

**Using Deep Learning Techniques to Predict Currency Pairs: A Case Study on USD/NGN**

**BY**

**Samir Salihu-Lukman**

**181103056**

**PROJECT REPORT**

**NILE UNIVERSITY OF NIGERIA, ABUJA**

**DEPARTMENT OF COMPUTER SCIENCE**

**JULY 2022**

**SUPERVISOR: Mr. Ahmed Adeniyi**

# DECLARATION

I hereby declare that this research work is the product of my effort undertaken under the supervision of Mr. Ahmed Adeniyi and has not been presented elsewhere for the award of a degree or certificate. All sources have been duly distinguished and appropriately acknowledged in the reference section.

Due to the imperfection of man, any mistake(s) or error(s) identified therein remains mine.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Samir Salihu-Lukman Date

181103056

# CERTIFICATION

This is to certify that this research study was undertaken by SAMIR SALIHU-LUKMAN 181103056 and has been read and approved by the undersigned signatories for meeting the requirement for the award of Bachelor of Science Honors (B.Sc. Hons) in Computer Science by the Department of Computer Science, Nile University of Nigeria, Abuja.

Mr. Ahmed Adeniyi

PROJECT SUPERVISOR

Prof. Nwojo Nnanna Agwu

HoD COMPUTER SCIENCE

Prof. Musa Muhammed

DEAN (FNAS)





Prof. Benjamin Okike Date

EXTERNAL SUPERVISOR

# DEDICATION

I dedicate this work to Allah (SWT), my source of inspiration, knowledge, and understanding, who gave me grace and strength throughout this program. I also dedicate this work to my parents, close friends, and lecturers for their support and encouragement.

# ACKNOWLEDGMENT

All praise, first and foremost, is to Allah (SWT), for His guidance and encompassing blessing.

My appreciation also goes to my parents for their constant support throughout the years and to my siblings for always being solid role models. I would also like to acknowledge my project supervisor, Mr. Ahmed, for all the assistance he rendered me and, all the stress he had to undergo by going through all my write-ups and making alterations and criticisms where necessary.

I would also like to thank all my friends especially Abdulhamid, Bilal, Haruna, Osusu, Sophie, and Aisha for their support throughout my project journey.

I would also like to thank Mr. Yinka of the Computer Engineering department for helping me understand and solve complicated errors in my code and Mr. David Clement for his initial input on how to go about this project.

# ABSTRACT

In the nation, there is interest in the price of the USD/NGN exchange rate. Deep learning has lately been used, among other contemporary methods, for autonomous feature selection and market prediction. However, the USD/NGN currency pair has not been considered a potential source of data for extracting characteristics in the trials described so far. A prognosis for this currency pair is crucial since it has an impact on Nigerians' lives all over the world. To extract characteristics for predicting the future of those markets, I use a CNN-based framework in this research that can be applied to a collection of data from many sources, including various marketplaces. The proposed framework has been used to forecast the direction of movement for the USD/NGN currency pair for the following day based on different sets of beginning factors.

# **Table of Contents**

[DECLARATION ii](#_Toc109662211)

[CERTIFICATION iii](#_Toc109662212)

[DEDICATION iv](#_Toc109662213)

[ACKNOWLEDGMENT v](#_Toc109662214)

[ABSTRACT vi](#_Toc109662215)

[1 INTRODUCTION 1](#_Toc109662216)

[1.1 Background of Study 1](#_Toc109662217)

[1.2 Problem statement. 7](#_Toc109662218)

[1.3 Project Justification 8](#_Toc109662219)

[1.4 Aim and objectives 9](#_Toc109662220)

[1.5 Scope and limitations 10](#_Toc109662221)

[1.6 Summary and report 10](#_Toc109662222)

[2 LITERATURE REVIEW 10](#_Toc109662223)

[2.1 Existing/Similar Systems 10](#_Toc109662224)

[2.2 Research Gap 15](#_Toc109662225)

[3 System Methodology 16](#_Toc109662226)

[3.1 Introduction 16](#_Toc109662227)

[3.2 Dataset 16](#_Toc109662228)

[3.3 Architecture 21](#_Toc109662229)

[4 Modalities and Results 27](#_Toc109662230)

[4.1 System implementation 27](#_Toc109662231)

[4.2 Results 27](#_Toc109662232)

[5 Conclusion 31](#_Toc109662233)

[5.1 Summary 31](#_Toc109662234)

[5.2 Future Improvements 31](#_Toc109662235)

[References 32](#_Toc109662236)

[APPENDIX 35](#_Toc109662237)

# Table of Figures

[Figure 1.1 Architecture of a Neural Network [6] 3](#_Toc110926037)

[Figure 1.2 Architecture of a Convolutional Neural Network [5] 4](#_Toc110926038)

[Figure 4.1 2D Model Results 26](#_Toc110926039)

[Figure 4.2 3D Model Results 27](#_Toc110926040)

# Table of Tables

[Table 3.1 Dataset Features [10] 17](#_Toc110926095)

[Table 3.2 2D Model Architecture 22](#_Toc110926096)

[Table 3.3 3D Model Architecture 22](#_Toc110926097)

# INTRODUCTION

## Background of Study

In this section, we cover the convolutional neural network, which is the core component of our system, the foreign exchange(forex) market, and deep learning.

### Foreign Exchange (FOREX)

### Foreign exchange, FX, or forex is the conversion of one currency into another. In a free economy, supply and demand determine the value of a nation's currency. To put it another way, a currency's value may be correlated with the value of another nation's currency, such as the U.S. dollar (USD), or a basket of currencies. In this instance, we make use of the currency pairs linked to the USD's value. The value of a nation's currency may also be decided by its government. On the other hand, several nations freely peg their currencies to those of other nations, leading them to fluctuate continuously. Take the Saudi Arabian Riyal (SAR), which pegs to the US dollar. [1]

The FX markets are the lifeblood of the world economy. The world's currencies are traded on this market. This is crucial because currencies enable us to buy products and services both locally and internationally. To conduct foreign trade and business, it is important to exchange currencies internationally. [2]

For instance, you or the company from which you acquire the cheese must pay the French in euros if you reside in the United States and intend to buy cheese from France (EUR). This implies that the American importer would have to convert the same amount of dollars (USD) into euros. When it comes to traveling, the same holds. A French visitor to the pyramids will not be able to pay with euros because they are not recognized in Egypt. The tourist's EUR must be changed to the local currency, in this example, the Egyptian pound, based on the going exchange rate (EGP). [2]

The marketplace for international markets is decentralized for foreign exchange, which is a unique feature. Rather than trading on a single centralized exchange, currency trading is done electronically over the counter (OTC), which means that all transactions occur over computer networks among traders worldwide. The market is open 24 hours a day, five days a week, and currencies are traded in every time zone in Frankfurt, Hong Kong, London, New York, Paris, Singapore, Sydney, Tokyo, and Zurich, among other important financial locations. This means that when the trading day in the United States finishes, the forex market in Tokyo and Hong Kong restarts. As a result, the currency market can be very lively at any time, with price quotes continuously fluctuating. [2]

Institutional firms and huge banks dominated the currency market in the past, acting on behalf of clients. However, in recent years, the market has become more retail-oriented, and traders and investors with a wide range of holding sizes have begun to participate; anyone with an internet connection and a bank account can participate.

### Deep Learning

One can consider machine learning to be a subset of deep learning. It focuses on the autonomous learning and development of computer algorithms. An adaptation of artificial neural networks called deep learning aims to imitate how people learn and reason. Before now, the complexity of neural networks was constrained by computer capability. Big Data analytics advancements have made it possible to build bigger, more potent neural networks, which let computers watch, comprehend, and respond to complex events more quickly than people. Deep learning has been useful for speech recognition, language translation, and image categorization. It can resolve any pattern recognition issue without requiring human assistance. [3]

It is a three or more-layer neural network. These neural networks aim to imitate the activity of the human brain by allowing it to learn from enormous amounts of data, albeit they fall far short of its capabilities. While a single-layer neural network may produce approximate forecasts, extra hidden layers could help to optimize or improve accuracy. [4]

Numerous AI apps and services rely on deep learning to improve automation by executing analytical and physical activities without the need for human participation. Daily goods and services (including digital assistants, voice-enabled TV remotes, and credit card fraud detection), as well as upcoming innovations, use deep learning technology (such as self-driving cars). [4]

#### Deep Learning vs. Machine Learning

The sorts of data it uses, and its learning techniques set deep learning apart from typical machine learning. Machine learning algorithms use structured, labeled data to make predictions, which implies that specific features are defined from the input model’s input data into tables. This is not to argue that it never uses unstructured data; rather, it simply implies that when it does, it typically pre-processes information to make it available in a structured way. [4]

Some of the data pre-processing required by machine learning are eliminated by deep learning. Feature extraction can be automated and unstructured data, such as text, and images, can be ingested and interpreted by these algorithms, eliminating the need for human experts. Imagine if we had a collection of pictures of different pets that we wanted to sort by "cat," "dog," or "hamster." Deep learning techniques can determine which traits, like ears, are most crucial in differentiating one species from another human-machine learning expert manually developed this feature hierarchy. [4]

The deep learning algorithm then changes and fits itself for accuracy via gradient descent and backpropagation, allowing it to generate more precise predictions about a fresh snapshot of an animal. [4]

Diverse types of learning, such as supervised, unsupervised, and reinforcement learning, are possible with machine learning and deep learning models. Supervised learning categorizes or predicts using labeled datasets; this involves some type of human interaction to appropriately classify input data. "Unsupervised learning, on the other hand, does not require labeled datasets; instead, it discovers patterns in the data and clusters them according to any differentiating criteria." Reinforcement learning is a learning process in which a model improves accuracy for completing an action in each scenario and receives feedback to increase the reward. [4]

#### How Deep Learning Works

Because most deep learning techniques use neural network designs, deep learning models are occasionally referred to as deep neural networks. Deep neural networks are neural networks with a significant number of hidden layers. Traditional neural networks only have 2-3 hidden layers; however, deep neural networks can have up to 150. Deep learning models are trained using massive amounts of labeled data and neural network topologies that extract features automatically from the data without the need for manual feature extraction. [5]

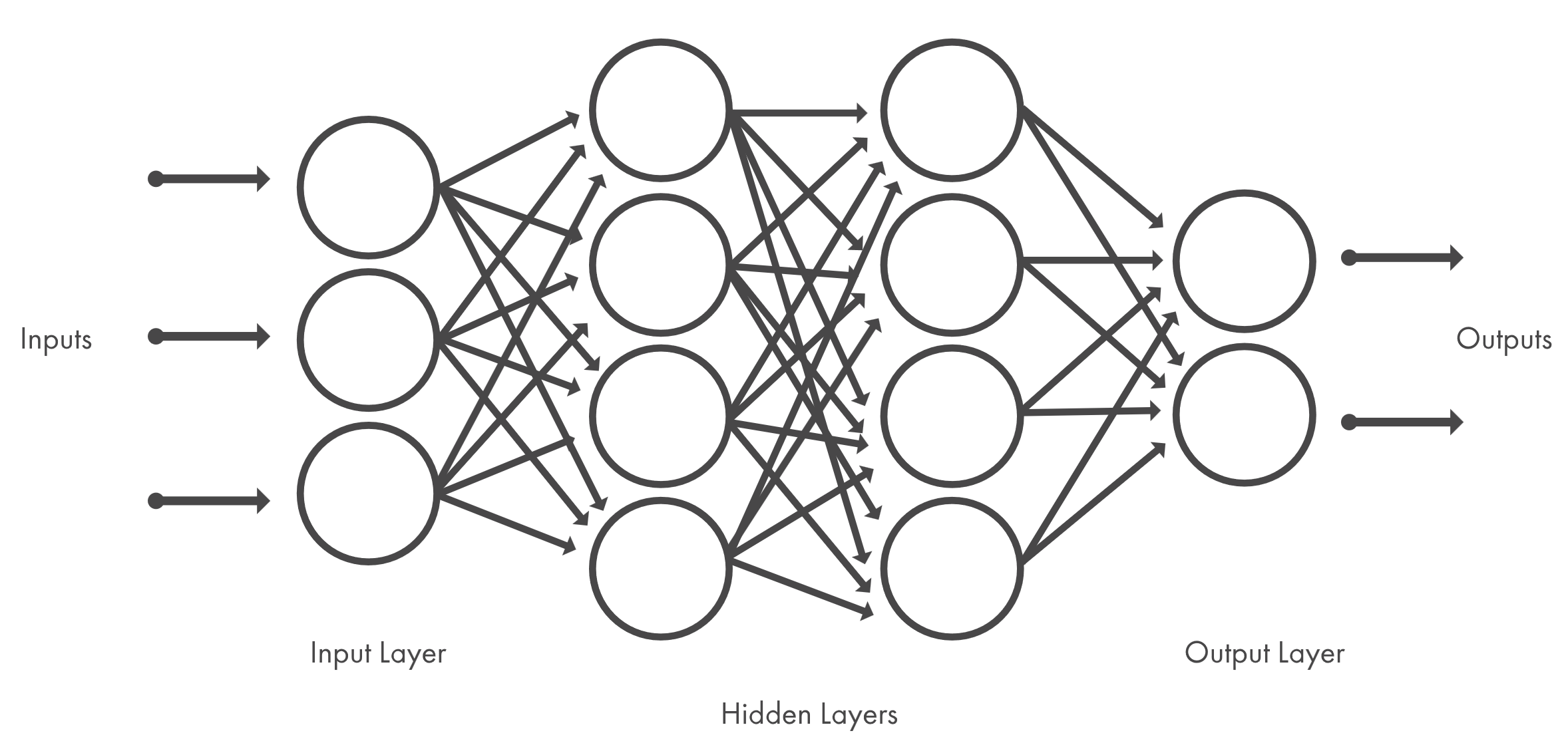


Figure 1.1 Architecture of a Neural Network [6]

One of the most well-liked deep neural network types is convolutional neural networks (CNN or ConvNet). A CNN is particularly suited to processing 2D data like photographs because it combines input data with learned features using 2D convolutional layers. You won't need to find out what features are used to identify photos because CNNs do away with the need for manual feature extraction. CNN uses direct feature extraction from images to run its business. The network trains on a series of photos, which leads to the discovery of the pertinent features rather than their pre-training. Thanks to this automatic feature extraction, deep learning models are very accurate for computer vision applications like object categorization. [5]

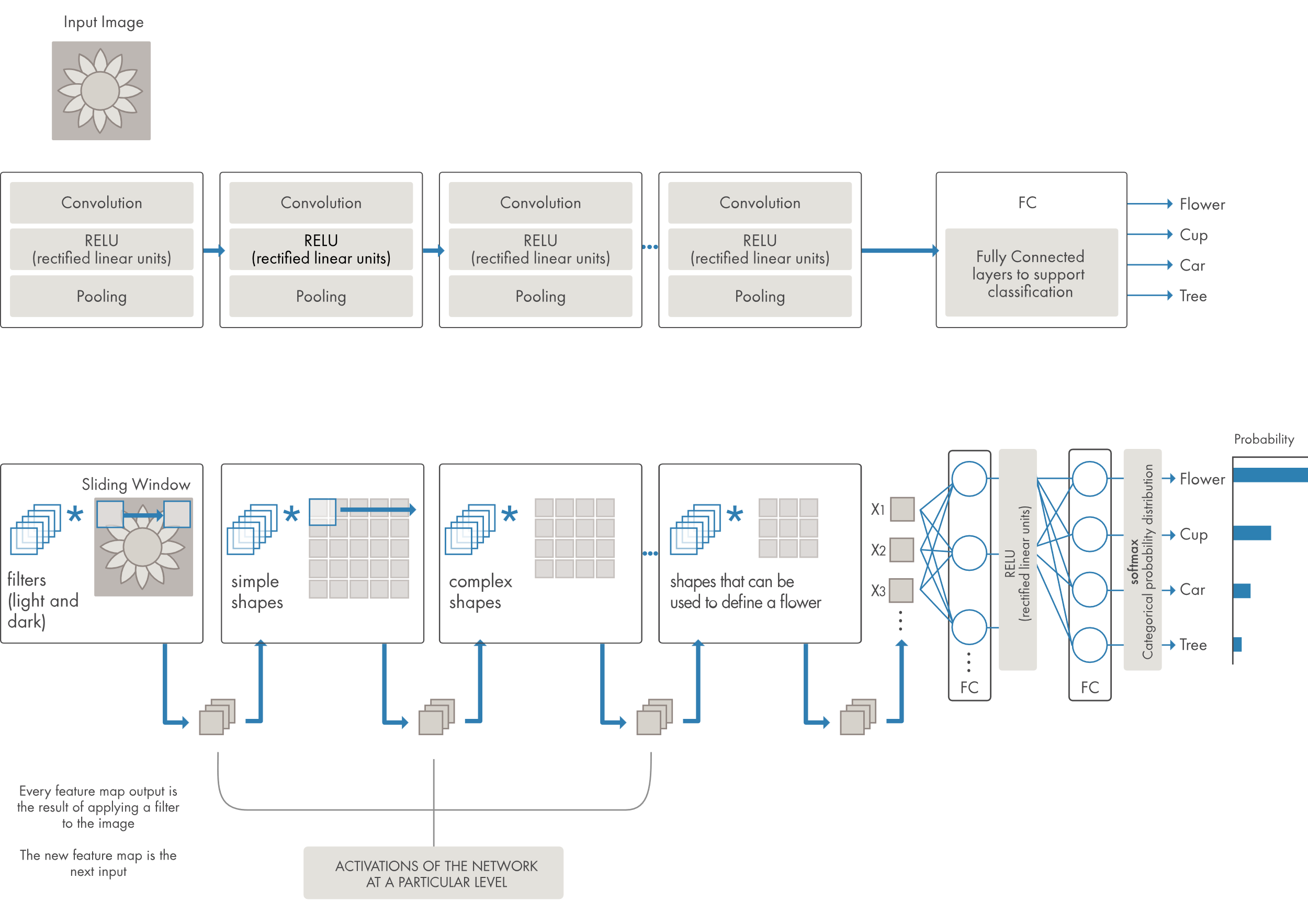


Figure 1.2 Architecture of a Convolutional Neural Network [5]

#### Deep Learning Applications

Deep learning applications in the real world are all around us, but they are usually so effectively integrated into products and services that users aren't aware of the intricate data processing going on in the background. Some of the real-world applications of deep learning include law enforcement, financial services, customer service, healthcare, automated driving, aerospace and defense, medical research, industrial automation, and electronics. [4] [5]

#### Deep Learning Hardware Requirements

Deep learning demands a massive amount of processing power. Graphic processing units (GPUs) with high performance are desirable because they can manage a big volume of operations in several cores with enough memory. Managing numerous GPUs on-premises, on the other hand, can place a significant strain on system resources and be extremely costly to expand. [4]

### Convolutional Neural Networks (CNN)

A convolutional neural network is a type of deep neural network that is often used to evaluate visual imagery in deep learning. When we think about neural networks, we usually think of matrix multiplications, but this is not the case with CNN. It employs a technique known as Convolution. Convolution is a mathematical procedure that produces a third function that expresses how the shape of one is affected by the other. However, we do not need to grasp what a CNN is or how it works to understand what it is. In the end, CNN's job is to compress the images into a format that is easier to manage while preserving elements that are important for a decent prediction. CNN models excel in recognizing patterns in pictures, such as lines. This feature should also be able to recognize trading chart trends. CNN can also find linkages between images that humans cannot see, and the topology of neural networks can help detect complicated interactions between features. [7]

Artificial Intelligence has made considerable progress in closing the gap between human and computer capabilities. "Researchers and hobbyists alike work on a variety of facets of the subject to achieve incredible results." The field of computer vision is one of several such disciplines. The goal of this field is to enable machines to see and perceive the world in the same way that humans do, and to use that knowledge for a variety of tasks such as image and video recognition, image analysis and classification, media recreation, recommendation systems, or natural language processing. Advancements in Computer Vision using Deep Learning have been built and developed through time, mostly using a single algorithm – the Convolutional Neural Network. [8]

The AI system, dubbed AlexNet (after its designer, Alex Krizhevsky), took first place in the 2012 ImageNet computer vision challenge with an incredible 85 percent accuracy. On the test, the runner-up received a respectable 74 percent. Convolutional Neural Networks were at the heart of AlexNet. CNNs have become an integral feature of many computer vision applications over the years and are thus covered in any online computer vision course. [7]

### Building Blocks of CNN Architecture

Convolution layers, pooling layers, and fully connected layers are among the building components of the CNN architecture. A typical architecture comprises one or more completely linked layers followed by a stack of many convolution layers and a pooling layer. Forward propagation refers to the process of transforming input data into output data using these layers. Although the convolution and pooling methods described are for 2D-CNNs, comparable operations can be done for 3D-CNNs as well. [9]

#### Convolution Layer

A convolution layer is a crucial element of said CNN architecture which facilitates feature extraction using a blend of linear and nonlinear processes, such as convolution and activation functions. "The Convolution Operation's goal is to extract high-level characteristics from the input image, such as edges." There is no need to limit Convolutional networks to just one convolutional Layer. The first convolutional layer is traditionally responsible for capturing Low-Level information such as edges, color, or gradient direction. "With the addition of layers, the architecture adjusts to the High-Level characteristics as well, giving us a network that understands the photos in the dataset in the same way that we do." [9] [8]

#### Convolution

A convolution is a form of linear operation used mostly for feature extraction in which a tiny array of numbers called a kernel is applied over the input, which is a tensor of numbers. At each point of the tensor, an element-wise product between each element of the kernel and the input tensor is calculated and summed to generate the output value in the corresponding place of the output tensor, referred to as a feature map. This technique is repeated with different kernels to generate an arbitrary number of feature maps that represent distinct features of the input tensors, different kernels may thus be regarded as different feature extractors. The size and number of kernels are two fundamental hyperparameters that determine the convolution operation. The former is usually 3 x 3, but it can also be 5 x 5 or 7 x 7. The depth of output feature maps is determined by the latter, which is arbitrary. [9]

The convolution method prevents the center of each kernel from overlapping the input tensor's outermost element, resulting in a smaller output feature map in terms of height and breadth. Padding, most commonly zero padding, is a technique for dealing with this problem that involves adding rows and columns of zeros on each side of the input tensor to fit the center of a kernel on the outermost element while maintaining the same in-plane dimension through the convolution operation. To keep in-plane dimensions so that more layers may be applied, modern CNN architectures commonly use zero padding. Without zero padding, following the convolution procedure, each succeeding feature map would be smaller. A stride is a separation between two consecutive kernel points, and it also specifies the convolution procedure. A stride of one is the most usual choice, but a stride of greater than 1 is occasionally used to accomplish feature map downsampling. A pooling process is an alternate method for downsampling. [9]

Weight sharing is an essential component of a convolution operation, kernels are shared among all image places. Convolution operations include the following properties because of weight sharing:

1. Allowing kernels to extract local feature patterns that are translation constant as they go across all image locations and detect learned local patterns.
2. Downsampling in combination with a pooling operation allows for the learning of spatial rankings of feature patterns, resulting in the capture of an ever-broader field of view.
3. In comparison to fully connected neural networks, model efficiency is improved by lowering the number of parameters to train. [9]

The procedure of training a CNN model for the convolution layer is to discover the kernels that perform great for a given job based on a training dataset, as discussed below. Kernels are the only parameters in the convolution layer that are automatically learned during the training process, however, the size of the kernels, number of kernels, padding, and stride are hyperparameters that must be defined before the training process begins. [9]

#### Nonlinear Activation Function

A nonlinear activation function is applied to the results of a linear operation like convolution. Although smooth nonlinear functions such as the sigmoid or hyperbolic tangent (tanh) function were formerly utilized since they are mathematical representations of biological neuron action, the rectified linear unit (ReLU) is now the most used nonlinear activation function. f(x) = max(0, x) [9]

#### Pooling Layer

A pooling layer performs a standard downsampling operation on the feature maps, reducing their in-plane dimensionality to introduce translation invariance to tiny shifts and distortions and reducing the number of learnable parameters. It is worth noting that none of the pooling layers have learnable parameters, although filter size, stride, and padding are hyperparameters in pooling operations, much like convolution operations. [9]

#### Max Pooling

Max pooling is the most common type of pooling procedure, which takes regions from the input feature maps, produces the largest value for each patch, and discards the rest. In practice, max pooling with a size 2 x 2 filter and a stride of two is usually utilized. The in-plane dimension of feature maps is reduced by a factor of two. "The depth dimension of feature maps, unlike the height and width dimensions, does not change." [9]

#### Global Average Pooling

A global average pooling operation is also worth mentioning. A global average pooling is an extreme sort of downsampling in which a feature map of height width is downsampled into a 1 x 1 array by simply taking the average of all the components in each feature map, while the depth of feature maps is preserved. Before the fully connected layers, this step is usually performed only once. The following are some of the benefits of using global average pooling:

1. Lowers the number of parameters that can be learned.
2. Allows the CNN to accept variable-size inputs. [9]

#### Fully Connected Layer

The final convolution or pooling layer's output feature maps are typically flattened, that is, converted into a 1D array of numbers (or vector), and linked to one or many fully connected layers, also known as dense layers, whereby each input is connected to each output by a learnable weight. Once the features that are extracted by the convolution layers and downsampled by the pooling layers are formed, they are transferred to the network's final outputs, such as the probabilities for each class in classification tasks, by a subset of fully connected layers. The number of output nodes in the final fully connected layer is usually equal to the number of classes. As previously mentioned, each fully linked layer is followed by a nonlinear function, such as ReLU. [9]

#### Last Layer Activation Function

The last fully connected layer's activation function is frequently distinct from the others. Each activity necessitates the selection of an appropriate activation function. The SoftMax function, which normalizes output true values from the last fully connected layer to target class probabilities, where each value ranges between 0 and 1 and all values total to one, is an activation function used in the multiclass classification problem. [9]

#### Dropout

We employed a technique called dropout, which was originally designed for training deep neural networks, in addition to pooling. The goal of the dropout strategy is to keep the model from learning too much from the training data. So, in each learning cycle during the training phase, each neuron has a chance of not being taught in that cycle equal to some dropout rate. This keeps the model from being excessively flexible, which aids the learning algorithm in coming to a model that is not overly fitted to the training data and can instead be generalized well for predicting unlabeled future data. [10]

## Problem statement.

The financial markets, where trillions of dollars are exchanged daily, are the lifeblood of the world economy. In several circumstances, a reliable projection of market behavior in the future would be quite helpful. [10]

FOREX markets play a key role in economic growth. The achievement of these economic objectives can be greatly aided by behavior analysis and future prediction. The use of forex market forecasts in trading systems, which typically include several modules for risk analysis, forecasting, and trading techniques, is another significant use. Maximizing the total profit over loss ratio in favor of the profits would be the aim of a trading module.

A prediction module, on the other hand, concentrates on the sub-problem of forecasting the direction of the markets, which may be an incredibly useful piece of knowledge in the process of trading stocks.

As a result, the accuracy of the module's forecasts will have an impact on both the performance of the trading module and the overall trading system. It is difficult to have a superior trading method without accurate forecasts. [10]

## Project Justification

Deep learning (DL) is a family of current techniques that can extract and predict features automatically. DL approaches have been demonstrated to be capable of progressively constructing meaningful complex features from raw data or simpler variables in a variety of disciplines, including machine vision and natural language processing. Because forex market behavior is complicated, unpredictable, and messy, extracting features that are useful enough to make predictions is a major difficulty, and DL is a viable solution. Deep learning algorithms used to predict stock markets include deep multilayer perceptron (MLP), restricted Boltzmann machine (RBM), long short-term memory (LSTM), autoencoder (AE), and convolutional neural network (CNN), so the hope is that this can also be applied to forex markets.

The range of factors that may be used into predictive modelling must be noted. Raw pricing information, technical indications generated from previous data, other markets with a connection to the target market, currency exchange rates, oil price, and a number of other factors might be helpful for a market prediction job. Unfortunately, aggregating such a broad variety of data in such a way that an autonomous market prediction algorithm can use it is usually a difficult undertaking. As a result, most of the existing research in this sector has concentrated on a range of technical indicators that represent a particular market's recent history.

Automatic feature extraction is another key issue in the field. The initial variables are basic since they were designed to be utilized by human experts, and even if they were selected by a finance expert with sufficient knowledge and expertise in this sector, they may not be the best feasible choices for machine prediction. In other words, the ideal autonomous stock market prediction method may derive useful features from a variety of variables that are useful for market prediction, train a prediction model based on those generated features, and then provide predictions using the model. The first stage of the procedure, which involves building a model to extract features from a collection of variables that contain information from earlier market records, is the subject of this study. [10]

This information contains initial basic factors such as raw price history, technical indications, and the change in those variables over time. Due to the complexity of the feature set that may be required for an accurate prediction and the variety of the input space, a deep learning algorithm like CNN is a viable solution for this feature extraction challenge.

I build my architecture on CNN because of its demonstrated capabilities in other domains. I employ the US Dollar to Nigerian Naira (USD/NGN) currency pair as a test example to show how CNN may be used in our proposed framework, CNNpred, to capture possible correlations between various variables for obtaining integrated features from a varied provided set of inputs [10]: Australian Dollar to U.S. Dollar (AUD/USD), European Euro to U.S. Dollar (EUR/USD), Great Britain Pound to U.S. Dollar (GBP/USD), U.S. Dollar to Japanese Yen (USD/JPY), additionally to other factors such the USD/NGN exchange rate, futures contracts, commodity prices, significant global market indexes, the prices of big U.S. firms, and treasury bill rates. [10]

In addition, the filters are created in a way that is suitable to the financial characteristics of the variables.

My work's primary contributions can be summed up as follows:

1. Using a CNN-based framework to combine numerous factors for extracting features and market forecasting. Because the behavior of financial markets is influenced by a variety of factors, it is critical to obtain as much information as possible. Our initial variable collection covers various elements of forex-related variables, and it can be simply expanded to cover more variables.
2. To aggregate and align a wide collection of input variables before training the network to extract useful features for forecasting each of the key forex markets, the research suggests a CNN. [10]

According to [11] the major disadvantages forex traders face is.

1. Changes in the exchange rate: Frequent fluctuations in currency exchange rates can be negative, as economic, and political influences can produce market price uncertainty.
2. Absence of centralized Exchanges: There is no central regulator or exchange to set base prices in the spot market, resulting in pricing differences amongst brokers.
3. Higher Leverage: Although traders can conduct forex transactions at a minimal cost, increased leverage can represent a bigger risk of loss if exchange rate swings impair earnings.

All these factors increase the risk of trading foreign currency markets but using a solid prediction algorithm can drastically lower the risk.

Forex trading can also be a daunting task and keeping track of factors that influence the market can be extremely time-consuming even experienced traders find this extremely difficult and tiring.

Even though the Forex market is the world's largest, most of the work on financial market prediction using deep learning focuses on the stock market and general market prediction.

## Aim and objectives

### Aim

To predict the rate of the USD/NGN currency pair for a specific period.

### Objectives

1. To research the best model/architecture that would better suit my problem domain.
2. To modify and finetune the model/architecture to my problem domain.
3. To evaluate the model/architecture.

## Scope and limitations

### Scope

The purpose of this study is to provide traders with a better understanding of market movements so they can make better trade decisions. It is not the intention to completely replace human analysis.

### Limitations

In financial market prediction systems, as in other prediction efforts, it is always impractical to assume high accuracy.

## Summary and report

With more than five trillion dollars transacted each day, Forex is the only financial market that is open 24 hours a day. There is currently no completely reliable approach for predicting market direction. To generate the best potential prediction regarding the market's direction, we proposed using a Convolutional Neural Network to discover patterns in the dataset.

# LITERATURE REVIEW

Since the use of deep learning approaches to predict foreign exchange markets is mostly unexplored, the reviewed literature is based on similar literature.

## Existing/Similar Systems

In this chapter, I will give a detailed analysis of existing/similar research works related to the scope of financial predictive models in deep learning.

### Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach [12]

In this study, they use a 2-D Convolutional Neural Network and introduce CNN-TA, a unique algorithmic trading model based on image processing characteristics. A total of fifteen different technical indicators with variable parameter values are employed to transform financial time series into 2-D visuals. [12]

Data is produced for 15 days by each indicator instance. 15x15-inch 2-D images are created, therefore. Depending on the hills and valleys in the original, each image is then marked as Buy, Sell, or Hold. various circumstances the results show that the trained model outperforms the untrained model over a lengthy out-of-sample period across various popular trading platforms when compared to the Buy & Hold and Buy & Sell strategies.

In this paper, they propose a novel approach for transforming a 1-D financial time series into a 2-D image-like data representation that can be utilized to drive deep convolutional neural networks in an algorithmic trading system. To create such a depiction, 15 distinct technical indicator instances with various parameter settings, each with 15 days, were updated to reflect the data in each column. [12]

Like this, the x-axis of each row includes a time series of data for each technical indicator that spans 15 days. Additionally, the rows are organized so that relevant indicators are grouped to satisfy the y-axis locality requirements. Consequently, 15x15 pixel images are produced and fed into the deep convolutional neural network. The 2-D representation of financial technical analysis time series data and feeding it as the input for a 2-D image classification based on deep CNN, namely CNN-TA, for a financial trading system, in their opinion, is unique because it has never been used not only for any trading system but also in any financial prediction model in the manner they suggest. Performance assessments show that such an approach works remarkably well even over long periods of time. The suggested approach beat Buy & Hold, popular technical indicator-based models, the most widely used neural network, MLP, and the most advanced deep learning time series forecasting model, LSTM, on both short and long out-of-sample periods. They believe the suggested model has potential, even though this is one of the first attempts to use such an unconventional technique. They also believe that model fine-tuning and parameter optimization could boost performance even further.

### Financial Time-series Data Analysis using Deep Convolutional Neural Networks [13]

This work presents a revolutionary financial time-series analysis tool based on deep learning techniques. Time-series data processing and financial market forecasting are the main topics of this study. The main contribution of this study is to enhance algorithmic trading.

This study suggests a fresh deep learning-based approach to financial time-series analysis. A spate of successful applications in artificial intelligence and multimedia fields, including image identification, robot vision, and natural language processing, have been made possible by the explosion in popularity of deep learning research in recent years. In this study, the researchers concentrate on processing and forecasting time-series data in financial markets. To extract numerical features using conventional feature extraction approaches in intelligent trading decision support systems, a variety of technical indicators and expert rules are used. The primary contribution of this paper is an improvement to the algorithmic trading framework using deep convolutional neural networks and planar feature representation methodologies (CNN). The proposed method is developed and benchmarked utilizing historical datasets from Taiwan Stock Index Futures. The results of the tests imply that deep learning is efficient in their trading simulation application and may have more modelling potential for challenging social science problems and noisy financial data. They believe that the deep learning framework and methodology they have provided will be applied to more creative applications in the upcoming financial technology (FinTech).

The main contribution of this paper is:

1. They create the mean average mapping method (MAM) and the double moving average mapping method to convert time series data into 2D images (DMAM). To be used in training, the altered photographs must retain all their original metadata and be identified by CNN.
2. With the help of their suggested altered photographs, CNN can compile helpful data and classify the price trend appropriately. CNN will be able to perform financial forecasts and other time series analyses, expanding its capabilities beyond 2D data display.

They consider this study to be an outstanding illustration of unique Financial Technology (FinTech) applications that have attempted to employ contemporary technology to address financial issues and generate novel applications. Artificial intelligence and big data analysis are crucial components of this developing research area since they help to uncover more hidden knowledge and advance service automation. They also think it can be used to a range of specialized, individual, and unmanned service designs, as well as intelligent trading systems including high-frequency trading and algorithmic trading.

### CNN-based stock market prediction using a diverse set of variables [10]

They propose a CNN-based framework in this paper for assessing data from many sources, including various markets. To identify characteristics that can be used to predict the future of such markets, using a range of beginning parameters, the suggested framework has been utilized to forecast the movement of the S&P 500, NASDAQ, DJI, NYSE, and RUSSELL indexes as well as the direction of the weather for the next day. The forecasts perform notably better when compared to the assessments' performance against the baseline of the latest technology. [10]

One of the major difficulties in the field of market prediction is feature extraction from financial data, and several solutions have been proposed to solve this problem. Convolutional neural networks (CNN) have recently been used for market prediction and autonomous feature selection, among other things.

The link across various markets, however, has gotten less attention in the experiments so far as a potential source of data for extracting features. In this study, they provide a framework based on CNN that can be used to extract features from a set of data from various sources, including various markets, to estimate the future of those markets. The suggested methodology has been used to forecast the direction of movement for the S&P 500, NASDAQ, DJI, NYSE, and RUSSELL indices the next day based on different sets of beginning data. The evaluations show a significant improvement in prediction performance when compared to cutting-edge baseline approaches.

Due to CNN's track record in other industries and its prior achievements in the field of market forecasting, they based their strategy on it. The S&P 500, NASDAQ, Dow Jones Industrial Average, NYSE, and RUSSELL stock market indices are used as a test case to show how CNN can be used in their proposed framework, CNNpred, to capture potential correlations between different variables for extracting combined features from a variety of input data. Other variables included in the test case include currency exchange rates, futures contracts, commodity price 90, and significant market indices. [10]

Additionally, they design their filters in a way that considers the characteristics of financial variables.

1. using a CNN-based framework to combine many variables for feature extraction and market forecasting. Since a variety of factors influence financial market behavior, it is crucial to obtain as much relevant information as you can. Their basic variable collection effectively addresses a variety of stock-related variables, and it is easy expandable to include additional potential variables.
2. According to their understanding, this is the first paper to propose a CNN that, prior to training the network to extract useful features for forecasting each of the key stock markets, combines and aligns a wide variety of variables as input using a 3-dimensional tensor.

### Predicting Financial Prices of Stock Market using Recurrent Convolutional Neural Networks [14]

In this study, they suggest merging architectures for a trading signal prediction that simultaneously benefits from CNN and RNN advantages. Their model is presented to financial time series forecasting signals via a CNN layer after being input into a GRU layer to capture long-term signal correlations. The GRU model outperforms traditional RNNs in sequential learning tasks and fixes the exploding and disappearing gradient issues. Using three datasets from the Hang Seng Indexes (HSI), the Deutscher Aktien Index (DAX), and the S&P 500 Index from 2008 to 2016, they compare their model against existing deep learning methodologies.

For a very long time, one of the most challenging issues in financial market analysis has been financial time-series prediction. In the past ten years, scientists in several domains of time-series prediction have focused a lot of attention on deep neural networks, an efficient data mining technique. Convolutional neural network (CNN) and recurrent neural network (RNN) models are now considered industry standards for financial forecasting. In this study, they propose combining architectures for the trading signal prediction that benefit from both CNN and RNN advantages simultaneously. To capture long-term signal relationships, their model is basically fed into a gated recurrent unit (GRU) layer. A CNN layer then presents their model to financial time series forecasting signals. In sequential learning tasks, the GRU model performs better than regular RNNs and resolves the exploding and vanishing gradients issues. They test their model against other deep learning techniques using three datasets for stock indices from 2008 to 2016: the Hang Seng Indexes (HSI), the Deutscher Aktienindex (DAX), and the S&P 500 Index. The studies' findings showed that the recommended GRU-CNN model had the highest prediction accuracy, with values of 56.2% on the HIS dataset, 56.1 on the DAX dataset, and 56.3 on the S&P500 dataset.

They develop two network topologies using a combination technique in this study, and their contributions are outlined below.:

1. They concentrate on improving the common gated recurrent unit (GRU) model, which addresses the standard RNNs' vanishing gradient and growing problem with the gating mechanism while maintaining the influence of LSTM (is a well-known variant of traditional RNNs).
2. To find financial marketing predictions based on return predictive signals, they suggested integrating GRU and CNN architecture.
3. They trained their model (GRU-CNN) using an attention mechanism and evaluated its performance against that of conventional deep learning models.
4. The results of their experiments show that their improved GRU-CNN model exceeds prior conventional methods in terms of prediction accuracy.

In statistics and economics, the current GRU-based model delivers good accuracy and increased returns. The suggested GRU-CNN model, however, performs somewhat better than the GRU-based model.

### Financial Markets Prediction with Deep Learning [15]

The dynamics of the financial markets are complicated, making them difficult to anticipate. Although several recent studies have employed machine learning techniques to forecast financial markets, the findings have not been financially rewarding. This research offers a novel one-dimensional convolutional neural networks (CNN) model to forecast financial market movement. As the specialized one-dimensional convolutional layers scan financial trading data over time, different types of data, such prices and volume, share parameters (kernels). Instead of employing conventional technical indicators, their program automatically extracts traits, avoiding biases brought on by the choice of technical indicators and pre-established correlations in technical indicators. To assess the effectiveness of their prediction model, Using historical trade data for six futures from January 2010 to October 2017, they carefully back-test it. The outcomes of their experiment demonstrate that their CNN model extracts more general and useful features than traditional technical indicators and delivers more consistent and lucrative financial performance than prior machine learning techniques. [10]

Recent evidence indicates that this is one of the first attempts to use deep CNN to forecast financial markets, and they rigorously back-test the model to determine its effectiveness. To assess their notion, six futures from the New York Mercantile Exchange and the Chicago Mercantile Exchange are chosen. Their 1-D CNN model surpasses prior methods based on Nearest Neighbour, SVM, and Deep Feedforward Networks in terms of average yearly return and robustness, according to back-test results (better Sharpe ratio).

Additionally, they observe that their 1-D CNN model, which does not use technical indicators as input, outperforms the model that does. This indicates that their 1-D CNN model is more capable than conventional technical indicators of extracting general and instructive properties. Their findings are consistent with the idea that conventional machine learning metrics, such as accuracy and F1 score, are inappropriate for predicting financial markets because different forms of prediction errors have varied effects on financial performance.

They propose a Weighted-F-Score, a modified F-measure score, to deal with this problem in this study. According to their back-test findings, Weighted-F-Score shows a significant association with both the Sharpe ratio and the average yearly return, with minimal cross-correlation values of 0.79 and 0.84, respectively.

Following is a summary of their contributions.:

1. **One-Dimensional Cross-Data Convolution.** Therefore, they are not immediately applicable to this study. Regular 2-D convolutions may not be able to convolve various aspects of historical financial trading data, such as price and volume, to provide useful conclusions.

Because it is unable to capture the characteristics needed to represent the combined distribution of elements in financial historical trade data, regular 1-D convolution is likewise useless. They provide Cross-Datatype 1-D convolution, a type of 1-D convolution, as a solution to the issue. Based on modified 1-D convolution, their model beats earlier approaches by 6.1 percent - 53.0 percent on average annual return and 53.0 percent - 199.0 percent on Sharpe ratio.

1. **Deep Features Auto-Extraction.** Technical indicators are frequently used as input in machine learning methods for financial market forecasting.

This paper suggests replacing common technical indications with deep features recovered using Cross-Datatype 1-D convolution to collect more all-encompassing and practical information. To the best of their knowledge, this is one of the first systems that extract features using CNN rather than customary technical indications. Their analysis demonstrates that this approach yields cutting-edge outcomes in both financial and machine learning benchmarks.

1. **Correlation between Finance and Machine Learning Metrics.** The effects of various prediction errors on trading in the financial markets vary.

The trader will lose more money, for instance, if the prognosis is for an increase while the price decreases or vice versa.

1. Because the many different forms of errors are not considered by traditional machine learning metrics like accuracy and F1 score, they only have a tenuous link with financial KPIs. To close the gap between machine learning and finance metrics, they suggest the Weighted-F-Score, a modified version of the F score for Type I and Type II mistakes.
2. They discovered a substantial connection between the Weighted-F-Score and financial indicators (the minimum Cross-correlation value is 0.79).

## Research Gap

The application of deep learning to forecast Forex markets has received very little study; instead, the stock market and other financial markets are the major areas of interest. Additionally, I discovered that there had been no investigation into the viability of forecasting the USD/NGN currency combination.

## Conclusion

In conclusion, I noticed that the usage of CNNs in some capacity is a recurrent feature in all the work studied, and I intend to reproduce the results of the reviewed literature.

# SYSTEM METHODOLOGY

## Introduction

In this chapter, I would be discussing my dataset, my architecture, and all the necessary information needed to complete this research.

## Dataset

The dataset employed in this paper is mostly influenced by the dataset created by [10], except for the fact that the dataset used in [10] is of five different U.S stock market indices.

In CNN models, I display our data as a matrix; the characteristics across several timesteps are displayed as a 2D array. Each of the five CSV files contains data for a different currency pair. Date and name columns in the supplied data help identify the currency pair's ticker symbol. The name column is deleted, while the date column is kept as the time index. The final values are numbers. [16]

As already said, my objective is to develop a model that can forecast the general trend of price changes on the Forex market. The USD/NGN currency pair's movement was predicted using my approach. In my prediction task, I use eighty-two variables to represent each day of each currency pair. While the other variables are economic factors that are replicated across all currency pairings in the data set, some of these variables are exclusive to a given currency pair. The various variables offered include fundamental variables, technical indicators, international stock market indices, the US dollar's exchange rate versus other currencies, commodities, information from key US market companies, futures contracts, and other crucial elements. Some of these factors are important because they show processes that have an impact on the Forex markets either directly or indirectly. While not representing causal links, other variables are nonetheless significant because they offer cues or signals that can help the system make short-term predictions about the direction of the markets. I'll quickly discuss each of the several groups in our variable set below; the table below contains more details.

1. Primitive Variable: The close price and the day of the week on which the prediction is to be created are the only primitive variables employed in this investigation.
2. Technical Indicators: To examine short-term price movement, technical analysts utilize indicators derived from past data on Forex prices and trading information. They are regularly consulted while conducting research on the currency market. An illustration of this kind of variable is the moving average.
3. World Stock Markets: The nature of economic globalization causes regular interaction between forex markets throughout the globe. This link is even more helpful when time zones are considered since it enables us to predict how a country's market will develop by keeping an eye on the markets of other nations. Think about how markets like China, Japan, and South Korea may affect the US market.
4. The Exchange Rate of U.S. Dollar: Multinational firms either distribute their products abroad or import their needs from several other countries. As a result, the earnings of these companies are impacted by changes in the US dollar's value in relation to other currencies like the Canadian dollar and the European Euro. The demand for these firms' shares changes along with the price of their stock whenever a change in earnings is revealed. Changes in demand for goods and services by international corporations also have an impact on domestic firm stock prices. As a result, both direct and indirect effects of currency exchange rates on stock prices exist.
5. Commodities: The price of commodities like gold, silver, oil, or wheat is another factor that may be used to predict how the Forex market would behave. Such information could offer perception into the world market. Researchers have discovered a connection between the forex and commodities markets, particularly during the US financial crisis of 2007–2008.

In addition, commodities have joined stocks and FX trading as crucial components of portfolios. This implies that knowledge of commodity prices can help forecast changes in Forex pricing.

1. Big U.S. Companies: Market indices are generated in the US utilizing a range of equities. This formula gives each stock a weight that is equivalent to its market share.

Or, to put it another way, when projecting market indices in the United States, large firms are more significant than small ones. Examples of this are Apple Inc. and Exxon Mobil Corporation. The use of US market index prices is crucial due to their impact on the US economy and, consequently, the US Dollar.

1. Futures Contracts: Contracts for future delivery of stocks, commodities, or other goods are known as futures contracts. These agreements outline the product's anticipated future worth. Investors are more inclined to buy stocks with a greater expected value than their current value. For instance, S&P 500 Futures, DJI Futures, and NASDAQ Futures may influence the current price of the S&P 500 and other indices.
2. Other Useful Variables: Other factors that can be used to anticipate the US stock market include Treasury bill rates, term spreads, and default spreads.

Table 3.1 Dataset Features [10]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Feature | Explanation | Category | Derivation |
| 1 | Day | Day of the Week | Elementary | Pandas |
| 2 | Close | Price Close | Elementary | Excel STOCKHISTORY Formula |
| 3 | MOM – 1 | “Return of 2 days” | Technical Analysis | Formula |
| 4 | MOM – 2 | “Return of 3 days” | Technical Analysis | Formula |
| 5 | MOM – 3 | “Return of 4 days” | Technical Analysis | Formula |
| 6 | ROC – 5 | “Rate of Change of 5 days” | Technical Analysis | Formula |
| 7 | ROC – 10 | “Rate of Change of 10 days” | Technical Analysis | Formula |
| 8 | ROC – 15 | “Rate of Change of 15 days” | Technical Analysis | Formula |
| 9 | ROC – 20 | “Rate of Change of 20 days” | Technical Analysis | Formula |
| 10 | EMA – 10 | “Exponential Moving Average of 10 days” | Technical Analysis | Formula |
| 11 | EMA – 20 | “Exponential Moving Average of 20 days” | Technical Analysis | Formula |
| 12 | EMA – 50 | “Exponential Moving Average of 50 days” | Technical Analysis | Formula |
| 13 | EMA – 200 | “Exponential Moving Average of 200 days” | Technical Analysis | Formula |
| 14 | DTB4WK | “4 Week Treasury Bill: Secondary Market Rate” | Additional | FRED |
| 15 | DTB3 | “3 Month Treasury Bill: Secondary Market Rate” | Additional | FRED |
| 16 | DTB6 | “6 Month Treasury Bill: Secondary Market Rate” | Additional | FRED |
| 17 | DGS5 | “5 Year Treasury Constant Maturity Rate” | Additional | FRED |
| 18 | DGS10 | “10 Year Treasury Constant Maturity Rate” | Additional | FRED |
| 19 | DAAA | “Moody’s Seasoned Aaa Corporate Bond Yield” | Additional | FRED |
| 20 | DBAA | “Moody’s Seasoned Baa Corporate Bond Yield” | Additional | FRED |
| 21 | TE1 | “DGS10 – DTB4WK” | Additional | FRED |
| 22 | TE2 | “DGS10 – DTB3” | Additional | FRED |
| 23 | TE3 | “DGS10 – DTB6” | Additional | FRED |
| 24 | TE5 | “DTB3 – DTB4WK” | Additional | FRED |
| 25 | TE6 | “DTB6 – DTB4WK” | Additional | FRED |
| 26 | DE1 | “DBAA – DAAA” | Additional | FRED |
| 27 | DE2 | “DBAA – DGS10” | Additional | FRED |
| 28 | DE4 | “DBAA – DTB6” | Additional | FRED |
| 29 | DE5 | “DBAA – DTB3” | Additional | FRED |
| 30 | DE6 | “DBAA – DTB4WK” | Additional | FRED |
| 31 | CTB3M | “Change in the market yield on U.S Treasury securities at 3-month Constant maturity, quoted on an investment basis” | Additional | FRED |
| 32 | CTB6M | “Change in the market yield on U.S Treasury securities at 6-month Constant maturity, quoted on an investment basis” | Additional | FRED |
| 33 | CTB1Y | “Change in the market yield on U.S Treasury securities at 1-Year Constant maturity, quoted on an investment basis” | Additional | FRED |
| 34 | OIL | “Relative change of oil price(WTI), Oklahoma” | Natural Commodity | FRED |
| 35 | OIL | “Relative change of oil price(Brent)” | Natural Commodity | Investing.com |
| 36 | OIL | “Relative change of oil price(WTI)” | Natural Commodity | Investing.com |
| 37 | GOLD | “Relative change of gold price(London markets)” | Natural Commodity | FRED |
| 38 | GOLD – F | “Relative change of gold price futures” | Natural Commodity | Investing.com |
| 39 | XAU – USD | “Relative change of gold spot U.S. Dollar” | Natural Commodity | Investing.com |
| 40 | XAG – USD | “Relative change of silver spot U.S. Dollar” | Natural Commodity | Investing.com |
| 41 | GAS | “Relative change in gas price” | Natural Commodity | Investing.com |
| 42 | SILVER | “Relative change in the silver price” | Natural Commodity | Investing.com |
| 43 | COPPER | “Relative change of copper future” | Natural Commodity | Investing.com |
| 44 | IXIC | “Return of NASDAQ Composite index” | Index | Yahoo Finance |
| 45 | GSPC | “Return of S&P 500 index” | Index | Yahoo Finance |
| 46 | DJI | “Return of Dow Jones Industrial Average” | Index | Yahoo Finance |
| 47 | NYSE | “Return of NY stock exchange index” | Index | Yahoo Finance |
| 48 | RUSSELL | “Return of RUSSELL 2000 index” | Index | Yahoo Finance |
| 49 | HIS | “Return of Hang Seng index” | Index | Yahoo Finance |
| 50 | SSE | “Return of Shang Hai Stock Exchange Composite index” | Index | Yahoo Finance |
| 51 | FCHI | “Return of CAC 40” | Index | Yahoo Finance |
| 52 | FTSE | “Return of FTSE 100” | Index | Yahoo Finance |
| 53 | GDAXI | “Return of DAX” | Index | Yahoo Finance |
| 54 | USD – Y | “Relative change of U.S. Dollar to Japanese Yen exchange rate” | Currency Pair | Yahoo Finance |
| 55 | USD – GBP | “Relative change of U.S. Dollar to Great Britain Pound exchange rate” | Currency Pair | Yahoo Finance |
| 56 | USD – CAD | “Relative change of U.S. Dollar to Canadian Dollar exchange rate” | Currency Pair | Yahoo Finance |
| 57 | USD – CNY | “Relative change of U.S. Dollar to Chinese Yuan exchange rate” | Currency Pair | Yahoo Finance |
| 58 | USD – AUD | “Relative change of U.S. Dollar to Australian Dollar exchange rate” | Currency Pair | Investing.com |
| 59 | USD – NZD | “Relative change of U.S. Dollar to New Zealand Dollar exchange rate” | Currency Pair | Investing.com |
| 60 | USD – CHF | “Relative change of U.S. Dollar to Swiss Franc exchange rate” | Currency Pair | Investing.com |
| 61 | USD – EUR | “Relative change of U.S. Dollar to European Euro exchange rate” | Currency Pair | Investing.com |
| 62 | USDX | “Relative change in U.S. Dollar index” | Currency Pair | Investing.com |
| 63 | XOM | “Return of Exon Mobil Corporation” | Company | Yahoo Finance |
| 64 | JPM | “Return of JP Morgan Chase & Co.” | Company | Yahoo Finance |
| 65 | AAPL | “Return of Apple Inc.” | Company | Yahoo Finance |
| 66 | MSFT | “Return of Microsoft Corporation” | Company | Yahoo Finance |
| 67 | GE | “Return of General Electric Company” | Company | Yahoo Finance |
| 68 | JNJ | “Return of Johnson & Johnson” | Company | Yahoo Finance |
| 69 | WFC | “Return of Wells Fargo & Company” | Company | Yahoo Finance |
| 70 | AMZN | “Return of Amazon.com Inc.” | Company | Yahoo Finance |
| 71 | FCHI – F | “Return of CAC 40 Futures” | Future | Investing.com |
| 72 | FTSE – F | “Return FTSE 100 Futures” | Future | Investing.com |
| 73 | GDAXI – F | “Return of DAX Futures” | Future | Investing.com |
| 74 | HSI – F | “Return of Hang Seng Index futures” | Future | Investing.com |
| 75 | NIKKEI – F | “Return of Nikkei index Futures” | Future | Investing.com |
| 76 | KOSPI – F | “Return of Korean stock exchange Futures” | Future | Investing.com |
| 77 | IXIC – F | “Return of NASDAQ Composite index Futures” | Future | Investing.com |
| 78 | DJI – F | “Return of Dow Jones Industrial Average Futures” | Future | Investing.com |
| 79 | S&P – F | “Return of S&P 500 index Futures” | Future | Investing.com |
| 80 | RUSSELL – F | “Return of RUSSELL Futures” | Future | Investing.com |
| 81 | USDX – F | “Return of U.S. Dollar Index Futures” | Exchange Rate | Investing.com |
| 82 | TNX | “Treasury Yield 10 Years” | Additional | Yahoo Finance |

## Architecture

The architecture of this paper is built around the work done on [16], by making improvements to [10]. This paper takes the architecture of [16] and implements the dataset we created, while also adjusting it to improve the accuracy of the model in tandem with our dataset.

Table 3.2 2D Model Architecture

|  |
| --- |
| Model: “sequential\_1” |
| Layer (type) | Output Shape | Param# |
| conv2d\_5 (Conv2D) | (None, 60, 1, 8) | 664 |
| Conv2d\_6 (Conv2D) | (None, 58,1 ,8) | 200 |
| Max\_pooling2d\_2 (MaxPooling 2D) | (None, 29, 1, 8) | 0 |
| Conv2d\_7 (Conv2D) | (None, 27, 1, 8) | 200 |
| Conv2d\_8 (Conv2D) | (None, 25, 1, 8) | 200 |
| Conv2d\_9 (Conv2D) | (None, 23, 1, 8) | 200 |
| Max\_pooling2d\_3 (MaxPooling 2D) | (None, 11, 1, 8) | 0 |
| Flatten\_1 (Flatten) | (None, 27, 88) | 0 |
| Dropout\_1 (Dropout) | (None, 27, 88) | 0 |
| Dense\_1 (Dense) | (None, 1) | 89 |
| Total params: 1,553 |
| Trainable params: 1,553 |
| Non-Trainable params: 0 |

Table 3.3 3D Model Architecture

|  |
| --- |
| Model: “sequential” |
| Layer (type) | Output Shape | Param# |
| conv2d (Conv2D) | (None, 5, 60, 8) | 664 |
| Conv2d\_1 (Conv2D) | (None, 1, 58,8) | 968 |
| Max\_pooling2d (MaxPooling 2D) | (None, 1, 29, 8) | 0 |
| Flatten (Flatten) | (None, 232) | 0 |
| Dropout (Dropout) | (None, 232) | 0 |
| Dense (Dense) | (None, 1) | 233 |
| Total params: 1,865 |
| Trainable params: 1,865 |
| Non-Trainable params: 0 |

### Pre-processing of Data

[16] before attempting to predict market direction, seeks to create a categorization label. By comparing the closing value of tomorrow to the closing value of today, one may estimate the market direction. If I read the data into a pandas DataFrame, I can obtain the percentage change, which is a sign that the market is rising. I alter the name to a single-time stride back as a result. I determine the closing index's percentage change and contrast it with the information from the day before. The data is then converted to 1 or 0, depending on whether the percentage change is positive or negative.

I open each of the folder's eight data files as a separate pandas DataFrame and save the results in a Python dictionary. With the column "Target" acting as the classification label and the other columns acting as input characteristics, the result is a DataFrame for each currency pair. [16] additionally, uniformizes the data using a common scaler. Instead of arbitrarily dividing the data into training and test sets, it is typically preferable to designate a cut-off point where the data before the cut-off is the training set and the data after the cut-off is the test set in time series issues. Although the training set serves as the focal point for the scaling, the entire dataset is subject to it.

### Data Generator

The architecture defined by [16] uses a set of N time steps rather than all the time steps at once to predict the direction of the market at step N+1. In this paradigm, the window of N time steps might start at any point. Simply put, we can produce a lot of overlapped DataFrames. To conserve memory, they next develop a data generator for training and validation. [16]

A generator in Python is a specific function that iteratively creates a stream of data rather than returning a result. A generator must create a batch of data inputs and a target to be used in Keras training. This generator is made to operate continuously forever. Therefore, an infinite loop that starts with True is used to construct the generator function. Each time, a DataFrame is chosen at random from the Python dictionary, and N time steps are taken using the pandas "iloc[start: end]" syntax to construct an input under the variable frame inside the training set's collection of time steps (i.e., the beginning section). This DataFrame was created using a 2D array. The target label is the label of the previous time step. The input data and label are then combined to create the list batch. Once it had accumulated enough for one batch size, it was then released from the generator. [16]

Sending a batch for training or validation is the data generator's final step. [16] then assigns a collection of input data—each a 2D array—and a list of target labels to variables X and y, converting them to NumPy arrays so their Keras model can handle them. They then use "np. expand dims()" to give the NumPy array X an additional dimension due to the network model's structure.

### The Model

The model used in [16] are two, the first being a 2D CNN model and the other being a 3D CNN model, both gotten from [10].

[10]’s input tensor for a 2D CNN model is of the form N x m x 1, where N is the total number of time steps and m is the total number of features in each time step. We use m = 82 and n = 60 in [16]. The model consists of three convolutional layers, the first of which applies to all features in each time step and has eight units. Because it is commonly accepted that three days would produce a trend in the trading markets, it is followed by a second convolutional layer that takes three consecutive days into account at once. Prior to being applied to a max-pooling layer and another convolutional layer, it is flattened into a one-dimensional array and applied to a fully connected layer with sigmoid activation for binary classification. [10] named the previously stated model "2D-CNNpred," and a second iteration is known as "3D-CNNpred."

The objective is to help in the forecast of a single index by considering not only the multiple components of a single market index but also by contrasting it with several other market indices. Data for one market index is represented using a 2D array. By stacking several similar data from various indices, a 3D array is produced. Even though the target label is the same, examining a different market could yield some more information that might help with prediction. The convolutional network had to be redefined and the data generators had to be adjusted suitably because the data's structure had changed.

### Training, Validation, and Test

[16] used accuracy, F1 score, and MAE as loss metrics to evaluate the model's quality. It is important to remember that the accuracy and recall ratios, which both take positive categorization into consideration, are the foundation of the F1 score. The F1 from the positive and negative categorization's mean is utilized in [16]. Clearly, it is the F1-macro metric.:

………. eq1

.……… eq2

........….. eq3

................. eq4

F1 = (((2 \* eqn1 \* eqn2) / (eqn1 + eqn2)) + ((2 \* eqn3 \* eqn4)/(eqn3 + eqn4)) ................ eq5

TN : True Negative

TP: True Positive

FN: False Negative

FP: False Positive

The fraction eqn1 is the total number of FP and TP true positives, respectively. Similarly, eqn2 is what you remember. The first term in the large parenthesis above refers to the standard F1 metric, which took positive classifications into consideration. The second word, known as the inverse, takes adverse classifications into account. There is no Keras equivalent to "sklearn.metrics.f1 score()," although it is available in sci-kit-learn. As a result, [16] used the code from this stack exchange question to create their own. [17]

[16]’s model was saved in the middle of the training so that it may be stopped and resumed later. We can use checkpoint features in Keras. They built a checkpoint path filename template and requested Keras to fill it with the validation F1 score and epoch number. We do this by monitoring the F1 measure of the validation, which should increase as the model gets better. They gave it the "mode="max" argument, therefore. Now, training their model should be straightforward. Two things need to be considered. They supplied the function f1macro as the metrics argument and "acc" as the accuracy parameter to the "build()" method. As a result, these two characteristics will be monitored during training. They refer to this measure as val f1macro in the checkpoint's monitor parameter since the function is called f1macro. Separately, they utilized the datagen() generator to provide the input data in the fit() method, as previously mentioned. In the training loop, batches are retrieved one by one from the generator that is formed when this function is invoked. In a similar vein, the generator also offers validation information. Considering that a generator's nature is to continuously deliver data. In the training process, they must define an epoch and explain how to do so. Keep in mind that with Keras, a batch corresponds to one gradient descent update iteration. Each cycle through all the data in a dataset is an epoch. After an epoch, validation should be performed. It is also an appropriate moment to run the checkpoint we put up previously. Because Keras lacks a way for estimating the size of the dataset from a generator, we must tell it how many batches it should process in a single epoch using the steps per epoch option. On the other side, the validation stages option shows how many batches are utilized in each validation step. Although the validation had no influence on the training, it gave them the metrics they were interested in. We evaluate the model with unseen data or the test set once it has finished training. The measure for the training set is updated after each batch, while the one for the validation set is only accessible at the end. They deterministically construct the test set from the dataset as opposed to producing it at random. Testgen() shares the same structure as datagen(), which they had already defined. The first dimension of the output data is the number of samples in a batch, except for datagen(), whereas it is the total number of test samples in testgen(). A floating point between 0 and 1 will be produced by using the prediction model because we are using the sigmoid activation function. We will translate this to 0 or 1 using the 0.5 criteria. Then, using scikit-learn techniques, the accuracy, mean absolute error, and F1 score are calculated (which accuracy is just one subtracted by the MAE).

# RESULTS AND DISCUSSIONS

## System implementation

This system was implemented using a convolutional neural network deep learning method. This method was used because of its demonstrated capabilities in other domains.

### Packages and tools used:

#### Software tools used are:

1. Google Colab – Used to implement the system and make improvements based on the proposed dataset
2. Python v 3.7 is the programming language used to implement the model. It is an interpreted language that contains multiple libraries.

#### Packages used are:

1. NumPy - This is a library in python which provides support for large multidimensional arrays and matrices and a vast collection of high-level mathematical functions to perform operations on these arrays and matrices.
2. Pandas - This is a library in python that provides the Data Frame object for storing datasets during manipulation it also provides reading & writing capabilities to ".CSV".
3. Scikit-Learn - This is a machine learning library in python that comes bundled with various machine learning algorithms and tools. It also provides data mining tools. It was us to implement Logistic Regression when creating the model.
4. TensorFlow - A free and open-source software library for artificial intelligence and machine learning is called TensorFlow. While it may be used to many other tasks, deep neural network training and inference are its main areas of study

#### System Requirements:

1. RAM – 16GB
2. CPU – intel corei7 7th generation
3. GPU – Nvidia GeForce 10 series/AMD Radeon series
4. Storage – 1TB HDD/256 SSD
5. OS – Windows/Linux/macOS

## Experimentation

Various Experiments were run on both the 2D model and 3D models, they include:

1. Increasing and reducing batch size ranging from 80 – 256.
2. Epoch ranging from 10 up to 40.
3. Droprate of between 0% to 20%.
4. Multiple loss functions experimented on including, mean squared error, mean bias error, cross-entropy loss, etc.

**Note:** At the end, the parameters with the highest accuracy in relation to the dataset was adopted.

## Results

Figure 4.1 2D Model Results

The graph shows the accuracy, F1 score and loss of the 2D predictive model over 10 instances.

Figure 4.2 3D Model Results

The graph shows the accuracy, F1 score and loss of the 3D predictive model over 10 instances.

## Summary

In this Chapter I showed how I implemented and experimented on my dataset and architecture. I also presented my results and discussed it.

# SUMMARY, RECOMMENDATION AND CONCLUSION

## Summary

I implemented this research using deep learning techniques, majority of which were convolutional neural networks. I also created my dataset consisting of 82 features varying in commodity prices, index features, currency prices, etc. The predictive algorithm was implemented using python, Tensorflow, scikit learn and a host of other python libraries. Google Colab was used to run the code, because a high-performance pc would be needed to run the experiment. I also ran a lot of experiments to try and achieve the best results possible on the created dataset.

It is impractical to anticipate a high degree of accuracy, as it is with any financial market forecast projects. Both the 2D and 3D models had an average accuracy of 53% for the USD/NGN currency pair over 10 instances of running the model, but the 3D model had a much better F1 score of 0.6 compared to the 2D model’s 0.4. This is already performing better than some of the works of literature reviewed. This research may be of some use but there is no magic formula to making money.

## Recommendation

Given the nature of the research and the time allocated to it, I believe that given enough time improvements can be made to:

1. The Dataset – More features could be added to improve the prediction model and a larger timeframe of the data can be explored.
2. The model - With enough time there would be ample room for experimentation to get even better results.

## Conclusion

I have created original datasets focused on the US Dollar, and I have used these datasets to attempt to predict the USD/NGN currency pair on a Modified Convolutional Neural Network-based architecture, and I have presented my results.

# References

|  |  |
| --- | --- |
| [1] | N. Lioudis, S. Anderson and P. Rathburn, "Investopedia," Dotdash Meredith, 21 July 2021. [Online]. Available: https://www.investopedia.com/ask/answers/08/what-is-foreign-exchange.asp#:~:text=Foreign%20exchange%2C%20or%20forex%2C%20is,to%20a%20basket%20of%20currencies.. [Accessed 9 May 2022]. |
| [2] | J. Chen, G. Scott and D. Constagliola, "Investopedia," Dotdash Meredith, 8 December 2021. [Online]. Available: https://www.investopedia.com/articles/forex/11/why-trade-forex.asp. [Accessed 10 June 2022]. |
| [3] | K. Reyes, "Simplilearn," Disqus, 21 February 2022. [Online]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-deep-learning. [Accessed 9 May 2022]. |
| [4] | I. C. E. "IBM," 1 May 2020. [Online]. Available: https://www.ibm.com/cloud/learn/deep-learning. [Accessed 11 June 2022]. |
| [5] | "MathWorks," MathWorks inc., [Online]. Available: https://www.mathworks.com/discovery/deep-learning.html. [Accessed 11 June 2022]. |
| [6] | "MATLAB for Deep Learning - MATLAB & Simulink," [Online]. Available: https://www.mathworks.com/solutions/deep-learning.html. [Accessed 24 July 2022]. |
| [7] | M. Mandal, "Analytics Vidhya," Analytics Vidhya, 1 May 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/. [Accessed 9 May 2022]. |
| [8] | S. Saha, "Towards Data Science," Medium, 15 December 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed 11 June 2022]. |
| [9] | R. Yamashita, M. Nishio, R. Kinh Gian Do and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights Imaging,* vol. 9, no. 4, pp. 611-629, 2018. |
| [10] | E. Hoseinzade and S. Haratizadeh, "CNNpred: CNN-based stock market prediction using a diverse set of variables," *Expert Systems with Applications,* vol. 129, no. 0957-4174, pp. 273-285, 2019. |
| [11] | P. Juneja, " Management Study Guide," Management Study Guide, [Online]. Available: https://www.managementstudyguide.com/advantages-and-disadvantages-of-forex-market.htm. [Accessed 9 May 2022]. |
| [12] | O. B. Sezar and A. M. Ozbayoglu, "Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to image conversion approach," *Applied Soft Computing,* vol. 70, no. 1568-4946, pp. 525-538, 2018. |
| [13] | J.-F. Chen, W.-L. Chen, C.-P. Huang, S.-H. Huang and A.-P. Chen, "Financial Time-Series Data Analysis Using Deep Convolutional Neural Networks," *7th International Conference on Cloud Computing and Big Data (CCBD),* pp. 87-92, 2016. |
| [14] | M. Zulqarnain, R. Ghazali, M. Ghouse, Y. Mazwin, H. Mohmad and I. Javid, "Predicting Financial Prices of Stock Market using Recurrent Convolutional Neural Networks," *International Journal of Intelligent Systems and Applications,* 2020. |
| [15] | J. Wang, T. Sun, B. Liu, Y. Cao and D. Wang, " Financial Markets Prediction with Deep Learning," *17th IEEE International Conference on Machine Learning and Applications (ICMLA),* pp. 97-104, 2018. |
| [16] | A. Tam, "Machine Learning Mastery," 15 November 2021. [Online]. Available: machinelearningmastery.com/using-cnn-for-financial-time-series-prediction/. [Accessed 15 June 2022]. |
| [17] | "Data Science Stack Exchange," 1 August 1966. [Online]. Available: https://datascience.stackexchange.com/questions/45165/how-to-get-accuracy-f1-precision-and-recall-for-a-keras-model. [Accessed 24 July 2022]. |

# APPENDIX

**2D Model Code:**

from google.colab import drive

drive.mount('/content/drive') %cd /content/drive/MyDrive/CNNpred-Keras-main

import os

import random

import numpy as np

import pandas as pd

import tensorflow as tf

from tensorflow.keras import backend as K

from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, Input

from tensorflow.keras.models import Sequential, load\_model

from tensorflow.keras.callbacks import ModelCheckpoint

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score, f1\_score, mean\_absolute\_error

DATADIR = "./Dataset2"

TRAIN\_TEST\_CUTOFF = '2015-11-26'

TRAIN\_VALID\_RATIO = 0.75

def recall\_m(y\_true, y\_pred):

    true\_positives = K.sum(K.round(K.clip(y\_true \* y\_pred, 0, 1)))

    possible\_positives = K.sum(K.round(K.clip(y\_true, 0, 1)))

    recall = true\_positives / (possible\_positives + K.epsilon())

    return recall

def precision\_m(y\_true, y\_pred):

    true\_positives = K.sum(K.round(K.clip(y\_true \* y\_pred, 0, 1)))

    predicted\_positives = K.sum(K.round(K.clip(y\_pred, 0, 1)))

    precision = true\_positives / (predicted\_positives + K.epsilon())

    return precision

def f1\_m(y\_true, y\_pred):

    precision = precision\_m(y\_true, y\_pred)

    recall = recall\_m(y\_true, y\_pred)

    return 2\*((precision\*recall)/(precision+recall+K.epsilon()))

def f1macro(y\_true, y\_pred):

    f\_pos = f1\_m(y\_true, y\_pred)

    # negative version of the data and prediction

    f\_neg = f1\_m(1-y\_true, 1-K.clip(y\_pred,0,1))

    return (f\_pos + f\_neg)/2

def cnnpred\_2d(seq\_len=60, n\_features=82, n\_filters=(8,8,8,8,8), droprate=0.1):

    "2D-CNNpred model according to the paper"

    model = Sequential([

        Input(shape=(seq\_len, n\_features, 1)),

        Conv2D(n\_filters[0], kernel\_size=(1, n\_features), activation="relu"),

        Conv2D(n\_filters[1], kernel\_size=(3,1), activation="relu"),

        MaxPool2D(pool\_size=(2,1)),

        Conv2D(n\_filters[2], kernel\_size=(3,1), activation="relu"),

        Conv2D(n\_filters[3], kernel\_size=(3,1), activation="relu"),

        Conv2D(n\_filters[3], kernel\_size=(3,1), activation="relu"),

        MaxPool2D(pool\_size=(2,1)),

        Flatten(),

        Dropout(droprate),

        Dense(1, activation="sigmoid")

    ])

    return model

def datagen(data, seq\_len, batch\_size, targetcol, kind):

    "As a generator to produce samples for Keras model"

    batch = []

    while True:

        # Pick one dataframe from the pool

        key = random.choice(list(data.keys()))

        df = data[key]

        input\_cols = [c for c in df.columns if c != targetcol]

        index = df.index[df.index < TRAIN\_TEST\_CUTOFF]

        split = int(len(index) \* TRAIN\_VALID\_RATIO)

        assert split > seq\_len, "Training data too small for sequence length {}".format(seq\_len)

        if kind == 'train':

            index = index[:split]   # range for the training set

        elif kind == 'valid':

            index = index[split:]   # range for the validation set

        else:

            raise NotImplementedError

        # Pick one position, then clip a sequence length

        while True:

            t = random.choice(index)     # pick one time step

            n = (df.index == t).argmax() # find its position in the dataframe

            if n-seq\_len+1 < 0:

                continue # this sample is not enough for one sequence length

            frame = df.iloc[n-seq\_len+1:n+1]

            batch.append([frame[input\_cols].values, df.loc[t, targetcol]])

            break

        # if we get enough for a batch, dispatch

        if len(batch) == batch\_size:

            X, y = zip(\*batch)

            X, y = np.expand\_dims(np.array(X), 3), np.array(y)

            yield X, y

            batch = []

def datagen(data, seq\_len, batch\_size, targetcol, kind):

    "As a generator to produce samples for Keras model"

    batch = []

    while True:

        # Pick one dataframe from the pool

        key = random.choice(list(data.keys()))

        df = data[key]

        input\_cols = [c for c in df.columns if c != targetcol]

        index = df.index[df.index < TRAIN\_TEST\_CUTOFF]

        split = int(len(index) \* TRAIN\_VALID\_RATIO)

        assert split > seq\_len, "Training data too small for sequence length {}".format(seq\_len)

        if kind == 'train':

            index = index[:split]   # range for the training set

        elif kind == 'valid':

            index = index[split:]   # range for the validation set

        else:

            raise NotImplementedError

        # Pick one position, then clip a sequence length

        while True:

            t = random.choice(index)     # pick one time step

            n = (df.index == t).argmax() # find its position in the dataframe

            if n-seq\_len+1 < 0:

                continue # this sample is not enough for one sequence length

            frame = df.iloc[n-seq\_len+1:n+1]

            batch.append([frame[input\_cols].values, df.loc[t, targetcol]])

            break

        # if we get enough for a batch, dispatch

        if len(batch) == batch\_size:

            X, y = zip(\*batch)

            X, y = np.expand\_dims(np.array(X), 3), np.array(y)

            yield X, y

            batch = []

def testgen(data, seq\_len, targetcol):

    "Return array of all test samples"

    batch = []

    for key, df in data.items():

        input\_cols = [c for c in df.columns if c != targetcol]

        # find the start of test sample

        t = df.index[df.index >= TRAIN\_TEST\_CUTOFF][0]

        n = (df.index == t).argmax()

        # extract sample using a sliding window

        for i in range(n+1, len(df)+1):

            frame = df.iloc[i-seq\_len:i]

            batch.append([frame[input\_cols].values, frame[targetcol][-1]])

    X, y = zip(\*batch)

    return np.expand\_dims(np.array(X),3), np.array(y)

data = {}

for filename in os.listdir(DATADIR):

    if not filename.lower().endswith(".csv"):

        continue # read only the CSV files

    filepath = os.path.join(DATADIR, filename)

    X = pd.read\_csv(filepath, index\_col="Date", parse\_dates=True)

    # basic preprocessing: get the name, the classification

    # Save the target variable as a column in dataframe for easier dropna()

    #print(X)

    name = X["Name"][0]

    del X["Name"]

    #print(X.shape)

    cols = X.columns

    X["Target"] = (X["Close"].pct\_change().shift(-1) > 0).astype(int)

    X.dropna(inplace=True)

    # Fit the standard scaler using the training dataset

    index = X.index[X.index < TRAIN\_TEST\_CUTOFF]

    index = index[:int(len(index) \* TRAIN\_VALID\_RATIO)]

    #print(X.loc[index, cols])

    scaler = StandardScaler().fit(X.loc[index, cols])

    # Save scale transformed dataframe

    X[cols] = scaler.transform(X[cols])

    data[name] = X

seq\_len = 60

batch\_size = 128

n\_epochs = 50

n\_features = 82

model = cnnpred\_2d(seq\_len, n\_features)

model.compile(optimizer="adam", loss="mae", metrics=["acc", f1macro])

model.summary()  # print model structure to console

checkpoint\_path = "./2D-models/cp2d-{epoch}-{val\_f1macro:.2f}.h5"

callbacks = [

    ModelCheckpoint(checkpoint\_path,

                    monitor='val\_f1macro', mode="max",

                    verbose=0, save\_best\_only=True, save\_weights\_only=False, save\_freq="epoch")

]

 model.fit(datagen(data, seq\_len, batch\_size, "Target", "train"),

          validation\_data=datagen(data, seq\_len, batch\_size, "Target", "valid"),

          epochs=n\_epochs, steps\_per\_epoch=400, validation\_steps=10, verbose=1, callbacks=callbacks)

test\_data, test\_target = testgen(data, seq\_len, "Target")

test\_out = model.predict(test\_data)

test\_pred = (test\_out > 0.5).astype(int)

print("accuracy:", accuracy\_score(test\_pred, test\_target))

print("MAE:", mean\_absolute\_error(test\_pred, test\_target))

print("F1:", f1\_score(test\_pred, test\_target))

**3D Model Code:**

from google.colab import drive

drive.mount('/content/drive')

%cd /content/drive/MyDrive/CNNpred-Keras-main

import os

import random

import numpy as np

import pandas as pd

import tensorflow as tf

from tensorflow.keras import backend as K

from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, Input

from tensorflow.keras.models import Sequential, load\_model

from tensorflow.keras.callbacks import ModelCheckpoint

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score, f1\_score, mean\_absolute\_error

DATADIR = "./Dataset3"

TRAIN\_TEST\_CUTOFF = '2015-11-26'

TRAIN\_VALID\_RATIO = 0.75

  def recall\_m(y\_true, y\_pred):

      true\_positives = K.sum(K.round(K.clip(y\_true \* y\_pred, 0, 1)))

      possible\_positives = K.sum(K.round(K.clip(y\_true, 0, 1)))

      recall = true\_positives / (possible\_positives + K.epsilon())

      return recall

def precision\_m(y\_true, y\_pred):

    true\_positives = K.sum(K.round(K.clip(y\_true \* y\_pred, 0, 1)))

    predicted\_positives = K.sum(K.round(K.clip(y\_pred, 0, 1)))

    precision = true\_positives / (predicted\_positives + K.epsilon())

    return precision

def f1\_m(y\_true, y\_pred):

    precision = precision\_m(y\_true, y\_pred)

    recall = recall\_m(y\_true, y\_pred)

    return 2\*((precision\*recall)/(precision+recall+K.epsilon()))

def f1macro(y\_true, y\_pred):

    f\_pos = f1\_m(y\_true, y\_pred)

    # negative version of the data and prediction

    f\_neg = f1\_m(1-y\_true, 1-K.clip(y\_pred,0,1))

    return (f\_pos + f\_neg)/2

def cnnpred\_3d(seq\_len=60, n\_stocks=5, n\_features=82, n\_filters=(8,8), droprate=0):

    "3D-CNNpred model according to the paper"

    model = Sequential([

        Input(shape=(n\_stocks, seq\_len, n\_features)),

        Conv2D(n\_filters[0], kernel\_size=(1,1), activation="relu", data\_format="channels\_last"),

        Conv2D(n\_filters[1], kernel\_size=(n\_stocks,3), activation="relu"),

        MaxPool2D(pool\_size=(1,2)),

        Flatten(),

        Dropout(droprate),

        Dense(1, activation="sigmoid")

    ])

    return model

def datagen(data, seq\_len, batch\_size, target\_index, targetcol, kind):

    "As a generator to produce samples for Keras model"

    # Learn about the data's features and time axis

    input\_cols = [c for c in data.columns if c[0] != targetcol]

    tickers = sorted(set(c for \_,c in input\_cols))

    n\_features = len(input\_cols) // len(tickers)

    index = data.index[data.index < TRAIN\_TEST\_CUTOFF]

    split = int(len(index) \* TRAIN\_VALID\_RATIO)

    assert split > seq\_len, "Training data too small for sequence length {}".format(seq\_len)

    if kind == "train":

        index = index[:split]   # range for the training set

    elif kind == 'valid':

        index = index[split:]   # range for the validation set

    else:

        raise NotImplementedError

    # Infinite loop to generate a batch

    batch = []

    while True:

        # Pick one position, then clip a sequence length

        while True:

            t = random.choice(index)

            n = (data.index == t).argmax()

            if n-seq\_len+1 < 0:

                continue # this sample is not enough for one sequence length

            frame = data.iloc[n-seq\_len+1:n+1][input\_cols]

            # convert frame with two level of indices into 3D array

            shape = (len(tickers), len(frame), n\_features)

            X = np.full(shape, np.nan)

            for i,ticker in enumerate(tickers):

                X[i] = frame.xs(ticker, axis=1, level=1).values

            batch.append([X, data[targetcol][target\_index][t]])

            break

        # if we get enough for a batch, dispatch

        if len(batch) == batch\_size:

            X, y = zip(\*batch)

            yield np.array(X), np.array(y)

            batch = []

def testgen(data, seq\_len, target\_index, targetcol):

    "Return array of all test samples"

    input\_cols = [c for c in data.columns if c[0] != targetcol]

    tickers = sorted(set(c for \_,c in input\_cols))

    n\_features = len(input\_cols) // len(tickers)

    t = data.index[data.index >= TRAIN\_TEST\_CUTOFF][0]

    n = (data.index == t).argmax()

    batch = []

    for i in range(n+1, len(data)+1):

        # Clip a window of seq\_len ends at row position i-1

        frame = data.iloc[i-seq\_len:i]

        target = frame[targetcol][target\_index][-1]

        frame = frame[input\_cols]

        # convert frame with two level of indices into 3D array

        shape = (len(tickers), len(frame), n\_features)

        X = np.full(shape, np.nan)

        for i,ticker in enumerate(tickers):

            X[i] = frame.xs(ticker, axis=1, level=1).values

        batch.append([X, target])

    X, y = zip(\*batch)

    return np.array(X), np.array(y)

data = {}

for filename in os.listdir(DATADIR):

    if not filename.lower().endswith(".csv"):

        continue # read only the CSV files

    filepath = os.path.join(DATADIR, filename)

    X = pd.read\_csv(filepath, index\_col="Date", parse\_dates=True)

    # basic preprocessing: get the name, the classification

    # Save the target variable as a column in dataframe for easier dropna()

    name = X["Name"][0]

    del X["Name"]

    cols = X.columns

    X["Target"] = (X["Close"].pct\_change().shift(-1) > 0).astype(int)

    X.dropna(inplace=True)

    # Fit the standard scaler using the training dataset

    index = X.index[X.index < TRAIN\_TEST\_CUTOFF]

    index = index[:int(len(index) \* TRAIN\_VALID\_RATIO)]

    scaler = StandardScaler().fit(X.loc[index, cols])

    # Save scale transformed dataframe

    X[cols] = scaler.transform(X[cols])

    data[name] = X

for key, df in data.items():

    df.columns = pd.MultiIndex.from\_product([df.columns, [key]])

data = pd.concat(data.values(), axis=1)

seq\_len = 60

batch\_size = 128

n\_epochs = 20

n\_features = 82

n\_stocks = 5

model = cnnpred\_3d(seq\_len, n\_stocks, n\_features)

model.compile(optimizer="adam", loss="mae", metrics=["acc", f1macro])

model.summary() # print model structure to console

checkpoint\_path = "./3D-models/cp3d-{epoch}-{val\_f1macro:.2f}.h5"

callbacks = [

    ModelCheckpoint(checkpoint\_path,

                    monitor='val\_f1macro', mode="max",

                    verbose=0, save\_best\_only=True, save\_weights\_only=False, save\_freq="epoch")

]

model.fit(datagen(data, seq\_len, batch\_size, "USDNGN", "Target", "train"),

          validation\_data=datagen(data, seq\_len, batch\_size, "USDNGN", "Target", "valid"),

          epochs=n\_epochs, steps\_per\_epoch=400, validation\_steps=10, verbose=1, callbacks=callbacks)

test\_data, test\_target = testgen(data, seq\_len, "USDNGN", "Target")

test\_out = model.predict(test\_data)

test\_pred = (test\_out > 0.5).astype(int)

print("accuracy:", accuracy\_score(test\_pred, test\_target))

print("MAE:", mean\_absolute\_error(test\_pred, test\_target))

print("F1:", f1\_score(test\_pred, test\_target))