise in the feature fusion layer. The sum of the outputs are then fed through a fully connected layer to return the next prediction.

All of these models can easily be used in conjunction, such as in a larger hybrid or ensemble model, or swapped out using our modular design of these models. Most of the models can perform similar tasks, and can easily be swapped with one line of code. This allows users to experiment with different deep learning architectures and models without needing to deal with implementation or changing other parts of the pipeline to adjust for the new model.

Training Executor and Model Evaluation

Lastly, FinDL also includes a general training executor to train models and functions to evaluate model performance during training. The training executor component works independent of whichever model design the user creates and task the model is designed to answer. This allows users to freely create models using FinDL and easily train the models using our training executor. Training parameters, such as optimizer, loss function, and training epochs can all be configured and changed easily to allow for more effective model evaluation and hyperparameter tuning. Our training executor works with any PyTorch optimization and loss functions, which gives users a wide range of ways to optimize model training and performance, without needing to implement any complex algorithm themselves. Additionally, FinDL includes a built in visualization component to help users quickly plot losses per epoch during model training to see how well the model trained and converged. All of these components help users build a pipeline using flexible components that can be interchanged and fine tuned to maximize performance for time series tasks.

Results

Instead of implementing an entire deep learning pipeline from scratch, which requires extensive knowledge in deep learning models and frameworks to implement effectively, users can easily build their own pipelines using FinDL's modularized components. FinDL reduces the amount of code needed to create, train, and evaluate deep learning models by providing the tools needed to put together a working pipeline. This allows users to focus more on model designs and fine-tuning models, reducing the amount of time and resources people may spend on learning and using more complex machine learning frameworks.

To demonstrate FinDL's capabilities, we implement a deep learning pipeline using a TreNet model using FinDL components. Using raw time series stock data from the New Stock Exchange dataset, we convert this into trend durations and slope and prepare the data to be consumed by the model. We then configure our TreNet model, choose an optimizer and loss function, and train our model using FinDL's training executor. Figure 4 shows the code we used to build and run the pipeline.

In addition, we also created pipelines for CNN, LSTM, and TreNet and trained each model. For the CNN and LSTM models, we used FinDL to create a pipeline to predict the next subsequent stock price. For the TreNet model, we used FinDL to predict the next trend duration and slope. We trained each model separately for 2000 epochs and plotted their losses per epoch using FinDL's visualization component.

```python
# Load and pre-process data
dl = DataLoader(data_config)
la = LinearApproximation(dl.data, la_hparams)
data = la.process_data()

# Normalize and linearly approximate data
trends, points = preprocessing.convert_data_points(data)
data_scaler = Scaler.MultiScaler(2)
scaled_trends, scaled_points = data_scaler.fit_transform([trends, points])

split_data = preprocessing.preprocess_trends(scaled_trends, scaled_points, device, feature_cfg)

# Create model
model = TreNet.TreNet(device, LSTM_params=LSTM_hparams, CNN_params=CNN_hparams, TreNet_hparams)
loss_fn = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=training_config['lr'])

# Train model
trainer = train_models.TrainingExecutor(model=model, optim=optimizer, loss=loss_fn)
trainer.train(split_data.get("X_train"), split_data.get("y_train"), X_val=split_data.get("X_valid"),
y_val=split_data.get("y_valid"))
train_loss, val_loss = trainer.losses["train"], trainer.losses["valid"]

# Visualize loss during training
loss_visuals.visualize_loss([train_loss, val_loss], visual_configs)
```

Fig 4. Library API creating TreNet pipeline

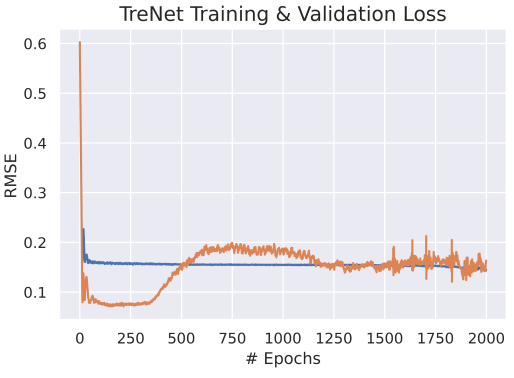

Fig 5. TreNet Training and Validation Loss

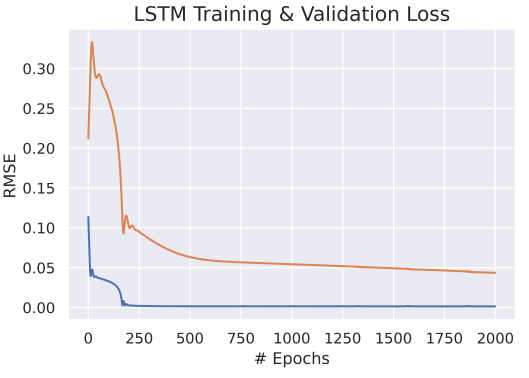

Fig 6. LSTM Training and Validation Loss

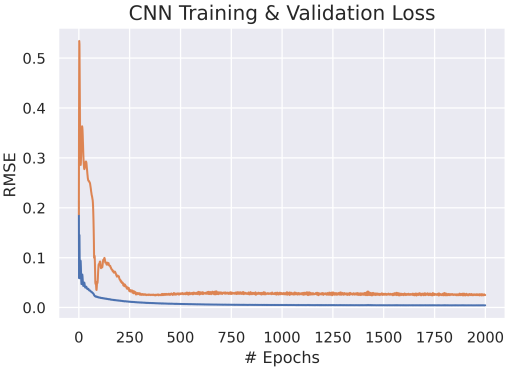

Fig 7. CNN Training and Validation Loss

The resulting visualizations are plotted in figures 5, 6, and 7, with the blue line representing training loss and the orange line representing validation, and we observed that each model converged fairly quickly. In the TreNet model, the validation loss for predicting future trend data dropped from 0.6028 to 0.1504, and converged at the 1250th epoch. For the LSTM model, the validation loss when predicting the next stock price dropped from 0.332 to 0.0518 and converged at the 1250th epoch. Lastly, the CNN model had a validation model of 0.5342 which dropped to 0.0273, and converged at the 500th epoch. By using components from FinDL, we were able to swap models and change prediction tasks fairly easily with minimal amounts of changes.

Conclusion

In this paper, we introduced FinDL, a library that provides an end-to-end machine learning pipeline for time series data related tasks in finance. Users have the ability to utilize state-of-the-art deep learning models without significant machine learning experience and knowledge. FinDL provides callable data loader and preprocessor functions, as well as time-series forecasting models and training and evaluation functions. Users are able to implement the entire deep learning pipeline and generate promising and consistent results with just a few lines of code using FinDL. In the future, we will continue building upon the library that we have and implement more models and introduce additional features to cover a wider range of deep learning prediction tasks in the finance domain.

References

[1] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021).

[2] Cody, 2018, Stock Exchange Data. Retrieved November 13th 2022 from https://www.kaggle.com/datasets/mattiuzc/stock-exchange-data

[3] E. Keogh, S. Chu, D. Hart and M. Pazzani, "An online algorithm for segmenting time series," Proceedings 2001 IEEE International Conference on Data Mining, 2001, pp. 289-296, doi: 10.1109/ICDM.2001.989531.

[4] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[5] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, D. Bahdanau, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014.

[6] Lin, Tao, Tian Guo, and Karl Aberer. "TreNet: Hybrid neural networks for learning the trend in time series." Proceedings of the twenty-sixth international joint conference on artificial intelligence. No. CONF. 2017

[7] Livieris, I.E., Pintelas, E. Pintelas, P. A CNN–LSTM model for gold price time-series forecasting. Neural Comput Applic 32, 17351–17360 (2020).

[8] Pra, Marco Del. "Time Series Forecasting with Deep Learning and Attention Mechanism." Medium, Towards Data Science, 5 Nov. 2020, https://towardsdatascience.com/time-series-forecasting-with-deep-learning-and-attention-mechanism-2d001fc871fc.

[9] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[10] Sagheer, Alaa, and Mostafa Kotb. "Time series forecasting of petroleum production using deep LSTM recurrent networks." Neurocomputing 323 (2019): 203-213.

[11] Zeroual, Abdelhafid, et al. "Deep learning methods for forecasting COVID-19 time-Series data: A Comparative study." Chaos, Solitons Fractals 140 (2020): 110121.

[12] Zhang, Xu, Chen Li, and Yasuhiko Morimoto. "A multi-factor approach for stock price prediction by using recurrent neural networks." Bulletin of networking, computing, systems, and software 8.1 (2019): 9-13.