Addressing the Energy Efficiency of Deep Learning Algorithms in Finance

by

Tom Maxwell Potter

August 21, 2022

A dissertation submitted in part fulfilment
of the requirements for the degree of

Master of Science

of

University College London

Scientific and Data Intensive Computing
Department of Physics and Astronomy

Declaration

I, Tom Maxwell Potter, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the dissertation.

Abstract

This thesis investigates the efficiency of deep learning training algorithms, exploring how deep neural networks can be trained in a way that minimises energy consumption and training data requirements. For the first time, these energy and data-efficient training algorithms will be applied to financial modelling, demonstrating how the computational cost of deep learning in finance can be reduced. The financial sector has long been associated with negative environmental, social, and governance (ESG) impacts, including being a major contributor to global carbon emissions. Despite the attempts by some to prioritise *sustainable finance*, the recent expansion of financial technology—incorporating new, expensive methods such as *deep learning* (DL)—has only worsened the energy consumption attributed to this industry, accelerating its carbon emissions. In an attempt to address these negative impacts, this project aims to develop an energy-efficient DL system for financial modelling. This research will explore *Green AI* methods that attempt to reduce the energy expended training DL models and apply these for the first time to models used in finance. To exemplify the benefits of these methods, a performant financial volatility model will be developed that not only produces accurate results but prioritises generating this performance in an efficient manner, minimising the energy and data resources required during training. This system aims to demonstrate that the principles of Green AI are applicable within the financial sector, furthering the scope of sustainable finance by improving the sustainability of deep learning for finance and, hence, minimising the ESG impacts of the financial sector.

Outline of research:

- Chapter 3: *Methodology*. This research contains three core studies.

  - Section 3.4: *Baseline finanical volatility model*. This research will commence with an analysis of the resource requirements of typical systems in the field of deep learning for finance. A particular focus will be given to the domain of financial volatility modelling, as this is a major application of deep learning in finance, and the *long short-term memory* (LSTM) networks typically exploited for such tasks. This analysis aims to act as an exemplar of the computational cost of typical deep learning algorithms used in finance.

  - Section 3.5: *Energy-efficient training extensions*. Several adaptations to the model training process are made that prioritise reducing the energy consumed by the system (quantified as total training time) whilst mainaining an accurately performing model. Specifically, *progressive training*, and *mixed-precision* will be explored, proving the feasibility of efficient training algorithms within the chosen domain.

  - Section 3.6: *Data-efficient training extensions*. Additional adaptations are made that

reduce the data requirements of training, further lowering resource requirements and mitigating the computational costs of data storage and processing at data centres.

- Chapter 4: *Results & discussion*. An analysis is made between the baseline and extended models, comparing their performance and efficiency, and discussing the overall success of the application of Green AI to finance.

Contributions to science:

1. *Expanding the applications of Green AI.* The first application of Green AI to the finance sector, further demonstrating the utility and importance of Green AI in lowering the environmental cost of deep learning.

2. *Reducing the environmental impact of financial technology.* The improvement of sustainable finance to include the new research domain of *sustainable deep learning for sustainable finance.*

3. *Improving the inclusivity of finance.* Lowering the bar-to-entry to engage in deep learning for finance, allowing more individuals to leverage financial technology and analytics.

Keywords: Green AI, Green Deep Learning, Energy Efficiency, Data Efficiency, Sustainable Finance, Financial Technology, Financial Volatility Modelling, Long Short-Term Memory

Acknowledgements

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Topic & Background

Escalating concern about how industrialisation has contributed to climate change has lead to many industries coming under increased pressure to monitor and rectify their environmental impact. This pressure is typically directed towards high carbon industries that constitute the major pollutant sectors of the economy, such as transport, energy supply, and agriculture. For example, recent estimates suggest that of the 33.5 billion tons of carbon dioxide emissions generated globally in 2018, with 8 billion tons attributed to the transport sector (IEA, 2022). These concerning figures have rightly sparked increased international discussion surrounding global carbon emissions and sustainability, such as the 2021 *United Nations Climate Change Conference* (COP26).

1.1.1 Sustainable Finance

The finance sector has long been closely associated with sustainability concerns, being a major contributor to global carbon emissions both directly and indirectly. The most visible environmental impact of the financial industry is its direct emissions from business practices such as the distribution of cash through the economy (e.g. cash transport and ATM power consumption), card payment processing centres, and everyday operational costs (Hanegraaf et al., 2018). However, indirect emissions—attributable to services such as investing and lending—have been estimated to contribute over 700 times more to the carbon footprint of the financial industry than all direct emissions (Power et al., 2020). This form of carbon emissions, entitled *financed emissions* by Power et al. (2020), includes practices such as financing fossil fuel companies—who have received $3.8 trillion in funding from global banks since the *Paris Agreement* in 2016 (Kirsch et al., 2022). In their survey of 700 global financial institutions, Power et al. (2020) estimated that the production of over 1.04 billion tons of carbon dioxide was attributable to financed emissions in 2020 (approximately 3.1% of global emissions). Furthermore, a recent report by *Greenpeace* and the *WWF* concluded that the combined carbon emissions of the largest banks and investors in the UK in 2019 totalled 805 million tons, which (if consolidated into its own country) would rank 9th in the global list of total emissions per country (Greenpeace, 2021). This figure is 1.8 times higher than the total emissions of the UK (455 million tons), and almost 90% of the global emissions from commercial aviation (918 million tons) in the same year (Graver et al., 2020).

The increasing awareness of the negative *environmental, social, and governance* (ESG) impacts of the finance industry, highlighted by studies such as those of Power et al. (2020) and Dennis et al. (2021), has led researchers to investigate financial practices that prioritise the *sustainable development goals* (SDGs). These initiatives, known as *sustainable finance*, aim to incorporate ESG impacts and SDGs into financial decisions to improve sustainability. In particular, a focus is given to those practices that produce a measurable positive improvement to the social and environmental impact of the financial industry, global economy, and wider society.

The discussion around sustainable finance largely began with an examination of the benefits of investing pension funds in a socially responsible way by Ferris and Rykaczewski. Since the turn of the century, research has begun to focus on environmental sustainability, considering factors such as climate change and renewable energy (Van Der Laan and Lansbury, 2004), and exploring environment-focused practices such as *climate finance* (Hogarth, 2012) and *carbon finance* (Aglietta et al., 2015). Recent research within sustainable finance has attempted to put into practice the sustainability goals set out by the Paris Agreement, ESG factors, and SDGs. This work, commonly exploring sustainable investment fields such as *impact investing* (Agrawal and Hockerts, 2021) and *ESG investing* (Alessandrini and Jondeau, 2020), has significantly increased the prominence of sustainable finance. In their review of 936 research papers, Kumar et al. (2022) found that almost 70% of sustainable finance research had been published between 2015–2020, and an exponential trend was exhibited in the increase in papers being published each year; additionally, they found that in 2020, $400 billion of new sustainability funds were raised on capital markets. Thus, it is clear the scope and impact of sustainable finance is currently on the rise, predominantly driven by a renewed focus on the ESG impacts of financial practices, resources, and investments.

1.1.2   Financial Technology and the Issues with Sustainable Finance

Whilst the $400 billion raised in sustainability funds seems impressive, in the same year the total US equity market value was over $40 trillion (Ltd, 2022), meaning globally only 0.98% of the value of the US equity market alone was raised. Furthermore, recent research has uncovered the prevalence of investment *greenwashing* (Popescu et al., 2021), where institutions misleadingly classify their practices and investments as sustainable without credible data to back up their claims (and often excluding data that would suggest the opposite). In their review, Cunha et al. (2021) raised similar concerns, asserting that research into sustainable finance is currently "excessively fragmented". These issues indicate that whilst attention is growing around the sustainability of financial practices, this domain is still not widely recognised, and further work and research are still necessary to increase the adoption of sustainable methods and tools within finance.

An additional concern is that the tools used to conduct financial practices are becoming increasingly resource-hungry at a pace exceeding the adoption rate of sustainable finance. A clear exemple of this is the increased adoption of technology throughout the finance industry. Recently, a surge of developments in financial technology (*Fintech*) has revolutionalised financial methods and practices, from those used within large financial institutions and increasing number of Fintech startups, to groups of academic researchers. This Fintech revolution has transformed many aspects of finance, promising to enhance and automate existing financial services, and deliver new, innovative financial products. Palmié et al. (2020) assert that a significant reason behind accelerating

adoption of Fintech in recent years is a new reliance upon *artificial intelligence* (AI). Driven by the promise of increased automation and computing power, the utilisation of AI within the financial sector has been rapidly expanding over recent years, becoming a core component of many of the financial products and services used today: from accurate real-time financial fraud detection (Sadgali et al., 2019), to automated analysis of financial statements (Amel-Zadeh et al., 2020).

1.1.3 Deep Learning for Finance

Because of their power and potential, AI methods have been used to produce state-of-the-art results over a plethora of scientific problems and research fields: from DeepMind's *AlphaFold* (Jumper et al., 2021)—the first programmatic solution to the age-old protein folding problem in Biology— to Google's *PaLM* (Chowdhery et al., 2022)—a cutting-edge human language model that delivers breakthrough results in multi-step arithmetic and common-sense reasoning (a major step towards *artificial general intelligence*). This research typically revolves around the use of *machine learning* (ML), where large collections of data are used to train computational models how to perform certain tasks independently (Samuel, 1959). Recent innovations in ML have increasingly taken advantage of *deep learning* (DL) methods, which use large, complex models to produce state-of-the-art performance (Witten et al., 2017).

These recent DL successes have driven an increased adoption of AI and DL further afield, such as within the finance industry. In fact, it is estimated that 70% of financial institutions rely upon ML (Gokhale et al., 2019), and that global spending on AI is predicted to double in value by 2024, from \$50 billion in 2020 to approximately \$110 billion (Nassr et al., 2021).

In a recent survey of financial professionals, Chartis Research Staff (2019) found that 44% of respondents cited "greater accuracy of process and analysis" as a key motivation behind their adoption of AI methods. This superior accuracy provided by ML and DL methods provides a compelling alternative to traditional statistical models. Hence, the promise of increased accuracy of performance is a major driving factor behind recent ML adoption within Fintech.

1.1.4 The Issues with Deep Learning

Whilst cutting-edge DL models push the boundaries of computational accuracy, few of these systems prioritise the efficient use of energy and data. This has led to DL inflicting a great cost upon the environment, as the energy-intensive algorithms, long training processes, and the power-hungry data centres they utilise inflict a high carbon footprint (Lacoste et al., 2019). Schwartz et al. (2019) assign the label *Red AI* to these accurate but energy-intensive DL models, defining such work as "research that seeks to improve accuracy through the use of massive computational power while disregarding the cost". Schwartz et al. explain how the performance of these systems is often achieved through the use of extensive computational resources, such as complex models with vast parameter sets, large collections of data, and power-hungry computer hardware. This trend is illustrated by Bender et al. (2021) through progression of recent language models: whilst the 2019's state-of-the-art model *BERT* (Devlin et al., 2018) used 340 million parameters and a 16GB dataset, the leading models of 2020 (*GPT-3* by Brown et al. (2020)) and 2021 (*Switch-C* by Fedus et al. (2021)) utilised 175 billion and 1.57 trillion parameters respectively, and data sets of size 570GB

and 745GB. In fact, between 2012 and 2018 the computational resources used to train cutting edge models increased by a factor of $300,000$, outpacing *Moore's Law* (Amodei and Hernandez, 2018).

The intense computational load of these DL models does not come for free; the large parameter and data sets mean training, storing, and computing with these models draws a significant amount of energy—referred to by Bietti and Vatanparast (2019) as *data waste*. Partly due to this inefficiency, the data centres upon which DL models rely for storage and cloud computing become a significant hidden contributor to carbon emissions (Al-Jarrah et al., 2015). Studies such as Masanet et al. (2020) and Malmodin and Lundén (2018), 2020 have estimated that processing at these data centres consumes around 200–300TWh of electricity a year, with the cost of data transmission exceeding this at up to 340TWh per year (IEA, 2022). These estimates suggest that global data centres use more electricity than the majority of countries in the world, ranking above both Australia and Spain (EIA, 2019), and account for around 1–1.5% of global electricity consumption (rising to 2.4% when including transmission costs). Furthermore, this energy is likely not entirely carbon-neutral; Cook et al. (2017) showed that of their total electricity demand, *Amazon Web Services* only powered 12% through renewable sources, and *Google Cloud* 56% (with the latter figure ranging between 4% and 94% depending on location). These figures are also somewhat unrepresentative of the true carbon emissions of such facilities, as they report net emissions including carbon offsetting, which Schwartz et al. (2019) argue has negligible impact on mitigating environmental costs. Thus, these large energy budgets generate considerable carbon emissions, resulting in the use of data centres coming alongside significant environmental detriment. Specifically, research suggests that in 2018, cloud computing at data centres generated the equivalent of 31 million tons of carbon dioxide (Hockstad and Hanel, 2018) in the US alone, equaling the total emissions attributable to electricity generation in California (IEA, 2022).

Beyond these general figures, Strubell et al. (2019) showcased the carbon emissions specifically produced by training DL models. They found that training the language model BERT, which utilised 110M parameters and trained for 96 hours over 16 TPUs, produced the equivalent of 1438 lbs of CO_2—the same as a trans-American flight. Strubell et al. also found that whilst the *Evolved Transformer* of So et al. (2019) improves state-of-the-art accuracy in English-German translation by 0.1 *BLEU* (a common metric of translated text quality), if implemented on GPU hardware this model could cost \$3.2 million to train (or \$147,000 if using TPUs), and generate $626,155$lbs of CO_2 (almost five times the lifetime emissions of an average car).

The large financial cost associated with intensive DL systems also inflicts a great social cost. As the models used at the forefront of DL research get larger and more complex, the price of storing the model and its training data, as well as the cost of running its training process, becomes prohibitively high (Schwartz et al., 2019). This financial barrier restricts who can engage in cutting-edge research to only those with the backing of a large institution. Thus, this lack of accessibility drives a *"rich get richer"* cycle of research funding (Strubell et al., 2019) where only research driections within the interest of these institutions receive enough funding. This not only stifles creativity, but leaves the allocation of who benefits from the development of these systems, and who bears the negative side effects, to a handful of large corporations. In particular, Bender et al. (2021) highlight this disconnect between the benefits of energy-intensive DL research and the environmental consequences it inflicts (using the example of language models (LMs)): "is it fair or

just, for example, that the residents of the Maldives [. . .] pay the environmental price of training and deploying ever larger English LMs".

Thus, whilst DL been shown to provide state-of-the-art computational accuracy across a range of fields, its environmental impact cannot be ignored. This poses sustainability concerns for the accelerating reliance on ML in Fintech, contributing further to the negative ESG impacts of the financial sector; furthermore, this genertates a clear conflict between the growing use of DL in Fintech, and the growing need for sustainable finance.

Both of these fields provide great utility to the finance sector: Fintech (including DL) provides innovative services to consumers and accurate tools for institutions, and sustainable finance ensures the industry inflicts minimal ESG costs. Moreover, DL has been shown to have utility for work in sustainability further afield, such as improving the efficiency of exploiting renewable energy sources (Daniel et al., 2021), and even within sustainable finance itself, for example using ML to analyse the ESG factors of potential investments (Mehra et al., 2022). Hence, a compromise between the use of DL Fintech and the prioritisation of sustainable finance must be reached.

1.1.5 Green AI

The gap between the utility of DL systems (provided by their superior accuracy) and their environmental consequences has recently started to gather attention from ML researchers. In what has become known as *Green AI* (Schwartz et al., 2019), new research has begun to consider how to mitigate the negative ESG impacts of ML by improving the efficiency of DL models. These efficient models reduce the energy and resources required for training and deployment, minimising the carbon emissions they contribute towards.

Alongside *mobile computing*, which prioritises energy-efficient methods due to the hardware constraints of mobile devices, the research domains of *Natural Language Processing* (NLP) and *Computer Vision* (CV) are currently the predominat fields focussing on Green AI, as both these areas exploit highly complex models, and require efficient real-time processing once deployed. NLP focuses on the processing and understanding of language, typically using large, complex DL models in an attempt to match and exceed human accuracy in language modelling (e.g. using language models to make predictions about text sequences). CV typically uses complex models with expensive methods—such as the *convolution operation*—to analyse, classify, and map visual environments. Within these fields, Green AI developers have been working to promote the utilisation of efficient methods that reduce the energy, time, and data requirements of model training by using parameters, data, and operations more intelligently (without producing a significant drop in accuracy). However, these methods have yet to garner significant attention outside of the specific research domains of NLP, CV, and mobile computing. Therefore the utility of these methods to improving the sustainability of ML has largely not been demonstrated further afield, meaning their potential benefits to reduce the energy consumption and carbon emissions of ML have yet to be seen on a wide scale.

1.2 Research Motivations

As demonstrated, the energy-intensive nature of high-performance DL systems inflicts several environmental and social issues, including their significant carbon footprint and financial cost. For this reason, the accelerating adoption of ML and DL within Fintech raises concerns about increasing the negative ESG impacts of the financial industry. Namely, the side effects of these complex DL models produced by their large parameter and data sets and long training processes (and hence high energy budgets) are in direct conflict with the goal of sustainable finance to reduce the negative ESG impacts of the financial sector. However, the benefits of DL to finance (and indeed sustainable finance) have been shown, such as improved fraud detection (Sadgali et al., 2019) and ESG data analysis (Mehra et al., 2022), making their continued adoption inevitable.

Therefore, to minimise the negative ESG impacts of the financial sector in light of the recent Fintech revolution (and solidify the benefits of sustainable finance), the adoption of Green AI principles and methods is paramount. In the first study of its kind, this thesis investigates how the energy-efficient Green AI methods can be adapted for DL in finance. This research aims to demonstrate how Fintech (in particular DL for finance) does not have to come with significant environmental and social costs, and how the models and methods used in this field can coincide with the SDGs of sustainable finance. To exemplify the promise of introducing Green AI to Fintech and sustainable finance, energy-efficient methods will be applied to one of the most popular applications of DL in finance: *financial risk modelling*. Specifically, a popular and important area of financial risk analysis known as *volatility forecasting* will be explored, developing a DL-based model that accurately predicts the future volatility of the *S&P 500* market, but takes minimal energy and data to train. Volatility forecasting is commonly used to give insight into the risk of a financial market or asset (French et al., 1987); it is also been extensively explored by researchers to showcase the prediction capabilities of DL (Zhang et al., 2022). Hence, as risk modelling is often cited as a core application of DL in finance (Thakkar and Chaudhari, 2021), volatility forecasting has been chosen as the specific application within Fintech to demonstrate the promise of applying Green AI methods to the finance industry.

By showing the utility of Green AI within Fintech, this thesis aims to advance the field of sustainable finance as a whole, creating a new field of *sustainable deep learning for sustainable finance*. This work is the first study to address the conflict of interest between sustainable finance and the growing reliance of Fintech on DL systems with high energy consumption and concerning ESG impacts. Hence, the following research aims to demonstrate how the energy and data-efficient methods proposed by Green AI research can be exploited within the DL systems used in finance, increasing the scope and impact of sustainable finance by allowing it to use helpful DL systems without compromising SDGs, and reducing the emissions of Fintech.

1.3 Contributions to Science

This research contributes to scientific literature and the finance industry in a number of ways:

1. *Expanding the applications of Green AI.* This thesis is the first application of Green AI principles and methodology to the field of finance. This will further demonstrate the utility

of Green AI by proving that such methods can provide compelling performance with a lower environmental cost in new domains outside of the existing research focus on NLP, CV, and mobile computing, expanding the scope of this field.

2. *Reducing the environmental impact of financial technology.* The following work combines the research fields of FinTech, deep learning for finance, sustainable finance, and Green AI, to create the new research domain of *sustainable deep learning for sustainable finance.* This further improves the ESG impact of the financial sector beyond previous work in sustainable finance by mitigating the conflict between the utility of DL models for sustainability modelling and the intrinsic carbon footprint of these energy-intensive systems.

3. *Improving the inclusivity of finance.* The reduction in the financial, environmental, and social cost of DL for finance also increases the inclusivity of this field. Improving the energy and data efficiency of these models educes resource requirements and lowers the bar-to-entry to using DL in finance, allowing more industry players, developers, and individual traders to utilise the advantages brought by Fintech and DL.

1.4   Research Objectives

The initial objective of this research is to analyse the efficiency of a baseline DL model used for financial volatility forecasting. In particular, the resource requirements in terms of training time and training dataset size are inspected. A particular focus is given to the use of *long short-term memory* (LSTM) networks, as these are the typically used DL model for sequence forecasting tasks in general, and recent research into volatility forecasting with DL (Xiong et al., 2015a). The second objective is to then implement energy-efficient training process for the baseline model, adapting the traditional training algorithm originally utilised such that the total training time is reduced. The final objective is to build an alternative data-efficient training algorithm for the implemented DNN, which facilitates the same performance within the chosen financial domain but requires less training data.

1.5   Research Structure

The aforementioned research will be organised as follows:

- Chapter 2: *Background & Literature Review.* The relevant literature to this research is first explored to give an overview of the key concepts and methodologies utilised in the following experimentation. This begins with a general exploration of machine learning, which then leads to a more refined discussion of the use of DL in finance, and the specific DL methods and concepts focussed upon within this thesis, giving a clear outline of the domain of financial volatility forecasting. Following this, the review explores the research domain of Green AI, with a focus on the specific methods and practices used in Green AI research which will go on to be a core aspect of this thesis.

- Chapter 3: *Methodology.* The methods used to conduct the three core studies of this research will then be outlined.

- Section 3.4: *Baseline Volatility Model Implementation*. An explanation and analysis of the baseline volatility model implemented, exemplifying the typical models, methods, and resource requirements of this domain.

- Section 3.5: *Energy-Efficient Training Extensions*. Once the baseline model has been implemented, several energy-efficient adaptations to its training process shall be explored. This section gives a complete overview of these methods, explaining how they work, the motivations behind their use, and their utility in decreasing the energy consumption of training.

- Section 3.6: *Data-Efficient Training Extensions*. The utilised methods to reduce the data requirements of training are then be explored, discussing their origins within Green AI and explaining how they aim to reduce the data requirements and computational cost of DNNs.

- Chapter 4: *Results & discussion*. An extensive analysis of the benefit of Green AI methods to this domain is then made, quantifying the facilitated reduction in time and energy (and hence carbon emissions). This compares both the accuracy of the final model and each energy and data-efficient extension to the original baseline model. The resource requirements required for training will then be compared to evaluate any accuracy-efficiency compromises made. A general conclusion is then settled upon as to the utility and viability of Green AI methods to Fintech.

Chapter 2

Background & Literature Review

To understand how to align Fintech with the goals of sustainable finance, a detailed understanding of DL must be conveyed, exploring the underlying methods and potential areas for efficiency improvements. This chapter achieves this by introducing the relevant background literature within the field, beginning with a general summary of the components and characteristics of ML and DL methods, before focussing on the specific models and algorithms used within volatility forecasting. The second part of the chapter outlines the field of Green AI, its motivations, applications, and the relevant tools and practices used in this domain to improve the energy efficiency of DL, acting as the basis for the experimentation that follows.

2.1 Machine Learning

Machine learning, first discussed by Arthur Samuel in his 1953 exploration of "mechanical brains" (Samuel, 1959), is a subset of artificial intelligence concerned with using data to allow computer programs to independently learn how to complete a given task without explicit information about the rules of the given problem. This learning typically involves a *training* procedure, where the program is supplied with instances of experience from the training dataset D describing the problem. The program learns from D how to perform the task by taking input of a data instance and outputting the result it believes is the correct response. This result is then evaluatied through some performance metric (typically computed through a *loss function*). For example, in a *prediction* task, the program takes input of a collection of values and outputs the value it predicts should come next: e.g. Xiong et al. (2015a) use a dataset of S&P 500 values and Google domestic trends to predict future stock market volatilities. Alternatively, in a *classification* task, a data point is input into the program to be assigned to a distinct class: e.g. Sadgali et al. (2019) use financial transaction data to classify whether a partiuclar transacition is fradulent or not. Such tasks are typically trained through *supervised learning*, where the correct result (known as the *label*) is provided to the program after it has produced its output, and the correctness of its solution is evaluated such that subsequent outputs can be adapted to optimise performance.

2.1.1 Neural Networks

Since the conception of machine learning, a core goal has been the development of algorithms that can accurately mimic the processing mechanisms of the human brain. These algorithms, known as *artificial neural networks* (ANNs), typically implement neural models that recreate a version of the complex communication system between neurons in the brain. The first implementation of such a model was Frank Rosenblatt's *Mark I Perceptron* (Rosenblatt, 1958), which attempted to perform binary classification of images. Each pixel in the input image was represented as a single value in a 2D matrix of input neurons known as the *input layer*. These values were then passed to the single internal neuron through *weighted channels* to compute the weighted sum of all input values, the result of which was passed through an *activation function* to normalise the result to zero or one—representing an output classification of 'class zero' or 'class one'.

2.1.2 Deep Learning

Whilst Rosenblatt's simple model produced underwhelming results, its design became a fundamental building block of the larger, more complex ANNs used today. Namely, modern networks consist of wider and deeper *model architectures* containing more internal neurons arranged in multiple layers. Recent work has pushed the boundaries of neural network performance by designing deeper and deeper architectures—known as *deep neural networks* (DNNs)—to establish the field of deep learning. These deep networks perform computations by passing information between the input layer $l^{(0)}$ and output layer $l^{(L-1)}$ along weighted channels between each neuron; the *activation value* $a_n^{(i)}$ assigned each neuron n in layer i is determined through the dot product between the weights $W^{(i-1,i)}$ of channels entering n and the values $x^{(i-1)}$ of neurons in the preceding layer $i-1$ connected through those channels (Witten et al., 2017). This product is then summed with a *bias* term $b^{(i)}$, and passed through a chosen activation function α (Equation 2.1).

$$a_n^{(i)} = \alpha(W^{(i-1,i)} \cdot x^{(i-1)} + b^{(i)}) \tag{2.1}$$

This computation is performed over all layers of the network, starting with the input layer—whose neurons take the values of the input vector—and propagating through to the output layer—the values of which are the final network output. During training, the values of the network's parameters (i.e. the elements of weight matrix W and bias vector b) are adjusted as the model learns, manipulating its outputs to more closely align with the true labels.

The full training process is typically implemented over a predetermined number of iterations over the entire training dataset. Within each iteration (known as an *epoch*) batches of data of a given size (referred to as the *batch size*) are sampled and input into the network to improve its performance. The network's parameter set is then updated using *backpropagation*, where the gradient of loss function with respect to the parameters is determined by propagating the error of the network on the current input backwards through all its layers (Zaras et al., 2022). These parameters can then be adjusted in the direction of the negative gradient through *gradient descent* (towards the minimum of the loss function).

2.1.3 Sequence Modelling Problems

In machine learning, a *sequence* is an ordered set of data elements, each of uniform type and dimension: for example, a stream of text can be seen as a sequence of strings (i.e. words), or a video can be modelled as a sequence of pixel matrices (i.e. frames of the video). Mathematically, temporal sequences—typically referred to as *time series*—can be denoted by their data elements x and time indices t:

$$seq = x_1, x_2, \ldots, x_t, \ldots, x_T. \tag{2.2}$$

Sequence learning (or *sequence modelling*) uses ANNs to model, analyse, and generate these sequences. The problem is common in fields such as NLP: for example, Meta AI's *M2M-100* (Fan et al., 2020a), the first language model that can directly translate between any pair of 100 different languages. A common example of such a task is the *sequence-to-sequence* problem, where a model both takes input of, and outputs, a data sequence. This problem typically involves either *translation*, where the M elements in the original sequence are converted into a new sequence of length N, or *prediction*, where the original sequence of elements $x_1, \ldots, x_M$ is used to predict a subsequent sequence $x_{M+1}, \ldots, x_{M+N}$. *Machine translation* is one such application, in which a model takes input of a sequence of text written in one language and generates a new output sequence in a different language (e.g. Meta AI's M2M-100). Prediction tasks include applications such as *time series forecasting*, where sequence elements are predicted for future timesteps, in such domains as stock price prediction (Darapaneni et al., 2022). In this case, the output sequence $y = x_{t+1}, \ldots, x_{t+n}$ is computed based upon the preceding time series $x_0, \ldots, x_t$, building a model M_θ (parameterised by θ) that approximates the mapping $y = f(x_0, \ldots, x_t | \theta)$ between past and future time series.

2.2 Recurrent Neural Networks

2.2.1 Neural Networks for Sequence Modelling

Forecasting sequences typically poses a challenge for traditional ANNs due to their fixed input structure that requires all elements of an input sequences to be fed into the network at the same time, with one element per neuron in the input layer. Hence, the input would contain no intrinsic sense of order, severely limiting the amount of analysis possible by the network, as the order of a sequence is crucial to understanding and forecasting them (Tsantekidis et al., 2022). The chosen input size would also pose a challenge for a network with a fixed input structure, as the number of input neurons would have to be made adequately large to accommodate the possible variations in sequence lengths. Additionally, learning a sequence's characteristic behaviour requires persistence of information independent from its location within the sequence. Namely, the network must contextually use previous understandings to inform the current instance, wheverver they appeared in the preceding sequence. This is not possible with a traditional ANN as the parameters of channels connected to different input neurons are not shared. Therefore, when the parameters of a specific neuron and its channels are adjusted after observing an important pattern in part of a sequence, this information will only prove useful if that patten appears again in the exact same location on

the input neurons, as if the pattern occurs at a different position it will be input through different neurons that have not been previously learned to deal with this subsequence.

2.2.2 Recurrent Networks

To overcome these issues, *recurrent neural networks* (RNNs) were conceived that use recurrent loops to allow varying input lengths and the persistence of learned sequential information (Sharma et al., 2022). These networks take input of a single sequence element x_t at each timestep t over a sequence, using this and previously observed elements to inform the internal state h_t, which acts as an internal memory. This state is updated sequentially to incorporate information learned from the current input x_t through the function g, parameterised by $\theta^{(g)} = \{W^{(g)}, U^{(g)}, b^{(g)}\}$, where $W^{(g)}$ and $U^{(g)}$ are weight matrices, and $b^{(g)}$ a bias vector(Sharma et al., 2022):

$$h_t = g(h_{t-1}, x_t | \theta^{(g)}) \tag{2.3}$$
$$= W^{(g)} \cdot h_{t-1} + U^{(g)} \cdot x_t + b^{(g)}. \tag{2.4}$$

The state h_t is then output along a recurrent loop that connects the output of the network at time $t-1$ to the input of the network at the next timestep t. This allows information to be passed between successive versions of the same neural network at different points in time. At the final timestep T, the state h_T is output as the final network output. The concept can be demonstrated by unrolling the network in time, shown in Figure 2.1. Tsantekidis et al. (2022) note that to avoid confusion is important to remember that the sequence of networks depicted in illustrations such as Figure 2.1 are in fact just the same network only at different points in time, with the connecting arrows between each signifying that the output at time $t-1$ is being passed on to the same network at the subsequent timestep t.
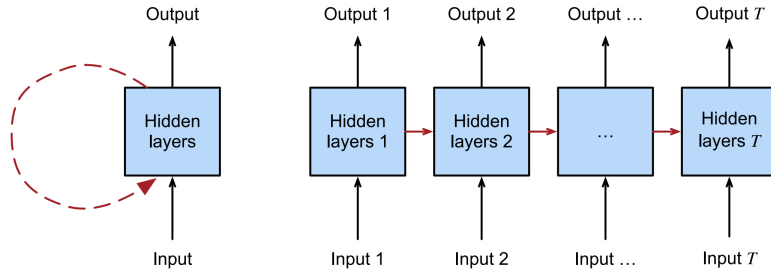
Figure 2.1: Diagram of an RNN at a single timestep t (left) and unrolled in time between timesteps $t = 1$ and $t = T$ (right) (Zhang et al., 2021)

To capture more information from the input sequences, RNNs can consist of multiple internal layers (shown in Figure 2.2), allowing them to learn higher-dimensional internal representations (Bengio, 2009). In this case, each layer i retains its own internal state $h_t^{(i)}$, which is passed on to both the next layer to compute its state $h_t^{(i+1)}$ (at the same input timestep) and reccured back into itself at the subsequent timestep to compute the state $h_{t+1}^{(i)}$ (Zhang et al., 2021).
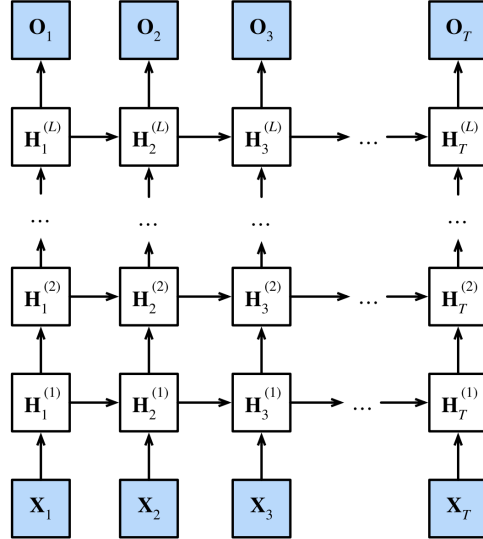
Figure 2.2: Diagram of a multi-layer RNN (showing layers 1 to L) with each layer unrolled in time between timesteps 1 and T (Zhang et al., 2021)

2.2.3 Training

Training of RNNs can be conducted through a similar backpropagation process to that used on traditional DNNs. This is done through computing the final output of the network on a given sequence and evaluating the loss function between this and the true result to compute the network's error. The error is then propagated backwards through both the internal layers of the network and through time to the network state at each timestep of the input sequence—and hence is referred to as *backpropagation through time* (Zhang et al., 2021). This allows the training procedure to adjust both the network parameters (weights W and biases b) and the parameters $\theta^{(g)}$ of the state function g.

2.2.4 Applications

Recurrent networks have been employed by most recent research into sequence modelling. In their review of the applications of RNNs within this domain, Lipton et al. (2015) explain how NLP tasks and time-series forecasting are the most common avenues explored by researchers utilising RNNs. These two areas have been extensively surveyed in literature; for example, Hewamalage et al. (2021) survey the current and future applications of RNNs for time series forecasting, highlighting the recent success of this model architecture at forecasting competitions, such as an RNN-based model by Smyl (2020) winning The M4 Competition in 2019 with cutting-edge accuracy nearly 10% greater than the utilised baseline (Makridakis et al., 2020).

2.3 Long Short-Term Memory

RNNs have been shown to be incredibly effective at modelling short-term dependencies within an input sequence. In many cases, however, to understand the full context of a sequence longer-term dependencies need to be taken into account. Unfortunately, Hochreiter (1991) and Bengio et al.

(1994) have shown that RNNs cannot accurately capture or exploit these long-term dependencies, as contextually relevant elements observed many timesteps ago can be forgotten prematurely. This issue is known as the *short-term memory problem* of RNNs. Whilst several solutions to this problem have been proposed—such as *skip connections* and *leaky recurrent units*—the most widespread approach is the use of *Long Short-Term Memory* (LSTM), presented by Hochreiter and Schmidhuber (1997).

2.3.1 Long Short-Term Memory Networks

An LSTM network is an RNN that uses multiple internal states and parameter matrices to model both short-term and long-term dependencies within sequences. Similarly to an RNN, this architecture contains a core network consistent of one or more layers—where each layer is a distinct LSTM cell—and information is passed both between these cells (from the input to the output layer of the network) and between timesteps (reccuring the output at time $t-1$ to be input at time t). However, instead of a single recurrent state h_t, the LSTM contains two: a short-term memory state h_t (analogous to that of an RNN) and long-term memory state c_t used to retain information from far back in the input sequence.
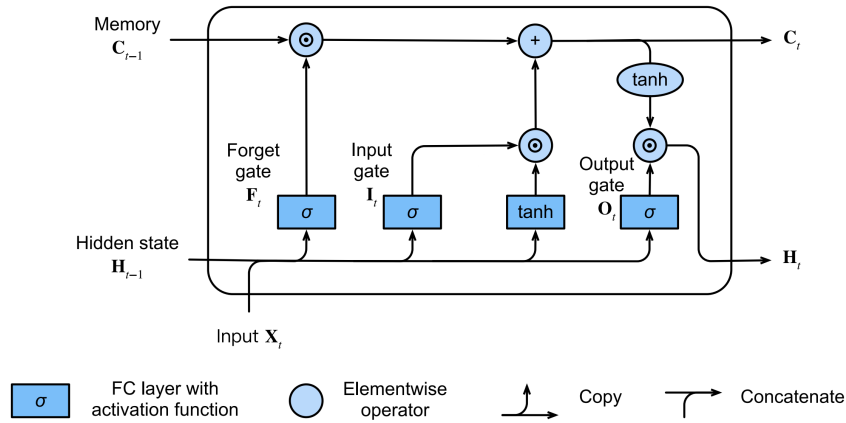
Figure 2.3: Diagram of a the arrangement of logical gates within an LSTM cell (Zhang et al., 2021)

At each timestep, the preceding long-term state c_{t-1} is input into the LSTM cell (alongside the current sequence element x_t) to compute the new state c_t. These updates are achieved through several logical *gates* within the network (Figure 2.3). Firstly, the *forget gate* is used to decide what information to discard from the previous state c_{t-1}. The previous short-term state h_{t-1} and input element x_t are multiplied with the forget gate's weight matrices $W^{(F)}$ and $U^{(F)}$ and added to its bias $b^{(F)}$, the result of which is passed through the *sigmoid* activation function to produce the intermediate state value $\tilde{c}_t^{(F)}$, shown in Equation 2.5 (Zhang et al., 2021).

$$\tilde{c}_t^{(F)} = \sigma(W^{(F)} \cdot h_{t-1} + U^{(F)} \cdot x_t + b^{(F)}) \tag{2.5}$$

Secondly, when new relevant information is seen in the input sequence, the *input gate* is used to add it to the long-term memory state c_t. This process similarly uses the short-term state h_{t-1}

and current input x_t, but evaluates both a sigmoid and *hyperbolic tangent* activation function in parallel on these inputs to decide both which values of c_{t-1} must be updated (through the sigmoid evaluation) and by how much (the tangent evaluation). The result of these two operations are then multiplied together to give the second intermediate state value $\tilde{c}_t^{(I)}$, shown in Equation 2.6 (Zhang et al., 2021).

$$(\tilde{c}_t^{(I)})_\sigma = \sigma(W^{(I,0)} \cdot h_{t-1} + U^{(I,0)} \cdot x_t + b^{(I,0)}) \tag{2.6}$$

$$(\tilde{c}_t^{(I)})_{\tanh} = \tanh(W^{(I,1)} \cdot h_{t-1} + U^{(I,1)} \cdot x_t + b^{(I,1)}) \tag{2.7}$$

$$\tilde{c}_t^{(I)} = (\tilde{c}_t^{(I)})_\sigma \cdot (\tilde{c}_t^{(I)})_{\tanh} \tag{2.8}$$

Both intermediate state values $\tilde{c}_t^{(F)}$ and $\tilde{c}_t^{(I)}$ are combined to give the new updated long-term state c_t (Equation 2.9).

$$c_t = \tilde{c}_t^{(F)} \cdot c_{t-1} + \tilde{c}_t^{(I)} \tag{2.9}$$

The third and final gate used by LSTMs is the *output gate*, which decides what each cell outputs. This is used as both the short-term state h_t and the final network output y. The output gate uses the newly computed long-term state c_t, input x_t, and previous short-term state $ht-1$. A biased weighted sum of h^{t-1} and x_t is computed and fed into the sigmoid function, which is then multiplied by the hyperbolic tangent function evaluated on c_t (Equation 2.10). Similarly to within the input gate, the sigmoid evaluation determines the values of h_{t-1} to be updated, and the tangent evaluation determines by how much (Zhang et al., 2021).

$$(h_t)_\sigma = \sigma(W^{(O)} \cdot h_{t-1} + U^{(O)} \cdot x_t + b^{(O)}) \tag{2.10}$$

$$(h_t)_{\tanh} = \tanh(c_t) \tag{2.11}$$

$$h_t = (h_t)_\sigma \cdot (h_t)_{\tanh} \tag{2.12}$$

This system of gates occurs in every LSTM cell, passing information from the input to output layer of the network at every timestep in the form of the short and long-term states. When the network reaches the final timestep, the short-term state is output as the final network output y; note, if the sequence modelling problem involves outputting a generated sequence of length N (e.g. forecasting multiple future values), this is produced one element at a time over N additional timesteps that take no input and output the value $y_i = h_{T+i}$.

2.3.2 Applications

Whilst reviewing modern RNN architectures, both Lipton et al. (2015) and Yu et al. (2019) assert that most state-of-the-art applications within the field of sequence learning use a LSTM-based model, largely due to this architecture's ability to model both long and short-term dependencies in sequences. Due to the superior accuracy provided by this architecture, LSTMs have been widely used in both NLP and time series forecasting. For example, Shi et al. (2022) recently compared the

performance of a number of networks (including both traditional RNNs and LSTMs) for predicting Bejing air quality over time. They concluded that LSTMs generated more accurate predictions than simpler RNNs, and demonstrated their improved long-term memory by showing LSTMs outperform other networks even when the context window available to the model is small. This architecture has also been shown to premit the accurate forecasting of financial markets, with research highlighting its ability to accurately capture the complex dependencies within highly variable financial variables (Li and Tam, 2017). The success of LSTMs within this domain has been demonstrated across a variety of financial modelling tasks, such as forecasting commodity prices (Ly et al., 2021), stock market indices like the S&P 500 (Fjellström, 2022), currency pairs on the Foreign Exchange (Qi et al., 2021), and the fluctuations of financial market risk (Du et al., 2019).

2.3.3 The Efficiency of Recurrent Networks

The use of RNNs and LSTMs has revolutionalised sequence learning, permitting models with cutting-edge accuracy across a spectrum of applications, including financial risk forecasting (Du et al., 2019). However, further research into recurrent architectures has shown that the sequential processing methods vital to their performance also make these models highly inefficient.

The tradeoff between accuracy and efficiency within RNNs has been explored by several recent research papers aiming to reduce the complexity and energy consumption of these architectures. In their study of compression techniques for LSTM models, Wang et al. (2018) identify the efficiency of such architectures as a core limiting factor restricting their use in resource-constrained domains. This is similarly identified by Zarzycki and Ławryńczuk (2021) in their exploration of LSTM-based predictors for chemical reactors; they concluded that whilst the number of parameters utilised by an LSTM was directly proportional to its modelling performance, higher complexity models inflicted a significant computational cost.

Numerous sources, such as Cao et al. (2017), Feliz (2021), and Zhang et al. (2021), have deduced that this inefficiency is directly caused by the sequential nature of RNNs. They assert that the many dependencies within LSTM cells—both between cells and to the input sequence—restrict the way in which these models can be trained and used. Specifically, cells in an LSTM network take input from both the input sequence and preceding cells; each cell in every network layer and timestep receives both an input vector and two memory states, introducing both temporal dependencies (between cell states at each timestep) and layerwise dependencies (between the output and input of each network layer). Both Cao et al. (2017) and Feliz (2021) identify that this means LSTMs cannot effectively exploit parallelisation, as variables with sequential dependencies must be computed serially. Thus, these networks cannot fully take advantage of the computational speedups provided by specialised hardware such as *Graphics Processing Units* (GPUs) and *Tensor Processing Units* (TPUs) that have been effectively used to parallelise other DNNs, making training long and energy-intensive (Zhang et al., 2021). Furthermore, these arcitectures require significant amounts of energy to store and fetch parameters (Feliz, 2021), and amount large memory costs calculating large intermediate values during state updates (Cao et al., 2017).

2.4 Financial Risk Modelling

Predicting stock price movements is a common example of sequence-to-sequence modelling that has received great attention within the field of deep learning for finance, with numerous reviews extensively exploring the area (such as Berat Sezer et al. (2019) and Jiang (2021)) and innovative approaches continually pushing the boundaries of prediction accuracy (Darapaneni et al., 2022). Despite this plethora of research, it is still a challenging task to accurately model price movements in real financial markets due to the complexity of financial systems, non-linear relationships between financial variables, and high-frequency variations in values (Timmermann and Granger, 2004).

2.4.1 Financial Risk & Volatility

The challenges associated with market price prediction mean that within the finance industry a more helpful use case of sequence-to-sequence modelling is forecasting financial risk. In their white paper explaining the applications and benefits of ML for financial modelling, LaPlante and Rubtsov (2019) exemplify this sentiment, asserting that ANNs provide the most utility for highlighting potential risks, not explicitly modelling variable movements. In fact, a recent survey of financial professionals by Chartis Research Staff (2019) found that 70% of respondents use ML tools in the financial risk sector, majoritively for analysing "market risk for the trading book" (51% of respondents) and "market risk for the banking book" (44%). This trend is echoed by Peng and Yan (2021) in their review of Fintech, DL, and financial risk; they assert that modelling risk is currently in high demand due to the large fluctuations, uncertainty, and high volatility exhibited in financial markets, making risk an important consideration when making financial decisions. Furthermore, Mashrur et al. (2020) highlight that financial market risk typically exhibits identifiable trends as risk patterns reoccur in cycles, facilitating accurate modelling based upon historical data.

For these reasons, modelling financial market risk is a popular application of DL. This typically involves the development of statistical models to forecast trends within different risk measures such as *value-at-risk* (VaR) and *volatility* (Peng and Yan, 2021). VaR estimates the value an asset is likely to lose over a given time period, to a specified confidence level, returning a value that indicates the maximum expected loss over that period at that confidence level. Alternatively, volatility can be defined as the scale of fluctuations in the pricing of an asset within a financial market over time and is intrinsically related to the risk associated with that asset (Cavalcante et al., 2016).

Cavalcante et al. (2016) assert that volatility is a vital measure to financial analysis, as it is a key indicator of the state of many economic and external factors that affect financial markets (such as investor sentiment and political conditions). This is because changes to these factors directly affect the volatility of a market; for instance, in times of economic crisis or political instability, volatility tends to rise, meaning large price variations of assets are likely, and the market is considered high risk (Berat Sezer et al., 2019). Because of this relationship between volatility and risk, Tino et al. (2001) highlight that volatility changes are typically used as buy and sell signals to investors; Ge et al. (2022) similarly assert market volatility dictates many decisions of market players. Hence, forecasting market volatility is a popular modelling domain. In fact, whilst analysing models for the financial derivatives market, Ozbayoglu et al. (2020) found that most research into ML and DL-based modelling in this field either focuses on pricing or volatility estimation. This popularity

is further demonstrated through the plethora of academic reviews specifically surveying this field, such as those by Poon and Granger (2003) and Ge et al. (2022), and the spectrum of recently proposed DNN-based forecasting models.

2.4.2 Measures of Volatility

The review of Ge et al. (2022) and recent exploration of volatility modelling by Tino et al. (2001) both provide comprehensive overviews of the most commonly used measures quantifying volatility: *historical volatility*, *realised volatility*, and *implied volatility*. Implied volatility (IV) is computed given a specific option price, computing the expected volatility associated with that price. Alternatively, realised volatility (RV) is a measure computed directly from the price movements of a market and hence is the volatility actually realised in the underlying financial market. Historical volatility (HV) is a type of realised volatility, which computes the volatility over a preceding time period based upon market closing prices.

Due to the differences in what these measures represent, each is computed in a slightly different way. Since IV represents an expectation of volatility, it cannot be computed directly from financial market data; instead, IV is estimated through an option pricing model such as the *Black-Scholes model* (Black and Scholes, 1973). Options models are used to estimate the price of an option (a financial contract between buyers and sellers) under the current market conditions. When inverted, these models can additionally be used to compute IV by inputting a given option price (Tino et al., 2001). On the contrary, RV is computed directly from market price movements over a time window $\tau_1 \rightarrow \tau_2$ (Ge et al., 2022). This measure is typically calculated as the variation in logarithmic returns r_t of an asset (Equation 2.13) over the specified time period. Most commonly, this time window covers returns observed in the past, which is the HV of that period. When surveying various volatility measures, Ge et al. (2022) found HV to be the most popular, primarily because it provides an intuitive metric that is simple to compute. The most common method of computing HV is to approximate the volatility v_t over the N values over the interval $t \colon \tau_1 \rightarrow \tau_2$ as the standard deviation of logarithmic returns between these times; this is shown in Equation 2.14, where $\mu(\tau_1, \tau_2)$ is the mean return over the time interval (Ge et al., 2022). This method of computing HV is used across a wide range of applications within volatility modelling, such as the ANN-based model of Lahmiri (2017) that forecasts the volatilities of the exchange rates between currency pairs.

$$r_t = \log \left(\frac{c_t}{c_{t-1}} \right) \tag{2.13}$$

$$v_t = \sqrt{\frac{1}{N} \sum_{t'=\tau_1}^{\tau_2} \left(r_{t'} - \mu(\tau_1, \tau_2) \right)^2} \tag{2.14}$$

In their review of 35 volatility forecasting ANNs, Ge et al. (2022) found HV was the most popular metric, constituting 71% of research papers; this was followed by other RV measures (17%), and finally IV (9%). Additionally, Ge et al. (2022) explored the most common domains for forecasting volatility, finding the S&P 500 market to be the most popular (accounting for 12 out of the 35 research papers), primarily due to its accessibility; the price of oil (7 papers) and metal (6 papers) were also found to be common modelling applications. However, a vast spectrum of models have

been developed outside of these domains; for instance, Ge et al. (2022) also explored the use of IV, RV, and HV within the context of stocks, bonds, and other financial indices. Hence, the selection of a volatility measure is typically a domain-specific choice, based upon the different benefits and drawbacks associated with each metric. For example, IV has been seeing increased adoption from institutional investors, hedge funds, and banks (Neftci, 2008), and is used to define major volatility indices such as the *Chicago Board Options Exchange's Volatility Index* (VIX). However, due to its reliance upon an options pricing model, IV requires options data, and hence cannot be used in other domains. There are also challenges associated with RV and HV; whilst RV has been shown to tend towards an estimate indistinguishable from the latent volatility (Andersen et al., 2001), this accuracy requires data with a high sampling frequency. Additionally, several issues unanimously encompass all volatility estimates, including price irregularities at the tail-ends of return distributions (discussed in detail by Ozbayoglu et al. (2020)) and the complex dependencies and transient relationships between financial variables (Timmermann and Granger, 2004).

2.4.3 Forecasting Financial Volatility

Due to the utility of modelling risk, forecasting market volatility has become a popular domain within both computational finance research (Ozbayoglu et al., 2020) and the finance industry (Chartis Research Staff, 2019). Traditionally, financial institutions and researchers have exploited *generalised autoregressive conditional heteroscedasticity* (GARCH) models to forecast volatility. GARCH models have been shown to be accurate and reliable, with research such as that of Lahmiri (2017) demonstrating the ability of these systems to capture the characteristics of financial time series. These models calculate the volatility of a market from a sequence of logarithmic returns $r_0, r_1, \ldots, r_t$, approximating the volatility v_t at time t as the conditional variance σ_t^2 of returns over a specified preceding time period. Namely, GARCH uses the time series $r_0, r_1, \ldots, r_t$ to predict the volatility v_t, building up a volatility time series $v_0, v_1, \ldots, v_t$. Early work in this field by Akgiray (1989) showed that GARCH models fit training data and forecast volatility more accurately than other models in its class. Similar studies, such as that of Hansen and Lunde (2005), have also demonstrated the impressive forecasting accuracy of GARCH models within the foreign exchange market, and shown GARCH's ability to effectively determine clusters of high and low volatility. Largely because of this accuracy, GARCH has become one of the most frequently utilised methods of estimating the volatility of financial time series. Even despite modern methods, this model is still a popular choice, often being used as a baseline to judge the performance of newly proposed volatility forecasting models; for example, Rodikov and Antulov-Fantulin (2022) used GARCH to contextualise the results of their DNN-based system. The calculations inherent to GARCH have also recently been used in several hybrid models to complement the computation of volatility within a larger system. In fact, Ge et al. (2022) found that of 35 volatility forecasting models surveyed, 14 focussed upon hybrid models, with GARCH being the most commonly appended method.

2.4.4 Deep Learning for Volatility Forecasting

Whist traditional forecasting methods have been shown to produce accurate estimations of the risk of financial markets, a significant amount of recent work has focussed on applying DL to this domain; for example, Chartis Research Staff (2019) found that 70% of financial institutions use

ML for risk forecasting, most commonly for "market risk". Through analysing the publication rate of research papers in 2018, Berat Sezer et al. (2019) found that volatility forecasting was in the top five uses of DL within financial research. Additionally, in their review of ML-based volatility models, Ge et al. (2022) asserted that almost all recent work on forecasting financial volatility relied on ML and DL methods, largely due to their improved accuracy and ability to fit complex time series. DNNs are used to forecast volatility through sequence-to-sequence learning; a time series of logarithmic returns $r_0, \ldots, r_t$ and of previously calculated volatilities $v_0, \ldots, v_t$ are used to predict the following volatility sequence $v_{t+1}, \ldots v_{t+n}$ starting at timestep $t+1$ (predicting v_{t+1}) and continuing to forecast a specified number of steps into the future (up to $t + n$) through sequence-to-sequence prediction. Namely, the volatility model M_θ is developed that computes the output sequence $y = v_{t+1}, \ldots, v_{t+n}$ by approximating the mapping $y = f(r_0, \ldots, r_t, v_0, \ldots, v_t | \theta)$ between past and future time series.

This performance improvement has been exemplified through several extended studies; for example, Zhang et al. (2022) demonstrated the superiority of DNNs at handling complex interactions and dependencies between financial variables, as their high-dimensional parameter sets allow them to act as better function approximators. Through an evaluation of volatility forecasting models over 23 different stocks, Rahimikia and Poon (2020) found that DL gives stronger forecasting power than autoregressive methods like GARCH. Rodikov and Antulov-Fantulin (2022) drew a similar conclusion, finding that an LSTM-based model could generate a higher test accuracy than other well-known models. They also found that RNN architectures were effective at estimating financial market variables like volatility as they do not need to know the parameters of the underlying distribution of the variable being forecast. This is different from models such as GARCH, which uses *maximum likelihood estimation* to approximate the parameters of the return and variance functions of the market. Furthermore, Tino et al. (2001) assert that GARCH has been shown not to be able to facilitate notable profits for investors, whereas RNN-based models show promise in facilitating a statistically significant profit for market players using them within their trading strategy.

Whilst some researchers opt for alternative architectures (such as the CNN-based model of Chen et al. (2018)), the improved accuracy and potential profit offered by RNNs and LSTMs have established them as a core focus of DL models for volatility forecasting. In their survey of 35 research papers, Ge et al. (2022) found that 9 out of the 21 pure models utilised an RNN, as these networks are a natural fit for time series data. This research typically explores the use of RNNs for forecasting realised volatility within the S&P 500 market, as Bucci (2020) showed that RNNs outperform all other traditional methods in this domain. Their experimentation demonstrated the ability of an LSTM to capture long-term dependencies within the financial time series, which enabled this architecture to produce accurate forecasts during highly volatile periods. Additionally, Xiong et al. (2015b) incorporated Google domestic trend data into their RNN-based model for the S&P 500, finding that RNNs are more robust against noise in the time series data.

Hence, it is clear that the use of DL (in particular RNNs and LSTMs) is an expanding field within financial volatility forecasting. This is primarily due to the increased forecasting accuracy provided by DNNs, as these models have been shown to produce class-leading performance across markets (such as the S&P 500) and volatility metrics (most notably RV).

2.5 Green AI

Complex DL systems are seeing increased adoption across financial modelling and further afield, primarily due to their superior accuracy. However, many of these implementations ignore the efficiency concerns associated with DL, such as the memory and energy inefficiency of recurrent networks. The ignorant use of these resource-hungry Red AI models inflicts large ESG costs, as their extreme computational load generates thousands of pounds of carbon emissions over training (Strubell et al., 2019), and limits who can research and employ within high-performance ML systems (Bender et al., 2021). These impacts have spurred recent attention around Green AI, where DL energy and data-efficient models are developed in an attempt to reduce resource requirements, mitigate environmental costs, and promote inclusivity (Schwartz et al., 2019).

2.5.1 Quantifying the Energy Efficiency of Deep Learning

As a simple initial approach to mitigating the ESG cost of DL, many proponents of Green AI suggest that new research papers presenting cutting-edge DNNs should report their training time and resource requirements. Strubell et al. (2019) assert that this should allow future work looking to utilise existing methods to conduct better cost-benefit analyses. In their analysis of 60 DL research papers, Schwartz et al. (2019) found that 90% of papers in the ACL Anthology, 80% of those in the NeurIPS conference, and 75% in the CVPR conference cited accuracy improvements as the main contribution of their work, with only 10% of ACL papers and 20% of CVPR papers contributing a new efficiency result. Schwartz et al. (2019) argue that this demonstrates the lack of reporting surrounding the energy and data efficiency of DL models, highlighting that describing performance contextually with respect to training budgets improves both the sustainability and inclusivity of this work by allowing future work to be compared despite fewer training resources.

Despite the consensus on the importance of reporting the resource requirements of these systems, a ubiquitous hardware-independent measure of the computational cost of a DL model has yet to be agreed upon. Schwartz et al. (2019) describe how the expense $Cost(R)$ of processing the result R using a DNN is proportional to the cost of processing a single instance I, the size D of the training dataset, and the number of hyperparameters H that require tuning (Equation 2.15). They survey several proposed metrics for quantifying this cost, including runtime, parameter count, electricity usage, and carbon emissions; however, they highlight that each of these poses its own challenges, such as the hardware-dependent nature of electricity usage and difficulty in measuring carbon emissions. Schwartz et al. (2019) conclude that reporting the total number of *floating-point operations* (FLOPs) to generate the result is the most reliable way to quantify the computational cost, and hence energy efficiency, of a system.

$$Cost(R) = I \cdot D \cdot H \tag{2.15}$$

Schwartz et al. (2019) use the competing models ResNet (he-2015) and ResNeXt (xie-2017) to exemplify how reporting FLOPs can contextualise performance advancements with model efficiency: whilst ResNeXt provided a 0.5% accuracy boost over the *ImageNet* dataset (becoming 2017's leading *ImageNet top-1 accuracy* score), it required 35% more FLOPs. Amodei and Hernandez (2018) echo these benefits of reporting FLOP; however, they highlight that computing the FLOP

count of a program is not always trivial. They suggest that in the absence of being able to directly compute the FLOPs of a given DL model, its computational load can be approximated through its GPU utilisation over training; they assert that the computational cost of training $Cost(T)$ in *petaFLOP/s-days* (pfs-days) is given by the product of the number of GPUs used N_{gpu}, the processing power per GPU in FLOP/s P_{GPU}, the training time in days T_{days}, and the estimated average utilisation of these GPUs U, typically taken as 33% (Equation 2.16). To exemplify this approximation, Amodei and Hernandez (2018) calculated that whilst 2012's leading image classifier AlexNet (Krizhevsky et al., 2012) had an approximate computational cost of 0.0058pfs-days, five years later the innovative DNN *Xception* (Chollet, 2016) had a cost of 5.0pfs-days (an increase by a factor of 862).

$$Cost(T) = N_{gpu} \cdot P_{gpu} \cdot T_{days} \cdot U \qquad (2.16)$$

Further work has attempted to explicitly quantify the carbon emissions associated with ML systems. Methods such as the *Machine Learning Emissions Calculator* of Lacoste et al. (2019) have been proposed that compute the CO_2-equivalents (CO_2-eq) of an ML system, measuring the environmental detriment caused by the energy expended training a given model, allowing a direct insight into its environmental cost.

This work demonstrates that quantifying the computational, energy, and carbon cost of DNNs is an expanding field within DL research and Green AI, solidifying quantification methods as a vital first step to raising awareness about the ESG impacts of DL, and promoting the use of energy-efficient models.

2.5.2 Efficient Neural Network Architectures

Once the ESG impact of DNNs has been thoroughly quantified, new Green AI systems can be developed that minimise their computational cost. As Schwartz et al. (2019) identify, the cost associated with generating a result from a DNN is proportional to that of processing a single instance I. Hence, much research into Green AI has focussed on how the architecture of DNNs can be redesigned to minimise computational complexity and the energy expended to process data instances.

Quantisation

Due to the publicity surrounding the record-setting performance of complex DNNs, these models are alluring tools for industry players looking to upgrade their workflows (Chartis Research Staff (2019) finding that 44% of financial firms cited "greater accuracy" as the motivation behind adopting ML). However, in many cases, the upper echelons of performance are neither necessary nor beneficial. Kumar et al. (2020) highlight that in resource-constrained applications such as on mobile devices, state-of-the-art DNNs are not feasible as they would quickly engulf the limited power and memory resources. For this reason, Green AI researchers have begun to explore how accuracy-efficiency tradeoffs can be made that produce significant decreases in the energetic cost of DNNs by inflicting slight accuracy drops.

Much of this research, explored in surveys such as those of Xu et al. (2021) and Cai et al.

(2022), has focussed on decreasing computation and memory costs by using lower precision representations of variables and parameters. Typically this is implemented through *quantisation*, where the value of each variable is mapped to discrete quantisation levels that require fewer bits to store. In practice, *deterministic quantisation* is commonly exploited, with approaches ranging from *uniform quantisation*, where floating-point numbers are mapped to their closest low-bit fixed-point representations, to *clustering quantisation*, where parameters are clustered by value and replaced by the mean of their cluster (Xu et al., 2021). Kumar et al. (2020) demonstrate the performance of uniform quantisation by comparing the DNNs of Courbariaux et al. (2015) and Judd et al. (2016). Namely, *BinaryConnect* (Courbariaux et al., 2015) implements an extreme quantisation approach, using only single-bit representations, but inflicts an accuracy drop of 19%; whereas, *Stripes* (Judd et al., 2016) reduce variable representations from 32 to 8-bits whilst maintaining an accuracy drop of only 1%. Hence, quantisation is an incredibly adaptable technique and can be widely used to decrease the computational cost of models contextually based on their desired accuracy-efficiency tradeoffs.

Quantisation-Aware Training

These methods implement post-training quantisation, which improves the efficiency of models after they have been developed. Further research has explored quantisation-aware training, which aims to reduce the performance drop inflicted by quantisation by utilising it throughout training. For example, Fan et al. (2020b) quantised parameters during training to produce a model that achieved an ImageNet top-1 accuracy score of 80.0% (equivalent to 2017's highest accuracy model ResNeXt) using only 3.3MB of memory (only 3% of the memory required by ResNeXt). Furthermore, Cai et al. (2022) assert that this training process can be adapted even further to conduct *low-bit training*, where parameters, activation values, and error gradients are all quantised. *DoReFa-Net* (Zhou et al., 2016) implemented such low-bit representations, using 1-bit parameters, 2-bit activations, and 6-bit gradients to boost training speed whilst generating an accuracy comparable to AlexNet using 32-bit representations. This demonstrates that quantisation and low-bit representations can be effectively used to produce models with significantly lower memory constraints and energy consumption. Research has also been conducted into quantisation for RNNs and LSTMs: Hubara et al. (2016) explored the effectiveness of quantising weights and activations within recurrent networks, whilst He et al. (2016) proposed quantising LSTM gates. However, whilst investigating quantisation for RNNs, Ott et al. (2016) found that low-bit training was not ubiquitously effective in this context; thus, they concluded *mixed-precision training* should be used for RNNs, where weight matrices are quantised but activation values retain higher bit representations.

2.5.3 Efficient Model Training

Whilst the efficiency of a DNN's architecture is essential to minimising the energy expended on each data instance, it is also important to minimise the length of DNN training processes. Hence, a significant amount of research within Green AI focuses on energy-efficient training algorithms that attempt to reduce the number of iterations necessary to train an accurate model.

Transfer Learning

Possibly the most common method of reducing the length of DNN training is using a pre-trained model developed for an analogous task. This method—discussed by Strubell et al. (2019), Walsh et al. (2021), and Schwartz et al. (2019)—is known as *transfer learning*, which uses an existing pre-trained *base model*, and tunes its parameters over a new training dataset that covers a new domain. Since the base model has already learned a general ability to complete a similar task, the new target model requires only a short, inexpensive retraining process, where the model is fine-tuned to accurately capture information from the new domain. Both Strubell et al. (2019) and Walsh et al. (2021) identify that transfer learning significantly reduces the resources required to train a DNN and allows their application to fields with limited amounts of data. For example, Wang et al. (2020) used transfer learning to apply ResNet to the *Stanford Dogs 120* dataset, a small collection of only 20,580 images, making it only 1.7% of the size of the original ImageNet dataset used to train ResNet. The use of transfer learning allowed Wang et al. to develop a model that had an accuracy 11.07% greater than any other DNN developed for this domain and reduced the required training iterations by a factor of 10. This demonstrates that transfer learning is a simple but effective method for reducing the training cost of DNNs, which allows models to be developed for niche domains and minimises the energy required to train accurate ML models.

Initialisation

Instead of directly copying a base model, its parameter values can be used as a starting point from which training a new DNN can begin. This is an example of *initialisation*, which is the process through which the initial values in a network's parameter set are chosen. This is an important part of training, as Xu et al. (2021) found that the rate of convergence of a network's parameters (towards the minimum of the loss function) heavily depends upon their initial values. Furthermore, Hanin and Rolnick (2018) found that primitive techniques for initialising DNNs such as random initiation from a uniform or Gaussian distribution can often lead to slow training, or even result in parameter values never converging.

Both Hanin and Rolnick (2018) and Xu et al. (2021) evaluate alternative methods for initialising a DNN. There are a number of varying ways in which the parameters of an existing model can be used for initialising a new DNN, several of which are explored byHanin and Rolnick (2018) and Xu et al. (2021). *Feature-based initialisation* assigns borrowed parameter values to a subset of the new DNN's parameters and keeps these fixed during training (improving generalisability), whereas *fine-tuning-based initialisation* trains all new and borrowed parameters, providing greater specialisation. Alternatively, *supervised initialisation* pre-trains the new DNN over similar datasets or for analogous tasks, then reuses these low-level representations as a starting point for training high-level features in the target task over the target dataset. For example, Lin et al. (2020) initially pre-trained their language model as a general multilingual translator, before specifically applying the network to translate between language pairs. *Self-supervised initialisation* takes this concept further, allowing pre-training to be conducted without supervised data by learning general representations over unlabelled data (Peters et al., 2018).

Hence, this established research demonstrates the importance of selecting a suitable initialisation method when developing a new DNN based upon the chosen application and architecture, as the

starting point of training can drastically reduce the time and energy expended to generate an accurate model.

Progressive Training

Progressive training (Xu et al., 2021), also known as *greedy layer-wise pre-training* (Xu et al., 2018), advances the aforementioned concept of developing a DNN through a combination of pre-training and tuning. This training approach constructs a model sequentially layer-by-layer by iteratively adding a new layer and tuning its parameters. This builds the DNN in a bottom-up approach, whereby the network first trains lower layers to accurately represent low-level features of the input, then trains higher layers based upon the learned parameters of the preceding layers (keeping the parameters of the previously added layers fixed). After the desired number of layers has been added, all parameters are unfixed and their values tuned in a concluding training phase. In theory, this reduces the computational cost of training a DNN, as training shallow, single-layer networks is significantly simpler and less resource intensive than training full, deep networks (Xu et al., 2021). Furthermore, Xu et al. (2021) assert that the parameters of later layers are optimised faster, as values are updated based on information already known about the features of the input (learned while training previous layers). This approach has been successfully used by implementations such as Yang et al. (2020) to radically speed up training. Specifically, Yang et al. (2020) reduced the total training time required by the language model BERT by over 110% (from 85 to 40 hours) without degrading accuracy.

Progressive training is typically implemented through one of two methods: *supervised layer-wise pre-training* or *unsupervised layer-wise pre-training*. Supervised approaches, such as that of Ienco et al. (2019), train individual layers in a supervised manner; for instance, if building a forecasting model (such as in the case of Xu et al. (2018)), each layer is independently trained to predict the future elements of a sequence. Starting by training a shallow network consisting of only an input layer, one hidden layer, and an output layer, the model iteratively gets deeper; this is achieved by first removing (and saving) the output layer, then fixing the values of all previously trained parameters, adding a new hidden layer, appending the saved output layer, and finally training the parameters of this new layer. Once a network of the desired depth has been constructed, the parameters of all layers are unfixed, and a short training round is conducted over the full network, where the parameters found during pre-training are used as a starting point for deducing the optimal global parameter set.

Unsupervised approaches, such as the implementations of Xu et al. (2018) and Sagheer and Kotb (2019), train each additional layer as an *auto-encoder*. Autoencoders are a type of ANN that learn to output a condensed or encoded representation of its input; they are trained in an unsupervised manner by simply minimising the difference between the input vector and the network's output vector (Lopez Pinaya et al., 2020). In unsupervised layer-wise pre-training, network layers are also built up iteratively; however, each layer is trained as an unsupervised autoencoder, learning to output a representation of the input vector. This unsupervised learning is repeated over the entire pre-training process, resulting in an autoencoder network of the desired depth (Sagheer and Kotb, 2019). During the subsequent tuning process of the full network, the model is then repurposed to the desired task (e.g. forecasting sequence elements). Namely, the autoencoder output layer used

during pre-training is discarded, and a new output layer for the desired task is appended. With all parameters unfixed, the network then undergoes training, starting from the parameter set found during pre-training, and optimising the network's performance in the chosen domain (Sagheer and Kotb, 2019).

Research into layer-wise pre-training commonly centres around improving the efficiency of training recurrent networks, as RNNs have been identified as one of the most sensitive architectures to parameter initialisation. Xu et al. (2018) highlighted that poor training initialisation of RNNs can cause parameters to converge to sub-optimal values, and Ienco et al. (2019) found poorly initialised networks resulted in models with weak generalisability. Fortunately, both supervised and unsupervised layer-wise pre-training have been shown to combat these issues. Ienco et al. (2019) showcase how supervised layer-wise pre-training can be used to produce an RNN-based sequence classifier that exceeded the accuracy of comparable models in a number of applications. Both Xu et al. (2018) and Sagheer and Kotb (2019) demonstrate that unsupervised layer-wise pre-training is effective at improving training efficiency while preserving accuracy. By comparing LSTM models initialised with layer-wise pre-training to a simple randomised initialisation, Xu et al. (2018) exemplified the performance and efficiency benefits of this technique across a number of domains, including image recognition and sequence-to-sequence learning. Xu et al. (2018) found that unsupervised layer-wise pre-training of LSTMs induced faster convergence toward the optimal parameter set, as their initialised values were located close to local minima (of the loss function). They also found that the resultant model exhibited a smaller test error, suggesting this training method improves the generalisability of LSTMs. Similarly, Sagheer and Kotb (2019) compared unsupervised pre-training and random parameter initialisation, focussing on deep LSTMs for time series forecasting. They discovered that the layer-wise method produced better and faster convergence, especially when forecasting collections of correlated variables; this suggests unsupervised layer-wise pre-training is a good fit for forecasting sequences like financial time series that exhibit complex, non-linearly dependent variables.

Hence, progressive training has been widely demonstrated to be effective in decreasing the time and energy required to train DNNs (and in particular RNNs), solidifying it as a useful method within Green AI. Furthermore, the ability of unsupervised pre-training to develop networks that capture the complex dependencies between variables indicates that this method has significant potential to improve the efficiency of the high-performance models used in applications such as financial modelling.

2.5.4   Data Efficiency

A common trend in training state-of-the-art models is the utilisation of vast datasets; Bender et al. (2021) summarise much of this work as following a philosophy of "there's no data like more data". For example, the cutting-edge image classification model of Mahajan et al. (2018) produced record-breaking accuracy in its field, but achieved this through training over a dataset of 3.5 billion images, three orders of magnitude larger than the *Open Images Dataset* (of 9 million instances) typically employed by leading models in this field. Both Schwartz et al. (2019) and Bender et al. (2021) assert that these extreme datasets are a major problem with Red AI, highlighting that dataset size is a significant contributor to the computational and energetic cost of a model. Therefore,

some Green AI research has attempted to reduce the data requirements of training DNNs, aiming to minimise memory requirements for low-resource applications, and reduce the energy expended training DNNs.

Inefficient Use of Data

Both Bender et al. (2021) and Walsh et al. (2021) highlight how a significant portion of the energy used to train DL models is due to the inefficient use of data. Bender et al. (2021) assert that much cutting-edge research opts to naively feed expansive collections of data into their model, most of which is not beneficial to learning. They suggest that more time should be put into carefully curating specialised datasets for each model and domain, minimising the time and energy wasted on unhelpful data instances and instead using data more intelligently. Al-Jarrah et al. (2015) further note that current ML systems are not intelligent enough to efficiently deal with significantly increased data loads. They highlight that commonly used optimisation methods (such as gradient descent) require substantial computational work to locate global optima; thus, when applied to large datasets, these methods accumulate an extreme computational cost.

Furthermore, there can often be a conflict introduced between the size of architecture used and the amount of training data required, as small ANNs typically require more data to achieve comparable accuracy to deeper networks. Namely, Bender et al. (2021) found that models such as *ALBERT* (Lan et al., 2019) that attempt to recreate the performance of high-complexity networks using a smaller parameter set typically rely upon large amounts of data to force advances in accuracy during training beyond what would typically be possible with compact networks. This demonstrates that when considering the energy consumption of DNNs, a balance must be drawn between model size and dataset size, as both contribute to a model's computational cost (as shown in Equation 2.15).

Reducing Data Requirements

Several avenues have been proposed within Green AI for reducing the amount of training data needed by DNNs while preserving accurate performance. Xu et al. (2021) discuss a variety of dataset reduction techniques; these methods typically rely on pre-trained models analogous to those described in Section 2.5.3. Namely, pre-trained models can exploit self-supervised learning to initialise near-optimal parameters without needing labelled data. This reduces data requirements as energy does not need to be expended labelling the data required for pre-training, and the initialised model converges faster downstream during the full training stage (Xu et al., 2021). *Contrastive learning* (Chen et al., 2020) is raised by Xu et al. (2021) as a commonly used pre-training method for reducing data requirements in CV. This approach focuses on learning pairwise relationships between data instances, representing similar pairs close together in the data space and pushing divergent pairs far apart, thus allowing the dataset to be constricted. Xu et al. (2021) further highlight the effectiveness of *prompt learning* (Liu et al., 2021), where data instances are labelled with a discrete task-specific template known as a *prompt*, which can improve the efficiency of data sampling as a single prompt can represent up to 100 individual data instances.

Active Learning

Possibly the most popular and widely explored method for reducing the data required to train DNNs is *active learning*. This method, whose utility to DL has been extensively explored by surveys such as that of Ren et al. (2020), aims to reduce training costs by selecting a subset of data instances from the full data space that are believed to provide the most utility to the model's learning process.

Active learning directly adapts the training process of a DNN, moving away from the traditional approach of iteratively training a network over the full dataset D of N instances. Instead, active learning approaches split the training data D into smaller chunks, utilising it as either a *stream* or a *pool*. *Stream-based active learning* individually picks data instances from D and evaluates whether this instance is useful for training (in which case it is fed into the DNN) or not (Ren et al., 2020). More commonly, *pool-based active learning* is implemented; contrary to the stream-based approach, at each training iteration, this method selects a pool of the n_{sample} most useful instances from the full dataset, and trains the network over these (Ren et al., 2020). Specifically, pool-based approaches divide D into the pool set P and validation set V; at the start of training, the pool set $P^{(0)}$ is empty, and the validation set V_0 contains all elements of the full dataset D (i.e. $d \in D \implies d \in V^{(0)}$). To begin training, an initial pool $P^{(0)}$ of n_{sample} data instances (known as the seed) must be sampled from the full dataset; most simplistically, these are drawn at random. These sampled values are then moved from the initial validation set $V^{(0)}$ to the new pool; hence we have the new datasets $P^{(1)}$ (where $|P^{(1)}| = n_{sample}$) and $V^{(1)}$ (where $|V^{(1)}| = N - n_{sample}$). After the seed has been selected, the first round of training can be conducted, where the DNN is trained over pool $P^{(1)}$. This partly trained DNN is then used to predict the labels of the remaining $N - n_{sample}$ data points in the validation set; these labels are fed into an *importance function* scoring each instance in $V^{(1)}$ according to its utility to the learning process of the DNN. The n_{sample} instances with the highest importance score are then selected from the validation set and moved into the pool; thus, we have the new datasets $P^{(2)}$ and $V^{(2)}$ where $|P^{(2)}| = 2n_{sample}$ and $|V^{(2)}| = N - 2n_{sample}$. This process is repeated over multiple iterations of training, evaluating the importance function and expanding the pool set, until either a dataset size, iteration count, or accuracy threshold has been reached (Ren et al., 2020).

Xu et al. (2021) explain that active learning algorithms are typically categorised through the specific importance function they rely upon. *Uncertainty-based active learning* selects new instances to add to the pool based upon the uncertainty of the predicted labels assigned by the partly trained DNN. Alternatively, *expected-model-change-based active learning* selects data instances from the validation set that are expected to cause the largest change to the parameters or output vector of the DNN.

Both Ren et al. (2020) and Xu et al. (2021) assert that this training algorithm drastically increases the data efficiency of DNN training. Through analysing a spectrum of applications, Ren et al. (2020) found that active learning can theoretically achieve an exponential improvement in the amount of time required to label training data, as well as being able to successfully deal with high-dimensional data and train efficient but accurate models for sequential data. Xu et al. (2021) similarly assert that active learning significantly reduces the time and energy wasted on redundant data instances that do not benefit the DNN's learning. Furthermore, Ren et al. (2020) note that

the resources required to evaluate the importance function does not undermine the efficiency gains provided by the iterative learning process, as this function can be efficiently implemented to inflict a negligible computational cost.

Hence, previous research has demonstrated the utility of active learning in reducing the data constraints of training DNNs, and thus this method is of vital importance to work within Green AI. Furthermore, whilst active learning over time series introduces further complexity due to the temporal nature of variables, approaches such as Peng et al. (2017) and Zimmer et al. (2018) have demonstrated its feasibility in this field, meaning there is significant potential for active learning to help improve the data and energy efficiency of high-performance sequential models such as those used within volatility forecasting.

Chapter 3

Methodology

3.1 Introduction to Experiments

This study investigates how energy and data-efficient Green AI methods can be applied to DL for finance. Specifically, this research will explore how DNNs for financial volatility forecasting over the S&P 500 market can be trained that deliver high accuracy whilst prioritising efficiency, in an attempt to demonstrate how the resource requirements, energy expenditure, and environmental impacts of Fintech can be reduced. Hence, this experimentation highlights the importance of considering the ESG impacts of DL, and demonstrates that high-performance DNNs can be utilised in this domain without sacrificing the SDGs of sustainable finance. Additionally, this study intends to illustrate how these efficient methods can be applied outside of the the existing research applications of NLP, CV, and mobile computing, expanding the scope and impact of Green AI.

To illustrate the benefit of efficient DL to Fintech, two core research hypotheses are tested. Firstly, this study explores the question: *can using Green AI methods reduce the training time required to develop an accurate volatility forecasting model?* Experimentation into the hypothesis that Green AI methods can improve training efficiency aims to fill the research gap between the goals of sustainable finance and the expanding use of DL in Fintech, demonstrating how the energetic cost (and thus carbon emissions) of DL can be minimised. Secondly, this research examines the further question: *can Green AI methods reduce the data requirements of training an accurate volatility forecasting model?* This extended study explores the hypothesis that Green AI not only facilitates a direct reduction in training time and energy consumption, but additionally provides a way of negating the reliance of DL on broad, expansive datasets. By minimising data requirements, the cost of training DNNs can further be reduced, as less energy is expended labelling training data and learning over redundant instances that do not benefit performance (Schwartz et al., 2019). Furthermore, the use of smaller datasets improves the inclusivity of this field, as researchers and market players do not need to invest in expansive computer memory or cloud storage to train accurate volatility models (Strubell et al., 2019). Hence, the exploration of both hypotheses contributes to the field in several ways: expanding the applications of Green AI, reducing the environmental impact of Fintech, and improving the inclusivity of finance.

To investigate these research hypotheses, the following experimentation is divided into three core studies. Initially, the dataset over which volatilty forecasting is conducted is presented in Section 3.2, and the metrics used to analyse the accuracy and efficiency of the developed DL models and

algorithms are outlined in Section 3.3. Following this, the baseline model at the heart of this research is presented in Section 3.4; this is an LSTM-based financial market volatility forecasting DNN. The underlying methods utilised to build this model, the motivations behind their use, and the relation of this model to typical DNNs implemented within computational finance are all explored. Following this, Section 3.5 explores the Green AI methods implemented that attempt to improve the energy efficiency of the baseline model training process. A comprehensive explanation of these methods and their utility is given, exploring their potential to reducing the computational cost of DNN training. Finally, Section 3.6 explores the minimisation of data requirements through Green AI methods; an adapted training process is presented that aims to decrease the size model's training dataset, hence facilitating further computational cost, energy consumption, and carbon emission reductions.

3.2   Dataset

To evaluate the use of Green AI in finance, the specific domain of financial market volatility forecasting was chosen, as it is currently one of the most popular applications of DL within computational finance. This is demonstrated through the plethora of surveys and research papers published examining the use of DL for volatility forecasting, such as the extended review of Ge et al. (2022); in fact, volatility forecasting is within the top five most popular uses of DL within financial research (Berat Sezer et al., 2019). Furthermore, Chartis Research Staff (2019) demonstrated that this domain is a popular application of DL within the financial industry, finding that 70% of financial institutions use ML for risk analysis, most commonly for forecasting "market risk". This extensive use is primarily due to the superior accuracy provided by DL methods: Chartis Research Staff (2019) found that 44% of financial firms are motivated to use DL by the promise of "greater accuracy", and a multitude of recent work has demonstrated that DL outperforms traditional volatility forecasting methods (Rodikov and Antulov-Fantulin, 2022). For this reason, it is clear that volatility forecasting is a characteristic example of the use of DL within finance; hence, showing the applicability and benefit of Green AI within this domain acts as an exemplar of how the energy consumption of DL for finance in general can be reduced, and the ESG impacts of Fintech mitigated.
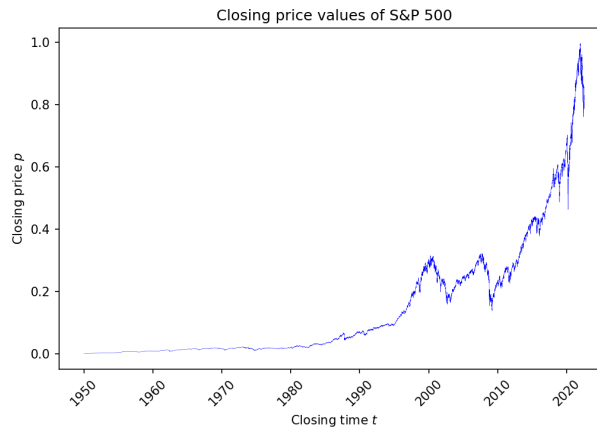

Figure 3.1: Time series of closing prices of the S&P 500 market used within the training dataset.

In their analysis of 35 research papers, Ge et al. (2022) found that DNNs were most commonly

used for forecasting the volatility of the S&P 500 market (accounting for 12 of the 35 works), primarily due to the accessibility of data. The *Standard & Poor's 500 Index* (S&P 500) is a stock market index capturing the price movements of 500 leading US companies. It is often regarded as one of the best indicators of the performance of the global stock market; for this reason, it is unsurprising that S&P 500 data has become a popular target for time series modelling (Thakkar and Chaudhari, 2021). Therefore, to ensure universal results charcateristic of this domain, S&P 500 data was chosen as the specific modelling domain over which financial market forecasting was conducted within this research.
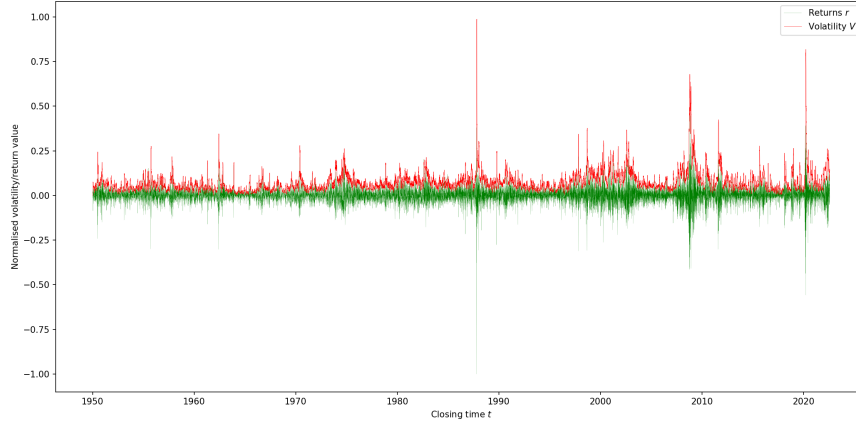
Figure 3.2: Time series of daily returns of prices over the S&P 500 market, overlaid with the computed historical volatility time series for the same period.

To enable sequence learning, the S&P 500 data is represented as a multivariate time series consisting of 18261 daily sampled timesteps between the fifth of January 1950 (1950–01–05) and the first of August 2022 (2022–08–01). The time series of closing prices over this period is shown in Figure 3.1. This period and sampling frequency were specifically chosen as daily data strikes an effective compromise between maintaining a reasonable dataset size and having a high enough frequency of data instances to permit detailed analysis (Rodikov and Antulov-Fantulin, 2022). Each timestep maintains the value of seven financial variables of the S&P 500 market: the daily *open*, *close*, *high*, and *low* prices, trading *volume*, associated *return*, and market *volatility* (Table 3.1). Prices are taken from the S&P 500 market and used to compute the return r_t as the logarithmic change in closing price from the previous day (Equation 2.13). From this, the historical volatility is then calculated as the standard deviation σ_t^2 of returns over the past 10 days; namely, σ_t^2 is computed from the time series $r_{\tau_1}, \ldots, r_{\tau_2}$ over the time interval $\tau_1 \to \tau_2$ where τ_2 is the current timestep t and τ_1 is 9 timesteps in the past. The HV is specifically calculated through Equation 2.14, iteratively constructing the volatility time series $v_0, \ldots, v_t$; this echoes the approach of Lahmiri (2017) who also developed an ANN for volatility forecasting. The time series of returns and volatilites over the full period is showcased in Figure 3.2. Historical volatility was chosen as the metric to be forecast as it was identified by Ge et al. (2022) as the most commonly explored way of measuring market volatility. Furthermore, HV has been utilised in many innovative explorations of this field, such as Rahimikia and Poon (2020) and Rodikov and Antulov-Fantulin (2022) who both demonstrated the exceptional forecasting accuracy of DNNs for this metric.

The implemented DNNs use the volatility time series $v_0, \ldots, v_t$ to forecast the daily historical

Timestep	Open	Close	High	Low	Volume	Return	Volatility
2022-08-01	0.855942	0.855188	0.856735	0.853413	0.309027	-0.025801	0.140913
2022-07-29	0.850728	0.857619	0.855739	0.849899	0.333186	0.128754	0.145031
2022-07-28	0.837990	0.845556	0.843038	0.831855	0.338870	0.110068	0.147466
2022-07-27	0.822442	0.835378	0.834864	0.823165	0.312798	0.235645	0.129777
2022-07-26	0.822814	0.813996	0.816945	0.814652	0.269089	-0.105961	0.126826

Table 3.1: The final 5 timesteps of the multivariate time series (in reverse chronological order).

volatility of the S&P 500 at a single future timestep $t + 1$ (generating the prediction value $\hat{v}_{t+1}$). To train and test the model, the full time series is broken down into individual sequences of 10 timesteps. Namely, each data instance is a small subsequence of the entire time series, comprised of 10 consecutive timesteps, where each timestep has 7 variables associated with it (Table 3.1). Hence, each data instance in the full dataset is in essence 7 small time series; one for the closing prices, returns, historical volatilities, and so on. Each of these variables was normalised through min-max scaling, restricting their values to the range 0–1 (except the returns, which were allowed to fluctuate between -1 and 1 to permit negative values); this scaling approach was similarly used by Rodikov and Antulov-Fantulin (2022), who highlighted that normalisation is essential to a stable and efficient training process.
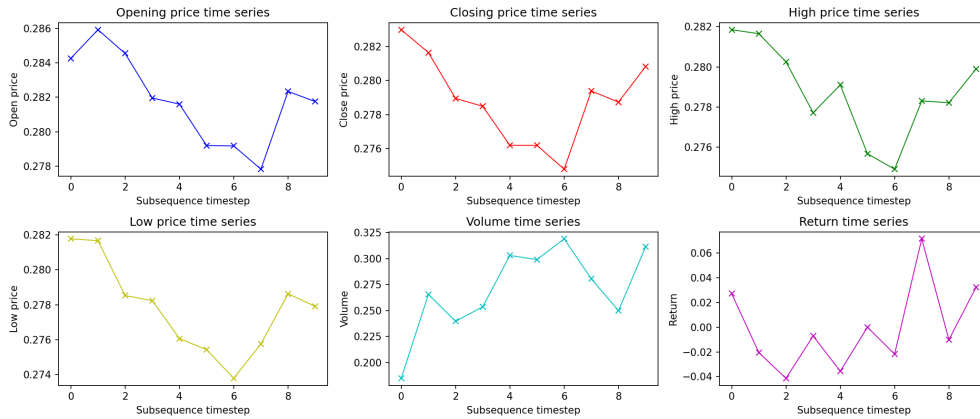
Figure 3.3: Subsequences comprising the multivariate time windowed time series for an example instance from the training dataset

The sequence learning task was then set up as a sequence-to-sequence problem, where each multivariate 10-step time series is input into the DNN, which output the predicted successor $\hat{y}_t = \hat{v}_{t+1}$. Thus, each pass through the model predicts the volatility of the S&P 500 market at timestep $t + 1$ through analysing the preceding multivariate time series over steps $(t - 9) \rightarrow t$ (i.e. the previous 10 closing prices, returns, volatilities, etc.). This is shown through Figure 3.3, which (for a randomly selected data instance) gives an example of subsequences that make up each multivariate 10-step time series, and Figure 3.4 demonstrating a volatility subsequence with the true label v_{t+1} to be predicted by the model.

This multivariate, time windowed approach was chosen as it has been effectively used by a host of popular studies on the use of DNNs for time series prediction, and in particular volatility forecasting.
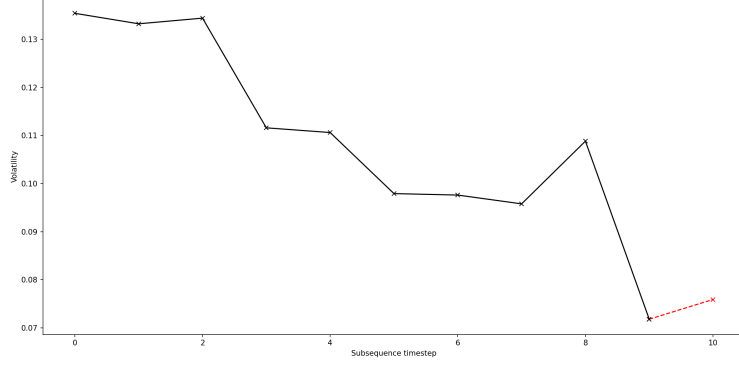
Figure 3.4: Volatility subsequence for an example data instance, including the true label to be predicted

For example, Xiong et al. (2015b) used 25-dimensional multivariate time series (incorporating daily open, close, high, and low prices, and returns) to demonstrate the prediction accuracy and robustness of their RNN-based model for the S&P 500. Furthermore, Xiong et al. (2015b), Bucci (2020), and Rodikov and Antulov-Fantulin (2022) all utilised a dataset consisting of time windowed subsequences; Xiong et al. (2015b) specifically made predictions using a 10-step rolling time window, as they found this to be consistent with the majority of comparable models in this field. Hence, a dataset has been developed in keeping with popular modelling implementations within this field.

3.3 Analysis Approach

To demonstrate the efficiency of the adapted training algorithms, and their ability to maintain model performance, each fully trained DNN was tested over the testing dataset D_{test}. This test set is constructed by implementing a train-test split of the full dataset D (of multivariate, time windowed sequences of S&P 500 data), where D is partitioned into the two distinct subsets D_{train} and D_{test} used for training and testing. In this research, an $80\colon 20$ split is implemented, using 80% of data instances for training and the remaining 20% for testing (determined experimentally to produce the best results); this split is demonstrated for the full (unwindowed) volatility time series in Figure 3.5.

3.3.1 Accuracy Metrics

After training, the DNNs predictions over D_{test} are analysed through a collection of statistical metrics that objectively quantify the modelling error ε and prediction accuracy. Namely, accuracy metrics are used to explicitly measure the divergence between the predicted volatility time series value $\hat{y}_t = \hat{v}_{t+1}$ and the true label $y_t = v_{t+1}$ enumerating the actual recorded HV at that future timestep. These metrics are important as they not only demonstrate the success of the baseline model, but will later be used to contextualise the effect of the implemented energy and data-efficient adaptations, highlighting their impact on performance.

One metric typically used by volatility forecasting models is the *coefficient of determination R^2* (Zhang et al., 2022), which computes the ratio between the estimated variance of the prediction error ε and the variance of the variable y_t being predicted, shown in Equation 3.1 for the predictions

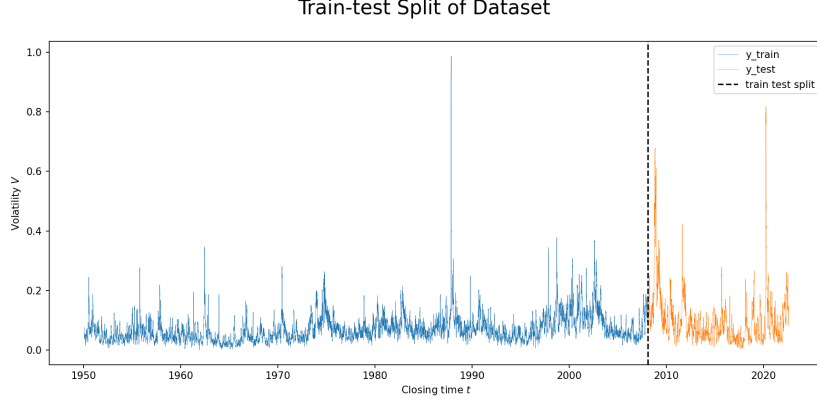Figure 3.5: Visualisation of the 80 : 20 train-test split dividing training instances (blue) and testing instances (orange)

$\hat{y}_t$ (and the sample mean of labels $\mu^{(y)}$). The value of R^2 for a given set of predictions represents the proportion of the variability in the label set that is correctly captured by the model's predictions. This value ranges between $-\infty$ and 1, with a score of 1 meaning the true labels have been perfectly represented. The coefficient of determination is an effective metric for demonstrating how well a model represents data, and has been used to showcase the performance of several volatility models such as the ANN implementation of Zhang et al. (2022). However, it also has several limitations. Most notably, this metric doesn't explicitly quantify whether a model is good or bad, it only shows whether the testing performance is noticeably better than that of a comparison *constant model* that outputs predictions simply as the sample mean of the observed inputs; hence, its results can sometimes be misleading.

$$R^2 = 1 - \frac{\sum_{t=0}^{n}(y_t - \hat{y}_t)^2}{\sum_{t=0}^{n}(y_t - \mu^{(y)}))^2} \tag{3.1}$$

Because of this limitation, several other statistical metrics are commonly used to quantify the prediction accuracy of DNNs. Two such measures are the *mean absolute error* (MAE) and *root mean squared error* (RMSE). Given the set of predictions $\hat{y}_t$ and labels y_t, MAE measures the average magnitude of the prediction error ε, taken as the mean absolute difference between $\hat{y}_t$ and y_t over the entire test set (Equation 3.2). RMSE also measures the magnitude of deviations, computing this as the standard deviation of prediction errors (Equation 3.3). Low MAE and RMSE values (close to 0) both indicate that the model is accurate; however, the squared component of RMSE gives increasing weight to large errors and hence is useful for penalising highly anomalous predictions more than slight deviations, whereas MAE gives a uniform representation of ε. One limitation of both MAE and RMSE is that they are scale-dependent; when predicting small values the magnitude of the computed ε will always be smaller than that generated predicting large values, irrespective of the model's accuracy. Hence, these metrics are typically utilised to compare various models over the same domain, such as by Rodikov and Antulov-Fantulin (2022) who used both MAE and RMSE to compare the performance of LSTMs and GARCH. The MAE and RMSE accuracy metrics will be used similarly in the research of this thesis, comparing the performance of the baseline DNN to the proposed DNNs using energy and data-efficient training processes.

$$MAE = \frac{1}{n}\sum_{t=0}^{n}|y_t - \hat{y}_t| \tag{3.2}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=0}^{n}(y_t - \hat{y}_t)^2} \tag{3.3}$$

To further quantify the accuracy of the implemented models, and address the inherent issues of preceding measurements, a scale-independent measure is required. One such metric is the *mean absolute percentage error* (MAPE) of predictions, which computes the absolute difference between $\hat{y}_t$ and y_t over the test set as a percentage average (Equation 3.4). This provides an objective and easily interpretable picture of the accuracy of a model and has been used to analyse numerous volatility forecasting DNNs, including by Xiong et al. (2015b) and Zhang et al. (2022). Hence, this research additionally computes the MAPE of each implementation to independently demonstrate the performance of the baseline DNN and efficient adaptations.

$$MAPE = \frac{1}{n}\sum_{t=0}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| * 100\% \tag{3.4}$$

3.3.2 Efficiency Metrics

Beyond generating accurate performance, the efficiency of the implementations is of vital importance to this research. An informative efficiency metric is required to demonstrate the computational cost of DNNs, and showcase the reduction facilitated by Green AI. Several efficiency metrics have been presented by Schwartz et al. (2019), who assert that the computational cost of a DNN is proportional to the cost of passing a single data instance through the network, the size of the utilised datasets, and the number of hyperparameters being tuned (Equation 2.15). Since the architecture used in this research is fixed, and the hyperparameters predetermined, these variables are not covered by the efficiency calculations analysed; however, dataset size is an important aspect of the efficiency of training a DNN, explored later in Section 3.6.

Amodei and Hernandez (2018) also explore the efficiency of DNNs, asserting that the cost of training a given network is dependent on the hardware through which this process is implemented, and the training time T_{days} (Equation 2.16). As the implemented experimentation all relies upon the same hardware, metrics analysing hardware efficiency are also redundant to our study. However, training time is an important aspect of the cost of training, and hence to quantify energy efficiencym the total time required to train a DNN with accurate performance is recorded. Training time is a simple but effective measure for highlighting the efficiency of a program, as more time typically means a higher computational cost. Although it is important to note that training time is not a completely representative metric, as it is reliant upon the underlying hardware and software dependencies of the system (Schwartz et al., 2019). However, since this efficiency metric is used for comparison of the relative differences in computational cost between implementations (all of which use the same underlying hardware and software), the calculation of training time is both valid and beneficial in this context.

The efficiency of a training algorithm can additionally be determined by inspecting how fast

and smooth the minimum of the loss function is converged upon. As outlined in Section 2.1.2, a DNN is trained by adjusting the network's parameter set in the direction of the negative gradient of the loss function, determined through backpropagation of the DNN's prediction error. Hence, the purpose of DNN training is to find the optimal parameter set that corresponds to the minimum of that loss function, meaning the DNN has an optimally low prediction error. The efficiency of a DL training process can therefore be determined by the speed and smoothness at which this minimum point is approached; this is known as the convergence rate of training, which calculates the size of the step taken towards the loss function's minimum by each epoch. Convergence rate is an important factor when considering the efficiency of a DL training algorithm, as the faster a DNN's parameters converge towards their optimal values, the more effective the early stages of training (as the DNN learns more over each epoch), and thus the less time, fewer calculations and data evaluations, and lower energy consumption is required to develop an accurate model. Hence, convergence rate is also computed to demonstrate the efficiency of each training implementation.

3.4 Baseline Volatility Model Implementation

3.4.1 Aims of Study

Before analysing the benefit of energy and data-efficient training adaptations, a baseline model that is representative of the chosen domain must be implemented. As the aim of this thesis is to demonstrate the utility of Green AI to Fintech, the baseline model must be a characteristic example of DNNs used in finance. This should ensure that the findings of this research are universal, demonstrating on a wide scale how the ESG impacts of DL for finance can be mitigated. With this in mind, a common application of DL within finance must be chosen, and a DNN-based model used that has an architecture most typical of existing approaches developed for this domain. The resource requirements of this model can then be analysed to demonstrate the typical computational, energetic, and carbon cost of DL models for finance. This research should give a specific example of the environmental impact of DL highlighted by Schwartz et al. (2019) and Strubell et al. (2019), and reinforce the concerning energy consumption and carbon emission figures presented by Hockstad and Hanel (2018) and Masanet et al. (2020).

3.4.2 Model Approach

To conduct volatility forecasting over the chosen domain, an LSTM architecture was chosen as RNNs have been shown to produce superior accuracy when forecasting time series (Fjellström, 2022). These architectures have been shown to effectively model both long and short-term dependencies in sequences, and capture complex dependencies between highly variable features. Furthermore, they have been successfully applied to forecasting financial market volatility, with Bucci (2020) showing that LSTMs outperform other traditional methods for forecasting RV over S&P 500 data. Hence, to act as a baseline model that is characteristic of this domain, an LSTM architecture was chosen.

Devising a specific configuration of hyperparameters for the LSTM model was similarly done by surveying the literature base, and constructing a hyperparameter set that was representative of typical models in this field. The hyperparameters of the model that were determined in this

manner are as follows: *number of layers*, *neurons per layer*, *learning rate*, *epochs*, and *batch size*. First, existing models were used to determine the number of LSTM layers to implement within the deep model. The reviewed DNNs were found to typically exploit between 1 (Bucci, 2020) and 5 (Kim and Won, 2018) layers; therefore, to strike a balance between these implementations, the model proposed in this thesis relies on 3 LSTM layers, each containing 24 neurons (Table 3.2). Since the output layer is simply used to return the predicted volatility $\hat{v}_{t+1}$, this layer consists of a single neuron fully connected to its preceding hidden layer, from which the network's single output value is read.

Layer	Type	Output Shape	Parameter Count
Input layer	Vector	$(batch\_size, 10, 7)$	-
Hidden layer 1	LSTM	$(batch\_size, 10, 24)$	3072
Hidden layer 2	LSTM	$(batch\_size, 10, 24)$	4704
Hidden layer 2	LSTM	$(batch\_size, 10, 24)$	4704
Output layer	FCN	$(batch\_size, 1)$	25

Table 3.2: Architecture of Baseline LSTM Model

To train the proposed LSTM model, the learning rate, epoch count, and batch size hyperparameters must also be determined. The surveyed models typically exhibited a learning rate of 0.001 (Zhang et al., 2022), training processes consisting of between 50 (Rahimikia and Poon, 2020) and 600 (Xiong et al., 2015b) epochs, and exploited a batch size of 32 (Xiong et al., 2015b). Hence, to implement a model that was both characteristic of these statistics and experimentally determined to produce the best performance, a learning rate of 0.001, epoch count of 100, and batch size of 32 were chosen (Table 3.3).

Window Size	Time Series Variables	Prediction Length	Layers	Neurons	Learning Rate	Epochs	Batch Size
10	7	1	3	24	0.001	100	32

Table 3.3: Hyperparameters of Baseline LSTM Model

3.4.3 Summary

Hence, a financial volatility forecasting model was developed over the S&P 500 market, which exploited a 3-layer LSTM architecture to generate accurate predictions over this sequence learning task. This model is trained over 100 epochs, recording the efficiency of training to give an insight into the computational cost of traditional DL algorithms, and set a baseline performance and efficiency level to be compared to the subsequently discussed energy and data-efficient algorithms.

3.5 Energy-Efficient Training Extensions

3.5.1 Aims of Study

Building upon the baseline approach and model, energy-efficient Green AI methods are introduced to the training process, aiming to minimise the energy expended to train an accurate DNN for volatility forecastinge. This research focuses on three avenues through which the training process of a DNN can be adapted to improve energy efficiency, all of which have been explored in literature surveying Green AI (Xu et al., 2021): mixed-precision training, supervised layer-wise pre-training, and unsupervised layer-wise pre-training. Initially, mixed-precision training will be explored, where network computations are quantised to utilise low-bit representations. This exploration aims to demonstrate how using lower precision data and variables can boost training speed whilst still producing an accurate model, thus showing that mixed-precision training is a viable mechanism for reducing the energetic cost of developing DNNs in finance. The experimentation then proceeds to explore progressive training methods; this begins by implementing a supervised pre-training process, that additionally aims to reduce traiing time and energy consumption. Unsupervised pre-training is then explored, where a more complex approach is taken to pre-train the proposed DNN to produce further computational cost reductions.

3.5.2 Mixed-Precision Training Approach

The first method presented by this research for reducing the energy consumption of DL is mixed-precision training. As explored in Section 2.5.2, quantisation is often used within Green AI to reduce resource requirements. These implementations reduce energy and memory costs by lowering the precision at which variables are stored and operations are evaluated (Fan et al., 2020b). In real-world applications, quantisation is often realised through mixed-precision training, where a combination of high-precision and quantised representations are used to balance efficiency and accuracy (Ott et al., 2016). This research aims to demonstrate how mixed-precision training can be used to reduce the computational and memory cost of training DNNs, mitigating the energy consumption and ESG impacts of DL in finance.

To implement mixed-precision training, the *Keras mixed-precision API* (Abadi et al., 2016) was exploited. This API enables ML developers to adapt the precision of variables and operations within an implemented DNN, changing the default 32-bit precision. In this research, the mixed precision API is used to specify a *'mixed_float16'* precision policy (Abadi et al., 2016), where a combination of 16-bit and 32-bit floating-point data types are used during training. Specifically, this exploits 16-bit floating-point variables for evaluating network computations, whilst maintaining a 32-bit floating-point representation for storage. Using lower precision reduces the amount of memory used by network operations, increasing the speed of calculations and reading from memory by minimising their computational load and allowing increased hardware acceleration. Specifically, 16-bit representations are utilised as modern GPUs and TPUs have been shown to run operations significantly faster when using 16-bit precision, as batches of data consume half the amount of memory when being passed through the DNN. High precision is still used for storing variables to maintain numerical stability, ensuring that the model maintains accurate performance. Hence, this implementation of mixed-precision training strikes a compromise between reducing the energetic

cost of network operations during training and preserving accurate performance.

The efficiency benefits provided by mixed-precision training were evaluated by using the same DNN as proposed in Section 3.4. First, the use of a 'mixed_float16' precision policy is specified; an identical network with the same LSTM-based architecture and hyperparameters of the baseline model (Tables 3.2 and 3.3) is then implemented using variables of the specified precision. This DNN is trained using the same training algorithm as in the baseline case, for the same number of epochs.

3.5.3 Supervised Layer-wise Pre-Training Approach

Pre-training is also a popular approach within Green AI, which aims accurate models to be developed more intelligently and efficiently. Progressive training is one such pre-training method, highlighted by Green AI researchers (such as Xu et al. (2021)) as being capable of reducing the length of DNN training through a computationally inexpensive pre-training phase. As discussed in Section 2.5.3, progressive training constructs a DNN layer-by-layer through an iterative process of adding new layers and training each individually. Research into this approach—such as the exploration of Ienco et al. (2019)—has shown its effectiveness at reducing the computational cost of training a DNN, as the inexpensive pre-training phase (which is computationally cheap since only single network layers are trained) allows a significant reduction in the number of epochs necessary to optimise the parameters of the full network. Both the low-cost pre-training phase and shortened full network training mean that progressive training is an effective way to reduce training time and energy consumption, providing a promising avenue for improving the ESG impact of DL in finance.

In this section of research, supervised layer-wise pre-training will be explored to demonstrate the benefits of progressive training. This method trains network layers individually using supervised learning over a labelled dataset of the desired modelling domain and task (Ienco et al., 2019). In our case, the time series dataset outlined in Section 3.2 is used, with each round of pre-training optimising a new layer's parameters to predict the HV $\hat{v}_{t+1}$ at timestep $t+1$ within a sequence. In this research, the base model at the start of pre-training consists of a single LSTM layer combined with a fully connected single-neuron output layer. To execute the training algorithm, each pre-training round (where a single layer is added and optimised) was conducted over 20 epochs. A further 30 epochs were then used to conclude training, where the parameters of the full network are tuned. This proportion of pre-training and tuning epochs was chosen to provide the best accuracy-efficiency tradeoff, minimising the total length of training but maintaining the model's forecasting performance.

The first pre-training round focuses on training the base network. This shallow network is trained over the dataset D_{train}, optimising the parameters of the single LSTM layer and the DNN's output layer. After 20 epochs, the parameters of the first layer are fixed, a new LSTM layer is added between the initial hidden layer and the output layer, and another 20 pre-training epochs are executed. This iterative process was repeated until a DNN with 3 LSTM layers was constructed, meaning this experimentation utilised the same architecture as in the baseline case. After the full network is pre-trained, final tuning begins, where all network parameters are unfixed and optimised over 20 additional epochs, starting from the initialisation values deduced during pre-training.

3.5.4   Unsupervised Layer-wise Pre-Training Approach

Although supervised layer-wise pretraining is a simple method through which progressive training can be realised, greater performance and efficiency can be achieved through using unsupervised layer-wise pre-training. Both Xu et al. (2018) and Sagheer and Kotb (2019) found this approach induced faster convergence toward the optimal parameter set and smaller test error than comparable LSTMs using traditional training algorithms. Hence, in an attempt to further improve the efficiency of DNN training in the chosen domain, an unsupervised layer-wise pre-training algorithm is also implemented.

Unsupervised layer-wise pre-training begins by initialising a similar single-layer LSTM base network. The base architecture was identical to the previously exploited base network, except a different output layer was utilised to enable unsupervised learning. The new output layer consisted of the same dimensions as the input layer such that the network could be trained as an autoencoder. Since the same training dataset of multivariate time series subsequences was utilised, the unsupervised pre-training approach trained each layer to reconstruct this subsequence, meaning the network learned to output a representation of the input time series (tuning its parameters to minimise the difference between each layer's inputs and outputs). Layer-wise pre-training was then implemnted, optimising each LSTM layer independently as an autoencoder over 7 epochs to result in a 3-layer DNN. The concluding full network tuning stage over 50 epochs then converts this autoencoder into a forecasting model that predicts the volatility $\hat{v}_{t+1}$ based upon each input time series. To achieve this, the autoencoder output layer is discarded, and a new output layer is appended that consists of a single fully-connected neuron, which is optimised (alongside the three hidden layers) over the tuning stage.

3.5.5   Summary

Hence, to demonstrate how Green AI can improve the energy efficiency of training DL models in finance, three algorithms were implemented: mixed-precision training, supervised layer-wise pre-training, and unsupervised layer-wise pre-training. To evaluate the success of each method, a DNN with the same architecture as presented in Section 3.4.2 was trained, recording the efficiency of the training process and the accuracy of the resulting model. This involved the calculation of the R^2, MAE, RMSE, and MAPE of test predictions, to deduce the accuracy of each developed model, and quantify the performance difference to the baseline model. The total training time and convergence rate were also recorded over the execution of each algorithm, which are then put into context against those computed for the baseline model, demonstrating the efficiency gains of the new model, and exemplifying the benefit of the adapted training algorithms for reducing the energy consumption, carbon emissions, and ESG impacts of DL-based financial volatility forecasting models.

3.6   Data-Efficient Training Extensions

3.6.1   Aims of Study

A core component of the computational cost of state-of-the-art DL is the recent trend of utilising extremely large datasets (Schwartz et al., 2019). This approach forces performance gains through

training a model over an expansive dataset, most of which is not beneficial to learning (Bender et al., 2021). Whilst this has been shown to facilitate impressive accuracy, current DNNs typically do not deal with data intelligently or efficiently (Al-Jarrah et al., 2015); this is because a significant amount of time and energy is wasted training over redundant data instances that do not benefit performance, accruing an alarming energetic cost.

Therefore, to further improve the energy efficiency of DNNs, this research focuses on the use of data-efficient training algorithms that minimise training data requirements and thus reduce memory and energy consumption. Specifically, this is achieved through adapting the training process to intelligently select (and train over) data instances that are the most beneficial to learning, discarding redundant instances. Thus, the size of the total dataset trained over is reduced, lowering the time and energy expended over training.

Furthermore, Bender et al. (2021) showed that compact networks with small parameter sets typically require increased amounts of training data to achieve comparable performance to larger models. Thus, the benefits of intelligent data-efficient training additionally aim to mitigate this requirement, allowing compact networks to be used without inflicting the further computational costs associated with large datasets.

3.6.2 Active Learning Approach

Possibly the most promising approach to improving the data efficiency of DNNs is active learning (Ren et al., 2020). As described in Section 2.5.4, active learning has been shown to drastically improve the data efficiency of training DNNs by reducing the time and energy wasted on redundant data instances (Xu et al., 2021), as well as achieving an exponential improvement in the time required to label training data, and successfully dealing with high-dimensional sequential data (Ren et al., 2020). The active learning training process implemented in this research aims to reduce computational costs by training over a subset of the data space containing instances believed to provide the most utility to the model's learning process. In this experimentation, pool-based active learning is implemented, which uses two datasets $P \subset D_{train}$ and $V \subseteq D_{train}$ to split useful and unuseful data according to an importance function. Beginning with an initial seed $P^{(1)}$ of n_{sample} instances, an iterative process is exploited where the DNN is trained over pool P and used to predict the outputs of all data instances in the validation set V; these predictions are then evaluated through the importance function, and the n_{sample} most important data instances moved into the pool before the next training round is conducted.

As noted by Xu et al. (2021), active learning algorithms can be classified through their importance function; in this research *greedy sampling* (GS) is employed. A specific pair of greedy sampling methods is used, known as *greedy sampling on the inputs* (GSx) and *greedy sampling on the outputs* (GSy). These methods, conceived by Wu et al. (2019), implement a performant greedy sampling approach that aims to increase the diversity of the input and output spaces of a model, improving training efficiency. In our implementation, GSx is used to specify the initial seed pool $P^{(1)}$, and GSy is used in the following training iterations to select data instances to be moved from the validation set $V^{(i)}$ to the pool set $P^{(i+1)}$.

GSx (Wu et al., 2019) is a method through which an initial seed pool $P^{(1)}$ is constructed by selecting n_{sample} data instances from the full dataset $V^{(0)} = D_{train}$ of size N (Wu et al., 2019).

First, the GSx algorithm selects the single data instance from the full validation set $V^{(0)}$ that is the shortest distance from the dataset's mean instance; in this case, the Euclidean distance is used, which finds the difference between multidimensional matrices. Once the data instance closest to the dataset's centroid is found, it moved from $V^{(0)}$ to $P^{(1)}$ and then used to choose the remaining $n_{sample} - 1$ data instances in the seed. This is conducted iteratively; to find the next seed instance x_i, we first compute the Euclidean distance between each data instance x_m already in the seed pool and each instance x_n remaining in $V^{(0)}$, constructing a 2D distance matrix (Equation 3.5). Then, for each instance x_n remaining in $V^{(0)}$ we select the corresponding seed value v_m that minimises the distance $x_n \rightarrow x_m$, building a 1D distance vector enumerating the closest seed instance to each remaining validation set element (Equation 3.6). Finally, the data instance x_n furthest from all existing seeds (i.e. the maximum of the distance vector) is selected and moved from $V^{(0)}$ into $P^{(1)}$. This process is repeated until we have a full seed $P^{(1)}$ of n_{sample} data instances, and a validation set $V^{(1)}$ of $N - n_{sample}$ instances. By adding seeds that are furthest from any others in the pool, GSx ensures that the seed pool used to initially train the DNN is as diverse in the data space as possible, giving the model a complete picture of the breadth of the full training dataset whilst only using a small subset of instances.

$$d_{n,m}^{(x)} = \|x_n - x_m\| \tag{3.5}$$

$$d_n^{(x)} = \min_m d_{n,m}^{(x)} \tag{3.6}$$

GSy (Wu et al., 2019) similarly aims to capture diverse instances, but alternatively focuses on the output space of the model M_θ that realises the function $y = f(x|\theta)$ (Wu et al., 2019). The GSy algorithm implements an iterative approach where data instances are selected from the validation set $V^{(i)}$ to add to the pool $P^{(i+1)}$ such that the prediction error of the function $f(x|\theta)$ over the total dataset is minimised. Namely, when considering an input instance x_n, if the model produces an output $y_n = f(x_n|\theta)$ that is noticeably distinct from the outputs generated from other data instances, the algorithm selects x_n to add to the training pool as it helps refine the parameters defining the model's predictions. GSy then works by the assertion that the more diverse the pool instances x_m collected, the more accurately the optimal parameters of the model can be determined. However, this algorithm requires a functioning model $f(x|\theta)$ to exist (such that model outputs can be generated and analysed), so it can only be used to populate a pool that has already been initialised with seed instances. Therefore, Wu et al. (2019) assert that to implement active learning, GSx should be used for determining the seed, followed by GSy to further populate the pool set at later training iterations.

Given a pool $P^{(1)}$ of n_{sample} data instances, and validation set $V^{(1)}$ of $N - n_{sample}$ instances, the GSy algorithm selects n_{sample} instances to move from $V^{(1)}$ to form the new datasets $P^{(2)}$ and $V^{(2)}$ believed to provide the most utility to training. This is done by first evaluating the model $f(x_n|\theta)$ over all data instances $x_n \in V^{(1)}$, to create the set of predicted labels $\hat{Y}^{(1)} = \{\hat{y}_n : \hat{y}_n = f(x_n|\theta) \forall x_n \in V^{(1)}\}$. Once all $N - n_{sample}$ predicted labels have been computed over the validation set $V^{(1)}$, a 2D distance matrix is constructed enumerating the distance (as the absolute difference) between each prediction $\hat{y}_n$ and all true labels y_m of the data instances x_m that already exist within

the pool set $P^{(1)}$ (Equation 3.7). From the distance matrix a 1D distance vector is constructed over the $N - n_{sample}$ outputs of the validation set $V^{(1)}$, enumerating the shortest distance of each prediction $\hat{y}_n$ to a true label y_m of an instance x_m in the pool set (Equation 3.9). The data instances in the validation set corresponding to the n_{sample} largest distances in the distance vector are then selected and moved from the validation set to the pool, creating the new datasets $P^{(2)}$ and $V^{(2)}$. This process is repeated at each training round, allowing the pool to be iteratively populated with data instances that stimulate diverse outputs from the model being trained. Thus, a DNN is developed that can produce a wide range of appropriate outputs to accurately represent a spectrum of different predictions (allowing the DNN to model a plethora of different possible scenarios), despite only being trained over a limited number of data instances.

$$d_{n,m}^{(y)} = |\hat{y}_n - y_m| \tag{3.7}$$

$$= |f(x_n|\theta) - y_m| \tag{3.8}$$

$$d_n^{(y)} = \min_m d_{n,m}^{(y)} \tag{3.9}$$

In their research, Wu et al. (2019) found that the use of GSx and GSy active learning algorithms improved the data efficiency of training and produced a model that exhibits impressive generalisability when compared to approaches using random sampling to select data instances. This demonstrates the effectiveness and robustness of active learning training that uses GSx and GSy. Hence, the experimentation into data-efficient training for DNNs presented in this thesis utilises GSx for picking the seed pool of $n_{sample} = 100$ data instances and GSy at each following training iteration i to select a further 100 data instances to move into the pool set. The full 3-layer DNN proposed in Section 3.4 is utilised over this iterative process, training over the expanding pool set for 100 training rounds.

3.6.3 Summary

Hence, to exemplify how Green AI can improve the data efficiency of training DNNs in finance, an active learning algorithm was implemented, initially selecting a seed pool using the GSx algorithm (Wu et al., 2019) to build an initial model, and iteratively building up the training dataset through the GSy algorithm (Wu et al., 2019). To evaluate the performance and efficiency of this training approach, the metrics outlined in Section 3.3 are utilised to inform a comparative analysis between the baseline algorithm and the active learning method. A close inspection of the amount of data utilised over the training process is made, demonstrating how effective active learning is at reducing the data requirements of training DNNs within the chosen financial domain. Whilst proposing the GSx and GSy algorithms, Wu et al. (2019) raise several limitations with this approach; most notably, they found that when the pool size during the early stages of training is too small (and too few instances are added at each iteration) the model being built exhibits very high variance, meaning its performance is not reliable. Therefore, to properly explore the benefits and drawbacks of active learning for improving the data and energy efficiency of DNN training, close inspection of the convergence rate of this training process is essential, to ensure it remains stable and robust.

Chapter 4

Results & Discussion

The following chapter explores the results of the presented studies. Initially, the data collected from each experiment is presented, summarising what is shown by the research findings, highlighting their significance, and examining their implications for the research hypotheses. Once all results have been outlined, a detailed discussion surrounding the success of each experiment is undertaken, evaluating how effective Green AI approaches are at reducing the computational cost of DNN training within the chosen application of financial volatility forecasting. The results of each experiment are presented and discussed with close reference to the accuracy and efficiency measures introduced in Section 3.3, quantifying both the performance and efficiency of each approach.

Each result was derived by tuning the hyperparameters of each training algorithm to maximise model accuracy whilst maintaining an efficiency greater than that of the baseline method. This implemented an accuracy-efficiency compromise that prioritised developing models with low prediction error, then evaluated the efficiency of the training process exploited to develop that model. To evaluate each training algorithm in a robust and reproducible manner, each was employed five times, optimising a model from scratch over a fixed number of epochs; the results of the most successful training round were then collected as the outcome of this research. This was to ensure that a representative result of the true ability of each training algorithm was obtained and that the outcome of this research was not biased or degraded by the stochastic nature of DL training algorithms, which can cause results to vary over different executions. To ensure reproducible and generalisable results, the model and training process is implemented in *Python* with the popular DL framework *TensorFlow* (Abadi et al., 2016). The implementations were run on *Google Colaboratory Pro* to achieve higher training performance by exploiting high-end ML hardware, executing each algorithm using a *Tesla T4* GPU.

4.1   General Predictive Performance

To demonstrate the success each training algorithm, each trained model was employed to forecast volatility over the test dataset. Figure 4.1 shows the forecasts of all models; each plot shows the predicted volatilities $\hat{v}_{t+1}$ over all timesteps of the testing period, contextualised against the true values v_{t+1}.

Figure 4.1 demonstrates that all models generate predictions that accurately follow the general trend of the true volatility sequence. However, close inspection of how the different models handle
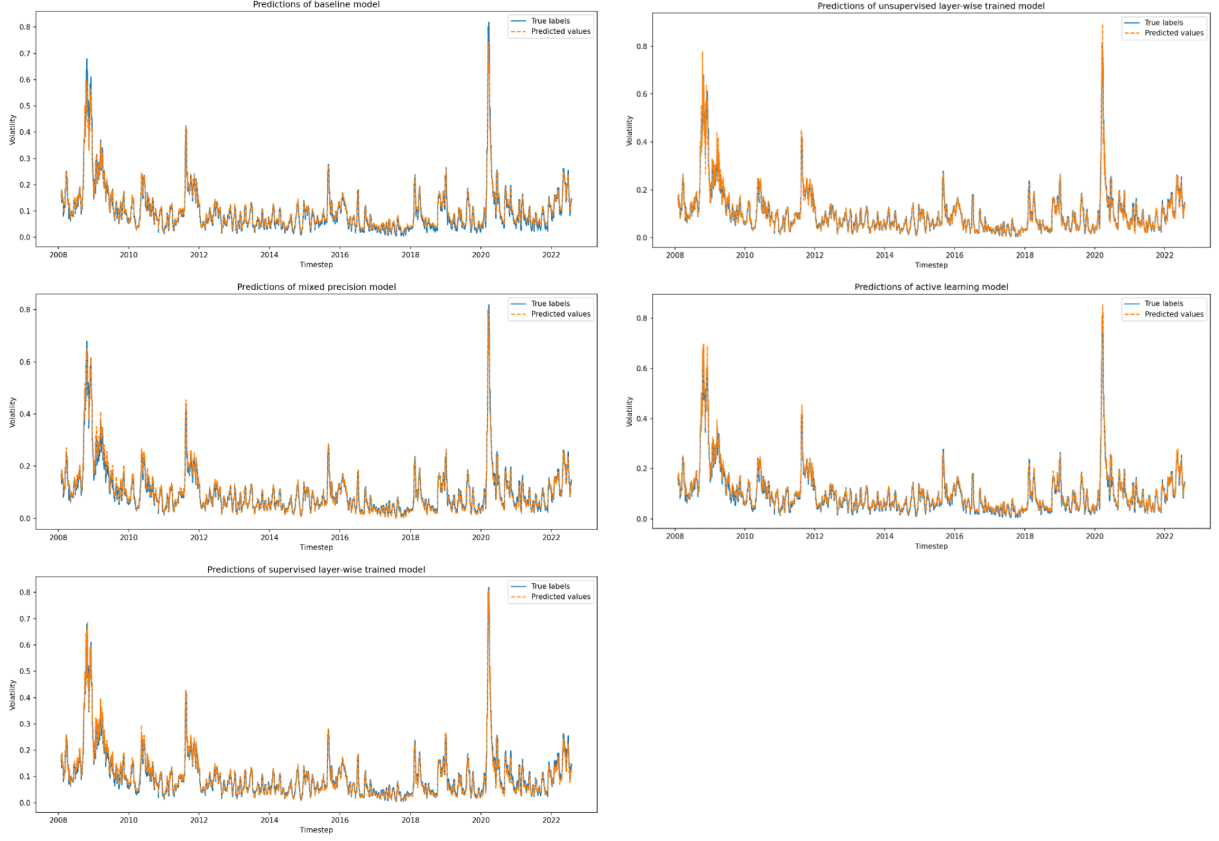
Figure 4.1: Predicted volatility time series of each tested model (orange dashed line) against the true volatility time series (blue) over testing dataset.

rapid volatility jumps highlights the differences in forecasting characteristics. If one inspects the significant rise in volatility in late 2008 (corresponding to the 2008 financial crisis), the baseline model is shown to underpredict the true volatility, undercutting the true peak by a fairly significant margin ($\hat{v}_{t+1} \approx 0.6$ when the true v_{t+1} is closer to 0.7). A similar characteristic can be seen in early 2020 (corresponding to the COVID19 pandemic) and soon thereafter, where the significant fluctuations in volatility are consistently underestimated by the DNN's predictions. This suggests that whilst the baseline DNN appears to model the smaller volatility changes very accurately, large, rapid changes in the true volatility result in less accurate predictions.

When the predicted time series of the adapted models are compared to that of the baseline, the differences in forecasting ability can be observed. Figure 4.1 shows that the models using supervised and unsupervised layer-wise pre-training most accurately represent both small and large fluctuations in volatility. These plots exhibit predicted volatility values $\hat{v}_{t+1}$ significantly closer to the labels v_{t+1} at times of high volatility (e.g. in 2008 and 2020). The forecast values of the model trained through unsupervised layer-wise pre-training appear slightly less accurate than those of its supervised counterpart, as its predictions seem to overestimate the volatility jumps in 2008 and 2020, however less extreme movements seem to be modelled very closely. The model developed through mixed-precision training also appears to generate predictions that lie closer to the true volatility than the baseline model, although with an accuracy slightly below that of the layer-wise

pre-trained models.

When considering the model trained through active learning, the forecast time series is also shown to accurately follow the path of the true volatility, with most fluctuations accurately captured. However, these predictions seem to slightly miss the true movements, as the scale of peaks and troughs seems to be consistently underestimated (underpredicting the value of $\hat{v}_{t+1}$ for peaks and overestimating it at troughs), suggesting this model is slightly less performant than other adapted approaches. However, the general trend of the predictions on a wider scale seems (at least) commensurate with those of the baseline model, indicating that the implemented active learning training process does not degrade forecasting performance.

4.2   Accuracy Results

To demonstrate the preformance of each trained model more precisely, Table 4.1 presents the results generated by evaluating the accuracy metrics presented in Section 3.3.1. This began with the baseline model; it was found to exhibit an impressive test accuracy, demonstrating the great ability of the LSTM architecture to model sequences such as the utilised financial volatility time series. Namely, the coefficient of determination of the baseline model was found to be 0.977414, showing that the sum of squared prediction errors between $\hat{v}_{t+1}$ and v_{t+1} is close to zero (Equation 3.1). This means that the true labels have been near-perfectly represented by the model, demonstrating the DNN trained by a traditional training process can generate excellent one-step-ahead forecasts of the market volatility of the S&P 500. Similarly, the baseline model produced impressive MAE and RMSE scores of 0.008407 and 0.013830, demonstrating that the average magnitude of the prediction error is near-zero. Importantly, both the MAE and RMSE are similar, indicating that the model is averse to uniformly distributed prediction errors (highlighted by the MAE) and highly deviant predictions (identified by the RMSE). The model also generated an impressive MAPE of 14.662217%, indicating that the DNN has fit the training data well.

Method	R^2	MAE	RMSE	MAPE (%)
Baseline	0.977414	0.008407	0.013830	14.662217
Mixed-precision	0.977372	0.009121	0.013843	12.126700
Supervised layer-wise pre-training	0.985642	0.007814	0.011027	9.932409
Unsupervised layer-wise pre-training	0.98406	0.00708	0.01162	8.80475
Active Learning	0.971119	0.010452	0.015639	11.08715

Table 4.1: Accuracy data for all methods, evaluated over the testing datset.

After certifying the performance of the baseline model, the accuracy achieved by alternative training methods could be contextualised. This exploration began with an analysis of mixed-precision training; impressively, the mixed-precision model was found to higher accuracy than the baseline across almost all metrics, generating a 15.10% lower MAE, 10.50% lower MAPE, 8.53% decrease in RMSE (Table 4.1). These scores indicate that the DNN exploiting mixed-precision

training was able to forecast market volatility more accurately than the baseline across both scale-dependent and scale-independent metrics.

Following this result, the two models that utilised progressive training were analysed. The first model, which exploited a supervised layer-wise pre-training process, exhibited higher forecasting accuracy than both the baseline and mixed-precision models that used traditional training algorithms. The optimised model produced an outstanding R^2 score of 0.985642, indicating that using supervised layer-wise pre-training facilitated an almost perfect correspondence between $\hat{v}_{t+1}$ and v_{t+1}; additionally, this model reduced the MAPE of the baseline model by 32.26% and the mixed-precision model by 18.09%. The improvement in accuracy was observed across the board, generating better R^2, MAE, RMSE, and MAPE scores than both the baseline and mixed-precision methods. The model using unsupervised layer-wise pre-training also exhibited impressive performance, generating predictions of similar accuracy to its supervised counterpart, and significantly outperforming the baseline with 0.6800% higher R^2, 15.78% lower MAE, 15.98% lower RMSE, and an exceptional decrease in MAPE of 39.95%. When directly comparing the models using supervised and unsupervised layer-wise pre-training, however, the relative advancements are not so explicit. The supervised approach produced marginally better performance across the R^2 and RMSE metrics: the R^2 of the supervised approach was 0.1607% higher and RMSE 5.103% lower. On the contrary, the unsupervised approach produced a model with a 9.393% smaller MAE and 11.35% smaller MAPE. This suggests that both supervised and unsupervised layer-wise pre-training facilitated accuracy gains over the baseline training algorithm, but each reduced prediction error differently: for example, the lower RMSE of the supervised pre-trained DNN indicates this model less frequently generates predictions that are highly incompatible with the true value. Thus, the accuracy results in Table 4.1 show that both supervised and unsupervised layer-wise pre-training are the most effective training methods for developing high-performance DNNs for volatility forecasting.

The active learning implementation produced impressive performance when evaluating through MAPE, exhibiting a 24.38% lower test MAPE than the baseline model. However, when considering other accuracy metrics, the model was not shown to exhibit such an accuracy benefit, having a slightly worse coefficient of determination (0.6440% lower) and noticeably increased test MAE (24.33% higher) and RMSE (13.08% higher). Thus, the performance of the active learning-based model was roughly equivalent to the baseline, outperforming it on some metrics but underperforming on others. When different parameterisations of the active learning training process were explored (Table 4.4), a few implementations did produce higher accuracy than the baseline; for example, when using 200 training iterations and a sample size of 70, a 26.83% lower RMSE and 34.72% lower MAPE were facilitated (but using 96.36% of the full training dataset). Hence, these results indicate that active learning can be used to develop a model with similar performance to a traditionally trained baseline, supporting the assertion that this method can be tuned to only inflict a marginal accuracy detriment.

4.3 Efficiency Results

4.3.1 Training Time Analysis

As outlined in Section 3.3.2, total training time is one of the most applicable metrics for quantifying the energy efficiency of DL training algorithms in this context; hence, to examine the success of each training algorithm the time (in seconds) taken to optimise the DNN's parameters was quantified by profiling each implementation, the results of which are exhibited in Table 4.2. This analysis found that the baseline training approach took a total of 198.602 seconds to optimise the DNN; this result was then used to contextualise the efficiency of each subsequent training method. Firstly, mixed-precision training was examined, which produced a fairly disappointing result: the algorithm took 18.98% longer to complete than in the baseline, despite using the same number of epochs. This suggests that utilising 16-bit floating-point representations for evaluating operations during training did not directly improve the energy efficiency of training the DNN.

Method	Total Training Time (s)
Baseline	198.602
Mixed-precision	236.306
Supervised layer-wise pre-training	154.952
Unsupervised layer-wise pre-training	130.292
Active learning	212.453

Table 4.2: Total training time required for each implemented training process to produce a model with the accuracy outlined in the preceding accuracy table.

On the contrary, both explored progressive training approaches were shown to generate a significant reduction in the total training time, with unsupervised layer-wise pre-training being the fastest to complete. The unsupervised method facilitated a 34.40% decrease in training time, whilst the supervised method improved on the baseline by 22.43%, but took 18.24% longer than its unsupervised counterpart. These results indicate that unsupervised layer-wise pre-training is the most efficient way to train DNNs, taking the least amount of time to produce an accurate model, and hence likely being the most energy-efficient approach. Furthermore, it demonstrates that supervised unsupervised layer-wise pre-training are both viable ways to improve the energy efficiency of DNN training, as they both took significantly less time to train an accurate forecasting model than the baseline algorithm.

When the training time of active learning is analysed, the results do not indicate that this is an effective method for reducing the overall length of training. Namely, the active learning algorithm using hyperparameters detailed in Section 3.6.2 marginally exceeded the total time spent training by the baseline approach by 6.974%. This suggests that active learning is not an effective method for directly minimising the length of training. Although, a small proportion of the active learning parameterisations tested did result in a reduction in training time over the baseline algorithm,

majoritively achieved by drastically limiting the number of training iterations and training dataset size, and hence in general also inflicted an accuracy drop (Table 4.4). Despite this result, it is not concerning that total training time was not reduced, as active learning majoritively focuses on improving the data efficiency of training, not explicitly the energy efficiency. This process instead intends to improve the ESG impacts of DL by reducing resource requirements, data storage and labelling costs, and the energy consumption of large data centres.

4.3.2 Convergence over Training

To further quantify the efficiency of each training approach, the convergence of the training error towards the minimum of the loss function was evaluated (Figure 4.2). The plots show the prediction error output by the loss function on input of the training dataset and on input of a smaller validation set (a collection of data instances held back from the training process) at each epoch.
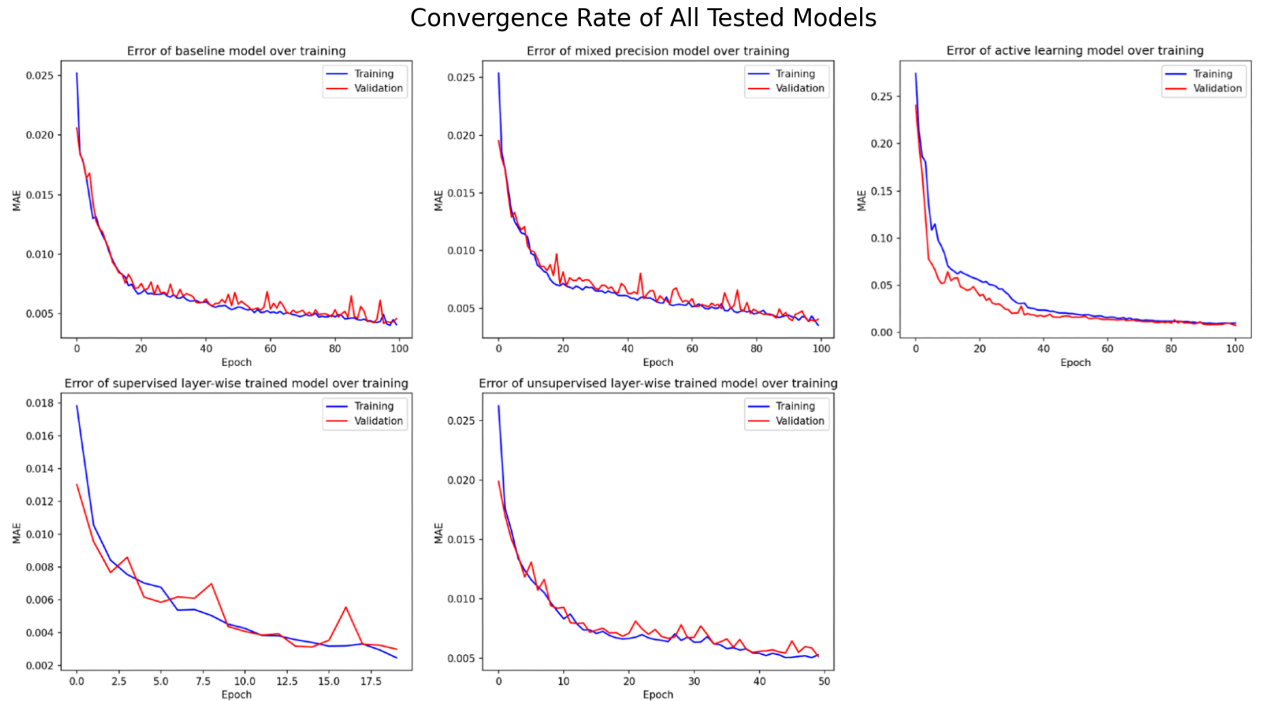
Figure 4.2: Convergence over each tested DNN training algorithm, shown as the progression of the output of the loss function (MAE) used at each epoch to determine the prediction error of the networkn (with a steeper gradient indicating a faster convergence rate).

s

When the convergence of the mixed-precision training process is compared to the baseline algorithm, a very similar trend can be seen; both exhibit closely aligned training and validation error curves, implying the models fit well to the training data, and both exhibit a significant decrease in MAE at the start of training with this tapering off in later epochs. The close similarities between both plots suggest that whilst introducing mixed-precision representations has not sped up the training process in terms of convergence per epoch, using these lower bit representations has not degraded the convergence rate of the prediction error towards the minimum of the loss function. Although, upon close inspection of the progression of the validation error over the mixed-precision training process, the curve appears to be slightly rougher than in the baseline case,

exhibiting marginally larger fluctuations around the general trend line of convergence over the training process. This suggests that the training process may be slightly less stable when using mixed-precision representations.

Comparison of the convergence of the two progressive training algorithms demonstrates the different approaches these methods take to developing an accurate model. Firstly, one can see that the pre-trained DNNs require significantly fewer epochs to optimise the full parameter set than the other training algorithms: the supervised and unsupervised methods only requiring 20 and 50 epochs respectively over the full network. Additionally, the initial epochs of both of these algorithms are shown to exhibit a much faster convergence, reducing both the training and validation error much more rapidly. Furthermore, the majority of their minimisation of prediction error within the first 10 epochs, whereas the same reduction is spread over the first 20 epochs in the baseline training process. For example, of the total reduction of validation error from ~ 0.013 to ~ 0.003 produced by the supervised approach, 90% occurs within the first 10 epochs (decreasing the MAE to ~ 0.004). Similarly, 80% of the total reduction in validation MAE ($\sim 0.02 \rightarrow \sim 0.005$) in the unsupervised case happens over the first 11 epochs of full network training ($\sim 0.02 \rightarrow \sim 0.008$). This indicates that progressive training is a significantly more efficient training approach as it induces faster learning within the DNN.

Analysis of the convergence over active learning similarly highlights its ability to induce faster learning within a DNN than the baseline algorithm. This is indicated through the incredibly sharp decline in both training and validation error over the early stages of training, generating approximately 83% of the total reduction in validation MAE over the first 8 epochs. Furthermore, whilst in the baseline case the convergence rate slows considerably after the initial decline in MAE, significantly plateauing after the first 20 epochs, during active learning the convergence rate stays higher for longer. Namely, a significant reduction in MAE is still observed up to approximately the 32nd epoch. This suggests that the learning process is more efficient as the network's parameters are optimised more rapidly in the early stages of training.

4.3.3 Efficiency Breakdown of Progressive Training

To further analyse the benefit provided by progressive training, a detailed study was conducted into the time taken by the supervised and unsupervised layer-wise pre-training algorithms, breaking down how these approaches improve training efficiency (Table 4.3). These results demonstrate the different approaches taken by the progressive training algorithms, laying out how much time (and how many epochs) was spent on pre-training and tuning.

This study found that supervised layer-wise pre-training was most effective when 20 epochs were used to pre-train each layer, with another 20 epochs used to optimise the full network; in comparison, the unsupervised approach generated the best performance when employing only 7 epochs per pre-training round, then a final tuning round of 50 epochs. Thus, in total the supervised method used 60 low-cost epochs (training only a single layer) and 20 high-cost epochs, whereas the supervised method only employed 21 low-cost epochs but required 50 high-cost epochs. Table 4.3 illustrates this divergence by showing the proportion of the total training time spent pre-training; the supervised approach allocated 69.63% to this stage, compared to only 33.59% during the unsupervised algorithm. Therefore, whilst unsupervised layer-wise pre-training had a shorter

Pre-Training Method	Pre-Training Time (s)	Pre-Training Epochs	Layer 1 (%)	Layer 2 (%)	Layer 3 (%)	Tuning Time (s)	Tuning Epochs	Total Time (s)
None	-	-	-	-	-	198.60	100	198.60
Supervised	107.89	20	32.6	29.0	38.4	47.066	20	154.95
Unsupervised	43.769	7	27.4	47.8	24.7	86.523	50	130.29

Table 4.3: The time taken by pre-training and full network tuning for both supervised and unsupervised layer-wise pre-training (including a breakdown of the time spent pre-training each of the 3 internal LSTM layers), compared to the total time taken by the baseline method.

total training time, supervised pre-training facilitated a greater reduction in the required number of full DNN tuning epochs to exceed the performance of the baseline method (exhibiting an 80% decrease in tuning epochs, compared to a 50% decreasing when using the unsupervised approach).

Table 4.3 also breaks down how each pre-training round contributed to the overall computational cost of training. Of the total pre-training time, the supervised method spent 32.6% on the first layer of the network, 29.0% on the second, and 38.4% on the final layer. Hence, the second hidden layer was faster to pre-train than the initial base model (suggesting the transfer of knowledge between these rounds), but the final LSTM layer was the most computational intensive of all (possible due to the higher-fidelity representations once the network deepens). A different trend was seen over unsupervised pre-training, where the second hidden layer was the most computationally intensive (using 32.6% of the pre-training time), and the final LSTM layer was the fastest (24.7% of pre-training); this is likely due to the 2nd LSTM layer building up the majority of the representation of the inputs within the autoencoder, meaning the 3rd hidden layer was not required to add as much to the representation learning process (simply building upon the knowledge of its preceding autoencoder layer).

4.3.4 Data Efficiency of Active Learning

Once the energy efficiency of the baseline, mixed-precision, and progressive training algorithms had been thoroughly explored, the data efficiency of active learning was examined. Since this approach focusses on simultaneously minimising both the training dataset size and the accuracy drop inflicted by this constriction, active learning can implement a spectrum of accuracy-efficiency compomises. To explore these varying compromises, different parameterisations of active learning algorithms were experimented with, manipulating the hyperparameters controlling the number of training iterations and sample size n_{sample}.

Table 4.4 outlines the results of this study, enumerating the different hyperparameters used and the resulting data efficiency, accuracy, and training time achieved. The results demonstrate the general trend that the greater the number of active learning iterations, and the more data added at each iteration, the higher the achieved model accuracy but the lower the data and time efficiency of training. For example, the hyperparameters $(N_{iters}, n_{sample}) = (200, 70)$ facilitated a highly performant DNN with a test MAPE of 9.57100 (a 34.72% decrease from the baseline model), but this required using most of the training data (96.36% of the full dataset) and a considerably

Method	Number of Iterations	Sample Size	Final Pool Size	Utilisation of Full Training Dataset (%)	RMSE	MAPE	Training Time (s)
Baseline	-	-	14600	100	0.013830	14.662217	198.602
Active Learning	*200*	70	14070	96.36	0.01012	9.57100	480.603
		45	9045	61.95	0.01492	10.47995	436.716
		20	*4020*	*27.53*	*0.01863*	*14.03773*	*222.579*
	150	90	13590	93.08	0.01404	12.25406	363.479
		55	8305	56.88	0.01595	13.57484	326.668
		20	3020	20.68	0.02579	25.93023	274.484
	100	140	14140	96.85	0.01740	15.03445	239.561
		80	*8080*	*55.34*	*0.015639*	*11.08715*	*212.453*
		20	2020	13.84	0.03527	29.82608	175.556
	50	250	12750	87.33	0.01735	13.69113	176.087
		135	6885	47.16	0.02580	22.39938	124.031
		20	1020	6.99	0.03883	41.60770	89.6995

Table 4.4: Results of the experimentation into how varying the number of training iterations and sample size effect the data efficiency, resulting accuracy, and training time of an active learning based training algorithm (using GSx and GSy). N.B. notable results are emphasised in italics.

longer training time (142.0% increase over the baseline).

However, by tuning these hyperparameters the ability of active learning to facilitate accurate performance whilst considerably reducing training data requirements was shown. Under the hyperparameters $(N_{iters}, n_{sample}) = (200, 20)$, this approach surpassed the performance of the baseline model (reducing the test MAPE by 4.259%) using only 27.53% of the amount of training data. Furthermore, whilst this did inflict a longer training time than the baseline, an increase of only 12.07% was observed, suggesting the direct computational cost was not significantly inflated. Another notable parameterisation $(N_{iters}, n_{sample}) = (100, 80)$ was able to produce a model with significantly improved MAPE (24.38% lower than the baseline) using only 55.34% of the training data, and only inflicting a 6.974% increase in training time. The impressive result solidified this parameterisation as the implementation used to compare the performance of active learning to the other implemented training algorithms in the preceding section.

4.3.5 Summary

Hence, mixed-precision training and both progressive training algorithms were shown to provide an accuracy benefit over the baseline model, with unsupervised layer-wise pre-training facilitating the greatest performance improvement. Additionally, active learning was shown to produce performance at least commensurate to that of the baseline. When considering the efficiency of each model, the mixed-precision algorithm generated dissapointing results, exhibiting similar convergence but a longer training time. However, both progressive training methods were able to significantly reduce training time, whilst active learning was able to considerably restrict the training dataset without

impacting performance; additionally, both progressive training and active learning achieved greater convergence rates, showing these algorithms learn more intelligently than the baseline approach.

4.4 Extended Discussion

Given the aforementioned results outlining the accuracy and efficiency of each training algorithm, an evaluation can be made surrounding how well each reduces the computational cost and resource requirements of training DNNs, demonstrating their utility for lowering the negative ESG impacts of Fintech.

4.4.1 Mixed-Precision Training

The implemented mixed-precision training approach attempts to reduce the computational cost of developing DNNs by lowering the precision at which operations within a DNN are evaluated. The experimentation found that whilst mixed-precision training did result in a higher accuracy model than the baseline, the training process took 18.98% longer, suggesting this approach did not directly improve energy efficiency. However, a nearly identical convergence rate towards the minimum of the loss function over this training process was demonstrated, suggesting that reducing the memory requirements of DNN training by using smaller 16-bit representations does not negatively impact the stability of training and performance of the resulting model.

Since the mixed-precision implementation was not shown to facilitate a reduction in training time, there is no evidence found in this research to suggest that mixed-precision representations are an effective method for improving the energy efficiency of training DNNs for financial volatility forecasting. This is a disappointing conclusion, as it does not echo the results of reduced precision implementations in other fields that have boosted training speed, such as DoReFa-Net (Zhou et al., 2016) which showed the benefit of low-bit training to CNN-based image classifiers. However, this is not that surprising, as DoReFa-Net inflicted a significant reduction in precision (using 1-bit parameters, 2-bit activations, and 6-bit gradients) which was not undertaken in this research; instead, 16-bit representations were used, which cannot be expected to facilitate the same efficiency improvement as that of Zhou et al. (2016). Furthermore, the presented findings more closely resemble those of Ott et al. (2016), who found that low-bit training was not ubiquitously effective for improving the efficiency of training RNNs; hence, the results presented in this thesis may suggest that LSTM-based volatility forecasting may be one such application.

The lack of efficiency benefit may also be explained by the model architecture and hardware utilised. Namely, the implemented DNN (and training dataset) is relatively small compared to state-of-the-art implementations developed in cutting-edge DL research and the expensive systems typically used by large institutions. Due to the overhead costs within model training, smaller models do not typically benefit from mixed-precision to the same extent as those with more complex architectures, as these overheads constitute a large portion of the training time. Furthermore, hardware accelerators (such as the GPUs relied upon in this study) do not provide as much benefit to smaller models, as the matrix multiplications are not as computationally intensive and so do not benefit as much from increased parallelisation. Therefore, whilst no training time reduction was witnessed in this research, this method may still hold promise in improving the energy efficiency

of the higher complexity systems used by financial institutions.

Hence, this experimentation did not uncover any benefit of mixed-precision training in improving the energy efficiency of DL in finance and thus did not confirm the research hypothesis. However, one limitation of this research is that the memory consumption over DNN training was not explored; thus, this research does not quantify any reductions to memory usage that may have been facilitated by mixed-precision representations, which could have revealed other efficiency improvements.

4.4.2 Progressive Training

On the contrary, both progressive training approaches were able to significantly reduce total training time whilst producing a DNN with higher prediction accuracy. Supervised layer-wise pre-training improved the MAPE of the baseline model by 32.26%, whereas the unsupervised approach generated a $\sim 15\%$ decrease in MAE and RMSE and an impressive 39.95% reduction in MAPE. These results indicate that the true volatility time series has been accurately represented by the predictions of the trained model, enabling accurate predictions. Furthermore, a high convergence rate is exhibited over both progressive training approaches, demonstrating the stability and robustness of the training algorithms. The significant reduction to training time in both cases (22.43% and 34.40% faster in the supervised and unsupervised cases respectively) also suggests that supervised and unsupervised layer-wise pre-training are successful at improving the energy efficiency of training DNNs in this context, as shorter training times likely correspond to lower energy consumption.

These results align with other research into progressive training completed further afield, similarly highlighting the efficiency of progressive training algorithms. For instance, Ienco et al. (2019) found supervised layer-wise pre-training improved the generalisability of sequence classifiers, which is similarly shown in the high testing accuracy of the presented models. Additionally, Xu et al. (2018) and Sagheer and Kotb (2019) found unsupervised layer-wise pre-training could induce faster convergence whilst improving accuracy, both of which are demonstrated in the results of this research. Hence, this study supports the findings of other research into progressive training, extending the known utility of this Green AI method to the field of financial market volatility forecasting.

Another core success shown by both algorithms was their ability to significantly reduce the number of epochs spent optimising the parameter set of the full network. The supervised approach reduced the required tuning rounds by 80% to 20 epochs and the supervised approach lowered it by 50% to 50 epochs. Since optimising the full 3-layer LSTM network is significantly more computationally intensive than optimising only a single LSTM layer (due to the higher number of parameters, calculations, and updates), this reduction in tuning likely facilitated a significant decrease in the computational cost of training. This was likely facilitated by the layer-wise pre-training stage inducing faster optimisation of the parameters, as the number of epochs in total over both the pre-training and tuning stages was still lower than in the baseline case. Thus, the low-cost pre-training epochs were shown to be effective at teaching the DNN how to make accurate predictions, allowing it to learn faster than possible in the traditional baseline algorithm. Furthermore, this efficiency benefit was likely facilitated by the transfer of useful information learned by each layer over pre-training to the subsequently added layer, allowing parameter updates to incorporate information already learned about the features of the input (as Xu et al. (2021) posited). This is

shown through layers added later during pre-training (in general) making up a lower proportion of the total pre-training time. During supervised pre-training, 29.0% of the total time was allocated to training the second LSTM layer, compared to 32.6% spent on the first. Similarly, the unsupervised approach expended 27.4% of pre-training on the first hidden layer and only 24.7% on the last. Thus, these results demonstrate that supervised and unsupervised layer-wise pre-training facilitate a more intelligent and efficient learning process.

One limitation of layer-wise pre-training is the extra computational cost of fixing the parameters of the previously pre-trained layers such that they are no longer optimised in subsequent pre-training rounds. When profiling the progressive training algorithms, this cost was included in the pre-training time recorded per layer; thus, the total time calculated is not solely attributed to model training. This means there is a slight divergence between what is included in the total training time statistic between the progressive training algorithms and the baseline. Additionally, more time is spent on auxiliary tasks during each subsequent pre-training round than when training the initial base network, as the first round does not fix any parameters. This could possibly contribute to why some hidden layers added during pre-training do not exhibit an increased training speed from the layer preceding it. However, these must be included in the total times attributed to progressive training to ensure a fair comparison to the baseline method which does not require these auxiliary tasks.

An interesting difference between the supervised and unsupervised approaches is that the supervised method allocated 69.63% of the total training time to pre-training, compared to only 33.59% during the unsupervised algorithm (which spent more time tuning). This distinction is likely due to the supervised algorithm training the DNN to forecast volatility from the start, whereas the unsupervised approach pre-trained the DNN as an autoencoder. Time series forecasting is known to be a challenging task, but this is not so much the case with representation learning, as training an autoencoder to solely minimise the reconstruction error between inputs and outputs is significantly simpler than forecasting future values. Thus, more epochs were required to produce an accurate DNN over supervised pre-training than in the unsupervised case. However, later stages of unsupervised layer-wise pre-training required the developed autoencoder to be converted into a forecasting model, hence requiring more full network tuning epochs.

The observed lower training times, fewer epochs, and computational cost savings indicate that both supervised and unsupervised layer-wise pre-training algorithms improve the energy efficiency of training a DNN for financial volatility forecasting. Whilst the observed reductions in training time have been on the scale of seconds in this research, when the model architecture and training length are scaled up to be comparable to the large, energy-intensive DL systems utilised in the upper echelons of computational finance research and by large financial institutions, these savings will likely also scale, allowing more significant reductions in training time (in the scale of hours and days) and carbon emissions. This adequately fills the research gap by proving that progressive training is an effective Green AI method for improving the energy efficiency of DL in finance, thus proving a possible route through which the carbon emissions of Fintech can be reduced, aligning Fintech more closely with the SDGs of sustainable finance. Moreover, this research has additionally improved the inclusivity of DL for finance, as shorter training times and smaller energetic costs lower the resource requirements of developing DNNs, reducing the expense of implementing such a

system, and hence allowing more financial players and researchers to reap the benefits of DL.

4.4.3 Active Learning

As discussed by Bender et al. (2021) and Walsh et al. (2021), a significant portion of the computational cost of high-performance DNNs can be attributed to their inefficient use of data. Thus, to further mitigate the ESG impacts of Fintech this research explores a data-efficient training algorithm known as active learning; this approach selects data instances more intelligently, reducing the training dataset size and minimising memory and energy costs.

The completed study demonstrated the spectrum of efficiency-accuracy compromises achievable by active learning, ranging from the parameterisation $(N_{iters}, n_{sample}) = (200, 20)$ which maintained a comparable test MAPE to the baseline model using only 27.53% of the data, to the parameterisation $(100, 80)$ which facilitated a 24.38% improvement to MAPE using only 55.34% of the training data (Table 4.4). This shows that active learning is an effective approach for significantly reducing the amount of training data required to develop accurate DNNs in the chosen context; this lines up with the research of Ren et al. (2020), who showcase the ability of active learning to deal with high-dimensional data in CV, and the results of Peng et al. (2017) and Zimmer et al. (2018) who both demonstrated benefits of active learning to reducing dataset sizes in other time series modelling domains. Hence, the presented research has built upon this preceding work to demonstrate that active learning is effective at improving the data efficiency of DNNs used in finance.

By reducing the training data used within each epoch, it is likely active learning also reduced the total computational cost of training. One method of approximating the computational cost of each epoch is by estimating the number of parameter optimisation iterations (i.e. applications of gradient descent over the full parameter set) per epoch. If we have a training dataset of N instances and utilise a batch size b, there will be a total of $\frac{N}{b}$ full parameter optimisation iterations at each epoch, as the DNN's parameter set is iteratively updated over the full training dataset one batch at a time. In the baseline case, there are $\frac{14600}{32} \approx 456.25$ parameter optimisation iterations per epoch. If this cost is to be computed for an active learning approach, the equivalent figure for training dataset size must be approximated as the average pool set size over the entirety of training. For the parameterisation $(N_{iters}, n_{sample}) = (100, 80)$ we begin with a seed pool of 80 instances, and finish training with a pool set of size 8080, meaning on average a pool of $N_{pool} = 4080$ instances was employed. Therefore, the active learning algorithm on average employs $\frac{4080}{32} \approx 127$ parameter optimisation iterations per epoch. Thus, the training dataset employed means the calculation of significantly fewer prediction errors and parameter updates, reducing the computational cost of each epoch and the full training process of optimising the network's parameter set.

Ren et al. (2020) assert that this reduction in training data is facilitated through more intelligent data selection. The presented findings support this assertion, as the active learning algorithm exhibits a higher convergence rate, maintained over a greater number of epochs; hence, this approach more rapidly optimises the parameter set, approaching the minimum of the loss function and optimal DNN parameterisation at a faster rate over the initial stages of training. The improved convergence is likely due to selected data allowing the DNN to learn more about its modelling task during each epoch, facilitating a faster and more efficient learning process. This indicates that the utilised GSx and GSy algorithms have successfully constructed a training pool that effectively

represents the input and output spaces of the model, allowing it to learn more generalisable performance early on, suggesting this algorithm more intelligently constructs a training dataset even over the complex, multivariate time series data used in finance.

This study achieved its aim to demonstrate how active learning can be used to produce equivalent (and even superior) performance to a traditional training approach whilst significantly reducing the training dataset size. By reducing data requirements, another possible avenue for reducing the ESG impacts of DL for finance has been highlighted. Whilst not attempting to directly reduce the energy consumption of training by minimising training time (as is the case with progressive training), active learning does facilitate a reduction in the energetic cost of DL indirectly, mitigating environmental concerns and improving the inclusivity of this field. This is because reducing dataset size lowers the computational cost of training (as evaluating a DNN over a smaller dataset is cheaper) and decreases data storage and labelling costs. Furthermore, the datasets employed in state-of-the-art DL systems and industrial applications are often multiple orders of magnitude larger than that used in this research; hence, the energy and carbon savings generated if the successes of this research are applied to higher complexity models will likely be even more impressive. Additionally, this approach has the potential to reduce the energy consumption of large data centres by removing the requirement to store expansive datasets, lowering the cost of transferring these datasets between servers, and reducing the cost of the algorithms executed there. This also lowers the bar-to-entry to training DL systems as developers do not need to gather or store vast collections of data to produce accurate models.

Thus, the contribution of this research into data efficiency to the field of computational finance and the financial industry is twofold; firstly, an approach for decreasing the environmental cost of DL for finance has been presented, and secondly, the financial cost of high-performance DL systems has been reduced, improving the inclusivity of Fintech. Furthermore, the successful application of active learning to the new domain of financial market volatility forecasting has expanded the known space of useful applications of Green AI, further demonstrating the utility of these methods, and providing a new avenue Green AI research can take focusing on improving the sustainability of DL in industry.

Chapter 5

Conclusions

5.1 Summary & Contributions

In summary, this research improves the efficiency of DNN training algorithms used within computational finance. Due to the significant energy consumption of DL algorithms, their expanding use within Fintech raises concerns about inflating negative ESG impacts of finance. These concerns have been addressed within the research fields of CV, NLP, and mobile computing, through new Green AI algorithms that prioritise efficient use of energy and data. Hence, to mitigate the environmental (and social) concerns associated with using DL in finance, and ensure DNNs can be used without compromising the SDGs of sustainable finance, this research has explored how Green AI can be applied within this field.

To apply Green AI algorithms to finance, a volatility forecasting model was implemented using a DNN with an LSTM architecture; this model was then optimised through efficient training algorithms that inflict a low computational cost (in terms of energy and data) whilst facilitating accurate performance. Mixed-precision training, progressive training, and active learning were explored, determining the efficiency of each algorithm. The experimentation found that progressive training was the most energy-efficient DL training algorithm, producing a model with higher performance over a shorter training period. Active learning was also shown to be incredibly beneficial to improving the data efficiency of training, further minimising the resource requirements of DL algorithms.

Based on these results, the contribution of this research is threefold. Firstly, applying Green AI to this new domain has expanded the known applications of efficient training algorithms, demonstrating the potential of progressive training and active learning to reduce the energy and carbon cost of DL in industrial applications. This proves that the benefits of Green AI are not only possible in state-of-the-art research domains like CV and NLP, or for low-resource systems such as mobile applications, as these algorithms are also effective at improving the sustainability of DL further afield. Hence, a new, impactful avenue of Green AI research has been created, exploring how these algorithms can be used to mitigate the ESG cost of DL in industry.

Secondly, addressing the energy and data efficiency of DL algorithms in finance proves that these systems can be used to conduct highly accurate modelling at a low computational cost. This aligns Fintech closer to the SDGs of sustainable finance, providing an avenue for mitigating the carbon emissions attributed to this field; this introduces a new approach towards sustainable

finance, namely *sustainable deep learning for sustainable finance*. Thirdly, lowering the resource requirements and computational cost of Fintech improves the social inclusivity of this field, lowing the bar-to-entry to engaging in computational finance research and to using DL algorithms to inform (and improve) the behaviour of individual financial market players.

5.2 Limitations & Further Work

The research presented within this thesis does exhibit several limitations; most notably, the majority of the efficiency analyses heavily rely on evaluating the total training time of each training algorithm. Whilst this is an effective metric for enumerating the general computational cost of an algorithm, it is not without its flaws (Schwartz et al., 2019). Hence, to give a more holistic analysis of the utility of Green AI within this domain, a wider host of efficiency metrics could have been used. Another limitation is the use of a smaller model architecture, dataset, and training length (and inferior computer hardware) in this research compared to state-of-the-art systems used within this field (which are responsible for the bulk of the ESG costs attributed to DL). Hence, it is unknown how exactly the energy savings demonstrated in this research will scale when Green AI is employed in higher complexity systems.

However, these limitations do not restrict the presented research findings, as it is likely the improvements to energy and data efficiency observed are scalable to higher complexity DL systems. Namely, progressive training is likely to improve the energy efficiency of training complex DNNs employed by large financial institutions. Additionally, the benefits to data efficiency provided by active learning could be used to reduce the storage, communication, and computing costs of the large servers employed by data centres.

Based on these limitations, future work in this field could improve the scope and impact of the presented research by further analysing the implemented Green AI algorithms using a wider spectrum of efficiency metrics, facilitating a more comprehensive overview of their computational cost. For example, each algorithm's memory consumption should be inspected, highlighting the reduction to memory costs facilitated by methods such as mixed-precision training. Explicitly quantifying the carbon emissions attributable to each algorithm would also be beneficial, possibly using the machine learning emissions calculator of Lacoste et al. (2019).

Furthermore, this research solely focused on improving the efficiency of DL training algorithms; however, both Bender et al. (2021) and Xu et al. (2021) assert that a significant proportion of the ESG impacts of DL algorithms are inflicted after these models have been deployed, not during training. These *inference* costs have been explored within Green AI, such as the research of Lacoste et al. (2019) and Cai et al. (2022). For example, *low-rank factorisation* (Xu et al., 2021) has been shown to effectively reduce the size of DNN architectures, reducing the computational cost of each computation made by the network. Hence, further work could exploit such methods to explore how the ESG impact of DL for finance could be reduced over the entirety of a model's lifetime.

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv e-prints*, art. arXiv:1603.04467, March 2016.

Michel Aglietta, Jean-Charles Hourcade, Carlo Jaeger, and Baptiste Perrissin Fabert. Financing transition in an adverse context: climate finance beyond carbon finance. *International Environmental Agreements: Politics, Law and Economics*, 15(4):403–420, Nov 2015. ISSN 1573-1553. doi: 10.1007/s10784-015-9298-1. URL https://doi.org/10.1007/s10784-015-9298-1.

Anirudh Agrawal and Kai Hockerts. Impact investing: review and research agenda. *Journal of Small Business & Entrepreneurship*, 33(2):153–181, 2021. doi: 10.1080/08276331.2018.1551457. URL https://doi.org/10.1080/08276331.2018.1551457.

Vedat Akgiray. Conditional Heteroscedasticity in Time Series of Stock Returns: Evidence and Forecasts. *The Journal of Business*, 62(1):55–80, 1989. ISSN 00219398, 15375374. URL http://www.jstor.org/stable/2353123.

Omar Y. Al-Jarrah, Paul D. Yoo, Sami Muhaidat, George K. Karagiannidis, and Kamal Taha. Efficient Machine Learning for Big Data: A Review. *Big Data Research*, 2(3):87–93, 2015. ISSN 2214-5796. doi: https://doi.org/10.1016/j.bdr.2015.04.001. URL https://www.sciencedirect.com/science/article/pii/S2214579615000271. Big Data, Analytics, and High-Performance Computing.

Fabio Alessandrini and Eric Jondeau. ESG Investing: From Sin Stocks to Smart Beta. *The Journal of Portfolio Management*, 46(3):75–94, 2020. ISSN 0095-4918. doi: 10.3905/jpm.2020.46.3.075. URL https://jpm.pm-research.com/content/46/3/75.

Amir Amel-Zadeh, Jan-Peter Calliess, Daniel Kaiser, and Stephen Roberts. Machine Learning-based Financial Statement Analysis. *Available at SSRN 3520684*, November 2020. doi: http://dx.doi.org/10.2139/ssrn.3520684.

Dario Amodei and Danny Hernandez. AI and Compute, May 2018. URL https://openai.com/blog/ai-and-compute/.

Torben G. Andersen, Tim Bollerslev, Francis X. Diebold, and Heiko Ebens. The Distribution of Realized Stock Return Volatility. *Journal of Financial Economics*, 61(1):43–76, 2001. ISSN 0304-405X. doi: https://doi.org/10.1016/S0304-405X(01)00055-1. URL https://www.sciencedirect.com/science/article/pii/S0304405X01000551.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, pages 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi: 10.1145/3442188.3445922. URL https://doi.org/10.1145/3442188.3445922.

Y. Bengio, P. Simard, and P. Frasconi. Learning Long-term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

Yoshua Bengio. Learning Deep Architectures for AI. *Found. Trends Mach. Learn.*, 2(1):1–127, jan 2009. ISSN 1935-8237. doi: 10.1561/2200000006. URL https://doi.org/10.1561/2200000006.

Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005-2019. *arXiv e-prints*, art. arXiv:1911.13288, November 2019.

Elettra Bietti and Roxana Vatanparast. Data Waste. *Harvard International Law Journal*, 61, December 2019. URL https://ssrn.com/abstract=3584251.

Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81 (3):637–654, 1973. ISSN 00223808, 1537534X. URL http://www.jstor.org/stable/1831029.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. *arXiv e-prints*, art. arXiv:2005.14165, May 2020.

Andrea Bucci. Realized Volatility Forecasting with Neural Networks. *Journal of Financial Econometrics*, 18(3):502–531, 06 2020. ISSN 1479-8409. doi: 10.1093/jjfinec/nbaa008. URL https://doi.org/10.1093/jjfinec/nbaa008.

Han Cai, Ji Lin, and Song Han. Chapter 4 - Efficient Methods for Deep Learning. In E.R. Davies and Matthew A. Turk, editors, *Advanced Methods and Deep Learning in Computer Vision*, Computer Vision and Pattern Recognition, pages 159–190. Academic Press, 2022. ISBN 978-0-12-822109-9. doi: https://doi.org/10.1016/B978-0-12-822109-9.00013-8. URL https://www.sciencedirect.com/science/article/pii/B9780128221099000138.

Qingqing Cao, Niranjan Balasubramanian, and Aruna Balasubramanian. MobiRNN: Efficient Recurrent Neural Network Execution on Mobile GPU. *arXiv e-prints*, art. arXiv:1706.00878, June 2017.

Rodolfo C. Cavalcante, Rodrigo C. Brasileiro, Victor L.F. Souza, Jarley P. Nobrega, and Adriano L.I. Oliveira. Computational Intelligence and Financial Markets: A Survey and Future Directions. *Expert Systems with Applications*, 55:194–211, 2016. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2016.02.006. URL https://www.sciencedirect.com/science/article/pii/S095741741630029X.

Chartis Research Staff. AI in RegTech: a quiet upheaval. Technical report, Chartis Research and IBM, February 2019. URL https://www.chartis-research.com/technology/artificial-intelligence-ai/ai-regtech-quiet-upheaval-10726.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. *arXiv e-prints*, art. arXiv:2002.05709, February 2020.

Yu-Ying Chen, Wei-Lun Chen, and Szu-Hao Huang. Developing Arbitrage Strategy in High-Frequency Pairs Trading with Filterbank CNN Algorithm. In *2018 IEEE International Conference on Agents (ICA)*, pages 113–116. IEEE, 2018.

François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv e-prints*, art. arXiv:1610.02357, October 2016.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph,

Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling Language Modeling with Pathways. *arXiv e-prints*, art. arXiv:2204.02311, April 2022.

Gary Cook, Jude Lee, Tamina Tsai, Ada Kongn, John Deans, Brian Johnson, and Elizabeth Jardim. Clicking Clean: Who is winning the race to build a green internet? Technical report, Greenpeace, 2017. URL https://www.greenpeace.org/usa/fighting-climate-chaos/click-clean/.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. *arXiv e-prints*, art. arXiv:1511.00363, November 2015.

Felipe Arias Fogliano de Souza Cunha, Erick Meira, and Renato J. Orsato. Sustainable Finance and Investment: Review and research agenda. *Business Strategy and the Environment*, 30(8):3821–3838, 2021. doi: https://doi.org/10.1002/bse.2842. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/bse.2842.

Chris Daniel, Anoop Kumar Shukla, and Meeta Sharma. Applications of machine learning in harnessing of renewable energy. In Prashant V. Baredar, Srinivas Tangellapalli, and Chetan Singh Solanki, editors, *Advances in Clean Energy Technologies*, pages 177–187, Singapore, 2021. Springer Singapore. ISBN 978-981-16-0235-1.

Narayana Darapaneni, Anwesh Reddy Paduri, Himank Sharma, Milind Manjrekar, Nutan Hindlekar, Pranali Bhagat, Usha Aiyer, and Yogesh Agarwal. Stock Price Prediction using Sentiment Analysis and Deep Learning for Indian Markets. *arXiv e-prints*, art. arXiv:2204.05783, April 2022.

Jon Dennis, Charlie Kronick, Raymond Dhirani, Anthony Field, Karen Ellis, and Becky Jarvis. The Big Smoke: the global emissions of the UK financial sector. Technical report, Greenpeace UK, WWF UK, and South Pole, May 2021. URL https://www.southpole.com/publications/the-big-smoke-the-global-emissions-of-the-uk-financial-sector.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv e-prints*, art. arXiv:1810.04805, October 2018.

Zhichao Du, Menghan Wang, and Zhewen Xu. On Estimation of Value-at-Risk with Recurrent Neural Network. In *2019 Second International Conference on Artificial Intelligence for Industries (AI4I)*, pages 103–106, 2019. doi: 10.1109/AI4I46381.2019.00034.

EIA. Total Energy Production 2019, 2019. Data retrieved from the U.S. Energy Information Administration, https://www.eia.gov/international/overview/world.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. Beyond English-Centric Multilingual Machine Translation. *arXiv e-prints*, art. arXiv:2010.11125, October 2020a.

Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Remi Gribonval, Herve Jegou, and Armand Joulin. Training with Quantization Noise for Extreme Model Compression. *arXiv e-prints*, art. arXiv:2004.07320, April 2020b.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *arXiv e-prints*, art. arXiv:2101.03961, January 2021.

Franyell Antonio Silfa Feliz. *Energy-Efficient Architectures for Recurrent Neural Networks*. PhD thesis, Universitat Politècnica de Catalunya, Spain, 2021. Supervised by Josè María Arnau Montañés and Antonio Gonzàlez Colás.

Stephen P. Ferris and Karl P. Rykaczewski. Social Investing and Portfolio Management. *Business & Society*, 25(1):1–7, 1986. doi: 10.1177/000765038602500101. URL https://doi.org/10.1177/000765038602500101.

Carmina Fjellström. Long Short-Term Memory Neural Network for Financial Time Series. *arXiv e-prints*, art. arXiv:2201.08218, January 2022.

Wenbo Ge, Pooia Lalbakhsh, Leigh Isai, Artem Lenskiy, and Hanna Suominen. Neural Network-Based Financial Volatility Forecasting: A Systematic Review. *ACM Comput. Surv.*, 55(1), jan 2022. ISSN 0360-0300. doi: 10.1145/3483596. URL https://doi.org/10.1145/3483596.

Nikhil Gokhale, Ankur Gajjaria, Rob Kaye, and Dave Kuder. AI Leaders in Financial Services: Common traits of frontrunners in the artificial intelligence race. Technical report, Deloitte Insights, August 2019. URL https://www2.deloitte.com/uk/en/insights/industry/financial-services/artificial-intelligence-ai-financial-services-frontrunners.html.

Brandon Graver, Dan Rutherford, and Sola Zheng. CO2 Emissions from Commercial Aviation: 2013, 2018, and 2019. Technical report, The International Council on Clean Transportation, October 2020. URL https://theicct.org/publication/co2-emissions-from-commercial-aviation-2013-2018-and-2019/.

Randall Hanegraaf, Nicole Jonker, S. Mandley, and Jelle Miedema. Life Cycle Assessment of Cash Payments. *SSRN Electronic Journal*, 01 2018. doi: 10.2139/ssrn.3267868.

Boris Hanin and David Rolnick. How to Start Training: The Effect of Initialization and Architecture. *arXiv e-prints*, art. arXiv:1803.01719, March 2018.

Peter R. Hansen and Asger Lunde. A Forecast Comparison of Volatility Models: Does anything beat a GARCH(1,1)? *Journal of Applied Econometrics*, 20(7):873–889, 2005. doi: https://doi.org/10.1002/jae.800. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/jae.800.

Qinyao He, He Wen, Shuchang Zhou, Yuxin Wu, Cong Yao, Xinyu Zhou, and Yuheng Zou. Effective Quantization Methods for Recurrent Neural Networks. *arXiv e-prints*, art. arXiv:1611.10176, November 2016.

Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent Neural Networks for Time Series Forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2020.06.008. URL https://www.sciencedirect.com/science/article/pii/S0169207020300996.

Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Master's thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Leif Hockstad and L Hanel. Inventory of US Greenhouse Gas Emissions and Sinks. Technical report, Environmental System Science Data Infrastructure for a Virtual Ecosystem, 2018.

J. Ryan Hogarth. The Role of Climate Finance in Innovation Systems. *Journal of Sustainable Finance & Investment*, 2(3-4):257–274, 2012. doi: 10.1080/20430795.2012.742637. URL https://www.tandfonline.com/doi/abs/10.1080/20430795.2012.742637.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *arXiv e-prints*, art. arXiv:1609.07061, September 2016.

IEA. Transport sector CO2 emissions by mode in the Sustainable Development Scenario, 2000-2030. Technical report, International Energy Agency, Paris, January 2022. URL https://www.iea.org/data-and-statistics/charts/transport-sector-co2-emissions-by-mode-in-the-sustainable-development-scenario-2000-2030.

Dino Ienco, Roberto Interdonato, and Raffaele Gaetano. Supervised Level-Wise Pretraining for Recurrent Neural Network Initialization in Multi-Class Classification. *arXiv e-prints*, art. arXiv:1911.01071, November 2019.

Weiwei Jiang. Applications of Deep Learning in Stock Market Prediction: Recent Progress. *Expert Systems with Applications*, 184:115537, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2021.115537. URL https://www.sciencedirect.com/science/article/pii/S0957417421009441.

Patrick Judd, Jorge Albericio, Tayler Hetherington, Tor M. Aamodt, and Andreas Moshovos. Stripes: Bit-Serial Deep Neural Network Computing. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12, 2016. doi: 10.1109/MICRO.2016.7783722.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly Accurate Protein Structure Prediction with AlphaFold. *Nature*, 596(7873):583–589, Aug 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2. URL https://doi.org/10.1038/s41586-021-03819-2.

Ha Young Kim and Chang Hyun Won. Forecasting the Volatility of Stock Price Index: A hybrid model integrating LSTM with multiple GARCH-type models. *Expert Systems with Applications*, 103:25–37, 2018. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2018.03.002. URL https://www.sciencedirect.com/science/article/pii/S0957417418301416.

Alison Kirsch, Grant Marr, Jason Opena Disterhoft, Henrieke Butijn, Johan Frijns, Maaike Beenes, Alberto Saldamando, Mea Johnson, Collin Rees, David Tong, Kyle Gracey, Lorne Stockman, Clement Faul, Maude Lentilhac, Ryan Cooper, Yann Louvel, Adele Shraiman, Ben Cushing, Julia Dubslaff, and Katrin Ganswindt. Banking on Climate Chaos: Fossil Fuel Finance Report 2022. Technical report, Rainforest Action Network, BankTrack, Indigenous Environmental Network, Oil Change International, Reclaim Finance, and Sierra Club, 2022. URL https://www.bankingonclimatechaos.org/.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Mohit Kumar, Xingzhou Zhang, Liangkai Liu, Yifan Wang, and Weisong Shi. Energy-Efficient Machine Learning on the Edges. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 912–921, 2020. doi: 10.1109/IPDPSW50202.2020.00153.

Satish Kumar, Dipasha Sharma, Sandeep Rao, Weng Marc Lim, and Sachin Kumar Mangla. Past, Present, and Future of Sustainable Finance: Insights from big data analytics through machine learning of scholarly research. *Annals of Operations Research*, Jan 2022. ISSN 1572-9338. doi: 10.1007/s10479-021-04410-8. URL https://doi.org/10.1007/s10479-021-04410-8.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the Carbon Emissions of Machine Learning. *arXiv e-prints*, art. arXiv:1910.09700, October 2019.

Salim Lahmiri. Modeling and predicting historical volatility in exchange rate markets. *Physica A: Statistical Mechanics and its Applications*, 471:387–395, 2017. ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2016.12.061. URL https://www.sciencedirect.com/science/article/pii/S0378437116310305.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv e-prints*, art. arXiv:1909.11942, September 2019.

Alex LaPlante and Alexey Rubtsov. Artificial Neural Networks in Financial Modelling. Technical report, Global Risk Institute, 55 University Avenue, Suite 1801, Toronto, March 2019. URL https://globalriskinstitute.org/publications/artificial-neural-networks-in-financial-modelling/.

Zhixi Li and Vincent W. L. Tam. A Comparative Study of a Recurrent Neural Network and Support Vector Machine for Predicting Price Movements of Stocks of Different Volatilites. *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, 2017.

Zehui Lin, Xiao Pan, Mingxuan Wang, Xipeng Qiu, Jiangtao Feng, Hao Zhou, and Lei Li. Pre-training Multilingual Neural Machine Translation by Leveraging Alignment Information. *arXiv e-prints*, art. arXiv:2010.03142, October 2020.

Zachary C. Lipton, John Berkowitz, and Charles Elkan. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv e-prints*, art. arXiv:1506.00019, May 2015.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *arXiv e-prints*, art. arXiv:2107.13586, July 2021.

Walter Hugo Lopez Pinaya, Sandra Vieira, Rafael Garcia-Dias, and Andrea Mechelli. Chapter 11 - Autoencoders. In Andrea Mechelli and Sandra Vieira, editors, *Machine Learning*, pages 193–208. Academic Press, 2020. ISBN 978-0-12-815739-8. doi: https://doi.org/10.1016/B978-0-12-815739-8.00011-0. URL https://www.sciencedirect.com/science/article/pii/B9780128157398000110.

Siblis Research Ltd. Total Market Capitalization of Public U.S. Companies, March 2022. Data retrieved from the Global Equity Valuations of Siblis Research, https://siblisresearch.com/data/us-stock-market-value/.

Racine Ly, Fousseini Traore, and Khadim Dia. Forecasting Commodity Prices Using Long Short-Term Memory Neural Networks. *arXiv e-prints*, art. arXiv:2101.03087, January 2021.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the Limits of Weakly Supervised Pretraining. *arXiv e-prints*, art. arXiv:1805.00932, May 2018.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2019.04.014. URL https://www.sciencedirect.com/science/article/pii/S0169207019301128. M4 Competition.

Jens Malmodin and Dag Lundén. The Energy and Carbon Footprint of the Global ICT and E&M Sectors 2010–2015. *Sustainability*, 10(9), 2018. ISSN 2071-1050. doi: 10.3390/su10093027. URL https://www.mdpi.com/2071-1050/10/9/3027.

Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith, and Jonathan Koome. Recalibrating Global Data Center Energy-use Estimates. *Science*, 367(6481):984–986, 2020. doi: 10.1126/science.aba3758. URL https://www.science.org/doi/abs/10.1126/science.aba3758.

Akib Mashrur, Wei Luo, Nayyar Zaidi, and Antonio Robles-Kelly. Machine Learning for Financial Risk Management: A Survey. *IEEE Access*, 8:203203–203223, 01 2020. doi: 10.1109/ACCESS.2020.3036322.

Srishti Mehra, Robert Louka, and Yixun Zhang. ESGBERT: Language Model to Help with Classification Tasks Related to Companies Environmental, Social, and Governance Practices. *arXiv e-prints*, art. arXiv:2203.16788, March 2022.

Iota Kaousar Nassr, Robert Patalano, Pamela Duffin, and Ed Smiley. Artificial Intelligence, Machine Learning and Big Data in Finance: Opportunities, Challenges and Implications for Policy Makers. Technical report, OECD, August 2021. URL https://www.oecd.org/finance/artificial-intelligence-machine-learning-big-data-in-finance.htm.

Salih N. Neftci. Chapter 15 - Volatility as an Asset Class and the Smile. In Salih N. Neftci, editor, *Principles of Financial Engineering (Second Edition)*, Academic Press Advanced Finance, pages 439–477. Academic Press, Boston, second edition edition, 2008. doi: https://doi.org/10.1016/B978-0-12-373574-4.50018-4. URL https://www.sciencedirect.com/science/article/pii/B9780123735744500184.

Joachim Ott, Zhouhan Lin, Ying Zhang, Shih-Chii Liu, and Yoshua Bengio. Recurrent Neural Networks With Limited Numerical Precision. *arXiv e-prints*, art. arXiv:1611.07065, November 2016.

Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer. Deep Learning for Financial Applications : A Survey. *Applied Soft Computing*, 93:106384, 2020. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2020. 106384. URL https://www.sciencedirect.com/science/article/pii/S1568494620303240.

Maximilian Palmié, Joakim Wincent, Vinit Parida, and Umur Caglar. The Evolution of the Financial Technology Ecosystem: An introduction and agenda for future research on disruptive innovations in ecosystems. *Technological Forecasting and Social Change*, 151:119779, 2020. ISSN 0040-1625. doi: https://doi.org/10.1016/j.techfore.2019. 119779. URL https://www.sciencedirect.com/science/article/pii/S0040162519310595.

Fengchao Peng, Qiong Luo, and Lionel M. Ni. ACTS: An Active Learning Method for Time Series Classification. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 175–178, 2017. doi: 10.1109/ICDE .2017.68.

Kuashuai Peng and Guofeng Yan. A Survey on Deep Learning for Financial Risk Prediction. *Quantitative Finance and Economics*, 5(4):716–737, 2021. ISSN 2573-0134. doi: 10.3934/QFE.2021032. URL https://www.aimspres s.com/article/doi/10.3934/QFE.2021032.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettle-moyer. Deep contextualized word representations. *arXiv e-prints*, art. arXiv:1802.05365, February 2018.

Ser-Huang Poon and Clive W.J. Granger. Forecasting Volatility in Financial Markets: A Review. *Journal of Economic Literature*, 41(2):478–539, June 2003. doi: 10.1257/002205103765762743. URL https://www.aeaweb.org/artic les?id=10.1257/002205103765762743.

Ioana-Stefania Popescu, Claudia Hitaj, and Enrico Benetto. Measuring the Sustainability of Investment Funds: A critical review of methods and frameworks in sustainable finance. *Journal of Cleaner Production*, 314:128016, 2021. ISSN 0959-6526. doi: https://doi.org/10.1016/j.jclepro.2021.128016. URL https://www.sciencedirect.com/sc ience/article/pii/S0959652621022344.

Joseph Power, Jordan McDonald, So Lefebvre, and Tom Coleman. The Time to Green Finance: CDP Financial Ser-vices Disclosure Report 2020. Technical Report DOE-SLC-6903-1, CDP Worldwide, Great Tower Street, London, 2020. URL https://www.cdp.net/en/research/global-reports/financial-services-disclosure-report-20 20.

Ling Qi, Matloob Khushi, and Josiah Poon. Event-Driven LSTM For Forex Price Prediction. *arXiv e-prints*, art. arXiv:2102.01499, January 2021.

Eghbal Rahimikia and Ser-Huang Poon. Machine Learning for Realised Volatility Forecasting. *SSRN Electronic Journal*, 10 2020. doi: 10.2139/ssrn.3707796.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. A Survey of Deep Active Learning. *arXiv e-prints*, art. arXiv:2009.00236, August 2020.

German Rodikov and Nino Antulov-Fantulin. Can LSTM outperform volatility-econometric models? *arXiv e-prints*, art. arXiv:2202.11581, February 2022.

Frank Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65 6:386–408, 1958.

I. Sadgali, N. Sael, and F. Benabbou. Performance of Machine Learning Techniques in the Detection of Financial Frauds. *Procedia Computer Science*, 148:45–54, 2019. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.20 19.01.007. URL https://www.sciencedirect.com/science/article/pii/S1877050919300079. THE SECOND INTERNATIONAL CONFERENCE ON INTELLIGENT COMPUTING IN DATA SCIENCES, ICDS2018.

Alaa Sagheer and Mostafa Kotb. Unsupervised pre-training of a deep lstm-based stacked autoencoder for multivariate time series forecasting problems. *Scientific Reports*, 9(1):19038, Dec 2019. ISSN 2045-2322. doi: 10.1038/s41598 -019-55320-6. URL https://doi.org/10.1038/s41598-019-55320-6.

A. L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959. doi: 10.1147/rd.33.0210.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *arXiv e-prints*, art. arXiv:1907.10597, July 2019.

Deepak Kumar Sharma, Mayukh Chatterjee, Gurmehak Kaur, and Suchitra Vavilala. 3 - Deep Learning Applications for Disease Diagnosis. In Deepak Gupta, Utku Kose, Ashish Khanna, and Valentina Emilia Balas, editors, *Deep Learning for Medical Applications with Unique Data*, pages 31–51. Academic Press, 2022. ISBN 978-0-12-824145-5. doi: https://doi.org/10.1016/B978-0-12-824145-5.00005-8. URL https://www.sciencedirect.com/science/article/pii/B9780128241455000058.

Jimeng Shi, Mahek Jain, and Giri Narasimhan. Time Series Forecasting (TSF) Using Various Deep Learning Models. *arXiv e-prints*, art. arXiv:2204.11115, April 2022.

Slawek Smyl. A Hybrid Method of Exponential Smoothing and Recurrent Neural Networks for Time Series Forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2019.03.017. URL https://www.sciencedirect.com/science/article/pii/S0169207019301153. M4 Competition.

David R. So, Chen Liang, and Quoc V. Le. The Evolved Transformer. *arXiv e-prints*, art. arXiv:1901.11117, January 2019.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and Policy Considerations for Deep Learning in NLP. *arXiv e-prints*, art. arXiv:1906.02243, June 2019.

Ankit Thakkar and Kinjal Chaudhari. A Comprehensive Survey on Deep Neural Networks for Stock Market: The need, challenges, and future directions. *Expert Systems with Applications*, 177:114800, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2021.114800. URL https://www.sciencedirect.com/science/article/pii/S0957417421002414.

Allan Timmermann and Clive W.J. Granger. Efficient Market Hypothesis and Forecasting. *International Journal of Forecasting*, 20(1):15–27, 2004. ISSN 0169-2070. doi: https://doi.org/10.1016/S0169-2070(03)00012-8. URL https://www.sciencedirect.com/science/article/pii/S0169207003000128.

P. Tino, C. Schittenkopf, and G. Dorffner. Financial Volatility Trading using Recurrent Neural Networks. *IEEE Transactions on Neural Networks*, 12(4):865–874, 2001. doi: 10.1109/72.935096.

Avraam Tsantekidis, Nikolaos Passalis, and Anastasios Tefas. Chapter 5 - Recurrent Neural Networks. In Alexandros Iosifidis and Anastasios Tefas, editors, *Deep Learning for Robot Perception and Cognition*, pages 101–115. Academic Press, 2022. ISBN 978-0-323-85787-1. doi: https://doi.org/10.1016/B978-0-32-385787-1.00010-5. URL https://www.sciencedirect.com/science/article/pii/B9780323857871000105.

Sandra Van Der Laan and Nina Lansbury. Socially Responsible Investing and Climate Change: Contradictions and Challenges. *Australian Accounting Review*, 14(34):21–30, 2004. doi: https://doi.org/10.1111/j.1835-2561.2004.tb00237.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1835-2561.2004.tb00237.x.

Paul Walsh, Jhilam Bera, Vibhu Saujanya Sharma, Vikrant Kaulgud, Raghotham M Rao, and Orlaith Ross. Sustainable AI in the Cloud: Exploring Machine Learning Energy Use in the Cloud". In *2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, pages 265–266, 2021. doi: 10.1109/ASEW52652.2021.00058.

Kafeng Wang, Xitong Gao, Yiren Zhao, Xingjian Li, Dejing Dou, and Cheng-Zhong Xu. Pay Attention to Features, Transfer Learn Faster CNNs. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxyCeHtPB.

Shuo Wang, Zhe Li, Caiwen Ding, Bo Yuan, Yanzhi Wang, Qinru Qiu, and Yun Liang. C-LSTM: Enabling Efficient LSTM using Structured Compression Techniques on FPGAs. *arXiv e-prints*, art. arXiv:1803.06305, March 2018.

Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. Chapter 10 - Deep Learning. In Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal, editors, *Data Mining (Fourth Edition)*, pages 417–466. Morgan Kaufmann, fourth edition edition, 2017. ISBN 978-0-12-804291-5. doi: https://doi.org/10.1016/B978-0-12-8042 91-5.00010-6. URL https://www.sciencedirect.com/science/article/pii/B9780128042915000106.

Dongrui Wu, Chin-Teng Lin, and Jian Huang. Active Learning for Regression using Greedy Sampling. *Information Sciences*, 474:90–105, 2019. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2018.09.060. URL https://www.sciencedirect.com/science/article/pii/S0020025518307680.

Ruoxuan Xiong, Eric P. Nichols, and Yuan Shen. Deep Learning Stock Volatility with Google Domestic Trends. *arXiv e-prints*, art. arXiv:1512.04916, December 2015a.

Ruoxuan Xiong, Eric P. Nichols, and Yuan Shen. Deep Learning Stock Volatility with Google Domestic Trends. *arXiv e-prints*, art. arXiv:1512.04916, December 2015b.

Jingjing Xu, Wangchunshu Zhou, Zhiyi Fu, Hao Zhou, and Lei Li. A Survey on Green Deep Learning. *arXiv e-prints*, art. arXiv:2111.05193, November 2021.

Kaisheng Xu, Xu Shen, Ting Yao, Xinmei Tian, and Tao Mei. Greedy Layer-Wise Training of Long Short Term Memory Networks. In *2018 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6, 2018. doi: 10.1109/ICMEW.2018.8551584.

Cheng Yang, Shengnan Wang, Chao Yang, Yuechuan Li, Ru He, and Jingqiao Zhang. Progressively Stacking 2.0: A Multi-stage Layerwise Training Method for BERT Training Speedup. *arXiv e-prints*, art. arXiv:2011.13635, November 2020.

Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation*, 31(7):1235–1270, 07 2019. ISSN 0899-7667. doi: 10.1162/neco _a_01199. URL https://doi.org/10.1162/neco_a_01199.

Adamantios Zaras, Nikolaos Passalis, and Anastasios Tefas. Chapter 2 - Neural Networks and Backpropagation. In Alexandros Iosifidis and Anastasios Tefas, editors, *Deep Learning for Robot Perception and Cognition*, pages 17–34. Academic Press, 2022. ISBN 978-0-323-85787-1. doi: https://doi.org/10.1016/B978-0-32-385787-1.00007-5. URL https://www.sciencedirect.com/science/article/pii/B9780323857871000075.

Krzysztof Zarzycki and Maciej Ławryńczuk. LSTM and GRU Neural Networks as Models of Dynamical Processes Used in Predictive Control: A Comparison of Models Developed for Two Chemical Reactors. *Sensors*, 21:5625, 08 2021. doi: 10.3390/s21165625.

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into Deep Learning. *arXiv e-prints*, art. arXiv:2106.11342, June 2021.

Chao Zhang, Yihuang Zhang, Mihai Cucuringu, and Zhongmin Qian. Volatility Forecasting with Machine Learning and Intraday Commonality. *arXiv e-prints*, art. arXiv:2202.08962, February 2022.

Xingyao Zhang, Haojun Xia, Donglin Zhuang, Hao Sun, Xin Fu, Michael B. Taylor, and Shuaiwen Leon Song. ν-lstm: Co-designing highly-efficient large lstm training via exploiting memory-saving and architectural design opportunities. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 567–580, 2021. doi: 10.1109/ISCA52012.2021.00051.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients. *arXiv e-prints*, art. arXiv:1606.06160, June 2016.

Christoph Zimmer, Mona Meister, and Duy Nguyen-Tuong. Safe Active Learning for Time-Series Modeling with Gaussian Processes. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 2735–2744, Red Hook, NY, USA, 2018. Curran Associates Inc.