Predicting signals for algorithmic cryptocurrency trading: A hybrid Convolutional Neural Network – Gated Recurrent Unit (CNN-GRU) architecture

Thanh-Nhan “Nicholas” Le
Transdisciplinary School
The University of Technology Sydney
Ultimo NSW 2007, Australia
nhanlt.31@gmail.com

***Abstract*—This paper proposes a hybrid Convolutional Neural Network – Gated Recurrent Unit (CNN-GRU) architecture for producing algorithmic trading signals for three cryptocurrency datasets: Bitcoin, Ethereum, and Binance Coin. It uses the standalone models, CNN and GRU, as benchmarks for comparing both classification and trading performances. The datasets are highly imbalanced, therefore an over- and under-sampling augmentation is employed to help the models train better. All models are trained and tested for each dataset separately, and significance tests for out-of-sample f1-scores and annualised returns are conducted using 30 experiments representing 30 data splits in a rolling window. Results on all datasets show that classification performance of CNN-GRU in terms of the f1-score is quite subpar comparing to that of the standalone models; however, this model achieves the highest mean annualised returns. This is found to be due to the hybrid model underperforming in recalling actual “Buy” and “Sell” decisions, but actually makes fewer wrong trades and catches more profitable trades when it predicts correctly. Although the outcomes are financially promising, the paper has not explored algorithmic trading in its full glory, so results are open to further improvements. Possible future works include employing other methods for improving imbalanced classification, more feature engineering and testing with different timeframes, and a more involved approach with feature explainability to improve confidence and reliability when the model is actually deployed in a live environment.**

***Index Terms*—cryptocurrency, signal, prediction, imbalanced, classification, Convolutional Neural Network, Gated Recurrent Unit, hybrid model**

I. Introduction

Cryptocurrencies have enjoyed widespread adoption in various sectors since the introduction of the first electronic currency system called Bitcoin in 2008. [2] In recent years, the cryptocurrency market has gained massive research interests in speculation and prediction, [1], [10], [18], [19] and perhaps more so from experienced investors with higher risk tolerance, due to their significantly greater volatility compared to traditional financial assets.

The domain of algorithmic trading is a mature one, and it sure enough includes cryptocurrency. The high volatility introduced by these assets can have very attractive profit potentials for medium- to high-frequency traders, [23] provided their system can learn the patterns well. There have been increasing academic research into deep learning applications for algorithmic trading in financial markets, [10]–[12], [14], [15], [17] and among them, hybrid architectures are getting considerable attention. [18], [19], [23], [25] Empirical results from employing hybrid architectures in these works generally show increased predictive performance compared to their single counterparts.

With this motivation in mind, this paper will experiment with a proposed Convolutional Neural Network - Gated Recurrent Unit (CNN-GRU) hybrid architecture to see if its performance is really better than that of the standalone CNN and GRU models. To this end, I will focus on predicting trading signals for three of the top cryptocurrencies by market capitalisation: Bitcoin (ticker: BTC), Ethereum (ticker: ETH), and Binance Coin (ticker: BNB), using historical price data and several technical indicators. The purpose is twofold: to assess the predictive performance of the proposed hybrid deep learning architecture, and to test said architecture in a simple yet realistic trading setup against the standalone models. The paper attempts to answer the following two questions:

Research Question 1: Does the hybrid CNN-GRU architecture lead to better classification performance than each architecture that is considered alone?

Research Question 2: Does the hybrid CNN-GRU architecture achieve better profitability than each architecture that is considered alone?

The **main contributions** of this paper are:

1) To offer empirical evidence on the use of a new hybrid deep learning architecture for algorithmic trading of cryptocurrencies. To the best of my knowledge, there are no current papers using CNN-GRU for directly classifying buying and selling signals for cryptocurrency data. The closest paper utilising this architecture is one by Kang, Lee & Lim, [25] but it was used for a regression task, i.e. price prediction. While predicting prices is a complex task and offering results on it is valuable, it does not directly translate to trading signals. This paper can provide a valuable asset that retail algorithmic traders can immediately implement in their system.

2) To expand on the limited amount of research on converting financial time-series data into 2-dimensional images to be used in convolutional components of deep learning networks. To date, one of the most prominent frameworks that take this approach is the Convolutional Neural Network with Technical Analysis (CNN-TA) [41] and the Convolutional Neural Network with Bar Images (CNN-BI) frameworks; [52] however, they only deal with the stock market. The cryptocurrency market is highly volatile, which presents a challenge that this paper will attempt to overcome.
3) To contribute an empirical record of a deep learning architecture on a highly imbalanced financial time series, which may be of value to other researchers who want to explore and improve within this space. Successful algorithmic trading works on imbalanced classification include Sezer & Ozbayoglu [41], and Parente, Rizzuti & Trerotola [9].

The paper will be structured as follows: first, related works on the use of traditional machine learning and deep learning architectures in the domain of cryptocurrency trading will be summarised. Next, I will briefly explain the theoretical framework of CNN and GRU, including how they learn to make predictions. The proposed CNN-GRU architecture is then explained in detail, followed by research methodology, experimental results, and some discussions. A conclusion with research limitations and future works clearly outlined will then wrap up the paper.

II. Related Works

A. Cryptocurrency

From the proposal of Bitcoin - the first decentralised currency system in 2008, [2] to it being in the top 10 most valuable assets in the world by market capitalisation (US$1.439 trillion as of March 2024,) [3] the cryptocurrency domain has taken the market by storm. Near-immediate transaction complete time across borders, complex cryptographic encryption enhancing network security, [2] [4] and public distributed ledgers (known as a blockchain) ensuring transparency [4] are some of the most compelling reasons for the increasing trust and adoption of cryptocurrencies. The cryptocurrency market has seen exponential growth in the last decade, in terms of the total number of coins on offer [5] and by raw market capitalisation. [7] It has attracted attention from investors and academics alike.

In recent literature, there are numerous studies on cryptocurrency applications in the trading and investment spaces. The most prominent areas of research include price, volatility, return, trend and/or market movement predictions for algorithmic trading, [15]– [28] portfolio construction and optimisation, [29]– [32] collective behaviour studies and bubble event analysis. [32]– [37]

In traditional financial markets, there are two main types of trading analysis, namely fundamental-based and technical-based. Fundamental analysis is based on the economic mechanics of the underlying assets, [39] while technical analysis is based primarily on chart patterns established in past trading data, such as price and volume, with the assumption that past behaviours inform future behaviours to some extent. [6] [38] This paper extensively focuses on technical-based algorithmic trading of cryptocurrencies, since I can focus on algorithm building and evaluation without needing extensive work in researching underlying fundamental factors for each cryptocurrency, which is beyond the scope of this paper.

B. Deep learning for cryptocurrency trading

Deep learning is a subset of machine learning based on deep neural networks, and the idea is to recognise complex and abstract layers of representations (features) from raw data [40] to reach a decision via automatic learning. There have been growing research interests from academics in using deep learning architectures for cryptocurrency trading, because of their general superiority in predictive power compared to statistical models and traditional machine learning methods. [15] – [28] Within neural network research, there are three popular architecture classes that have been used in the majority of works, namely the Multilayer Perceptron, (MLP) the Recurrent Neural Network, (RNN) and the Convolutional Neural Network. (CNN)

1) Multilayer Perceptron (MLP): The simplest form of deep learning architecture is a plain MLP consisting of only fully connected layers. Here, information from every neuron of the previous layer flows through to every other neuron in the next layer, and there can be one or more such layers, creating a dense network. The connecting edges between neuron layers form weight matrices, and optimising these matrices to improve predictions is part of the learning process for this type of network.

In cryptocurrency trading, MLPs have appeared in multiple research works, most of which used them in a "vanilla" fashion for benchmarking against more sophisticated models. Alonso-Monsalve, Suárez-Cetrulo, Cervantes & Quintana, [23] used MLPs for predicting market movements, which outperformed their radial basis function neural network (RBFNN) in three of the six cryptocurrencies tested. García-Medina & Aguayo-Moreno [20] used the MLP as a simple benchmark for their hybrid Long Short-Term Memory – Generalised Autoregressive Conditional Heteroskedasticity (LSTM-GARCH) model for volatility forcasting of a cryptocurrency portfolio, from which it was surprising that the MLP performed better in terms of the median heteroskedastic absolute error (MHAE) for all time horizons tested.

Akyildirim, Goncu & Sensoy, [27] on the other hand, considered MLP as one on the higher complexity level and compared it with traditional statistical models such as logistic regression (LR), support vector machines (SVM) and random forest (RF), from which it was evident that MLP did not perform as well as SVM and LR. On the contrary, Parente, Rizzuti & Trerotola [9] presented a very simple yet profitable MLP architecture in predicting trading signals, although they only compared it to a dummy model making random predictions. With these mixed results, it suggests that the success of a deep learning model may not entirely depend on its complexity; other aspects of training may also play an important role.

2) Recurrent Neural Networks (RNN): RNN is a variant of neural networks designed to model time-dependent (sequential) data, because of its keeping of a latent state that is updated as input at each time step is sent through the network. Unlike MLP, which cannot directly ingest variable-length data without conforming to the predefined shape of the input layer, [8] RNN can do so because the learning process happens one time step at a time.

In cryptocurrency trading literature, the most popular RNN variants adopted are the long short-term memory (LSTM) and the gated recurrent unit (GRU), which are versions of RNN with gating-controlled mechanisms to deal with vanishing/exploding gradients during training, which will be explained in more detail in Section III of this paper. Jaquart, Köpke & Weinhardt [1] employed LSTM and GRU in their ensemble architecture for market movement predictions, which beat the trivial buy-and-hold strategy in the out-of-sample annualised Sharpe ratio. Mittal & Bhatia [16] utilised LSTM models to predict the highest exchange rate of Bitcoin against the USD for future periods based on previous periods, and obtained a better RMSE than a multivariate linear regression. Similarly, Ammer & Aldhyani [10] used LSTM for price forecasting of four cryptocurrencies, and it was the best-performing in the root mean squared error (RMSE) among models such as LR, RF, and SVM.

3) Convolutional Neural Networks (CNN): CNNs enjoy tremendous success in the field of computer vision and image processing because of its ability to recognise local connectivity in the input data, which is a typical characteristic of images. While the CNN seems like an atypical use case for algorithmic trading, it makes natural sense when compared to a technical trader, because financial time series when plotted as price charts with overlaying indicators are nothing but images that move with time. The idea of a CNN making predictions on a multivariate financial time series is perfectly similar to that of a human trader analysing price charts to make trading decisions, thus it is natural to want to train the former to behave like, if not better than, the latter.

In literature, there are fewer papers using CNNs on their own in the cryptocurrency market, possibly owing to its unpopular use case. Cavalli & Amoretti [11] exploited a 1-dimensional CNN (1dCNN) to process BTC data including price actions, sentiment analysis and blockchain transaction records to make trend predictions, which produced higher-accuracy results compared to an LSTM. Zhang et al. [12] used CNN both as standalone and in composition with other RNNs such as LSTM and GRU for cryptocurrency price forecasting, from which it was shown that CNN outperforms other models in the standalone baseline, and composite models including the CNN improved regression performance. Hasan, Huyam Hasan, Salih Ahmed & Hamid Hasan [13] leveraged a CNN for Bitcoin price forecasting using price and public sentiment data, which achieved better regression metrics than the LSTM and the RNN.

It will be evident in Section II-C that the CNN is much more popularly used in composite architectures with other temporally-aware components such as RNN variants or attention modules, in order to leverage advantages of different network types in making trading decisions.

C. Hybrid deep learning for cryptocurrency trading

Within the deep learning space, hybrid models are getting even more popular due to their general outperformance against standalone architectures.

In regression modelling, Chen, Liao & Zhang [18] combined a CNN with 2 LSTM layers to forecast Bitcoin prices with the help of several macroeconomic indicators and other major coins' closing prices, and reported a mean absolute percentage error (MAPE) of 2.39%.

Liu, Xiao, Chen & Zheng [42] used a hybrid Long Short Term Memory and Gated Recurrent Unit (LSTM-GRU) for price prediction, which achieved better performance than the individual LSTM, and the best among applications utilising similar architectures.

Similarly, Tanwar et al. [17] proposed a parallel model using a LSTM-GRU and an LSTM in a single architecture, each processing a separate price dataset (namely Litecoin and Zcash), then concatenating the learners at the end. Their proposed model produced the lowest Mean Squared Error (MSE) compared to the standalone LSTM and GRU, and on all time windows tested.

Kang, Lee & Lim [25] proposed a 1-dimensional CNN combined with a stacked GRU (1dCNN-GRU) to predict the prices of Bitcoin, Ethereum and Ripple, which led to a decrease of the root mean squared error (RMSE) metric on all datasets compared to using the GRU model on its own. The authors of [25] attributed this improvement to the convolutional layers of the CNN being able to recognise local relationships in the data and turning them into distinct feature maps, which played down the effects of noise.

In the classification space, Alonso-Monsalve, Suárez-Cetrulo, Cervantes & Quintana [23] used a CNN-LSTM model along with technical indicators for classifying price movements of six pairs of cryptocurrencies, which outperformed the standalone CNN, as well as the MLP and the RBFNN.

Ortu, Uras, Conversano, Bartolucci & Destefanis [24] developed a hybrid multivariate attention-based LSTM network and fully convolutional network (MALSTM-FCN) along with several technical indicators for price classification of Bitcoin and Ethereum, achieving a superior accuracy score compared to MLP, LSTM and CNN.

Another advanced attention-based mechanism was used by Peng, Chen, Lin & Wang [19] in their CNN-LSTM hybrid structure to process multiple cryptocurrency datasets at two different timeframes: five minutes and one day. Multiple weight-sharing CNN-LSTM blocks were built in parallel, then connected to two attention layers to separately process frequency and currency aspects of the data. The strategy utilising this architecture outperformed in terms of excess returns compared to that using other deep networks, like MLP, CNN, GRU, and LSTM.

Similar to the parallel approach of [17], Livieris, Kiriakidou, Stavroyiannis & Pintelas [26] used multiple parallel CNN-LSTM architectures, each processing a separate cryptocurrency dataset, then concatenating the learned features at the end using a concatenate and several dense layers. This is compared to the standalone CNN-LSTM, where the authors of [26] reported that the parallel model was producing better classification metrics than the standalone model, while also keeping the best balance between sensitivity and specificity.

In reinforcement learning (RL), a very successful CNN-LSTM architecture was implemented by Peng, Ang & Lim [14] for trading the largest cryptocurrencies by market capitalisation, which beat the buy-and-hold strategy by a wide margin in terms of annualised return during both uptrends and downtrends.

There are vast opportunities for hybrid models to outshine their standalone counterparts, and the proposed model in Section IV-A will attempt to do that.

III. Preliminaries

This section outlines the mechanics behind each component architecture of the proposed model, which are the CNN and the GRU.

A. Convolutional Neural Network

The idea of a network capable of recognising patterns in input data was made popular in the early 1980s, with the proposal of the "neocognitron" model by Kunihiko Fukushima. [43] This was brought into modern application by Yann LeCun and his team of researchers in 1998 [44] for their task in image recognition, when the name of this type of architecture was known to be the CNN. Although this model had been more well-known in the computer vision field, it gathered perhaps a surprising amount of attention in algorithmic trading literature due to its ability to extract complex local patterns in the data and learn from those to make highly accurate predictions, much in the same way that technical-based trading is a game of pattern recognition. This is the reason why the architecture was chosen as one of the key components in the proposed model.

The typical structure of a modern CNN consists of one or more series of alternating convolutional and pooling layers (a pair of convolutional and pooling layer together forms a single "convolutional layer,") followed by one or more fully connected layers. The convolutional and pooling parts of the model are responsible for extracting feature maps from the input data, from which the fully connected parts are typically used for decision purposes. [45] Figure 1 shows an example architecture with 2 convolutional layers, followed by 3 dense layers.

1) Extracting the feature map: The feature maps are extracted using the convolution operation, which is a process whereby a "filter" (or "kernel") slides through the input data, convolving with each mini patch of the data as it does. Figure 2 shows an example 2-dimensional convolution between a 7×7 input matrix $\mathbf{I}$ and a 3×3 kernel matrix $\mathbf{K}$, with a stride of 1 and no zero-padding. As $\mathbf{K}$ slides through each 3×3 patch of $\mathbf{I}$, it elementwise multiplies with the patch, then adds the result to form an entry of the matrix $\mathbf{I} * \mathbf{K}$.

Following the convolution is usually a nonlinear activation, usually the rectified linear unit (ReLU):

$$\mathrm{ReLU}(x_{ij}) = \max(0, x_{ij}) \tag{1}$$

The data then typically goes through a subsampling operation to reduce its spatial dimensionality while retaining the most important information. This is usually done via either a MaxPool or an AveragePool (Equations 2 and 3 respectively; for 2d input):

$$\mathrm{MaxPool}(X)_{i,j} = \max_{a,b}(X_{s_x i+a, s_y j+b}) \tag{2}$$

$$\mathrm{AveragePool}(X)_{i,j} = \frac{1}{f_x f_y} \sum_{a,b} X_{s_x i+a, s_y j+b} \tag{3}$$

where i, j are the indices of the output, s_x, s_y the stride in each dimension, and f_x, f_y the filter size in each dimension.

2) Classification: After feature maps are extracted, they are flattened into a vector to be fed into one or more fully connected (dense) layers. For each layer ℓ except the final layer, the flattened feature $\mathbf{x}$ goes through a linear map with weights $\mathbf{W}_\ell$ and bias $\mathbf{b}_\ell$, then goes through a nonlinear activation ϕ:

$$\mathbf{a}_\ell = \phi\big(\mathbf{W}_\ell^\top \mathbf{x} + \mathbf{b}_\ell\big), \tag{4}$$

where $\mathbf{W}_\ell \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^m$, and $\mathbf{a}_\ell, \mathbf{b}_\ell \in \mathbb{R}^n$. The final layer activation, for this multiclass classification, is a softmax applied to each entry i of the final vector:

$$\mathrm{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{5}$$

B. Gated Recurrent Unit

The GRU, proposed in 2014, [47] is a gating variant of the recurrent neural network (RNN), which is itself a type of network used to model sequence data. Financial time series, which is the type that will be used in this paper, is one such type of data, so it makes sense to use a GRU layer in the proposed architecture.

Traditional RNNs learn by retaining a hidden state vector $\mathbf{h}$ through time. Let $\mathbf{W}_h$, $\mathbf{W}_x$ be shared weight matrices for the hidden state vector $\mathbf{h}$ at time $t-1$ and the input vector $\mathbf{x}$ at time t, respectively, $\mathbf{b}_h$ be the hidden state bias term, $\mathbb{1}$ be an indicator that returns 1 if $t \neq 0$, and ϕ be a nonlinear activation. The hidden state vector's update function for each time step t is given by

$$\mathbf{h}_t = \phi\big(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t + \mathbf{b}_h\big)\mathbb{1}\{t \neq 0\} \tag{6}$$

It is easy to see with Equation 6 that the updates capture dependencies all the way back to the start of time, which can

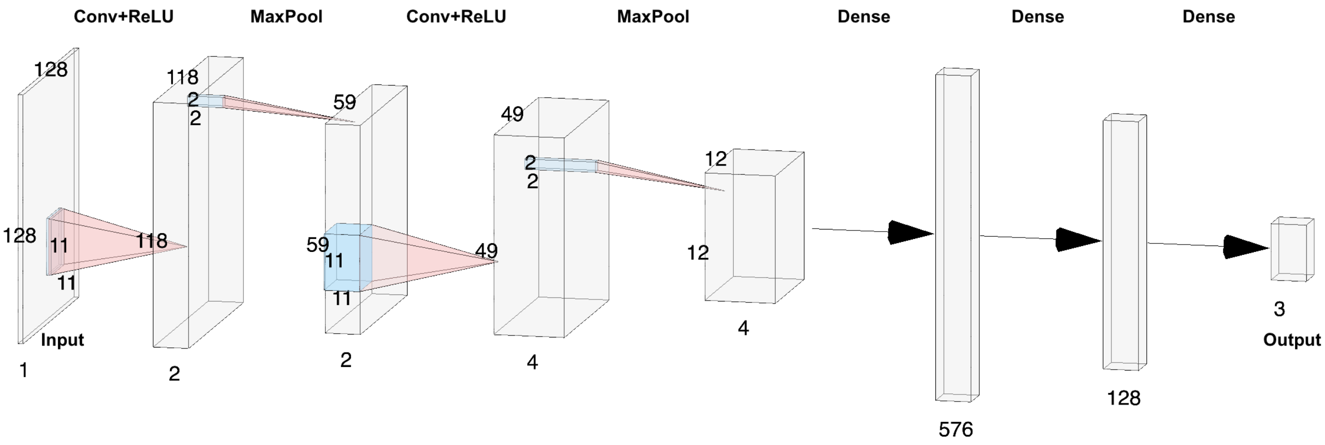
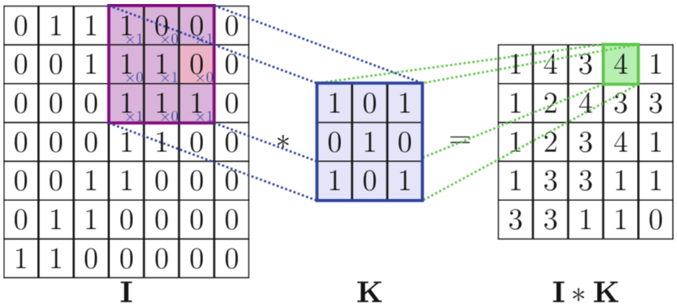


Fig. 1: An example architecture of a CNN. Diagram created on: https://alexlenail.me/NN-SVG/AlexNet.html



$$\underset{\mathbf{I}}{\begin{array}{|c|c|c|c|c|c|c|}\hline 0&1&1&1&0&0&0\\\hline 0&0&1&1&1&0&0\\\hline 0&0&0&1&1&1&0\\\hline 0&0&0&1&1&0&0\\\hline 0&0&1&1&0&0&0\\\hline 0&1&1&0&0&0&0\\\hline 1&1&0&0&0&0&0\\\hline\end{array}} * \underset{\mathbf{K}}{\begin{array}{|c|c|c|}\hline 1&0&1\\\hline 0&1&0\\\hline 1&0&1\\\hline\end{array}} = \underset{\mathbf{I*K}}{\begin{array}{|c|c|c|c|c|}\hline 1&4&3&4&1\\\hline 1&2&4&3&3\\\hline 1&2&3&4&1\\\hline 1&3&3&1&1\\\hline 3&3&1&1&0\\\hline\end{array}}$$

Fig. 2: An example of a 2-dimensional convolution operation. Source: [46]

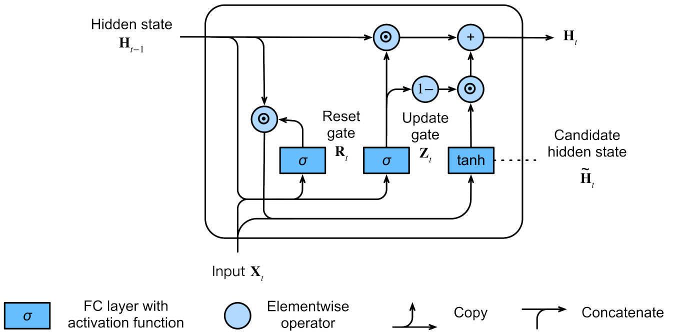


Fig. 3: A single GRU cell. Source: https://d2l.ai/chapter_recurrent-modern/gru.html

cause gradients to either vanish or explode during network training if the sequence is long. [48] The GRU offers a gating mechanism in an attempt to overcome this problem. The idea of GRU is to just keep paying attention to relevant information (update), and to disregard irrelevant information (reset). This simple update-and-reset mechanism allows for more important details to be captured and learned, thus greatly reduces gradient computations for unimportant details.

Figure 3 describes a GRU cell, which represents the inner workings of the network at a single time step t. It still consists of a hidden state $\mathbf{H}$ stored through time, but introduces a new *update gate* $\mathbf{Z}$ and a *reset gate* $\mathbf{R}$. Each of these gates takes as input the data $\mathbf{x}_t$ at time t and the previous hidden state $\mathbf{H}_{t-1}$, learns their associated weight matrices $\mathbf{W}$ and biases $\mathbf{b}$, then sends everything through a nonlinear activation σ, as in Equations 7 and 8:

$$\mathbf{Z}_t = \sigma\big(\mathbf{W}_{h,z}\mathbf{H}_{t-1} + \mathbf{W}_{x,z}\mathbf{x}_t + \mathbf{b}_z\big) \tag{7}$$

$$\mathbf{R}_t = \sigma\big(\mathbf{W}_{h,r}\mathbf{H}_{t-1} + \mathbf{W}_{x,r}\mathbf{x}_t + \mathbf{b}_r\big) \tag{8}$$

$\mathbf{R}_t$ then goes into the calculation of a candidate hidden state $\tilde{\mathbf{H}}_t$, where it decides how much of the previous hidden state to carry to the next time step via a Hadamard product. Another set of weights and biases for the candidate is also learned, as it still takes the data and previous hidden state as input. Everything is sent through the $\tanh$ activation, as follows:

$$\tilde{\mathbf{H}}_t = \tanh\big(\mathbf{W}_{h,h}(\mathbf{R}_t \odot \mathbf{H}_{t-1}) + \mathbf{W}_{x,h}\mathbf{x}_t + \mathbf{b}_h\big) \tag{9}$$

The final hidden state $\mathbf{H}_t$ is a convex combination of the previous hidden state and the candidate hidden state that was just calculated, to decide the ratio between the two in the new state:

$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t \tag{10}$$

C. GRU or LSTM?

It is important to note that GRU is not the first gating alternative to the RNN. An earlier attempt is the LSTM, [49] which is more complicated than the GRU in that it keeps an extra cell state, which stores information over longer periods of time (i.e. the long-term memory) as well as the hidden state (i.e. the short-term memory). It also has more gates than a GRU. Empirical results in recent literature [50] [51] show no signs of a preferred choice of model between the two for sequence modelling tasks. However, in terms of computational times, all literature points to GRU as the winner, since it is simpler than LSTM by design, hence the choice of GRU over LSTM in this paper. Also, the takeaway from Section II is that there are much fewer attempts at using GRU than LSTM as a component of a hybrid convolution-sequence model for algorithmic cryptocurrency trading in the literature, and to the best of my knowledge, no attempt at using GRU has been made in a classification task of that sort. These justify the use of GRU in the upcoming experimentations.

IV. Methodology

A. The proposed architecture: CNN-GRU

In literature, there are two ways to represent the input data for the CNN component: (1) as one-dimensional time series, which uses one-dimensional (1d) convolutional layers to convolve features across the temporal dimension, and (2) as two-dimensional images, which uses two-dimensional (2d) convolutional layers to convolve features across spatial dimensions. The 2d version has been used quite sparingly, yet quite successfully in the literature, for algorithmic trading purposes, [41] [52] [53] and this is possibly due to complexity reasons. However, transforming trading data into 2-dimensional images presents a novel approach to capturing price patterns in financial time series. In this way, the model is in the best position to behave like a human trader: recognising visual patterns presented in past price charts in a spatially-aware manner, to inform future trading decisions. Therefore, this will be the chosen approach for the paper.

It is also worth mentioning that the datasets used in this paper are highly imbalanced, with each of the two minority classes accounting for about 2% of the total number of observations. Works about the effects of imbalanced datasets on deep learning model performance are very limited, however, empirical results from Johnson & Khoshgoftaar, [54] who tested networks that are up to 50 layers deep for a dataset that has minority class representations ranging from 0.01 – 3.6%, show that deeper networks generally perform better than their shallow counterparts in the F1-score. Therefore, the proposed architecture will be comparable in depth to the best simple network in [54].

Figure 4 shows a visual diagram of the proposed CNN-GRU architecture. It contains:

- An input layer of size $L \times I$, where L is the number of time steps to consider in a rolling window, and I is the number of features for each dataset.
- Five convolutional blocks, each with four layers in the order of *Convolution* → *Batch normalisation* → *ReLU* → *Dropout*. Note that for all working layers except fully connected, batch normalisation is used as a means to speed up convergence in very deep networks, as evidently shown by Google researchers Ioffe & Szegedy [55] in their 2015 paper. For the convolutional layers, the filter size is fixed at 3×3, while the number of filters increases linearly for each block in multiples of 16, up to 80 filters in the last convolutional block. The choice of filter size is inspired by the successful CNN-TA and CNN-BI models for 2d image data, as proposed in [41] and [52], because each image will be relatively small. The choice of number of filters through each convolutional block is inspired by the successful work of [23], although their datasets seemed more balance than those in this paper. For each convolutional layer, ReLU is used as a nonlinear activation, and a dropout layer of 25% is added as regularisation against overfitting.
- A double-stacked GRU block of 2 GRU layers, followed by Batch normalisation and ReLU.
- Eight fully connected layers for classification.
- An output layer of 3 neurons, corresponding to 3 classes of the label.

Including the input and output layers, the proposed architecture stands at 34 layers, which is comparable to a "plain" CNN of 34 layers proposed in [54] in terms of depth ("plain" is in the sense that it does not use complex architectures such as residual blocks in deep residual networks (ResNet,) [56] which is one of those tested in [54].)

B. Comparison architectures

The proposed CNN-GRU architecture will be compared against their standalone counterparts, the CNN and the GRU. To enable the fairest comparison, the standalone architectures will be the same as the CNN-GRU, less the complement blocks. That means:

1) CNN: Similar to CNN-GRU, but the stacked GRU block is removed.

2) GRU: Similar to CNN-GRU, but the five convolutional blocks are removed.

The fully connected layers are still retained, because they are for classification purposes.

C. Data acquisition and technical indicators choice

The raw data for this study is retrieved from the Binance API. Binance is one of the largest cryptocurrency exchanges in the world, [60] and it provides a free API for extracting up to minutely trading data for its listed coins. Here, I use hourly trading data of the three major cryptocurrencies: Bitcoin (ticker: BTC), Ethereum (ticker: ETH), and Binance coin (ticker: BNB). The data is extracted from 2017-11-14 10:00:00 UTC to 2024-05-11 04:00:00 UTC. The starting datetime corresponds to the listing datetime for BNB, because it was the last to be listed. This is done so that all coins have the same start date, resulting in the same number of data points for each dataset.

For technical indicators, Table I outlines a set of popular trend, momentum and volume indicators that are used in this study, which reflects the types of technical indicators that have appeared in other algorithmic trading papers. [23] [57] [58]

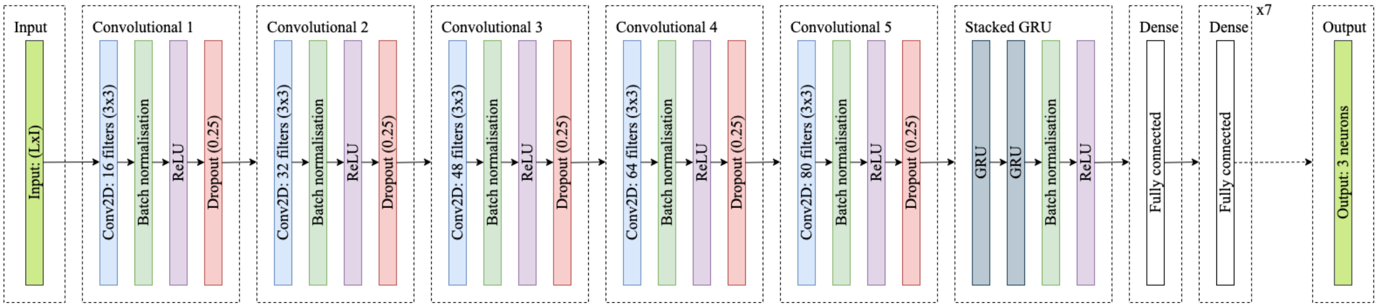


Fig. 4: The proposed CNN-GRU architecture. Diagram created on: https://app.diagrams.net/

[59] While this combination of technical indicators does not follow each of the existing papers exactly, (in fact it is hard to do so because this choice is somewhat subjective, and depends on the expertise and personal preferences of the trader) it ensures that it is consistent with most of them, so that results may be more comparable.

D. Data labelling

Algorithm 1 describes the process of labelling the dataset, which is motivated by the work of Sezer & Ozbayoglu. [41] The idea is to run a small sliding window of a fixed size over the source of calculation (which, in this case, is the "close" prices,) and label each window with the local minimum and maximum, when the middle index of that window coincides with one of its minimum/maximum. If the coincidence is at the minimum, a "Buy" label (encoded 1) is generated, and if it is at the maximum, a "Sell" label (encoded 2) is generated. For all other cases, a "Hold" label (encoded 0) is generated. Because of the keeping of a sliding middle index, the algorithm takes an odd number of window size as input for symmetry. A counter variable starts at the beginning of the loop, and calculations start when the counter indicates that there is enough data to cover the window.

E. Image creation

Figure 5 shows an example of image creation, which will be the final dataset that goes into training and testing. A time window of size L is determined, and an image of size $L \times I$ is created, where I is the total number of features. The label for this image is that corresponding to the last row of the image, which represents a manual trader's routine of looking at past data up to the current time step to make a trading decision. The procedure then increments by one time step, and creates the next image and label, until there is no data left.

F. Data splitting and pre-processing

The image data will be split into training, validation and testing subsets. Since the images are derived from time series, it is important that their temporal order be respected. Therefore, the splitting will be performed as follows:

1) Training set: 2017-11-14 10:00:00+00:00 to 2021-06-30 23:00:00+00:00, which covers the boom-and-bust period of the market in the first half of 2021.

Algorithm 1 Data labelling

```
Require: dataset, windowSize, source="close"
Ensure: windowSize is odd
numberOfRows ← length(dataset)
counter ← 0
dataset['label'] ← empty column
while counter ≤ numberOfRows do
    counter ← (counter + 1)
    if counter ≥ windowSize then
        windowBeginIndex ← counter − windowSize
        windowEndIndex ← windowBeginIndex + windowSize − 1
        windowMidIndex ← (windowBeginIndex + windowEndIndex) / 2
        closePrices ← getIndexOf(dataset[source], from windowBeginIndex to (windowEndIndex+ 1) )
        windowMaxIndex ← getMaxIndexOf(closePrices)
        windowMinIndex ← getMinIndexOf(closePrices)
        if windowMaxIndex = windowMidIndex then
            getIndexOf(dataset, windowMidIndex)["label"] = 2
        else if windowMinIndex = windowMidIndex then
            getIndexOf(dataset, windowMidIndex)["label"] = 1
        else
            getIndexOf(dataset, windowMidIndex)["label"] = 0
        end if
    end if
end while
return dataset
```

2) Validation set: 2021-07-01 00:00:00+00:00 to 2022-12-31 23:00:00+00:00, which covers the boom-and-bust period of the market from October 2021 to December 2022.

3) Testing set: 2023-01-01 00:00:00+00:00 to 2024-05-11 04:00:00+00:00, which covers the cryptocurrency consolidation cycle up to October 2023, then a booming period towards the end of the set.

In addition, $0-1$ normalisation will be performed per-image, because in this way, trends can be detected much better in periods with continuous and sharp increases/decreases, which

TABLE I: Technical Indicators and their Formulae

Indicator	**Type**	**Formula (for a single time step t)**
Exponential Moving Average (EMA)	Trend	$\text{EMA}_t(\mathbf{P}, n) = \frac{2}{n+1}\mathbf{P}_t + (1 - \frac{2}{n+1}) \times \text{EMA}_{t-1}(\mathbf{P}, n)$
Plus Directional Index (DI$^+$)	Trend	$\text{DI}_t^+(n) = 100\frac{\text{SDM}_t^+}{\text{ATR}_t}$
Minus Directional Index (DI$^-$)	Trend	$\text{DI}_t^-(n) = 100\frac{\text{SDM}_t^-}{\text{ATR}_t}$
Average Directional Index (ADX)	Trend	$\text{ADX}_t(n) = \frac{(n-1)\text{ADX}_{t-1} + \text{DX}_t}{n}$
Relative Strength Index (RSI)	Momentum	$\text{RSI}_t(n) = 100 - \frac{100}{1 + \frac{\sum_{i=0}^{n-1}(\text{Up})_{t-i}/n}{\sum_{i=0}^{n-1}(\text{Down})_{t-i}/n}}$
Moving Average Convergence Divergence (MACD)	Momentum	$\text{MACD}_t(n_1, n_2) = \text{EMA}_t(\mathbf{P}, n_1) - \text{EMA}_t(\mathbf{P}, n_2)$, for $n_1 < n_2$
MACD Signal Line (MACDSIG)	Momentum	$\text{MACDSIG}_t(n) = \text{EMA}_t\big(\text{MACD}(n_1, n_2), n\big)$, for $n_1 < n_2$
MACD Histogram (MACDHIST)	Momentum	$\text{MACDHIST}_t = \text{MACD}_t(n_1, n_2) - \text{MACDSIG}_t(n)$
Accumulation/Distribution Oscillator (ADOSC)	Momentum	$\text{ADOSC}_t(n_1, n_2) = \text{EMA}_t(\text{AD}, n_1) - \text{EMA}_t(\text{AD}, n_2)$, for $n_1 < n_2$
Money Flow Index (MFI)	Momentum & Volume	$\text{MFI}_t = 100 - \frac{100}{1+\text{MFR}_t}$
On-balance Volume (OBV)	Volume	$\text{OBV}_t = \text{OBV}_{t-1} + \begin{cases} V_t, & \mathbf{P}_t > \mathbf{P}_{t-1} \\ 0, & \mathbf{P}_t = \mathbf{P}_{t-1} \\ -V_t, & \mathbf{P}_t < \mathbf{P}_{t-1} \end{cases}$ (For first time step: $\text{OBV}_{t=0} = V_{t=0}$)

Notes:

- $\mathbf{P}$: price time series (which is typically the close price)
- C_t, L_t, H_t: close, low, and high prices
- V_t: trading volume
- n, n_1, n_2: number of lookback periods
- SDM$^{+/-}$: The smoothed Plus/Minus Directional Movement (DM$^{+/-}$), where $\text{DM}^+ = H_t - H_{t-1}$, and $\text{DM}^- = L_{t-1} - L_t$. Then for time step t: $\text{SDM}_t^{+/-}(n) = \left(\sum_{i=1}^n \text{DM}_i^{+/-}\right) - \left(\frac{\sum_{i=1}^n \text{DM}_i^{+/-}}{n}\right) + \text{DM}_t^{+/-}$
- ATR: Average True Range, acquired via the following calculations: $\text{TR} = \max(H_t - L_t, H_t - C_{t-1}, L_t - C_{t-1})$; $\text{ATR}_{t=0}(n) = \frac{1}{n}\sum_{t=1}^n \text{TR}_t$; $\text{ATR}_t(n) = \frac{(n-1)\text{ATR}_{t-1} + \text{TR}_t}{n}$ (where TR is the true range)
- DX: Directional Index, $\text{DX}_t = 100\frac{|\text{DI}^+ - \text{DI}^-|}{|\text{DI}^+ + \text{DI}^-|}$
- Up/Down: the upward or downward change in prices, where $\text{Up}_t = P_t/P_{t-1} - 1$, $P_t > P_{t-1}$, and $\text{Down}_t = P_t/P_{t-1} - 1$, $P_t < P_{t-1}$
- AD: Accumulation/Distribution line, acquired via the following calculations: $\text{FM}_t = \frac{(C_t - L_t) - (H_t - C_t)}{H_t - L_t}$; $\text{FV}_t = \text{FM}_t \times V_t$; $\text{AD}_t = \text{AD}_{t-1} + \text{FV}_t$ (where FM_t is the money flow multiplier, FV_t is the money flow volume)
- MFR: Money Flow Ratio, acquired via the following calculations: $\text{TP}_t = \frac{H_t + L_t + C_t}{3}$; $\text{MF}_t = \text{TP}_t \times V_t$; $\text{MFR}_t(n) = \frac{\sum_{i=1}^n (\text{MF}_i) \times \mathbb{1}\{\text{MF}_i \geq 0\}}{\sum_{i=1}^n (\text{MF}_i) \times \mathbb{1}\{\text{MF}_i < 0\}}$ (where TP is the typical price, MF is the money flow, $\mathbb{1}\{\cdot\}$ is an indicator function that returns 1 if conditions are true, otherwise 0)

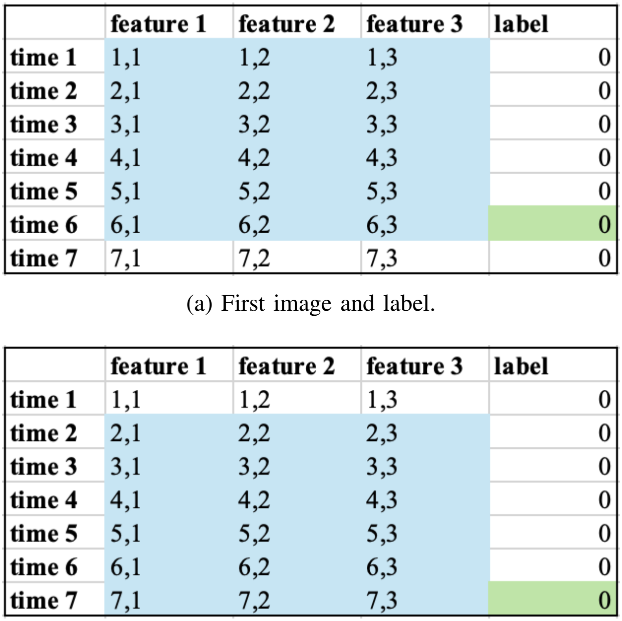
are very typical of the cryptocurrency market.

For the current labelling scheme, the percentage of "Hold" labels is around 96%, whilst each of the "Buy" and "Sell" labels only make up around 2%, which presents a significant class imbalance. Therefore, a resampling is performed on the training set in order to help the classifier learn better. To this end, at each training mini-batch, data will be augmented according to weightings that are inversely proportional to the number of samples in each class of the training set, so that the new subset for each mini-batch is simultaneously over- and under-sampled. This approach reduces overfitting on the minority classes by avoiding the creation of a balanced dataset via oversampling from the start, whilst also reducing the loss of information by merely undersampling the dataset from the start. Figure 6 visualises this sampling scheme.

G. Experiment procedures

The experiments are carried out according to the following protocol:

1) Pre-training settings: Hyperparameters to be shared across all models include learning rate, batch size, and the number of time windows L that will be the height of each training image. Because of computational constraints, the learning rate is fixed at 0.001, and the batch size is fixed at 256. The time window parameter L is experimented with values from 30 to 36 at an increment of 1, with a goal to optimise class recall rates for maximum profitability. It is found that incrementing L from 30 to 33 increases recall for the "Hold" class at a trade-off of lowering recall for the actual "Buy" and "Sell" classes, and incrementing from 34 to 36 lowers recall for all classes. Therefore, the final value of 33 is chosen for L.



	feature 1	feature 2	feature 3	label
time 1	1,1	1,2	1,3	0
time 2	2,1	2,2	2,3	0
time 3	3,1	3,2	3,3	0
time 4	4,1	4,2	4,3	0
time 5	5,1	5,2	5,3	0
time 6	6,1	6,2	6,3	0
time 7	7,1	7,2	7,3	0

(a) First image and label.

	feature 1	feature 2	feature 3	label
time 1	1,1	1,2	1,3	0
time 2	2,1	2,2	2,3	0
time 3	3,1	3,2	3,3	0
time 4	4,1	4,2	4,3	0
time 5	5,1	5,2	5,3	0
time 6	6,1	6,2	6,3	0
time 7	7,1	7,2	7,3	0

(b) Second image and label, incremented by one time step.

Fig. 5: An example of image creation from labelled data. Blue highlights depict the image pixels in an example window of 6 time steps, and green highlights depict the corresponding label for that image.

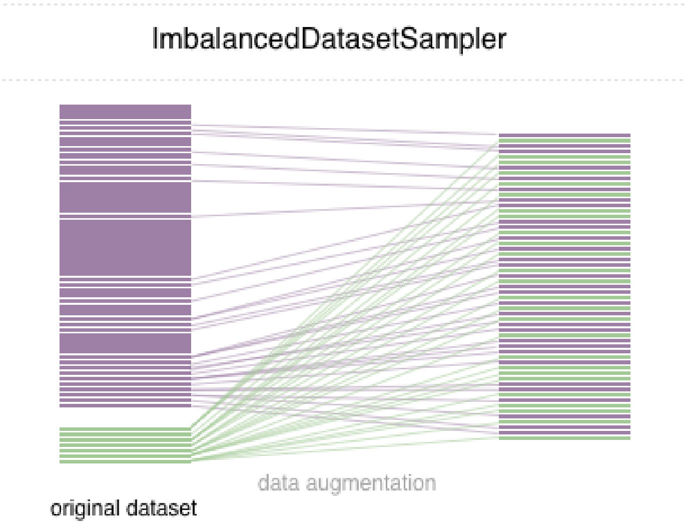


Fig. 6: Data augmentation technique from the Imbalanced-DatasetSampler object, developed using the PyTorch API. Original dataset on the left, augmented dataset on the right. Each augmented dataset represents a mini-batch, and each mini-batch is different due to random sampling. Source: https://github.com/ufoym/imbalanced-dataset-sampler

Because of the volatility of the cryptocurrency market, it is vital that the strategy possess the highest possible recall for the "Hold" class, while keeping recall rates reasonable for the actual decisions. This is because a system with a low "Hold" recall is due to make many trading decisions where it should not, and this is dangerous because it can lose money very quickly due to bad trading decisions, plus transaction fees.

2) Train-test loop: The main train-test loop involves 30 splits corresponding to 30 experiments. For the first split, the train and validation sets are of the same size as set in Section IV-F, but the test set is short by 30 days of data at the end. This essentially creates a sub-window within the whole dataset. Then, for each subsequent split, the whole window slides by 1 day, until the last split when it slides all the way to the end of the dataset. All models will be trained, validated and tested using these 30 splits, and their performance metrics will be recorded for each split. All performance statistics are covered in Section IV-H of this paper.

For each architecture, a maximum of 500 epochs are conducted, with early stopping employed if the validation loss does not improve. A learning rate decay callback is also employed as an extra layer to help stabilise the validation loss before the early stopper activates. For each architecture, the weights from the best epoch in terms of validation loss are saved for out-of-sample testing for that epoch, and its corresponding performance statistics are reported.

3) Statistical significance testing: After 30 train-test experiments are performed for the corresponding 30 splits, a paired t-test will be performed on the out-of-sample test statistics of each pair of models for each dataset to formally answer **Research Question 1** and **Research Question 2** (see Section I). To this end, two one-tailed paired t-tests for each dataset i (called $t\text{-test}_1^{(i)}$ and $t\text{-test}_2^{(i)}$) will be performed, each of which tests whether the data supports the alternative hypothesis that $\mu_{CG}^{(i)}$, the population mean of out-of-sample statistics for the CNN-GRU hybrid architecture for dataset i is higher than $\mu_C^{(i)}$ and $\mu_G^{(i)}$, which are the population means of out-of-sample statistics for the CNN and the GRU for dataset i, respectively. The paired version of the test is used since training and testing examples for each split are taken from subsets of the same dataset, therefore these subsets are not independent.

$$t\text{-test}_1^{(i)} = \begin{cases} \mathrm{H}_0: & \mu_{CG}^{(i)} \leq \mu_C^{(i)} \\ \mathrm{H}_1: & \mu_{CG}^{(i)} > \mu_C^{(i)} \end{cases} \tag{11}$$

$$t\text{-test}_2^{(i)} = \begin{cases} \mathrm{H}_0: & \mu_{CG}^{(i)} \leq \mu_G^{(i)} \\ \mathrm{H}_1: & \mu_{CG}^{(i)} > \mu_G^{(i)} \end{cases} \tag{12}$$

4) Profitability analysis: To answer **Research Question 2**, all models are taken for simulated trading, using the test dataset. The configurations for this analysis are the following:

- The transaction fee is 0.1% of order size per transaction. This is consistent with most exchanges' transaction fee model, and will assess model performance in a more realistic setting.

- For each buy trade, the order size is all of the available capital at the time the trade takes place. This serves to highlight model performance, while transaction fees will be directly deducted from the order total at each trade.
- No stop-losses will be implemented. This is to highlight model performance.
- Slippage will not be modelled, due to unavailability of the order book. This means that all orders are *market orders*, with the assumption that the transaction volume does not exceed the volume limit at that particular price.
- Short orders and margin trading are not allowed. The model will only buy or sell with its available capital, and will not profit during market downtrends.
- Execution is carried out in such a way that the model will not execute a duplicate buy order if it is currently in the market, and it will not execute a sell order if it is currently out of the market.

In practice, this procedure is called *backtesting*, and it is important to note that it only represents the "theoretical" earnings of a model, rather than its realistic earnings. This is due to many factors, including the lack of the order book, as well as tick data (that is, data that reflects changes in price as they happen). Liquidity also dictates model success, since fewer trades can be executed in an illiquid market. Adding these extra components require more extensive engineering and preferrably an event-driven trading system, which is far beyond the scope of this paper.

H. Evaluation metrics

For the train-test loop, the categorical cross entropy (CCE) loss will be the objective function to minimise for all models, per equation 13. Here, n is the number of data points, k is the number of classes, $\mathbb{1}_{ij}$ is an indicator that returns 1 if row i is correctly classified as class j and 0 otherwise, and p_{ij} is the probability of predicting class j for the i^{th} data point.

$$\text{CCE} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k}\mathbb{1}_{ij}\log(p_{ij}) \tag{13}$$

For classification evaluation, precision, recall and $\text{F}^{(\beta)}$-measure will be utilised in this paper, all of which rely on the number of true/false (T/F) positives/negatives (P/N) observed for each class (per Equations 14–16, where n is the total number of data points, y is the ground truth label, $\hat{y}$ is the predicted label, $\mathbb{1}$ is an indicator variable, and k is a single class.) For the individual experiments, per-class metrics and their macro average are reported. The β parameter in the $\text{F}^{(\beta)}$-measure is chosen to be 1, which denotes the popular F1-score.

$$\text{Precision}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FP}_k} \tag{14}$$

$$\text{Recall}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FN}_k} \tag{15}$$

$$\text{F}_k^{(\beta)} = (1+\beta^2)\frac{\text{Precision}_k \times \text{Recall}_k}{\beta^2\text{Precision}_k + \text{Recall}_k} \tag{16}$$

For imbalanced datasets, the accuracy metric is not reliable, because a naïve classifier only predicting the majority class would have achieved a very high yet pointless accuracy score. Thus, the metrics to be assessed in this paper will be the precision, recall, and F1-score, because it is only the actual trading decisions that will be of interest.

For statistical significance testing in the test set, the macro-average version of the F1-score (that is, the equally-weighted average of per-class F1-scores) will be used, for two reasons. First, it contains information about both precision and recall, which are of the same importance in trading decisions. Second, it treats F1-scores for all classes equally, which in the context of imbalanced datasets can mean treating the minority classes more favourably, which is intended.

$$\text{F}^{(\beta)}_{\text{macro avg}} = \frac{1}{k}\sum_{k=1}^{K}\text{F}_k^{(\beta)} \tag{17}$$

For profitability analysis, the annualised rate of return of a model is used. Where T denotes the ending index of the test period, let b_0 and b_T be the account balance at timesteps 0 and T, respectively; and N be the length of the testing period, in years. The annualised rate of return, denoted AR, is given by:

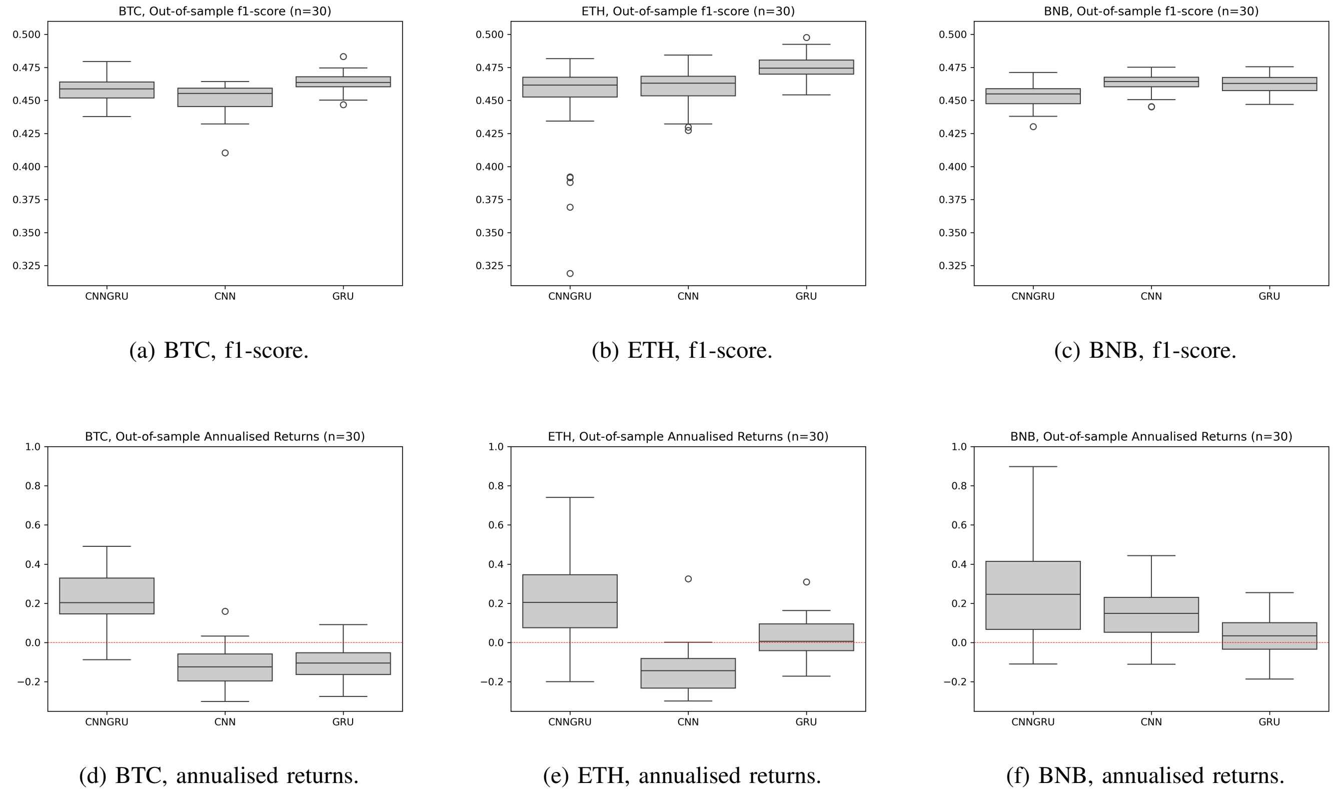
$$\text{AR} = \left(\frac{b_T}{b_0}\right)^{1/N} - 1 \tag{18}$$

V. Experimental Results and Discussion

Table II provides summary statistics of the macro-average precision, recall and f1-score of all models on all datasets, collected over 30 experiments. The full per-class statistics are reported in Table V of the Appendix. Corresponding visuals of the distributions of f1-scores are also given in Figures 7a–7c. Note that the y-axis limits of the f1-scores do not start at 0.0 and end at 1.0, which is the range for this metric. This means that the distributions of these metrics are zoomed in for better interpretation; when zoomed out there are minimal differences that can be seen.

It is observed that the hybrid CNN-GRU generally underperforms their standalone counterparts in the out-of-sample f1-scores. This is attributable to its lower recall rates in the Buy and Sell signals compared to the other models, (see Table V) leading to its lower mean recall overall, as reported in Table II. This means that the standalone models perform better in correctly recalling theoretical trading decisions, while the CNN-GRU only performs better in recalling the Hold decisions.

However, and perhaps surprisingly, this is where the CNN-GRU starts becoming more profitable than the single models, because its higher average "Hold" recall rates across all datasets (see Table V) mean that it is much less likely to make trades where it should not, despite the fact that it catches fewer true opportunities than the single models. Figures 7d–7f further show the distributions of annualised returns of the three models, from which it is visually clear that CNN-GRU is the only model that has its main distribution body wholly



(a) BTC, f1-score.

(b) ETH, f1-score.

(c) BNB, f1-score.

(d) BTC, annualised returns.

(e) ETH, annualised returns.

(f) BNB, annualised returns.

Fig. 7: Out-of-sample f1-scores (7a-7c) and annualised returns (7d-7f) boxplots from 30 experiments on different datasets. For each row corresponding to each metric, the y-axis upper and lower limits are synchronised for easier comparison. For row 2, red dashed line indicates zero return.

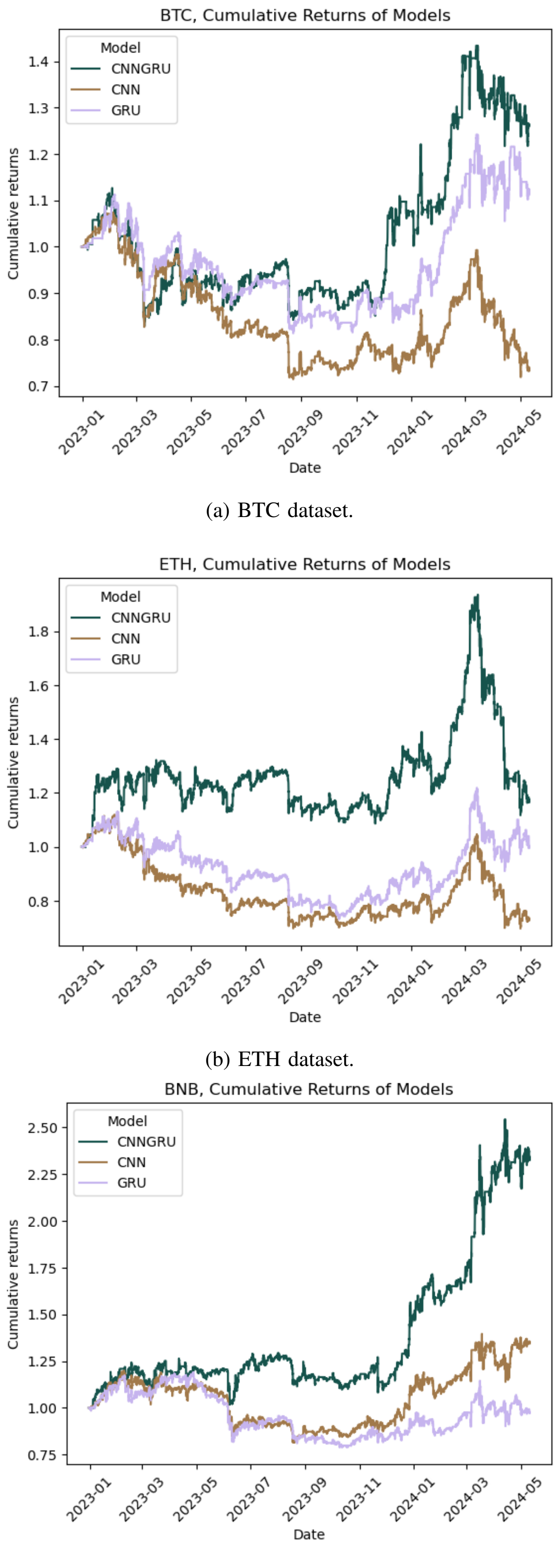
in the positive area (i.e. above the red zero dashed line) in the BTC and ETH datasets, while it achieves a greater mean distribution than the CNN in the BNB dataset. Table III lists the confusion matrices, which explain the numbers better in action. The following observations are made:

- For the BTC and BNB datasets, CNN-GRU avoids more false decisions, and identifies more profitable trades than the single models. These are reflected in the higher number of correctly predicted "Hold" decisions, albeit having a much lower number of correctly predicted "Buy" and "Sell" decisions. Although theoretical buy-sell decisions are of positive values, some are more profitable than others, and CNN-GRU was better at identifying these. The fact that CNN-GRU was able to avoid more false trades means that it keeps more of the profits that it makes, which have a compounding effect for the next trades due to the all-in order size.
- For the ETH dataset, CNN-GRU makes fewer true "Hold" decisions, and the fewest numbers of true trading decisions of all models. However, it is still the most profitable on average, which appears to agree with the insight above that the model is able to identify more profitable trades.

These results prove that for a strategy to be financially successful, it is more important to know when and for how long to hold the position than it is to catch all the "theoretical" ups and downs of the market. Although the single CNN and GRU models exhibit stellar performance in recalling the actual decisions, the hybrid architecture is more profitable on average by being able to avoid more false buy/sell decisions. Figure 8 visualises the normalised cumulative returns for each dataset, where each asset starts at 1.0 representing 100% equity. Although some fluctuations are seen in the beginning of the test period, the CNN-GRU gradually performs better towards the end.

Finally, Table IV shows significance testing results on two suites of tests: out-of-sample f1-scores and annualised returns. At the chosen significance level of $\alpha = 0.05$, in terms of the f1-scores, H_0 is only rejected for CNN-GRU vs. CNN in the BTC dataset, whereas for annualised returns it is strongly rejected on all datasets and all model pairs. Therefore, for these three datasets, CNN-GRU shows strong profitability performance albeit having subpar classification performance.

However, it is vital here to note that this trading performance only considers one market period, which is a mixed regime consisting of a sideways market (that is, prices going up and down within a narrow range) from January to October 2023, then an uptrend and downtrend towards the end of



(a) BTC dataset.

(b) ETH dataset.

(c) BNB dataset.

Fig. 8: Cumulative returns curve of three models on three datasets, tested from 2023-01 to 2024-05. The models with median performance in the experiments are taken for simulated trading.

TABLE II: Performance statistics for each dataset collected over 30 experiments. Reporting macro average on the testing set only. Original statistics are up to 8 decimal points; truncated version rounded up where the fifth decimal point value is 6 or higher. For example, 0.4456**6**270 is rounded up to 0.445**7**. Within a single dataset, bold value indicates the highest mean for that metric.

Model	Macro Average	Mean	Var	Min	Median	Max
		BTC dataset				
CNN-GRU	Precision	**0.4310**	0.0001	0.4091	0.4292	0.4726
	Recall	0.5386	0.0020	0.4427	0.5294	0.6454
	f1-score	0.4585	<0.0001	0.4377	0.4587	0.4794
CNN	Precision	0.4216	<0.0001	0.4062	0.4222	0.4297
	Recall	**0.7228**	0.0025	0.5995	0.7289	0.7946
	f1-score	0.4515	0.0001	0.4105	0.4553	0.4643
GRU	Precision	0.4288	<0.0001	0.4213	0.4288	0.4411
	Recall	0.7127	0.0019	0.6259	0.7060	0.7798
	f1-score	**0.4636**	<0.0001	0.4468	0.4637	0.4832
		ETH dataset				
CNN-GRU	Precision	0.4229	0.0005	0.3655	0.4296	0.4527
	Recall	0.5752	0.0036	0.4758	0.5691	0.7112
	f1-score	0.4474	0.0013	0.3190	0.4617	0.4816
CNN	Precision	0.4274	<0.0001	0.4140	0.4270	0.4435
	Recall	0.7343	0.0023	0.6057	0.7389	0.8260
	f1-score	0.4607	0.0002	0.4274	0.4630	0.4843
GRU	Precision	**0.4349**	<0.0001	0.4245	0.4340	0.4513
	Recall	**0.7411**	0.0017	0.6481	0.7529	0.8099
	f1-score	**0.4753**	<0.0001	0.4543	0.4745	0.4976
		BNB dataset				
CNN-GRU	Precision	0.4254	<0.0001	0.4106	0.4249	0.4493
	Recall	0.5530	0.0030	0.4831	0.5357	0.7524
	f1-score	0.4540	<0.0001	0.4302	0.4549	0.4711
CNN	Precision	**0.4273**	<0.0001	0.4180	0.4274	0.4408
	Recall	0.6732	0.0023	0.5763	0.6706	0.7573
	f1-score	**0.4626**	<0.0001	0.4451	0.4644	0.4752
GRU	Precision	0.4268	<0.0001	0.4187	0.4270	0.4365
	Recall	**0.6904**	0.0014	0.5763	0.6907	0.7549
	f1-score	0.4622	<0.0001	0.4471	0.4628	0.4756

2024. This inherent limitation is covered more in Section VI. Looking at Figure 8 again, it should now be clear why all models' equity curves are mainly fluctuating around the zero-return mark for the first 10-11 months; and performance only looks different afterwards. It is extremely difficult for trend-following strategies, such as the one employed in this paper, to profit from sideways markets. This should be duly noted during deployment, because having the wrong expectations can be financially costly.

TABLE III: Confusion matrices for different datasets and different models on the test set. Statistics are collected from the models achieving median performance in the experiments, and the test set is the same for all models, with start and end dates given in Section IV-F.

			BTC dataset			*ETH dataset*			*BNB dataset*		
			Predicted								
			Hold	Buy	Sell	Hold	Buy	Sell	Hold	Buy	Sell
CNN-GRU	**Actual**	Hold	10693	379	350	8951	1320	1167	10606	410	408
		Buy	168	75	0	103	131	1	150	92	0
		Sell	182	0	61	121	0	114	181	0	61
CNN		Hold	9009	1093	1320	9579	881	978	9908	883	633
		Buy	60	183	0	81	154	0	71	171	0
		Sell	55	0	188	77	0	158	129	0	113
GRU		Hold	10104	520	798	9564	985	889	9749	914	761
		Buy	118	125	0	56	179	0	81	161	0
		Sell	103	0	140	67	0	168	114	0	128

TABLE IV: Statistical significance tests of f1-scores and annualised returns across all models and datasets. All tests are conducted on samples collected over 30 experiments, using a one-tailed paired Student's t-test with 29 degrees of freedom, with significance level $\alpha = 0.05$. Tests are for H_1's that $\mu_{CG}^{(i)} > \mu_{C}^{(i)}$ and $\mu_{CG}^{(i)} > \mu_{G}^{(i)}$ for each dataset i, per Section IV-G3. Tests with bold p-values indicate a successful rejection of H_0 at the specified significance level α.

		BTC dataset		*ETH dataset*		*BNB dataset*	
		t-stat.	p-val.	t-stat.	p-val.	t-stat.	p-val.
Test 1: Out-of-sample f1-scores							
CNN-GRU vs.	CNN	2.4043	**0.0114**	-1.7340	0.9532	-3.9329	0.9998
	GRU	-2.1821	0.9813	-3.9131	0.9997	-3.5014	0.9992
Test 2: Out-of-sample annualised returns							
CNN-GRU vs.	CNN	9.6616	<**0.0001**	7.5320	<**0.0001**	2.5842	**0.0075**
	GRU	9.6314	<**0.0001**	5.1420	<**0.0001**	4.4442	<**0.0001**

VI. Conclusion and Future Work

This study has formally tested the statistical viability of the hybrid CNN-GRU architecture against their standalone counterparts, the CNN and the GRU. It was shown that while there is room for improvement in classification performance, the profitability potential of the hybrid system is worth pursuing. That said, this study has some limitations for both finance and data science aspects, namely:

1) Finance: The study does not consider algorithmic trading in its full glory. While profitability is one of the key success metrics, this study did not wholly focus on designing a profitable trading system. As indicated, profitability analysis can be considered a backtest on historical data, the results of which are merely theoretical. It has not taken into account other aspects of algorithmic trading, including but not limited to risk modelling, liquidity, slippage during live runs, and execution speed, all of which are very important for the success of trading systems. A further limitation is that the backtests were not run on different market conditions; they were only performed on subsets of the dataset that come from the same market regime, which was the general sideways market throughout 2023, then a brief uptrend and correction into 2024. This limits the paper from exploring model performance in other periods where volatility can eat up a large portion of gains, rendering the models unprofitable. It also affects investors' confidence that the model can achieve an all-star performance in any conditions, and thus they may refrain from using it until there is more data.
2) Data science: In classification terms, this paper is not considered successful. The hybrid architecture is hypothesised as being able to capture more nuances in the data, yet underperforms in more aspects than originally expected. This is due to a heavy imbalance in class distributions that are not yet fully resolved, plus a noise-ridden cryptocurrency market, leaving all models struggling to learn the class differences. Also, the complex nature of the networks, limited computing resources, and limited timeline of this paper meant that there were not a lot of room for experimenting with different combinations of hyperparameters, so the winning combination may still be waiting to be explored. A general consensus among researchers generally, and in this paper specifically, is that hybrid networks take longer to train than simple networks, and for obvious reasons. Computational complexity is generally seen to increase with performance, especially in very deep networks, and the complexity-performance tradeoff is inherently unsustainable, even by current standards. [61]

So what does it leave us with? The following is a list for possible future works in this space, the scope of some of which can be worth an entire paper:

1) To explore other advanced methods for tackling class imbalance, such as using Generative Adversarial Networks to oversample the minority class. This is an established technique in the literature, and has been used in various fields such as finance, cybersecurity, and healthcare. [62] The most popular method is minority oversampling, while a hybrid method has also been used in [63], which is empirically shown to be more effective.
2) To experiment with more advanced feature engineering, such as using multiple timeframe granularities in order to leverage lower- and higher-level information for best executions. Currently, this paper explores the hourly timeframe for all datasets (that is, every row of data represents a 1-hour summary of the prices and indicators for that hour). Experienced traders may analyse multiple lower and higher timeframes in parallel, such as 30-

minute and 4-hour, to get a broader view of market behaviours and reap a better trading performance. [65]

3) To test performance in different and preferrably isolated conditions, such as in downtrends or sideways markets only, in order to assess model reliability in these specific markets. This is a crucial step to take, since established trading firms may otherwise refrain from employing models they do not fully understand.
4) To employ explainable AI practices, such as using SHapley Additive exPlanations (SHAP) values (as computational resources permit) [64] to see which features contribute the most and least to signal predictability. This contributes further to feature selection, which can reduce computations and focuses on the most important indicators. The explainability of black-box algorithms such as neural networks has been left in the dark for the majority of research papers encountered thus far, so SHAP presents a solution to open up insights. However, SHAP is computationally intensive even for traditional machine learning algorithms, thus will take time and hardware resources.

These future works hopefully bring us back to the promising line of results that other researchers have published about hybrid architectures, in that they outperform standalone models in classification performance.

Acknowledgement

I would like to express my sincerest thanks to Dr. Ali Anaissi, Transdisciplinary School, the University of Technology Sydney, for his invaluable inputs and advice for me to be able to complete this research project.

Appendix

Full summary statistics of per-class precision, recall, and F1-scores for all models and all datasets

Table V lists per-class performance statistics for all models on all datasets. Unweighted averaging of each row of Table V along the class dimension yields Table II.

References

[1] P. Jaquart, S. Köpke, and C. Weinhardt, "Machine learning for cryptocurrency market prediction and trading," *The Journal of Finance and Data Science*, vol. 8, pp. 331–352, Nov. 2022, doi: https://doi.org/10.1016/j.jfds.2022.12.001.

[2] S. Nakamoto, "Bitcoin: a Peer-to-Peer Electronic Cash System," Oct. 2008. Available: https://bitcoin.org/bitcoin.pdf

[3] "Assets ranked by Market Cap - CompaniesMarketCap.com," *companiesmarketcap.com*. https://companiesmarketcap.com/assets-by-market-cap/

[4] A. Gorkhali, L. Li, and A. Shrestha, "Blockchain: a literature review," *Journal of Management Analytics*, vol. 7, no. 3, pp. 321–343, Jul. 2020, doi: https://doi.org/10.1080/23270012.2020.1801529.

[5] J. Zhang, K. Cai, and J. Wen, "A Survey of Deep Learning Applications in Cryptocurrency," *iScience*, vol. 27, no. 1, pp. 108509–108509, Jan. 2024, doi: https://doi.org/10.1016/j.isci.2023.108509.

[6] R. Gençay, "The predictability of security returns with simple technical trading rules," *Journal of Empirical Finance*, vol. 5, no. 4, pp. 347–359, Oct. 1998, doi: https://doi.org/10.1016/s0927-5398(97)00022-4.

[7] "Crypto Market Cap Charts," *CoinGecko*. https://www.coingecko.com/en/global-charts

[8] J. Sakuma, T. Komatsu, and R. Scheibler, "MLP-based architecture with variable length input for automatic speech recognition," *openreview.net*, Jan. 2022, Accessed: Apr. 06, 2024. [Online]. Available: https://openreview.net/forum?id=RA-zVvZLYIy

[9] M. Parente, L. Rizzuti, and M. Trerotola, "A profitable trading algorithm for cryptocurrencies using a Neural Network model," *Expert Systems with Applications*, vol. 238, pp. 121806–121806, Mar. 2024, doi: https://doi.org/10.1016/j.eswa.2023.121806.

[10] M. A. Ammer and T. H. H. Aldhyani, "Deep Learning Algorithm to Predict Cryptocurrency Fluctuation Prices: Increasing Investment Awareness," *Electronics*, vol. 11, no. 15, p. 2349, Jul. 2022, doi: https://doi.org/10.3390/electronics11152349.

[11] S. Cavalli and M. Amoretti, "CNN-based multivariate data analysis for bitcoin trend prediction," *Applied Soft Computing*, p. 107065, Dec. 2020, doi: https://doi.org/10.1016/j.asoc.2020.107065.

[12] Z. Zhang, H.-N. Dai, J. Zhou, S. K. Mondal, M. M. García, and H. Wang, "Forecasting cryptocurrency price using convolutional neural networks with weighted and attentive memory channels," *Expert Systems with Applications*, vol. 183, p. 115378, Nov. 2021, doi: https://doi.org/10.1016/j.eswa.2021.115378.

[13] S. H. Hasan, S. Huyam Hasan, M. Salih Ahmed and S. Hamid Hasan, "A Novel Cryptocurrency Prediction Method Using Optimum CNN," *Computers, Materials & Continua*, vol. 71, no. 1, pp. 1051–1063, 2022, doi: https://doi.org/10.32604/cmc.2022.020823.

[14] A. Peng, S. L. Ang, and C. Y. Lim, "Automated Cryptocurrency Trading Bot Implementing DRL," *Pertanika Journal of Science and Technology*, vol. 30, no. 4, pp. 2683–2705, Sep. 2022, doi: https://doi.org/10.47836/pjst.30.4.22.

[15] R. Miura, Lukáš Pichl, and Taisei Kaizoji, "Artificial Neural Networks for Realized Volatility Prediction in Cryptocurrency Time Series," *Lecture Notes in Computer Science*, pp. 165–172, Jul. 2019, doi: https://doi.org/10.1007/978-3-030-22796-8_18.

[16] R. Mittal and M. P. S. Bhatia, "Incorporating Deep Learning Techniques for Predicting the Prices and Identifying Most Influential Cryptocurrencies," *Social Science Research Network*, Apr. 03, 2018. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3364971 (accessed Mar. 15, 2024).

[17] S. Tanwar, N. P. Patel, S. N. Patel, J. R. Patel, G. Sharma, and I. E. Davidson, "Deep Learning-Based Cryptocurrency Price Prediction Scheme With Inter-Dependent Relations," *IEEE Access*, vol. 9, pp. 138633–138646, 2021, doi: https://doi.org/10.1109/access.2021.3117848.

[18] S. Chen, Z. Liao, and J. Zhang, "Forecasting the Price of Bitcoin Using an Explainable CNN-LSTM Model," Communications in computer and information science, pp. 93–101, Jan. 2024, doi: https://doi.org/10.1007/978-981-97-0065-3_7.

[19] P. Peng, Y. Chen, W. Lin, and J. Wang, "Attention-based CNN–LSTM for high-frequency multiple cryptocurrency trend prediction," *Expert Systems with Applications*, vol. 237, pp. 121520–121520, Mar. 2024, doi: https://doi.org/10.1016/j.eswa.2023.121520.

[20] A. García-Medina and E. Aguayo-Moreno, "LSTM–GARCH Hybrid Model for the Prediction of Volatility in Cryptocurrency Portfolios," *Computational Economics*, Mar. 2023, doi: https://doi.org/10.1007/s10614-023-10373-8.

[21] Y. Yi, M. He, and Y. Zhang, "Out-of-sample prediction of Bitcoin realized volatility: Do other cryptocurrencies help?," *The North American Journal of Economics and Finance*, vol. 62, p. 101731, Nov. 2022, doi: https://doi.org/10.1016/j.najef.2022.101731.

[22] V. D'Amato, S. Levantesi, and G. Piscopo, "Deep learning in predicting cryptocurrency volatility," *Physica A: Statistical Mechanics and its Applications*, vol. 596, p. 127158, Jun. 2022, doi: https://doi.org/10.1016/j.physa.2022.127158.

[23] S. Alonso-Monsalve, A. L. Suárez-Cetrulo, A. Cervantes, and D. Quintana, "Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators," *Expert Systems with Applications*, vol. 149, p. 113250, Jul. 2020, doi: https://doi.org/10.1016/j.eswa.2020.113250.

[24] M. Ortu, N. Uras, C. Conversano, S. Bartolucci, and G. Destefanis, "On technical trading and social media indicators for cryptocurrency price classification through deep learning," *Expert Systems with Applications*, vol. 198, p. 116804, Jul. 2022, doi: https://doi.org/10.1016/j.eswa.2022.116804.

[25] C. Y. Kang, C. P. Lee, and K. M. Lim, "Cryptocurrency Price Prediction with Convolutional Neural Network and Stacked Gated

TABLE V: Per-class out-of-sample performance statistics of all models on all datasets. Unweighted averaging of each row along the class dimension yields Table II. Results collected over 30 experiments. Original statistics are up to 8 decimal points; truncated version rounded up where the fifth decimal point value is 6 or higher. For example, 0.4456**6**270 is rounded up to 0.445**7**. Within a single class of a single dataset, bold value indicates the highest mean for that metric.

Model	**Macro Average**	**Mean**			**Var**			**Min**			**Median**			**Max**		
Class:		*0*	*1*	*2*	*0*	*1*	*2*	*0*	*1*	*2*	*0*	*1*	*2*	*0*	*1*	*2*
BTC dataset																
CNN-GRU	Precision	0.9708	**0.1729**	**0.1492**	<0.0001	<0.0001	<0.0001	0.9654	0.1432	0.1117	0.9703	0.1704	0.1452	0.9777	0.2356	0.2169
	Recall	**0.9209**	0.3730	0.3219	0.0005	0.0065	0.0077	0.8594	0.1965	0.1572	0.9274	0.3529	0.3069	0.9744	0.5261	0.5641
	f1-score	**0.9450**	0.2320	0.1986	0.0001	0.0002	0.0004	0.9148	0.2018	0.1419	0.9486	0.2299	0.2016	0.9699	0.2550	0.2383
CNN	Precision	**0.9836**	0.1488	0.1323	<0.0001	<0.0001	<0.0001	0.9747	0.1093	0.0913	0.9838	0.1488	0.1312	0.9894	0.1834	0.154
	Recall	0.8198	**0.6771**	0.6716	0.0014	0.0134	0.0112	0.7250	0.4323	0.4348	0.8224	0.6941	0.6960	0.8940	0.8755	0.9210
	f1-score	0.8937	0.2416	0.2193	0.0004	0.0002	0.0002	0.8359	0.1942	0.1661	0.8957	0.2439	0.2206	0.9326	0.2716	0.2400
GRU	Precision	0.9827	0.1670	0.1369	<0.0001	<0.0001	<0.0001	0.9763	0.1357	0.1144	0.9820	0.1672	0.1391	0.9880	0.2210	0.1496
	Recall	0.8402	0.6122	**0.6858**	0.0006	0.0130	0.0101	0.7958	0.2458	0.5165	0.8431	0.6064	0.6753	0.8848	0.8382	0.8559
	f1-score	0.9056	**0.2583**	**0.2270**	0.0002	0.0001	0.0001	0.8811	0.2327	0.2006	0.9075	0.2605	0.2297	0.9294	0.2828	0.2458
ETH dataset																
CNN-GRU	Precision	0.9740	0.1550	0.1397	<0.0001	<0.0001	<0.0001	0.9682	0.0574	0.0631	0.9733	0.1637	0.1434	0.9829	0.2169	0.1857
	Recall	**0.8830**	0.4437	0.3990	0.0058	0.0146	0.0159	0.5923	0.1552	0.2009	0.9081	0.4353	0.4062	0.9575	0.6667	0.7543
	f1-score	**0.9243**	0.2194	0.1986	0.0021	0.0017	0.0011	0.7373	0.1055	0.1144	0.9401	0.2334	0.2059	0.9628	0.2622	0.2468
CNN	Precision	0.9849	0.1598	0.1374	<0.0001	0.0003	0.0003	0.9760	0.1179	0.0846	0.9849	0.1573	0.1409	0.9926	0.2064	0.1709
	Recall	0.8295	0.6646	**0.7087**	0.0015	0.0086	0.0181	0.7460	0.4229	0.3587	0.8310	0.6726	0.7169	0.9135	0.8991	0.9912
	f1-score	0.9000	0.2554	0.2268	0.0005	0.0002	0.0004	0.8487	0.2084	0.1560	0.9019	0.2526	0.2310	0.9437	0.2823	0.2529
GRU	Precision	**0.9851**	**0.1646**	**0.1551**	<0.0001	0.0002	0.0001	0.9786	0.1381	0.1252	0.9860	0.1621	0.1542	0.9905	0.1953	0.1801
	Recall	0.8482	**0.6991**	0.6760	0.0006	0.0101	0.0093	0.7988	0.5088	0.3612	0.8469	0.7185	0.6746	0.9075	0.8546	0.8565
	f1-score	0.9113	**0.2643**	**0.2504**	0.0002	0.0002	0.0002	0.8838	0.2373	0.2184	0.9111	0.2637	0.2523	0.9417	0.2929	0.2701
BNB dataset																
CNN-GRU	Precision	0.9718	**0.1685**	0.1358	<0.0001	0.0002	0.0004	0.9675	0.1302	0.1051	0.9706	0.1720	0.1305	0.9860	0.1880	0.1913
	Recall	**0.9078**	0.4174	0.3338	0.0011	0.0128	0.0107	0.7776	0.2678	0.2189	0.9182	0.3812	0.3017	0.9490	0.8333	0.6463
	f1-score	**0.9383**	0.2359	0.1878	0.0003	0.0002	0.0003	0.8695	0.1847	0.1509	0.9436	0.2380	0.1897	0.9588	0.2709	0.2214
CNN	Precision	0.9798	0.1601	0.1421	<0.0001	0.0001	0.0002	0.9732	0.1199	0.1240	0.9794	0.1612	0.1387	0.9861	0.1849	0.1845
	Recall	0.8586	0.6332	0.5279	0.0008	0.0129	0.0143	0.7983	0.3431	0.2193	0.8614	0.6508	0.5202	0.9111	0.8458	0.7362
	f1-score	0.9149	**0.2530**	0.2201	0.0002	0.0002	0.0001	0.8823	0.2100	0.2004	0.9168	0.2568	0.2192	0.9415	0.2735	0.2443
GRU	Precision	**0.9810**	0.1552	**0.1441**	<0.0001	0.0001	0.0002	0.9731	0.1327	0.1233	0.9809	0.1569	0.1433	0.9859	0.1732	0.1845
	Recall	0.8497	**0.6333**	**0.5882**	0.0006	0.0062	0.0131	0.8043	0.5212	0.2961	0.8517	0.6360	0.6115	0.9069	0.8250	0.7718
	f1-score	0.9104	0.2480	**0.2282**	0.0002	0.0001	0.0001	0.8856	0.2217	0.1952	0.9117	0.2486	0.2293	0.9388	0.2672	0.2446

Note: **Class 0**: Hold; **Class 1**: Buy; **Class 2**: Sell.

Recurrent Unit," *Data*, vol. 7, no. 11, p. 149, Oct. 2022, doi: https://doi.org/10.3390/data7110149.

[26] I. E. Livieris, N. Kiriakidou, S. Stavroyiannis, and P. Pintelas, "An Advanced CNN-LSTM Model for Cryptocurrency Forecasting," *Electronics*, vol. 10, no. 3, p. 287, Jan. 2021, doi: https://doi.org/10.3390/electronics10030287.

[27] E. Akyildirim, A. Goncu, and A. Sensoy, "Prediction of cryptocurrency returns using machine learning," *Annals of Operations Research*, vol. 297, pp. 3–36, Apr. 2020, doi: https://doi.org/10.1007/s10479-020-03575-y.

[28] S.-H. Sung, J.-M. Kim, B.-K. Park, and S. Kim, "A Study on Cryptocurrency Log-Return Price Prediction Using Multivariate Time-Series Model," *Axioms*, vol. 11, no. 9, p. 448, Sep. 2022, doi: https://doi.org/10.3390/axioms11090448.

[29] S. Hashemkhani Zolfani, H. Mehtari Taheri, M. Gharehgozlou, and A. Farahani, "An asymmetric PROMETHEE II for cryptocurrency portfolio allocation based on return prediction," *Applied Soft Computing*, vol. 131, p. 109829, Dec. 2022, doi: https://doi.org/10.1016/j.asoc.2022.109829.

[30] T. Cui, S. Ding, H. Jin, and Y. Zhang, "Portfolio constructions in cryptocurrency market: A CVaR-based deep reinforcement learning approach," *Economic Modelling*, p. 106078, Oct. 2022, doi: https://doi.org/10.1016/j.econmod.2022.106078.

[31] R. Ren, M. Althof, and W. K. Hardle, "Financial Risk Meter for Cryptocurrencies and Tail-Risk Network Based Portfolio Construction," *The Singapore Economic Review*, Jun. 2022, doi: https://doi.org/10.1142/s0217590822480010.

[32] N. James and M. Menzies, "Collective Dynamics, Diversification and Optimal Portfolio Construction for Cryptocurrencies," *Entropy*, vol. 25,

no. 6, p. 931, Jun. 2023, doi: https://doi.org/10.3390/e25060931.

[33] D. Stosic, D. Stosic, T. B. Ludermir, and T. Stosic, “Collective behavior of cryptocurrency price changes,” *Physica A: Statistical Mechanics and its Applications*, vol. 507, pp. 499–509, Oct. 2018, doi: https://doi.org/10.1016/j.physa.2018.05.050.

[34] L. Yarovaya, R. Matkovskyy, and A. Jalan, “The effects of a ‘black swan’ event (COVID-19) on herding behavior in cryptocurrency markets,” *Journal of International Financial Markets, Institutions and Money*, vol. 75, p. 101321, Mar. 2021, doi: https://doi.org/10.1016/j.intfin.2021.101321.

[35] Z. Liu and R. Zhang, “An Empirical Study on Herd Behavior in Cryptocurrency Trading,” *Journal of Computer Information Systems*, pp. 1–15, Jun. 2023, doi: https://doi.org/10.1080/08874417.2023.2223175.

[36] M. S. R. Chowdhury, D. S. Damianov, and A. H. Elsayed, “Bubbles and crashes in cryptocurrencies: Interdependence, contagion, or asset rotation?,” Finance Research Letters, p. 102494, Oct. 2021, doi: https://doi.org/10.1016/j.frl.2021.102494.

[37] S. Zhang, D. Zhang, J. Zheng, W. Aerts, and D. Xu, “Plus Token and investor searching behaviour – A cryptocurrency Ponzi scheme,” *Accounting & Finance*, vol. 63, no. 4, pp. 4713-4728. Jul. 2023, doi: https://doi.org/10.1111/acfi.13128.

[38] F. Fang et al., “Cryptocurrency trading: a comprehensive survey,” *Financial Innovation*, vol. 8, no. 1, pp. 1–59, Feb. 2022, doi: https://doi.org/10.1186/s40854-021-00321-6.

[39] M. Wei, I. Kyriakou, G. Sermpinis, and C. Stasinakis, “Cryptocurrencies and Lucky Factors: The value of technical and fundamental analysis,” *International Journal of Finance & Economics*, Jul. 2023, doi: https://doi.org/10.1002/ijfe.2863.

[40] Y. LeCun, Y. Bengio, and G. Hinton, “Deep Learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: https://doi.org/10.1038/nature14539.

[41] O. B. Sezer and A. M. Ozbayoglu, “Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach,” *Applied Soft Computing*, vol. 70, pp. 525–538, Sep. 2018, doi: https://doi.org/10.1016/j.asoc.2018.04.024.

[42] Y. Liu, G. Xiao, W. Chen, and Z. Zheng, “A LSTM and GRU-Based Hybrid Model in the Cryptocurrency Price Prediction,” *Communications in computer and information science*, pp. 32–43, Nov. 2023, doi: https://doi.org/10.1007/978-981-99-8104-5_3.

[43] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, Apr. 1980, doi: https://doi.org/10.1007/bf00344251.

[44] Y. Lecun, L Eon Bottou, Y. Bengio, and Patrick Haaner Abstract—, “Gradient-Based Learning Applied to Document Recognition,” *PROC. OF THE IEEE*, 1998, Available: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

[45] C.-C. . J. Kuo, “Understanding convolutional neural networks with a mathematical model,” *Journal of Visual Communication and Image Representation*, vol. 41, pp. 406–413, Nov. 2016, doi: https://doi.org/10.1016/j.jvcir.2016.11.003.

[46] M. Vakalopoulou, Stergios Christodoulidis, N. Burgos, Olivier Colliot, and V. Lepetit, “Deep Learning: Basics and Convolutional Neural Networks (CNNs),” Neuromethods, pp. 77–115, Jan. 2023, doi: https://doi.org/10.1007/978-1-0716-3195-9_3.

[47] K. Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio, “On the Properties of Neural Machine Translation: Encoder-Decoder Approaches,” *arXiv (Cornell University)*, Sep. 2014, doi: https://doi.org/10.48550/arxiv.1409.1259.

[48] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994, doi: https://doi.org/10.1109/72.279181.

[49] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[50] N. Gruber and A. Jockisch, “Are GRU Cells More Specific and LSTM Cells More Sensitive in Motive Classification of Text?,” *Frontiers in Artificial Intelligence*, vol. 3, Jun. 2020, doi: https://doi.org/10.3389/frai.2020.00040.

[51] F. M. Shiri, T. Perumal, N. Mustapha, and R. Mohamed, “A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU,” *arXiv.org*, Jun. 01, 2023. https://arxiv.org/abs/2305.17473

[52] O. Sezer and A. Ozbayoglu, “Financial Trading Model with Stock Bar Chart Image Time Series with Deep Convolutional Neural Networks,” *Intelligent Automation and Soft Computing*, p. -1–1, 2018, doi: https://doi.org/10.48550/arXiv.1903.04610.

[53] J.-H. Chen and Y.-C. Tsai, “Encoding candlesticks as images for pattern classification using convolutional neural networks,” *Financial Innovation*, vol. 6, no. 1, Jun. 2020, doi: https://doi.org/10.1186/s40854-020-00187-0.

[54] J. M. Johnson and T. M. Khoshgoftaar, “Survey on deep learning with class imbalance,” Journal of Big Data, vol. 6, no. 1, Mar. 2019, doi: https://doi.org/10.1186/s40537-019-0192-5.

[55] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *arXiv.org*, 2015. https://arxiv.org/abs/1502.03167

[56] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *arXiv (Cornell University)*, Dec. 2015, doi: https://doi.org/10.48550/arxiv.1512.03385.

[57] Y. Kara, M. A. Boyacioglu, and Ö. K. Baykan, “Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange,” *Expert Systems with Applications*, vol. 38, no. 5, pp. 5311–5319, May 2011, doi: https://doi.org/10.1016/j.eswa.2010.10.027.

[58] E. Hoseinzade and S. Haratizadeh, “CNNpred: CNN-based stock market prediction using a diverse set of variables,” *Expert Systems with Applications*, vol. 129, pp. 273–285, Sep. 2019, doi: https://doi.org/10.1016/j.eswa.2019.03.029.

[59] M. U. Gudelek, S. A. Boluk, and A. M. Ozbayoglu, “A deep learning based stock trading model with 2-D CNN trend detection,” *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, Nov. 2017, doi: https://doi.org/10.1109/ssci.2017.8285188.

[60] “Binance: Buy/Sell Bitcoin, Ether and Altcoins — Cryptocurrency Exchange,” *Binance*. https://binance.com/.

[61] N. Thompson, K. Greenewald, K. Lee, and G. Manso, “THE COMPUTATIONAL LIMITS OF DEEP LEARNING,” 2020. Available: https://ide.mit.edu/wp-content/uploads/2020/09/RBN.Thompson.pdf

[62] R. Sauber-Cole and T. M. Khoshgoftaar, “The use of generative adversarial networks to alleviate class imbalance in tabular data: a survey,” *Journal of Big Data*, vol. 9, no. 1, Aug. 2022, doi: https://doi.org/10.1186/s40537-022-00648-6.

[63] B. Zhu, X. Pan, S. vanden Broucke, and J. Xiao, “A GAN-based hybrid sampling method for imbalanced customer classification,” *Information Sciences*, Jul. 2022, doi: https://doi.org/10.1016/j.ins.2022.07.145.

[64] S. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” arXiv.org, Nov. 24, 2017. https://arxiv.org/abs/1705.07874v2

[65] S. Deng and A. Sakurai, “Foreign Exchange Trading Rules Using a Single Technical Indicator from Multiple Timeframes,” 2013 27th International Conference on Advanced Information Networking and Applications Workshops, Mar. 2013, doi: https://doi.org/10.1109/waina.2013.7.