

https://dspace.org/

Anomaly detection using time series forecasting with deep learning

Mathonsi, Thabang

2022
https://hdl.handle.net/10539/37141

Downloaded from a DSpace repository. https://dspace.org/

THESIS

Anomaly Detection Using Time Series Forecasting with Deep Learning

Author:
Thabang MATHONSI

Supervisor:
Prof. Terence VAN ZYL



UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

submitted to
the Faculty of Science, in fulfilment of the requirements for the degree
of
Doctor of Philosophy

in the

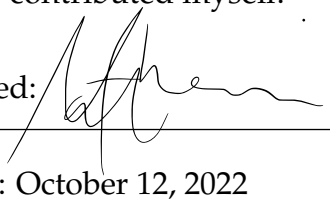
school of Computer Science and Applied Mathematics

October 12, 2022

Declaration of Authorship

I, Thabang MATHONSI, declare that this thesis titled, "Anomaly Detection Using Time Series Forecasting with Deep Learning" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: October 12, 2022

UNIVERSITY OF THE WITWATERSRAND

Abstract

Faculty of Science
school of Computer Science and Applied Mathematics

Doctor of Philosophy

Anomaly Detection Using Time Series Forecasting with Deep Learning

by Thabang MATHONSI

Anomaly detection is increasingly researched by the academic community because of its growing importance in applications such as, for instance, monitoring sensor readings in autonomous vehicles, or diagnosing potential medical risks in health data. This thesis presents solutions to the anomaly detection problem with the aid of multivariate time series forecasting, uncertainty quantification, and explainable and interpretable artificial intelligence. Challenges with existing deep learning-based time series anomaly detection approaches include (i) large sets of parameters that may be computationally intensive to tune for non-parsimonious models, (ii) returning too many false positives rendering the techniques impractical for use, (iii) requiring labelled datasets for training which are often not prevalent in real life, (iv) temporal dependence inherent in the data, as well as (v) complex dependency and cross-correlation between the covariates that may be hard to capture. An interpretable statistics and deep learning-based hybrid anomaly detection method is introduced, which overcomes these challenges. By systematically building anomaly detection machinery that is firstly good at forecasting, secondly satisfactorily quantifies the associated uncertainty, and thirdly, is interpretable, the presented methodology suggests that hybrid models show significant promise in terms of accuracy, efficiency, and practical usage. The extensive experimental results indicate that interpretable hybrid approaches can have a significant impact on anomaly detection as a research field, and the interpretability of such models can be useful for practitioners.

Acknowledgements

Thank you to my research supervisor, Prof. Terence van Zyl, for your time and expertise.

Thank you to my wife, Matankiso, for your love, support, and continued understanding. To my daughter Manelisi, you are my compass. Thank you to my sister Nonhlanhla for always believing in me.

In memory of my parents, Titi and Makhutso Mathonsi.

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iii

1 Introduction 1
- 1.1 Focus Themes . . . 1
  - 1.1.1 Anomaly Detection . . . 2
  - 1.1.2 Multivariate Forecasting and Uncertainty Quantification 2
  - 1.1.3 Interpretability and Explainability . . . 3
- 1.2 Thesis Introductory Background . . . 4
  - 1.2.1 Research Problem Background . . . 4
  - 1.2.2 Research Problem Specifics . . . 5
- 1.3 Problem Statement . . . 7
- 1.4 Significance and Motivation . . . 7
- 1.5 Research Aims and Objectives . . . 8
- 1.6 Thesis . . . 9
- 1.7 Research Questions . . . 9
- 1.8 Delineations, Limitations and Assumptions . . . 9
- 1.9 Derived Ph.D. Research Output . . . 10
- 1.10 Outline . . . 10

2 Research Methodology Overview 12
- 2.1 Research Design . . . 12
- 2.2 Results Data . . . 13
  - 2.2.1 Data Selection and Motivation . . . 13
  - 2.2.2 Air Quality Dataset . . . 13
  - 2.2.3 Power Consumption Dataset . . . 14
  - 2.2.4 Bike Sharing Dataset . . . 14
  - 2.2.5 Bearings Dataset . . . 15
  - 2.2.6 Webscope Benchmark Dataset . . . 15

2.2.7 Our World in Data COVID-19 Dataset . . . 15
2.2.8 Power Systems Machine Learning . . . 19
2.2.9 Ontario COVID-19 Testing Dataset . . . 20
2.2.10 Deaths Involving COVID-19 . . . 21
2.2.11 Nordpool Power Demand . . . 21
2.2.12 Large-scale Annotated Dataset for Energy Anomaly Detection . . . 21
2.2.13 Conclusion . . . 22
2.3 Results Analysis . . . 22
2.3.1 Forecasting . . . 23
2.3.2 Uncertainty Quantification . . . 24
2.3.3 Anomaly Detection . . . 24
2.3.4 Statistical Significance . . . 25
2.4 Conclusion . . . 25

3 Deep Learning and Statistical-based Models for Anomaly Detection 27
3.1 Introduction . . . 28
3.1.1 Recent Advances and the Current State-of-the-Art . . . 29
3.1.2 Contribution . . . 31
3.2 Methodology . . . 32
3.2.1 Forecasting . . . 32
AQD . . . 32
HPC . . . 32
BSD . . . 32
NBD . . . 32
YWB . . . 34
3.2.2 Preprocessing . . . 34
3.2.3 Algorithms . . . 35
Classical Statistical Models . . . 35
Deep Learning Models . . . 35
Computational Methods for Prediction Interval Construction . . . 41
3.2.4 Analysis . . . 42
AQD . . . 42
HPC . . . 43
BSD . . . 43
NBD . . . 44
YWB . . . 44

Overall . . . 44
3.2.5 Evaluation . . . 45
3.2.6 Recurrent Layer Dropout . . . 45
3.2.7 Bootstrapping . . . 46
3.2.8 Anomaly Detection . . . 47
3.3 Results and Discussion . . . 49
3.3.1 Forecasting . . . 49
3.3.2 Prediction Intervals . . . 49
3.3.3 Anomaly Detection . . . 52
3.4 Conclusion . . . 52

4 Statistics and Deep Learning-based Hybrid Model 55
4.1 Introduction . . . 56
4.1.1 Recent Advances and the Current State-of-the-Art . . . 58
Long Short-Term Memory in Morbidity and Mortality Modelling . . . 58
Statistics and Deep Learning-based Hybrid Models . . . 59
Multivariate Exponential Smoothing . . . 60
4.1.2 Motivation . . . 60
4.1.3 Contribution . . . 61
4.2 Methodology . . . 61
4.2.1 Preprocessing layer . . . 62
4.2.2 Deep Learning Layer . . . 64
4.2.3 Benchmarks . . . 67
4.2.4 Model Hyperparameter Tuning . . . 68
4.2.5 Metrics . . . 70
4.3 Results and Discussion . . . 70
4.3.1 COVID-19 Positive Testing Percentage by Age Group . . . 70
4.3.2 Deaths Involving COVID-19 by Fatality Type . . . 72
4.3.3 COVID-19 Morbidity and Mortality within a Localized Context . . . 74
Forecast Performance for SADC . . . 75
Prediction Interval Performance for SADC . . . 77
Forecast Performance for South Africa . . . 79
Prediction Interval Performance for South Africa . . . 81
Effects of Variability on Model Performance . . . 84
4.4 Conclusion . . . 86

5 Hybrid Model for Interpretable Anomaly Detection **88**
5.1 Introduction . . . 89
5.1.1 Recent Advances and the Current State-of-the-Art . . . 90
Explainable Anomaly Detection . . . 91
Explainability Systems and Methods . . . 92
5.1.2 Motivation . . . 94
5.1.3 Contribution . . . 95
5.2 Methodology . . . 96
5.2.1 Problem Statement for Main Experimental Study . . . 96
5.2.2 Univariate Studies . . . 97
5.2.3 Algorithms . . . 97
Classical Models . . . 98
Deep Learning-based Models . . . 99
5.2.4 Anomaly Detection . . . 101
5.2.5 Interpretability . . . 102
5.2.6 Analysis . . . 103
5.3 Results and Discussion . . . 104
5.3.1 Univariate Modelling and Anomaly Detection . . . 104
5.3.2 Multivariate Anomaly Detection . . . 106
5.3.3 Interpretabilty and Explainability . . . 109
5.4 Conclusion . . . 111

6 Conclusion **113**
6.1 Findings and Research Output . . . 113
6.2 Discussion . . . 114
6.3 Future Work and Closing Remarks . . . 116

A Supplementary Algorithms and Theory **118**
A.1 Introduction . . . 118
A.2 The Perceptron Training Rule . . . 118
A.3 The Delta Training Rule and Gradient Descent . . . 120
A.4 Stochastic Gradient Descent . . . 123
A.5 Backpropagation . . . 124
A.6 Bootstrapping . . . 125
A.6.1 Nonoverlapping Block Bootstrap . . . 126
A.6.2 The Sieve Bootstrap . . . 126

B Code Implementation Details **129**
B.1 Introduction 129
B.2 Deep Learning and Statistical-based Models for Anomaly Detection 129
B.3 Statistics and Deep Learning-based Hybrid Model 130
B.4 Hybrid Model for Interpretable Anomaly Detection 130

Bibliography **131**

List of Figures

3.1 Separate Variable Plots of the Daily Power Consumption Data Spanning Four Years. 33
3.2 Bearing Sensor Test Frequency Data 33
3.3 Flowchart Diagrams Depicting the Deep Learning Model Architecture Configurations for AQD. 43
3.4 Dropout RMSE Score Distributions. 46
3.5 Flowchart Diagram Depicting the Proposed Deep Learning Model Architectures. 49
3.6 Overall Daily RMSE for Household Power Consumption Dataset. 51
3.7 Point Forecasts and Constructed Prediction Intervals for Bike Sharing Dataset. 52
3.8 F_1-Score Distributions for Each Dataset and All the Models. (A) AQD. (B) BSD. (C) HPC. (D) NBD. (E) YWB. 53
3.9 *Ed*-Score Distributions for Each Dataset and All the Models. (A) AQD. (B) BSD. (C) HPC. (D) NBD. (E) YWB. 53

4.1 MES-LSTM Data Flowchart. 65
4.2 MES-LSTM Architecture Diagram. 67
4.3 Forecast Accuracy Aggregated Across All Age Groups. 72
4.4 Forecast Accuracy Aggregated Across All Fatality Types. . . . 74
4.5 Forecast Accuracy in the SADC Region. (**A**) sMAPE. (**B**) RMSE. 76
4.6 Prediction Interval Accuracy in the SADC Region. (**A**) MIS_α ($\alpha = 0.05$). (**B**) MIS_α ($\alpha = 0.1$). (**C**) MIS_α ($\alpha = 0.2$). (**D**) CS_α ($\alpha = 0.05$). (**E**) CS_α ($\alpha = 0.1$). (**F**) CS_α ($\alpha = 0.2$). 78
4.7 Forecast Accuracy Distribution Boxplots for All Trials in South Africa. (**A**) sMAPE for *total cases*. (**B**) RMSE for *total cases*. (**C**) sMAPE for *total deaths*. (**D**) RMSE for *total deaths*. 80
4.8 Forecast Accuracy in South Africa. (**A**) sMAPE. (**B**) RMSE. . . 80
4.9 Prediction Interval Accuracy in South Africa. (**A**) MIS_α ($\alpha = 0.05$). (**B**) MIS_α ($\alpha = 0.1$). (**C**) MIS_α ($\alpha = 0.2$). (**D**) CS_α ($\alpha = 0.05$). (**E**) CS_α ($\alpha = 0.1$). (**F**) CS_α ($\alpha = 0.2$). 82

4.10 MES-LSTM Forecast Accuracy Ranked for Each Country in the SADC Region. (**A**) sMAPE for *total cases*. (**B**) sMAPE for *total deaths*. 85
4.11 MES-LSTM Prediction Interval Accuracy (Normalized MIS_α) Ranked for Each Country in the SADC Region. (**A**) *total cases* ($\alpha = 0.05$). (**B**) *total cases* ($\alpha = 0.1$). (**C**) *total cases* ($\alpha = 0.2$). (**D**) *total deaths* ($\alpha = 0.05$). (**E**) *total deaths* ($\alpha = 0.1$). (**F**) *total deaths* ($\alpha = 0.2$). 85

5.1 Area Under the ROC Curve Distribution Boxplot for All Trials. 105
5.2 Area Under the PR Curve Distribution Boxplot for All Trials. . 106
5.3 Area Under the ROC Curve Distribution Boxplot for All Trials. 107
5.4 Area Under the PR Curve Distribution Boxplot for All Trials. . 108
5.5 Mean Discovery Score for Explainer Techniques Applied to Top Four Anomaly Detectors. (**A**) LIME. (**B**) SHAP. 110

A.1 A Perceptron. 120
A.2 Error Surface for Linear Unit with Two Weights. The Vertical Line Indicates the Direction of Steepest Descent. 121
A.3 A Sigmoidal Threshold Unit. 124

List of Tables

1.1 Derived Publications and Research Output Associated with this Thesis. 10

2.1 Air Quality Data Description. 13
2.2 Power Consumption Data Description. 14
2.3 Bike Sharing Data Description. 15
2.4 OWID COVID-19 Dataset Sourcing Summary. 16
2.5 OWID COVID-19 Dataset Feature List. 16
2.6 Power Systems Machine Learning Data Description. 20
2.7 All Datasets and their Respective Experimentation Use-cases. 22

3.1 Dropout RMSE Score Distributions. 45
3.2 Anomaly Data Structure Summary. 47
3.3 Model Error Scores. 49
3.4 Prediction Interval Performance for Air Quality Data. 50
3.5 Prediction Interval Performance for Bike Sharing Data. 50
3.6 Prediction Interval Performance for Household Power Consumption Data. 50
3.7 Prediction Interval Performance for Bearings Data. 51
3.8 Prediction Interval Performance for Webscope Benchmark Data. 51

4.1 Configuration Grid for Deep Learning Layer Hyperparameter Search Space. 69
4.2 Forecast Error Averaged Over All Trials for each Age Group. . 71
4.3 Forecast Error Averaged Over All Trials and Aggregated for All Age Groups. 71
4.4 Forecast Accuracy Averaged Over All Trials for each Fatality Type. 73
4.5 Forecast Accuracy Averaged Over All Trials for each Fatality Type. 73
4.6 Forecast Accuracy Averaged Over All Trials for *total cases* in the SADC Region. 75

4.7 Forecast Accuracy Averaged Over All Trials for *total deaths* in the SADC Region. . . . 76
4.8 Prediction Interval Accuracy Averaged Over All Trials for *total cases* in SADC ($\alpha = 0.05$). . . . 77
4.9 Prediction Interval Accuracy Averaged Over All Trials for *total deaths* in SADC ($\alpha = 0.05$). . . . 77
4.10 Forecast Accuracy Distribution for All Trials for *total cases* in South Africa. . . . 79
4.11 Forecast Accuracy Distribution for All Trials for *total deaths* in South Africa. . . . 79
4.12 Prediction Interval Accuracy Distribution for All Trials for *total cases* in South Africa ($\alpha = 0.05$). . . . 81
4.13 Prediction Interval Accuracy Distribution for All Trials for *total deaths* in South Africa ($\alpha = 0.05$). . . . 81
4.14 Student's t-test for Forecast RMSE (H_0: Benchmark Forecasts Outperform MES-LSTM). . . . 83
4.15 Student's t-test for Forecast sMAPE (H_0: Benchmark Forecasts Outperform MES-LSTM). . . . 83
4.16 Student's t-test for Prediction Interval MIS_α (H_0: Benchmark Prediction Intervals Outperform MES-LSTM). . . . 83
4.17 Student's t-test for Prediction Interval Coverage (H_0: Benchmark Prediction Intervals Outperform MES-LSTM). . . . 83
4.18 Diebold-Mariano Test for Forecast Accuracy (H_0: Benchmark Forecasts Outperform MES-LSTM). . . . 84

5.1 Summary of Changes from Rocket to MiniRocket [210]. . . . 100
5.2 Area Under Receiver Operating Characteristic Curve for All Models Over All Trials. . . . 104
5.3 Area Under Precision-Recall Curve for All Models Over All Trials. . . . 105
5.4 Area Under Receiver Operating Characteristic Curve for All Models Over All Trials. . . . 107
5.5 Area Under Precision-Recall Curve for All Models Over All Trials. . . . 108
5.6 Student's t-test for Testing Significance of auROC Performance Results (H_0: Benchmark Models Outperform MES-LSTM). . . . 109
5.7 Student's t-test for Testing Significance of auPR Performance Results (H_0: Benchmark Models Outperform MES-LSTM). . . . 109

5.8 Mean Discovery Score for LIME Applied to Top Four Anomaly Detectors. 110
5.9 Mean Discovery Score for SHAP Applied to Top Four Anomaly Detectors. 110

List of Algorithms

1 Cascade-Correlation Neural Network Process [1]. 37
2 Echo State Network Reservoir Computing. 42
3 Residual Bootstrapping. 46
4 Anomaly Detection. 48
5 The Sieve Bootstrap. 127

List of Abbreviations

AQD	Air Quality Dataset
AR	autoregressive
auPR	area under Precision-Recall
auROC	area under Receiver Operating Characteristic
Bi-LSTM	Bidirectional Long Short-Term Memory
BNN	Bayesian Neural Network
BP	backpropagation
BSD	Bike Sharing Dataset
CBOSS	Contractable Bag of Symbolic-Fourier Approximation Symbols
COTE	Collective Of Transformation-based Ensembles
COVID-19	Coronavirus Disease of 2019
CS$_\alpha$	Coverage Score
dDTW	dependent Dynamic Time Warping
DeepAR	Deep Autoregressive Recurrent neural network
DeepLIFT	Deep Learning Important FeaTures
DM	Diebold-Mariano
ed-LSTM	encoder-decoder Long Short-Term Memory
Ed-**score**	Early detection score
ES	Exponential Smoothing
ESN	Echo State Networks
ES-RNN	Exponential Smoothing Recurrent Neural Network
FPR	False Positive Rate
GAP	Global Average Pooling
GD	Gradient Descent
GiniReg	Gini Regularization
Grad-CAM	Gradient-weighted Class Activation Mapping
GRU	Gated Recurrent Units
HIVE-COTE	Hierarchical Voting Collective of Transformation-based Ensembles
HPC	Household Power Consumption
ICU	Intensive Care Unit
iDTW	independent Dynamic Time Warping
IF	Isolation Forests
IS$_\alpha$	Interval Score
LEMNA	Local Explanation Method using Nonlinear Approximation
LGB	Light Gradient Boosting
LIME	Local Interpretable Model-agnostic Explanations
LORE	Local rule-based explanations

LReLU	Leaky Rectified Linear Units
LSM	Liquid-state Machines
LSTM	Long Short-Term Memory
L2X	Learning to explain
MAE	Mean Absolute Error
MAPE	Mean Absolute Prediction Error
MAPLE	Model Agnostic Supervised Local Explanations
MC	Monte Carlo
MC-DCNN	Multi-Channels Deep Convolutional Neural Networks
MDS	Mean Discovery Score
MES-LSTM	Multivariate Exponential Smoothing Long Short-Term Memory
MiniRocket	Minimal random convolutional kernel transform
MIS$_\alpha$	Mean Interval Score
MLR	Multiple Linear Regression
MRI	Magnetic Resonance Imaging
M4	Makridakis-4 Forecasting Competition
M5	Makridakis-5 Forecasting Competition
NARNN	Nonlinear Autoregression Neural Network
NAB	Numenta Anomaly Detection Benchmark
NBD	NASA's Bearings Dataset
NBEATS	Neural Basis Expansion Analysis
NBEATSx	Neural Basis Expansion Analysis with exogenous factors
NNEuclid	one-Nearest neighbour with Euclidean distance
OLS	Ordinary Least Squares
OWID	Our World in Data
PIG	Path Integrated Gradients
ppv	proportion of positive values
PR	Precision-Recall
PSML	Power Systems Machine Learning
RC	Reservoir Computing
ReLU	Rectified Linear Units
ResNet	Residual Network
RESP	Responsibility Scores
RISE	Random Interval Spectral Ensemble
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
RuleReg	Rule Regularization
SADC	Southern African Development Community
SARIMAX	Seasonal Autoregressive Integrated Moving-Average with Exogenous Regressors
SBRL	Scalable Bayesian Rule Lists
SGD	Stochastic Gradient Descent
SHAP	Shapley additive explanations
sMAPE	symmetric Mean Absolute Percentage Error
sMIS$_\alpha$	scaled Mean Interval Score

SSE	Sum of Squared Errors
STC	Shapelet Transform Classifier
std	results standard deviation
STORN	stochastic recurrent networks
SVR	Support Vector Regressor
TapNet	Time series attentional prototype Network
TPR	True Positive Rate
VAE	variational autoencoders
VARMAX	Vector Autoregression Moving-Average with Exogenous Regressors
w.r.t.	with respect to
WYB	Yahoo Webscope Benchmark

List of Symbols

A_i	algorithm
$A_i\,(a)$	time taken for initial detection of anomaly a in D by algorithm A_i
a	anomaly
$\{a\}_D^{A_i}$	set of anomalies detected by algorithm A_i in dataset D
$\boldsymbol{a}$	time series
b	beginning index for window W
$\boldsymbol{b}$	time series
B_i	bootstrap blocks
C	correlation
$\mathcal{C}$	disturbances
card	cardinality
c_t	cell state
D	dataset
d	dilation
$\mathcal{D}_p$	path distance
E	squared error
e	end index for window W
E_t	the residual output error observed at t
$\bar{E}_t$	values of E_t averaged over J
F	unknown population
f	forget gate
$\mathcal{F}(\cdot)$	activation function
$\boldsymbol{f}_{dr}$	output functions of the dynamic reservoir nodes
$\boldsymbol{f}_{out}$	output functions of the output units
g	update gate
$\mathcal{G}_N$	sampling distribution
$\mathcal{H}$	model
$\boldsymbol{h}$	hidden vector sequence
$\mathrm{idx}(A_i\,(a))$	timestamp index
J	set of all training sequences
j	training sequence
k	number of covaraites
l	input gate
L_t	prediction interval lower bound at time t
l_t	level
M	teacher collection matrix
m	forecast horizon
N	total in-sample observations

$\mathcal{N}$	normal distribution
o	output gate
$\mathcal{O}(\cdot)$	oder of magnitude
P	seasonality length
q	number of bootstrap blocks
$\boldsymbol{q}$	concatenation of the (current) input, internal, and (previous) output vectors
$\mathcal{R}$	rank of an attribute's feature importance
S	Long Short-Term Memory size
$\mathcal{S}$	population statistic
$\mathcal{S}_N$	sample statistic
s_t	seasonal effect
$\hat{s}_{t,t+i}$	seasonal effect estimate
T	matrix or vector transposition
t	time
$\mathcal{U}$	uniform distribution
$\boldsymbol{u}$	activation states
U_t	prediction interval upper bound at time t
V	cascade correlation candidate unit's value
v	output predictands
$\bar{V}$	values of V averaged over J
$\mathcal{W}$	window
$\boldsymbol{w}$	weights
W_{bp}	weight matrix backprojection weight matrix for connections projecting back from the output unit to the internal
W_{dr}	fixed weight matrix inside dynamic reservoir
W_{hh}	matrix of weights from one hidden layer to the next hidden layer
W	composite weight matrix
W_{in}	weight matrix for connections between input nodes and internal units
$W_{h\hat{y}}$	matrix of weights from the last hidden layer to the output layer
W_{out}	weight matrix connections from the dynamic reservoir to output units
W_{xh}	matrix of weights from the input to the hidden layer
$\boldsymbol{x}^*$	bootstrap sample
$\boldsymbol{x}$	input sequence
$\hat{\boldsymbol{y}})$	output vector sequence
$\boldsymbol{X}_t$	matrix input
y_t	post-sample ground-truth value at time t
$\hat{y}_t$	the estimated forecast at time t
y_*	last identified anomalous observation
z	bus
$\mathcal{Z}$	set of buses in the entire system
S	sigmoid function

$\mathbb{1}$	indicator function
$\odot$	Hadamard product
∇	gradient
α	level of significance
β	highest features in terms of importance
γ	smoothing constant
δ	smoothing constant
ε	residuals
ϵ	some threshold
ζ_t	noise term centered at zero with constant variance
η	learning rate
$\{\vartheta\}_i$	bootstrap permutation
$\hat{\imath}_t$	estimated level
κ	principal contributing covariate
$\lvert\lambda_{\max}\rvert$	spectral radius
μ	mean
ξ	event type
ϱ	smoothing constants
ρ	warping path
σ	standard deviation
σ_N	sample standard deviation
ς	kernel length
$\hat{\tau}_t$	trend estimate
Ω	compact interval
Υ	compact interval

Chapter 1

Introduction

The main themes that this thesis focuses on are anomaly detection, time series forecasting, uncertainty quantification of said point forecasts using prediction intervals, and interpretability or explainability of the models and their inferences. Specifically, the research presented in this thesis is concerned with (i) using forecasts and their associated prediction intervals for multivariate anomaly detection and (ii) using model interpretability to demystify the predominantly black box nature of deep learning models. Each of these concepts are introduced in turn. Unless stated otherwise, the discussion is to be understood as pertaining specifically to time series.

1.1 Focus Themes

The central theme of this research is concerned with anomaly detection in the multivariate setting. In particular, multivariate forecasts and their associated quantified uncertainty are used as a mechanism for building machinery for interpretable anomaly detection. In this section, important concepts and related research are introduced. The background theory employed as some of the building blocks for the presented methodology is also discussed.

For one, forecasting using deep learning and classical statistical methods is considered. Secondly, uncertainty quantification is conducted for said forecasts in the form of prediction intervals. Some deep learning models are limited in that they do not have built-in constructs for deriving prediction intervals [2]. In such cases, one technique is to revert to computational means to synthesize the uncertainty quantities associated with each out-of-sample prediction.

Thirdly, anomaly detection in general is considered, as it is the main theme of the presented research. In particular, the presented research focuses on anomaly detection in the context of explainable and interpretable models.

1.1.1 Anomaly Detection

Anomaly detection is the act of finding nonconformity within data structures, and within multivariate time series data in particular. The non-conforming patterns are termed anomalies.

Some of the earliest explorations of anomaly detection in data arose from the statistics community [3]. Today, there is a plethora of industrial applications where anomaly detection is crucial and integral [4]. An example is with financial services providers, where unusually excessive transactions are flagged as anomalous for anti-fraud purposes. Similar preventative measures are taken in the insurance industry. Furthermore, there are similar risk-averse techniques applied to industries such as manufacturing, telecommunication, information technology, and others [5]. The usefulness of said techniques is in enabling the automatic deployment of pre-emptive responses that can be triggered in the event that anomalous activity is detected.

Other notable examples of applications include the health care sector, where a Magnetic Resonance Imaging (MRI) scan with anomalies may be interpreted as indicating malignant tumours [6], or fault detection in safety critical systems where, for example, anomalous readings from a self-driving vehicle could signify a fault in some component of the car [7]. These examples illustrate the importance of anomaly detection as a field of study, and for the interested reader, Chandola, Banerjee, and Kumar [8] review other disciplines where anomaly detection has been studied and its useful demonstrated.

1.1.2 Multivariate Forecasting and Uncertainty Quantification

Time series forecasting with the aid of machine learning tools has been conducted as far back as the works of Hu [9], whose application domain is weather prediction. Since then, new machine learning models have been developed, with advances dramatically improving output accuracy [10]–[13]. Machine learning-based forecasting applications include estimating realized volatility in finance [14] or projecting macroeconomic factors [15].

Makridakis, Spiliotis, and Assimakopoulos [16] noted an "*...equally important concern is that in addition to point forecasts, [machine learning] methods must also be capable of specifying the uncertainty around them, or alternatively providing confidence intervals. At present, the issue of uncertainty has not been included in the research agenda of the machine learning field, leaving a huge vacuum that must be filled as estimating the uncertainty in future predictions is as important as*

the forecasts themselves." Although one may argue that the quoted statement is false, there is strong evidence that in comparison, there has not been as much academic research output from the fields of uncertainty quantification in general, and probabilistic forecasting and prediction intervals in particular, as there have been in forecasting and anomaly detection, for instance [17].

The importance of quantifying predictive uncertainty can be illustrated in the autonomous vehicle setting, for instance, where it would suffice to state "*This vehicle will fail today.*" It is far more informative however to state, "*This specific vehicle component will fail within the next three to five hours, with 95% certainty.*" This example, which illustrates the importance of uncertainty quantification associated with point forecasts, extends to a lot of disciplines where forecasting is critical [2]. Quantifying the uncertainty adds more value to the forecast [18]. Uncertainty quantification also forms a pillar theme of the presented research as the forecasts and prediction intervals are used to detect anomalies in time series data.

1.1.3 Interpretability and Explainability

For stakeholders, it may not be sufficient to know that a specific observation is anomalous [19]. When the above example is extended to say, if more vehicles were to fail in the same manner as the one described, preventative measures become critical to avoid further potential harm. Explaining *why* an observation is anomalous can assist in unmasking underlying causality and prevent future recurrence [20]. This process of unmasking touches on the interpretability and explainability of deep learning models, which also forms a pillar theme of the presented research.

All of the above topics remain open research questions with fundamental real-world applications [21]–[23]. Challenges with existing deep learning-based anomaly detection approaches include (i) large sets of parameters that may be computationally intensive to tune for non-parsimonious models, (ii) returning too many false positives rendering the techniques impractical for use, (iii) requiring labelled datasets for training which are often not prevalent in real life, (iv) temporal dependence inherent in the data, as well as (v) complex dependency and cross-correlation between the covariates that may be hard to capture [24], [25].

In the rest of this chapter, deeper exposition is offered relating to each of the core concepts introduced above, and the problems with existing approaches are discussed and highlighted.

1.2 Thesis Introductory Background

Hawkins [26] classifies an observation as anomalous if it "*deviates so significantly from other observations as to arouse suspicion that a different mechanism generated it.*" Anomalies or outliers occur in instances where there are intentional malicious acts or system failure brought on by some operational defects. Real-time detection of such anomalies is crucial for aiding processes involving decision-making [27].

The terms *novelty detection* and *anomaly detection* are often used interchangeably [28]. However a distinction can be drawn between the two disciplines and more related research areas such as *rare event* analysis and *outlier detection* [29]. Novelty detection centers around the identification of observations conforming to yet undiscovered patterns and is usually characterised by one class in the training data representing the normal observations [29]. Rare event analysis can be thought of as an early binary time series classification problem where all classes are present in the training set [30]. Outlier detection can be static or dynamic in terms of temporal dependence within the data, and it is most prevalent in the unsupervised classification domain [31].

In some settings, the methods related to each of the three highlighted disciplines related to anomaly detection have been applied interchangeably [29]. Because of key methodological differences, the research presented in this thesis can be contextualised by limiting the scope to anomaly detection in a semi-supervised to supervised setting constrained by high class imbalance.

In the rest of this section, the research topic and the problem background are discussed in a broad sense, and then a definition of the research problem is offered.

1.2.1 Research Problem Background

Admittedly, several statistics-based models have enjoyed some success with anomaly detection problems [32]. If said classical probability-based techniques have been applied successfully, one may ask is there a need for deep learning-based anomaly detection at all? In addition to the problems highlighted above, further motivation is given through the three following reasons.

Firstly, classical anomaly detection methods usually share similar drawbacks. Statistical models are typically only useful in the univariate case [33], [34]. These univariate models have been extended and applied to multidimensional data using techniques such as the following. Li, Pedrycz, and

Jamal [35] propose a transformation based on Fuzzy C-Means clustering, which reduces the problem dimension space and enables univariate analysis as a proxy. A Hidden Markov Model is then applied for the inference stage. One drawback of Li, Pedrycz, and Jamal's approach is it neglects the full utility of the data's correlation information.

Secondly, the scalability attribute of deep learning-based algorithms has been demonstrated to extend to other domains including, for example, community detection. In this application domain, deep learning techniques outperform their classical counterparts such as spectral clustering in tasks involving high dimensional network data [25], [36]. Other domains this scalability attribute has been shown are image classification, recommender systems and semantic segmentation [37]. So, another motivation for deep learning is the hypothesis that it can scale well to applications involving large-scale anomaly detection.

Another concern with classical anomaly detection models is their usual dependence on labelled datasets [38]. These ground truth labels are also known as golden labels. Acquisition of such data in real life poses challenges such as the inexpensive process of labelling, or assuming the risk of incorporating human error in the labels, which would contaminate any further studies [39]. Even in instances where the obvious drawbacks do not occur and the classical models are trained successfully, most real-life problems are dynamic in nature and would require continuous retraining of the tuned models to ensure generalizability to new anomaly types [24].

As a final point, in a recent edition of one of the most contested global forecasting competitions, the Makridakis-4 (M4) Forecasting Competition [33], pure deep learning models are reported to outperform statistical methods at uncertainty quantification [33]. The ability of deep learning-based methods to output superior prediction intervals is also evidenced by studies conducted by Smyl [34] in the unidimensional setting and Mathonsi and van Zyl [40] in higher dimensions, for instance. So, another reason offered herein for the adoption of deep learning for anomaly detection tasks is the hypothesis that producing superior prediction interval bounds may lead to better performance at anomaly detection tasks.

1.2.2 Research Problem Specifics

Due to the challenges detailed above, the most general formulation of the anomaly detection problem is difficult to solve [41]. Notably, most of the

existing anomaly detection techniques are only able to solve some subset of the problem specifically formulated. Any such formulation is a result of numerous factors, including the inherent characteristics of the data, the scale of available labelled data, or even the specific type of anomalies to be detected, etc [42]. These factors that drive problem formulation are usually determined by the anomaly detection application domain. As anomaly detection scenarios usually suffer from a lack of sufficient labelled examples to train, a simple classification algorithm will fail and more innovative approaches are required [29].

A naïve method for anomaly detection is to simply raise an alarm whenever a pre-specified threshold is breached [43]. Such threshold is domain and use-case specific and is determined through experimentation or standards derived from historical observations. This level-based method reduces the time for manual monitoring but has the disadvantage of possibly raising many false alarms, especially in domains concerned with datasets that exhibit the non-stationarity property [44]. Even more problematic are anomalies which cannot be detected by thresholds but rather require the observation of multiple joint characteristics such as contextual anomalies, for instance [23].

Binary classification is the simplest method in terms of its underlying model and computational complexity [45]. However, a simple classification-based model requires an adequate amount of data samples for each of the normal as well as the anomalous class [45]. Unfortunately, in real-world applications, obtaining sufficient anomalous data samples is difficult [39]. The modelling may require training exclusively on normal examples, and subsequently detecting previously unseen patterns in new data samples, such as in the semi-supervised setting [46], [47].

Deep learning provides tools for handling immense amounts of data with distributed algorithms [48]. Furthermore, deep learning has the power to model complicated behaviour based on unknown underlying rules from a specific domain [49]. In general, a complex domain makes it very hard to implement an expert system, where task-specific assumptions are made based solely on a set of manually defined rules which requires extensive work by engineers with sufficient domain knowledge. In contrast, a deep learning algorithm can automatically derive the complex set of underlying rules from the data itself and encode them directly into an algorithm [25]. Subsequently, the resulting algorithm can potentially result in a more robust process for anomaly detection, as well as require less manual work by domain experts.

Furthermore, it may be argued that this ability of deep learning models to capture complex dependencies in a black-box fashion is among the biggest criticisms of deep learning [19]. In applications that require industry-wide regulation, it is important to understand how decisions are arrived at and how inferences are derived [50]. As such, unpacking the black-box nature of such models forms an additional thematic pillar of the research presented in this thesis.

In the rest of this chapter, the significance of this research is detailed, and the objectives are stated along with the aims to solve the specified problem set, and the research questions are outlined. Any limitations and assumptions that are relevant are also stated in this chapter, before an outline is offered describing the structure of the rest of the thesis.

1.3 Problem Statement

Having a fast, accurate, and interpretable anomaly detection model will allow real-time anomaly detection [51], and enable prevention of future recurrence due to understanding why such anomalies occur in the first place [52]. Deep learning does present several advantages over classical methods such as for example, better accuracy and the ability to handle larger datasets [36]. However, deep learning models are usually plagued by time-consuming inferencing, being computationally expensive to train and deploy, and the models are usually hard to interpret [4]. In response to these problems, this thesis presents a derivation, implementation and evaluation of a novel interpretable statistics and deep learning-based hybrid anomaly detection model.

1.4 Significance and Motivation

There are various works in academia that tackle each of the proposed steps, i.e. forecasting, quantifying uncertainty, detecting anomalies, and interpreting the model and the model inferences being covered by, for example, Makridakis, Spiliotis, and Assimakopoulos [53], Makridakis, Spiliotis, Assimakopoulos, *et al.* [54], Ruff, Kauffmann, Vandermeulen, *et al.* [55], and Guidotti, Monreale, Ruggieri, *et al.* [21] respectively. In this thesis, a consolidated approach is presented, which aims to unify these distinct streams of research.

There exists a need for efficient, effective, and interpretable techniques to model multivariate time series data with respect to various tasks [56]. One area where such models can be useful is for financial institutions, for

instance. In the financial sector, regulations mandate transparency. Furthermore, beyond accurate forecasts with quantifiable uncertainty, there is also a strong motivation for model explainability and interpretability [57].

1.5 Research Aims and Objectives

This research aims to develop a hybrid model that is parsimonious, thus computationally inexpensive to train with real-time inference. Such a model can be developed by extending classical statistical techniques and combining them with deep learning architectures. One aim is to systematically develop said novel model for detecting anomalies through empirically evaluating it at each phase of the proposed development pipeline: (i) multivariate forecasting, (ii) uncertainty quantification using prediction intervals, (iii) anomaly detection using said forecasts and associated prediction intervals, and (iv) interpreting the model and the model input space using explainable artificial intelligence techniques.

The above aims of this research are achieved through the following objectives:

- systematically review and implement some existing deep learning architectures, for example, the recurrent and reservoir classes, and variations thereof;
- build forecast machinery for multivariate time series data and construct associated prediction interval bounds, sometimes relying on computational synthesisation in instances where no standard implementations of probabilistic distributions are available within the deep learning models;
- using sourced datasets, empirically analyse the performance of deep learning and classical statistics-based models at forecasting tasks, construction of prediction intervals, and anomaly detection;
- develop a hybrid model and run similar empirical experiments based on real-world applications;
- analyse and discuss the results of the empirical evaluation;
- introduce analytic theory underpinning the methodology of said interpretable statistics and deep learning-based hybrid model;
- using explainable artificial intelligence techniques, add a layer of interpretability to the anomaly detection model; and

- using a novel evaluation metric, quantify how useful an anomaly detection model can be to human agents working with model inferences in a specific domain.

1.6 Thesis

The research hypothesis is that a statistics and deep learning-based hybrid model may be efficient and effective at multiple tasks related to multivariate time series modelling, i.e. (i) forecasting, (ii) uncertainty quantification using prediction intervals, (iii) anomaly detection using said forecasts and associated prediction intervals. In addition, the model may also be able to embed the useful functionality of interpretability.

1.7 Research Questions

The research questions include

1. How does one best construct prediction intervals for produced point forecasts where no probabilistic distributions exist within the forecasting model?
2. How does one merge the best characteristics of deep learning and classical models to form an interpretable hybrid model for multivariate forecasting, uncertainty quantification, and anomaly detection, that inherits the least possible negative traits from each?
3. How useful would interpretability and explainability be if added to such a model?
4. How would such a model perform compared to classical and deep learning counterparts at the aforementioned tasks?

1.8 Delineations, Limitations and Assumptions

There are recent advances with regard to hierarchical forecasting, where pure deep learning models have been shown to outperform pure statistical methods [58], [59]. Hierarchical forecasting falls beyond the scope of this thesis, however, the research herein also focuses on investigations in the multivariate with exogenous variables forecasting setting.

1.9 Derived Ph.D. Research Output

As previously discussed, this work is composed of several distinct but related research topics. Three experiments are presented, each focused on some aspects of the core topics. In large part, some aspects of the first two experiments of the work presented are adapted from my Ph.D. research articles that are published in double-blind peer-reviewed scientific journals and peer-reviewed conference proceedings. The final experiment is adapted from the latest results and forms part of a paper that has been submitted to an upcoming conference which features proceedings publication. Table 1.1 lists all the derived Ph.D. research publications and output associated with this thesis and the chapters they each relate to.

TABLE 1.1: Derived Publications and Research Output Associated with this Thesis.

Derived Output	**Chapter**	**Type**
Mathonsi and van Zyl [40]	3	Conference proceedings
Mathonsi and van Zyl [60]	3	Journal article
Mathonsi and van Zyl [61]	4	Journal article
Mathonsi and van Zyl [62]	5	Pre-print under review
https://github.com/zulucomputer/MES_LSTM	3, 4, 5	open-source code repository

1.10 Outline

The rest of this thesis is set up as follows. Chapter 2 outlines the research methodology, including a detailed description of the datasets employed for the empirical analysis component, as well as measures of accuracy employed and statistical significance tests conducted. Chapter 3 features a discussion of existing deep learning and statistical architectures, their design and training rules, and offers background theory. Chapter 3 and the two subsequent chapters, each detail a phase of the proposed experimentation pipeline. They offer a review of work done in the fields of forecasting, uncertainty quantification, anomaly detection, interpretability, and related fields. Each of these chapters expands the specific topic's literature review and methodology within the context of the main theme, as well as highlights the novel contribution at that specific sub-theme. Each of the three chapters can be viewed as a distinct, self-contained body of work, while also being interlinked and forming a contribution to the main topics. Results are presented in each of the

three chapters, as well as specific contextual conclusions. The thesis is concluded in Chapter 6 with a discussion that includes highlights from the research findings, summarizing key insights, offering perspectives, and discussing avenues for future research.

Chapter 2

Research Methodology Overview

This chapter presents an overview of the experiments conducted in this research. The design of the presented research methodology is discussed at a high level of abstraction. The datasets used in each experiment are presented here, and motivation for their selection is offered. This chapter also serves to discuss the analysis conducted on results presented in each of the three subsequent chapters (Chapters 3 - 5).

2.1 Research Design

The experimentation is segmented into three steps. First, the performance of deep learning and classical statistical methods is considered at (i) forecasting, (ii) uncertainty quantification, and (iii) anomaly detection. Second, a novel extension of a hybrid approach is presented and its performance at forecasting tasks and uncertainty quantification is analysed against pure deep learning and pure statistical methods, with an application to morbidity and mortality modelling. Third, said hybrid approach is further extended to an interpretable anomaly detection model, and its performance is tested against benchmark anomaly detection models, with an application to renewable energy in decarbonized grids.

Where there are overlapping concepts, say for instance a performance metric that is used in all the separate experiments, it is presented here. If the concept is unique to a particular experiment, it is presented in the relevant experimentation chapter. The literature review covering the state-of-the-art and methodology for each segment is expanded in the relevant chapter.

2.2 Results Data

2.2.1 Data Selection and Motivation

The chosen datasets come from a diverse array of sectors. Data from various sectors are used, ranging from climate to space aviation. The most important aspect when selecting the datasets for forecasting and uncertainty quantification, in particular, is ensuring they are distinctly different in terms of size, frequency, spatial and temporal resolution, application domain, and the number of variables. In terms of anomaly detection, in particular, the datasets are selected based on factors such as ensuring varied composition of normal to anomalous ratios of observations. This data sourcing methodology enables forecasting and uncertainty quantification for varied horizons, and anomaly detection under different settings, thus mitigating biases in model performance. All the datasets are also freely available and have been used extensively in related research. More focused motivation is offered in relation to specific experiments in the sections that follow.

2.2.2 Air Quality Dataset

One dataset is the Beijing Air Quality Dataset (AQD) [63] from UCI. It contains measurements of hourly weather and pollution level for five years spanning January 1st, 2010 to December 31st, 2014 at the United States embassy in Beijing, China. Meteorological data from the Beijing Capital International Airport are also included. The variable of interest is the level of pollution in the air (pm2.5 concentration), while Table 2.1 details the complete feature list.

TABLE 2.1: Air Quality Data Description.

Variable	Description
pm2.5	pm2.5 concentration
DEWP	dew point
TEMP	temperature
PRES	pressure
cbwd	combined wind direction
Iws	cumulated wind speed
Is	cumulated hours of snow
Ir	cumulated hours of rain

2.2.3 Power Consumption Dataset

The second dataset is UCI's Household Power Consumption (HPC) [64], a multivariate time series dataset detailing a single household's per-minute electricity consumption spanning December 2006 to November 2010.

With the date and time fields collapsed into a unique identifier, the remaining variables are described in Table 2.2.

TABLE 2.2: Power Consumption Data Description.

Variable	Description
global_active_power	total active power consumed (kW)
global_reactive_power	total reactive power consumed (kW)
global_intensity	minute-averaged current intensity (A)
sub_metering_1	active energy for the kitchen (W h)
sub_metering_2	active energy for laundry (W h)
sub_metering_3	active energy for climate control systems (W h)

A new variable is created

$$\text{sub_metering_4} = \frac{100}{6} \times \text{global_active_power} - \sum_{i=1}^{3} \text{sub_metering_}i, \quad (2.1)$$

that represents the per-minute active energy consumption not already captured by sub_meterings 1, 2, and 3. Active power is generated and consumed in the circuit, whereas reactive power flows back and forth within the electrical grid components like a pendulum but is theoretically useless. Watanabe, Stephan, and Aredes [65] offer a comprehensive explanation of these and related electrical engineering concepts.

2.2.4 Bike Sharing Dataset

Next, the UCI Bike Sharing Dataset (BSD) [66] is considered. It catalogues for each hour the number of bikes rented by customers of a bike-sharing service in Washington DC for the 2011 and 2012 calendar years. It includes features such as whether a particular day is a national holiday and the weather and seasonal information. Besides the date-time and index variables that are collapsed into a unique identifier, the dataset has 17 variables in total. A subset of variables is selected for the analysis in this thesis, as described in Table 2.3.

TABLE 2.3: Bike Sharing Data Description.

Variable	Description
holiday	boolean whether the day is holiday or not
weekday	day of the week
weathersit	1 if clear, few clouds, partly cloudy 2 if cloudy, mist, broken clouds, few clouds 3 if light snow or rain, thunderstorm, scattered clouds 4 if heavy rain, ice pallets, thunderstorm, snow mist, fog
hum	humidity
temp	normalized temperature in degrees Celsius
casual	count of casual users
registered	count of registered users
cnt	count of total rental bikes, sum of casual and registered

2.2.5 Bearings Dataset

The fourth dataset used is NASA's Bearings Dataset (NBD) [67]. It contains sensor readings on multiple bearings that are operated until failure under a constant load causing degradation. These are one-second vibration signal snapshots recorded at ten-minute intervals per file, where each file contains 20,480 sensor data points per bearing. It contains golden labels and has been considered extensively by other researchers in their experimentation; see, for example, Prosvirin, Islam, Kim, *et al.* [68].

2.2.6 Webscope Benchmark Dataset

The experimentation approaches are also evaluated using the Yahoo Webscope Benchmark dataset (YWB) [69], which is a highly cited benchmark data resource that has been used extensively in the research community [70]. It comprises large datasets with golden labels. YWB is particularly useful as it contains different kinds of anomalies, for instance, some are based on outliers and others on change points. The first benchmark, A1, is real site traffic to Yahoo and its related products. The other three, A2, A3, and A4, are synthetic. They all contain three months' worth of data points though of varying lengths, capture frequencies, seasonality patterns and the total number of labelled anomalies.

2.2.7 Our World in Data COVID-19 Dataset

Another resource utilized is the Our World in Data (OWID) coronavirus disease of 2019 (COVID-19) dataset [71], [72]. This dataset is aggregated from

various sources and includes historical data on the pandemic up to the date of publication, updated daily. A summary of the data and the aggregated data sources is shown in Table 2.4. Even though the actual databases are updated at daily, weekly, and other time frequencies, the aggregated datasets used in the study are all presented at a daily temporal resolution. This unified temporal resolution means no frequency harmonization is required.

TABLE 2.4: OWID COVID-19 Dataset Sourcing Summary.

Metrics	**Source**	**Updated**	**Countries**
Vaccinations	Official data[1]	Daily	217
Tests & positivity	Official data[1]	Weekly	136
Hospital & ICU	Official data[1]	Weekly	34
Confirmed cases	JHU CSSE COVID-19 Data	Daily	194
Confirmed deaths	JHU CSSE COVID-19 Data	Daily	194
Reproduction rate	Arroyo-Marioli F. *et al.*	Daily	184
Policy responses	Oxford Tracker	Daily	186
Other variables of interest	International organizations[2]	Fixed	240

The variables represent data related to confirmed cases, deaths, hospitalizations, intensive care unit (ICU) admissions, testing, as well as 56 other variables. Only a subset of the available variables is used, as detailed in the feature list shown in Table 2.5.

TABLE 2.5: OWID COVID-19 Dataset Feature List.

Variable	**Description**
total cases	Total confirmed cases of COVID-19
new cases	New confirmed cases of COVID-19
total cases[3]	Total confirmed cases of COVID-19[3]
new cases[3]	New confirmed cases of COVID-19[3]
total deaths	Total deaths attributed to COVID-19
new deaths	New deaths attributed to COVID-19
total deaths[3]	Total deaths attributed to COVID-19[3]
new deaths[3]	New deaths attributed to COVID-19[3]
ICU patients	COVID-19 patients in ICU on a given day
ICU patients[3]	COVID-19 patients in ICUs on a given day[3]

Continued on next page

[1] collated by the Our World in Data team
[2] including the United Nations, World Bank, and others
[3] per 1,000,000 people

Table 2.5 – continued from the previous page

Variable	**Description**
hosp patients	COVID-19 patients in the hospital on a given day
weekly ICU admissions	COVID-19 patients newly admitted to ICUs in a given week
weekly ICU admissions[3]	COVID-19 patients newly admitted to ICUs in a given week[3]
weekly hosp admissions	COVID-19 patients newly admitted to hospitals in a given week
weekly hosp admissions[3]	COVID-19 patients newly admitted to hospitals in a given week[3]
stringency index	Government Response Stringency Index: composite measure based on 9 response indicators
reproduction rate	Real-time estimate of the effective COVID-19 reproduction rate
total tests	Total tests for COVID-19
new tests	New tests for COVID-19 (only calculated for consecutive days)
positive rate	Share of COVID-19 tests that are positive, rolling 7-day average (inverse of tests per case)
tests per case	Tests conducted per new confirmed case of COVID-19, rolling 7-day average (inverse of positive rate)
total vaccinations	Total COVID-19 vaccination doses administered
people vaccinated	Total people who received at least one vaccine dose
people fully vaccinated	Total people who received all doses
new vaccinations	New COVID-19 vaccination doses administered (only calculated for consecutive days)
total vaccinations per hundred	Total COVID-19 vaccination doses administered per 100 people
people vaccinated per hundred	Total people who received at least one vaccine dose per 100 people

Continued on next page

Table 2.5 – continued from the previous page

Variable	**Description**
people fully vaccinated per hundred	Total people who received all doses prescribed by the vaccination protocol per 100 people
location	Geographical location
date	Date of observation
population	Population in 2020
population density	people divided by land area, measured in square kilometers
median age	Median age of the population, UN projection for 2020
aged 65 older	Share of the population that is 65 years and older most recent year available
aged 70 older	Share of the population that is 70 years and older in 2015
GDP per capita	Gross domestic product at purchasing power parity
extreme poverty	Share of the population living in extreme poverty
cardiovasc death rate	Death rate from cardiovascular disease in 2017
diabetes prevalence	Diabetes prevalence (% of population aged 20 to 79) in 2017
female smokers	Share of women who smoke, most recent year available
male smokers	Share of men who smoke, most recent year available
handwashing facilities	Share of the population with basic handwashing facilities
hospital beds per thousand	Hospital beds per 1,000 people
life expectancy	Life expectancy at birth in 2019
human development index	Composite average achievement in (*i*) a long, healthy life (*ii*) knowledge (*iii*) standard of living
excess mortality	Excess mortality P-scores for all ages

The OWID aggregated dataset is chosen over other available COVID-19 datasets because it does to some extent mitigate the concerns raised by Chandra, Jain, and Singh Chauhan [73], for example. These concerns include that the spread of any virus should be modelled using exogenous factors as well, such as the standard of living or the level of difficulty associated with accessing health services. It is not difficult to motivate that some exogenous factors not directly related to COVID-19 may be useful in the modelling process, and the OWID dataset is one such example. For instance, the OWID COVID-19 dataset includes covariates such as the *human development index*, an index tracking *extreme poverty*, and the general level of access to *handwashing facilities*.

2.2.8 Power Systems Machine Learning

Renewable energy from wind and solar farms can cause disturbances which may impede grid operational safety. Using sensors such as phasor measurement units, operators can assess the safety levels and pre-empt any compromise. Stakeholders, human agents, and domain experts are typically concerned with (*i*) *When is an event happening* (detection)*?* (*ii*) *What type of event is happening?* (classification from disturbances including *branch fault, branch tripping, bus fault, bus tripping, generator tripping, forced oscillation*); and (*iii*) *Where is the source of said event* (localization or explainability)?

The Power Systems Machine Learning (PSML) dataset [74] is used. The repository contains multiple machine learning-related tasks, but this thesis only focuses on the task concerning renewable power generation. The main aims of the selected task include early detection, accurate classification, and localization of dynamic disturbance events in decarbonized energy grids.

PSML is a combination of real-world load time series and synthesized active power time series of renewable generation, combined with real-world weather data. The renewable generation power is calculated based on the collected weather data of each load zone. The daily renewable power profile shows seasonal disparity, strong variation of renewable energy, and a significant load reduction during the emergence of the unprecedented COVID-19 global pandemic.

The dataset is presented at the millisecond, per-minute, and 5-minute temporal scales where the variables of interest are voltage, current, and power.

Only the millisecond time resolution is focused on, as a finer resolution offers more data points for training. The dataset was first published on 10 November 2021 and at the time of writing this there have been close to 5,000 downloads, indicating a keen interest in this resource from researchers and practitioners. A description of the data is given in Table 2.6.

TABLE 2.6: Power Systems Machine Learning Data Description.

Field	**Description**
Time	time in millisecond resolution
POWR # TO ## CKT ###	active power transferring in branch ### from bus # to bus #
VARS # TO # CKT ###	reactive power transferring in branch ### from bus # to #
VOLT ###	per-unit voltage magnitude at the bus ###
####.###.#	per-unit voltage magnitude of phase # at bus ### connecting to bus ####

PSML is chosen over others that are used as much or more widely in the research community from related research fields, such as, for instance, the Numenta Anomaly Detection Benchmark (NAB) [51] for several reasons. NAB does not readily offer metadata that can be used as a mechanism for explanation discovery. Furthermore, NAB presents some structural weaknesses that make it impractical for use, such as series with large missingness [75].

2.2.9 Ontario COVID-19 Testing Dataset

An experimental study is also conducted on the Ontario COVID-19 Testing Dataset (OTD) [76]. The data is aggregated from multiple sources including the Provincial Surveillance Information system, the Communicable Disease Reporting System and the Communicable Disease Outbreak Management system at Alberta Health Services, the Data Integration and Measurement Reporting database at Alberta Health Services, and the Provincial Immunization and Adverse Reaction to Immunization repository. OTD aims to track the positive testing rate by age group, and includes date, age group, and seven-day running average of percentage of positive tests in each age group. There are five age ranges grouped as $\{0-13, 14-17, 18-24, 25-64, \geq 65\}$.

2.2.10 Deaths Involving COVID-19

The ninth dataset considered in the Deaths Involving COVID-19 (DIC) [76] dataset from Ontario Health Ministry. The data aims to track the daily reported number of deaths involving COVID-19 by fatality type. The dataset is composed of variables including the reporting date of the deaths involving COVID-19, the total number of deaths confirmed as not involving COVID-19, the number of deaths with COVID-19 confirmed as the underlying cause of death, the number of deaths with COVID-19 having contributed to but not necessarily the main underlying cause, as well as the number of deaths where the cause of death is either unknown or missing. The *Cause of death unknown* variable details the deaths for COVID-19-positive individuals with cause of death still under investigation, or for which the public health unit has up to date been unable to determine with certainty the cause of death. The captured values may very well change later when the cause of death is confirmed either as either *COVID-19 as the underlying cause of death*, *COVID-19 contributed but not underlying cause*, or *COVID-19 unrelated*. The variables are renamed **unknown missing**, **covid**, **covid contrib**, and **no covid** respectively.

2.2.11 Nordpool Power Demand

Further experimentation is conducted on the Nordpool Power Demand dataset (NPD) [77], a real world univariate power demand tracking dataset from Sweden with normal and anomalous power consumption measurements. NPD includes measurements for the last six years with either hourly, daily, weekly, or monthly temporal resolution settings. The data contains both contextual and changepoint anomalies.

2.2.12 Large-scale Annotated Dataset for Energy Anomaly Detection

The final dataset considered in this research is the Large-scale Annotated Dataset for Energy Anomaly Detection (LEAD) [78]. The focus of the data is on energy use at commercial buildings and consists of 1,413 smart electricity meter data spanning over a year. The authors herald this dataset as the largest so far for annotated energy anomaly detection in the public domain.

Containing both changepoint and contextual anomalies, LEAD is adapted from data initially open-sourced at the Kaggle competition the Great Energy

Predictor III, conducted in 2019 [79]. This dataset includes one year of hourly meter readings from 1,636 non-residential buildings collected from 16 different sites worldwide. Also, included are contains building meta-data such as square footage, original building year, and building identifier [80]. LEAD is augment with weather information. The original dataset contains measurements from four different energy meter types i.e. electricity, chilled water, steam and hot water, and LEAD only focuses on electricity. The dimensionality is thus reduced from 1,636 buildings in the original data each with at most four-meter readings, to 1,413 electricity meters.

2.2.13 Conclusion

The above datasets are used for the listed tasks, i.e. forecasting, uncertainty quantification, anomaly detection, and model interpretability. Table 2.7 shows the tasks and the datasets applicable to each, as well as the relevant chapter that details the specific experiment.

TABLE 2.7: All Datasets and their Respective Experimentation Use-cases.

Dataset	Chapter	Forecasting	Uncertainty Quantification	Anomaly Detection	Explainability
AQD	3	x	x	x	
HPC	3	x	x	x	
BSD	3	x	x	x	
NBD	3	x	x	x	
WBD	3	x	x	x	
OWID	4	x	x		
PSML	5			x	x
OTD	4	x			
DIC	4	x			
NPD	5			x	
LEAD	5			x	

2.3 Results Analysis

This section details the analysis conducted on the experimental results. Only the formulation of analysis that overlaps for more than one experiment is detailed. Whichever techniques are only unique to one experiment are detailed in the relevant chapter.

2.3.1 Forecasting

For the point forecasts, one metric used is the symmetric Mean Absolute Percentage Error (sMAPE) [81],

$$\text{sMAPE} = \frac{2}{m} \sum_{t=1}^{m} \frac{|y_t - \hat{y}_t|}{|y_t| + |\hat{y}_t|}, \tag{2.2}$$

where y_t is the post-sample value of the predictand at time t, $\hat{y}_t$ the estimated forecast and m is the forecast horizon. As an error metric, sMAPE is a median-based error criterion, so it is also useful in instances where there are large amounts of outliers in the data. Lastly, sMAPE is symmetric on an absolute scale [82].

The Root Mean Squared Error (RMSE) is employed as a supplementary measure. This metric is useful as it gives predictive error in the same units as the forecast variable itself,

$$\text{RMSE} = \sqrt{\frac{1}{m} \sum_{t=1}^{m} (y_t - \hat{y}_t)^2}, \tag{2.3}$$

where there are m forecasts, y_t is the out-of-sample observation, and $\hat{y}_t$ the forecast at time t.

Because RMSE gives errors in the same units as the forecast variable itself, it is easily interpretable even for laymen. The other reason RMSE is employed as a supplementary prediction error metric is although symmetric on an absolute scale, sMAPE has been shown to penalize large positive errors more than negative ones on a percentage scale [83]. Both metrics are mean absolute differences between forecast and actual values. The key difference is how sMAPE is normalized.

For completeness, also mentioned here is the usage of Mean Absolute Error (MAE) [84] and the Sum of Squared Errors (SSE) for model training, but the appropriate usage context is described in more detail in later sections when they are first used,

$$\text{MAE} = \frac{1}{m} \sum_{t=1}^{m} |y_t - \hat{y}_t|, \tag{2.4}$$

$$\text{SSE} = \sum_{t=1}^{m} (y_t - \hat{y}_t)^2. \tag{2.5}$$

2.3.2 Uncertainty Quantification

For assessing the performance of the prediction intervals, the Mean Interval Score (MIS_α) [85] is used, averaged over all out-of-sample observations,

$$\text{MIS}_\alpha = \frac{1}{m}\sum_{t=1}^{m}(U_t - L_t) + \frac{2}{\alpha}(L_t - y_t)\,\mathbb{1}_{\{y_t<L_t\}} + \frac{2}{\alpha}(y_t - U_t)\,\mathbb{1}_{\{y_t>U_t\}}, \quad (2.6)$$

where U_t (L_t) is the upper (lower) bound of the prediction interval at time t, α is the significance level and $\mathbb{1}$ is the indicator function.

The MIS_α adds a penalty at the points where future observations are outside the specified bounds. The width of the interval at t is added to the penalty if any. Finally, this sum at the individual points is averaged.

For regression, Gal [2] uses the sample variance from the model output of multiple stochastic forward passes to compute predictive log-likelihood. The MIS_α measure is chosen here over the predictive log-likelihood [2] because in addition to penalizing observations that fall outside the prediction intervals, MIS_α also penalizes wide intervals. Narrow intervals are preferred for the novel anomaly detection technique introduced in Chapter 3.

As a supplementary metric, the Coverage Score (CS_α) is employed. CS_α indicates the percentage of observations that fall within the prediction interval,

$$\text{CS}_\alpha = \frac{1}{m}\sum_{t=1}^{m}\mathbb{1}_{\{U_t\le y_t\le L_t\}}. \quad (2.7)$$

With MIS_α the subscript α denotes explicit dependence on the significance level, whereas in the case of CS_α the dependence is implicit.

2.3.3 Anomaly Detection

For evaluating the performance when it comes to anomaly detection, precision, recall, and the F_1-score are employed. The F_1-score is the harmonic average of precision and recall, where one is the perfect score and zero is the worst. The *Ed*-score [86] is also used, which scores the early detection capability of an algorithm in the range zero (worst) to one (best). An increase (decrease) of say 5% in the *Ed*-score measure may be interpreted as a technique detecting an anomaly on average 5% of the time interval earlier (later). The *Ed*-score for an algorithm A_i for anomaly a that falls within window $\mathcal{W}$

for dataset D is given by

$$Ed\text{-score}\left(A_i\right) = \begin{cases} \dfrac{e - \text{idx}\left(A_i\left(a\right)\right)}{e - b} & \text{if } a \in \{a\}_D^{A_i} \\ 0 & \text{if } a \notin \{a\}_D^{A_i}, \end{cases} \tag{2.8}$$

where $\{a\}_D^{A_i}$ denotes the set of anomalies detected by algorithm A_i in dataset D, $A_i\left(a\right)$ is the time taken for the initial detection of anomaly a in D by algorithm A_i, $\text{idx}(A_i\left(a\right))$ is the index of that timestamp, and b and e are the beginning and end indices for window $\mathcal{W}$, respectively. The indices of the timestamp are used instead of the time difference as the datasets use different time steps between observations.

Receiver Operating Characteristic (ROC) curve is a graph-based illustration of a classifier or anomaly detector that communicates the model's accuracy. A ROC curve plots the True Positive Rate (TPR) and the False Positive Rate (FPR) on the same set of axes. The performance metric used to assess the anomaly detectors is the area under the ROC curve (auROC), where values nearest to one represent a good measure of separability.

Scoring anomaly detectors under an imbalanced class distribution might make the auROC metric worse than it should be [87]. A better performance metric to consider in this case is the Precision-Recall (PR) curve. In the anomaly detection experimentation sections, the area under the PR (auPR) curve is also reported which ranges from zero to one. A value close to one is preferred.

2.3.4 Statistical Significance

A Student's t-test [88] is conducted for testing the statistical significance of the results. A Diebold-Mariano (DM) [89] test is also conducted to compare the out-of-sample predictive skill of the models. The DM test is configured to use Mean Absolute Prediction Error (MAPE) [90],

$$\text{MAPE} = \frac{1}{m}\sum_{t=1}^{m}\left|\frac{y_t - \hat{y}_t}{y_t}\right|. \tag{2.9}$$

2.4 Conclusion

In the next chapter, as well as the two subsequent chapters, an exposition of the experimental studies is offered. Table 2.7 above indicates the road map.

Chapter 3 introduces a novel method of regression-based anomaly detection, that can be applied to both deep learning and classical statistics-based models. A comparison of the relative performance of models from the two streams (statistics and deep learning) is conducted within this novel anomaly detection framework.

In Chapter 4, a novel extension of a statistics and deep learning-based hybrid model for forecasting and uncertainty quantification is introduced. The hybrid model is applied to morbidity and mortality modelling, and its performance is compared to pure deep learning and pure statistical models.

Chapter 5 details a further extension of the novel hybrid method extension introduced in Chapter 4. The model is extended further to the domain of anomaly detection, with additional usefulness derived from interpretability. This extended interpretable model is applied to renewable energy generation in decarbonized grids, and its performance is compared to well-documented anomaly detection benchmark models.

Chapter 3

Deep Learning and Statistical-based Models for Anomaly Detection

The work adapted and presented in this chapter appears in the following publications: (*i*) Mathonsi and van Zyl [40] and (*ii*) Mathonsi and van Zyl [60]. T. Mathonsi participated in the conception of the research, formulated and implemented the methods, conducted experiments and analysed results, and was the principal participant in the writing of the manuscripts. T. van Zyl participated in the conception of the research and reviewed the manuscripts.

Summary

It has been shown that deep learning models can under certain circumstances outperform traditional statistical methods at forecasting tasks. Furthermore, various techniques have been developed for quantifying the forecast uncertainty (prediction intervals). In this chapter, prediction intervals constructed with the aid of deep artificial neural networks are utilized to detect anomalies in the multivariate setting. Challenges with existing deep learning-based anomaly detection approaches include (i) large sets of parameters that may be computationally intensive to tune, (ii) returning too many false positives rendering the techniques impractical for use, and (iii) requiring labelled datasets for training which are often not prevalent in real life. The proposed approach overcomes these challenges. The presented approach is benchmarked against the oft-preferred well-established statistical models. This chapter focuses on three deep learning architectures, namely, cascaded neural networks, reservoir computing, and long short-term memory recurrent neural networks. The findings are that deep learning outperforms (or at

the very least is competitive to) the latter.

3.1 Introduction

Time series forecasting with the aid of machine learning tools has been conducted as far back as the works of Hu [9], whose application domain is weather prediction. Since then, new machine learning models have been developed, with advances dramatically improving output accuracy [10]–[13]. Machine learning-based forecasting applications include estimating realized volatility in finance [14], or projecting macro-economic factors [15].

In the context of time series modelling, there are three main types of anomalies [8]. A changepoint anomaly is characterized as an observation that is far removed from the core distribution of the rest of the data. Geometrically, it can be understood as a point that is isolated and far from all the other points. Time series anomalies that manifest over a period rather than at single time points are known as range-based or collective anomalies. Finally, contextual anomalies are values that significantly deviate from the rest of the data points within the same context. As a result, an anomaly detected in one setting may otherwise be classified as normal if it occurred in a different context. For time series data, said context is usually temporal.

If one begins with a notion of well-defined normalcy, then an anomaly is a pattern that does not conform to *expected* normal behaviour. This definition introduces expectation, which is derived using probability distributions.

A simple approach to detect anomalies then, is to define a region or range that describes normalcy and classify everything outside this space as anomalous. However, this simple approach presents some challenges [8]:

- defining the normal region such that each and every possible normal point is encompassed is nontrivial, and the separating region boundary is in practice often imprecise, leading to misclassification especially for observation closest to the boundary of the constructed normal region;
- *adversarial anomalies* - when anomalies are malicious activities e.g. insurance fraud, perpetrators adapt the anomalous observations to make them appear normal;
- *non-stationarity* - the boundary for normal points may evolve, so one's initially devised notion of normalcy might be insufficient in the future;
- *class-imbalance* - it is hard to source labelled data necessary for training. Even in instances where such data is available, it may present

with very few anomalous observations compared to normal sequences which complicates training; and

- noise contained in the data may be similar to the anomalies, and overfitting becomes more of a challenge.

With the anomaly detection problem phrased as a supervised learning problem (applied to data where each observation is labelled), there are general steps usually followed in the existing literature. Although not identical in their methodologies, even the most recent advances are not dissimilar to approaches first adopted by Malhotra, Vig, Shroff, *et al.* [91], Chauhan and Vig [92] and Malhotra, Ramakrishnan, Anand, *et al.* [93], for instance. Generally, methodologies incorporate the following steps:

1. use a deep learning algorithm or architecture (e.g. LSTM or encoder-decoder) for generating forecasts, retain the errors as they are used later for anomaly detection;
2. assume the resultant errors follow a Gaussian (or other) distribution; and
3. learn a threshold by maximizing metrics such as precision, recall or some combination thereof on the validation set.

The challenge with this problem framing is that real-world datasets are often not labelled. Even in cases where they are, the labeling process is prone to human error. The errors could inadvertently be incorporated into the modelling process, resulting unreliable results and misguided post hoc decision making. Next, some relevant recent advances and the current state-of-the-art techniques are reviewed, and their advantages and disadvantages are discussed.

3.1.1 Recent Advances and the Current State-of-the-Art

Assendorp [94] examines deep learning methods for detecting anomalies within sensor data. Sensor data anomaly detection forms part of a larger research wave exploring anomaly detection in mechanical devices, which has applications in, for example, the manufacturing industries.

Sölch, Bayer, Ludersdorfer, *et al.* [95] apply variational autoencoders (VAEs) [96] composed of stochastic recurrent networks (STORNs) [97] on multivariate sensor data from a robot arm. The VAE is trained solely on normal data, whereas validation and testing is done with both normal and anomalous sequences that were produced by manually altering the robot's behaviour. At

each time step, the VAE outputs the variational lower bound and the prediction of the distribution at the next time step. The lower bound and median forecast are then used to calculate thresholds for anomaly detection.

For range-based or collective anomalies, Thi, Cao, and Le-Khac [98] train a LSTM on normal time series data, then forecast for each time step. They do not consider isolated time steps, but instead use prediction errors from multiple time steps for detecting collective anomalies. Zhu and Laptev [99] use Bayesian Neural Networks (BNNs) [100] and the Monte Carlo (MC) dropout framework to construct prediction intervals, inspired by the works of Gal [2] and Gal and Ghahramani [18].

One-Class Support Vector Machines represent some of the earliest machine learning-related attempts at solving anomaly detection phrased as a classification task [28]. If there are too few anomalous observations, they may be subsumed into the density distribution and erroneously classed as normal. Faced with severe class imbalance, Schölkopf, Williamson, Smola, *et al.* [28] restrict their attention to local density distributions.

Malhotra, Vig, Shroff, *et al.* [91] employ stacked Long Short-Term Memory (LSTM) [101] networks to detect anomalies in time series data. The network is trained on non-anomalous data, then used as a predictor over a multi-step forecast horizon. The likelihood of an observation being anomalous is assessed using a multivariate Gaussian distribution formed from the prediction errors. Later, Malhotra, Ramakrishnan, Anand, *et al.* [93] consider mechanical devices within the context of exogenous effects. An Encoder-Decoder scheme with an LSTM-based back-end is used for time-series behaviour reconstruction using only normal observations. The errors from the reconstruction process are then used to detect anomalies similarly to Malhotra, Vig, Shroff, *et al.* [91]. Schlegl, Seeböck, Waldstein, *et al.* [102] consider a deep convolutional generative adversarial network in an unsupervised setting that learns a manifold of normal anatomical variability, and apply it to the task of anomaly detection to aid disease detection and treatment monitoring.

Since graphs can be used to represent structural information [25], it is important to identify anomalous nodes or edges in a single graph, or an anomalous graph from a selection of graphs. Graph neural networks are a research area where deep learning is useful as it can handle the complexity of the graph data better than conventional anomaly detection techniques. Ma, Wu, Xue, *et al.* [103], for instance, offer a comprehensive review of cutting-edge deep learning techniques for graph-based anomaly detection.

There have been immense advances made in applying deep learning methods in fields such as forecasting [33], [104], but there is a relative scarcity of deep learning approaches for anomaly detection and related disciplines [105], [106].

There are also some comprehensive reviews, including, for instance, the works of Adewumi and Akinyelu [107], who survey deep learning-based methods for fraud detection, Kwon, Kim, Kim, *et al.* [105] who focus on techniques and applications related to cyber-intrusion detection, and as recent as works by Thudumu, Branch, Jin, *et al.* [106], who focus on the challenge of detecting anomalies in the domain of big data.

As a first step, one may consider semi-supervised approaches for anomaly detection. As evidenced by the literature highlighted above, training on normal sequences and then making inferences with data contaminated by anomalies has been successfully applied. Existing approaches are still prone to suffer from drawbacks detailed in Sections 1.2.1 and 1.2.2, for instance. This chapter builds upon the reviewed approaches and offers extensions as contribution to mitigate the aforementioned challenges.

3.1.2 Contribution

In this chapter, the novel contribution can be summarized as follows:

- a dynamic threshold-setting approach is introduced, which learns and adapts the applicable threshold as new information becomes available;
- no golden labels are required for training or detecting the thresholds, and subsequently detecting the anomalies.
- by testing methodological assertions on both outlier and step-change anomalies, two streams of research are consolidated, whereas previous researchers mostly focused on either one of these; and
- forecast machinery are critically analysed, competitive prediction intervals are constructed using computational means where no statistical methodology is readily applicable, then all of this information is used to detect anomalies.

One of the key differences between the proposed approach and some of the existing approaches detailed above, is that the proposed technique is intended to be model agnostic and may be applied to deep learning forecast machinery to augment it to applicability within a probabilistic forecasting framework.

3.2 Methodology

The presented approach utilizes three fundamental steps:

1. forecasting using deep learning and probability-based models;
2. deriving prediction intervals using computational means and built-in density distributions, respectively; and
3. using said prediction intervals to detect anomalies.

In this chapter, five of the datasets detailed in Section 2.2 are used, namely, the Air Quality Dataset (AQD), Household Power Consumption (HPC), Bike Sharing Dataset (BSD), Yahoo Webscope Benchmark (YWB), and NASA's Bearings Dataset (NBD). Each of the fundamental steps and intermediate ones are discussed below.

3.2.1 Forecasting

AQD

The forecast problem is framed as a supervised learning problem where, given the weather conditions and pollution measurements for the current and 23 prior hours, $t, t-1, t-2, \ldots, t-23$, the outcome is to forecast the pollution at the next hour $t+1$.

HPC

Given power consumption for four weeks, the goal is to forecast power consumption for the next week ahead. The observations are converted from per-minute to daily aggregates, as illustrated in Figure 3.1. Seven values comprise a single forecast, one value for each day in the standard week (beginning on a Sunday and ending Saturday). Each forecast is evaluated in this aggregated setting as well as with each time step separately.

BSD

Given all available information for BSD for the past 3 months, the goal is to forecast the total rental bikes for the coming week.

NBD

Given the sensor readings for the past 10 hours, the aim is to forecast the readings for the next hour. The mechanical degradation in the bearings occurs gradually over time, so the analysis is conducted using only data points

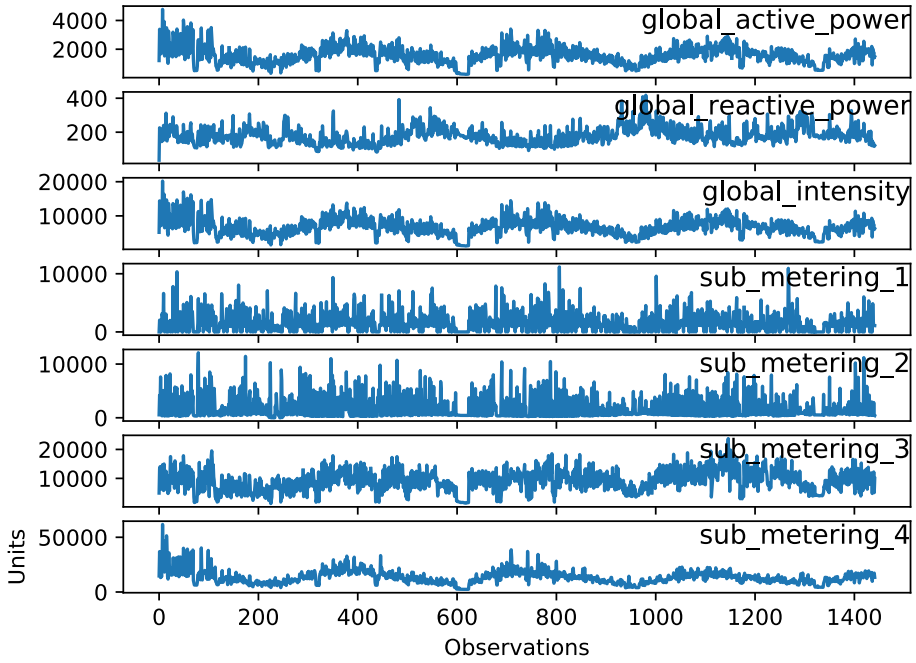


FIGURE 3.1: Separate Variable Plots of the Daily Power Consumption Data Spanning Four Years.

occurring every ten minutes instead of per-second observations. In total, there are 20,480 observations across each of the four bearings.

To easily detect the point where the bearing vibration readings begin oscillating dramatically, the signal is transformed from the time- to the frequency domain using a Fourier transform.

As depicted in Figure 3.2, there is an increase in the frequency amplitude for readings progressing to the complete degradation and subsequent failure of each bearing.

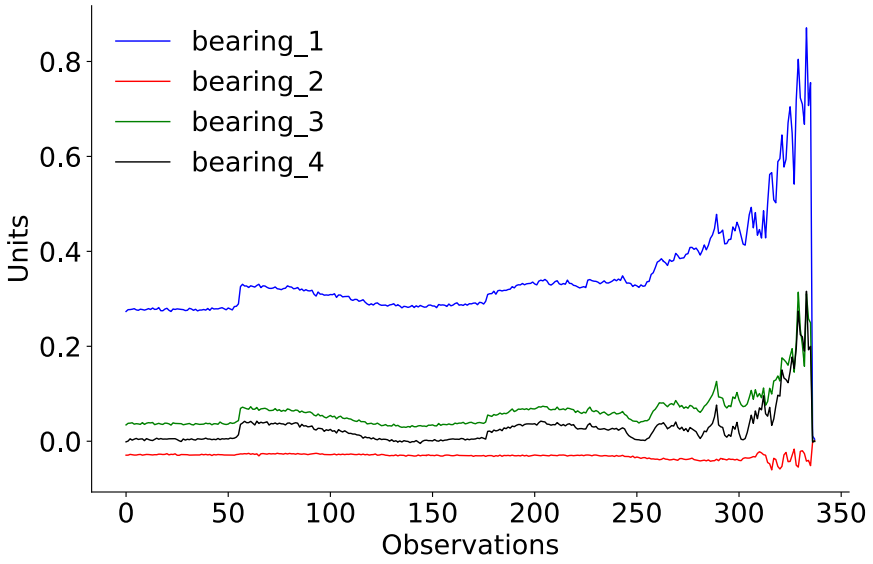


FIGURE 3.2: Bearing Sensor Data Test Frequency.

YWB

Lastly, for the YWB, the last five days' worth of observations are utilized to forecast traffic to the Yahoo entities for the next hour.

Note, all the above forecast problems can be re-framed using different window sizes. The chosen window sizes in this research are stated here for reproducibility. These windows were chosen using empirical studies with optimizing based on training accuracy.

For each dataset, the relevant forecast framework is used as described above and subsequently, the prediction intervals are constructed with all three deep learning algorithms and all three statistical algorithms.

To a great extent, the treatment of each dataset is not dissimilar. As such, the narrative that follows might overlap. For ease of exposition, all methods are the same unless explicitly stated.

3.2.2 Preprocessing

The following steps are undertaken in preparing the time series data for forecasting. The following points constitute the most used techniques for all the datasets. NA values are removed. NA is distinct from missing (blank) values. The latter, amounting to 4.664% for AQD and 1.252% for HPC, for instance, are imputed using the iterative imputer from the Scikit-learn module [108], which models each feature with missing values as a function of other features in a round-robin fashion. The imputer is set to parameterize only on past values for each iteration, thus avoiding information leakage from the future testing set.

In the case of BSD, the categorical values are label encoded, such as the overall weather situation, *weathersit*, and variables that state whether or not observations fall on a *weekday* and *holiday*. Label encoding is chosen over one-hot encoding for instances where there are many categories in total (more than five), as the former reduces memory consumption. Furthermore, the datasets are normalized.

All the datasets are composed of time series with different seasonality patterns. In such instances, artificial neural networks tend to struggle with seasonality [34], [109]. A standard remedy is to de-seasonalize the series first and remove the trend using a statistical procedure like the Seasonal and Trend decomposition using Loess (STL) [110].

For reproducibility, points unique to only HPC are also detailed, as it requires more preprocessing than the other datasets. Seven values comprise a

single forecast, one value for each day in a standard week. Each forecast time step is evaluated separately.

The data is divided into standard weeks beginning on a Sunday and ending Saturday. Pruning observations before the very first Sunday available in the data, December 17th 2006, and after November 20th 2010 (the last Saturday observed) yields 205 full standard weeks for training and testing.

There are only 205 samples available for training, and this may pose a problem for parameter-tuning [111]. The small data problem is circumvented by disregarding the standard week and using the prior 28 days to forecast the coming 7 days. This flattening and use of overlapping windows effectively transforms the 205 samples into a total of $1,418$.

In contrast, the testing framework remains unchanged, i.e. the prior four standard weeks are used to forecast the daily power consumption for the subsequent standard week. Lastly, 1.252% of the data which is missing is imputed, normalized, de-trended and de-seasonalized.

3.2.3 Algorithms

Classical Statistical Models

For benchmarking, the statistical methods used are Multiple Linear Regression (MLR) [112], Vector Autoregression Moving-Average with Exogenous Regressors (VARMAX) [113] and Seasonal Autoregressive Integrated Moving-Average with Exogenous Regressors (SARIMAX) [114].

Deep Learning Models

The deep learning architectures used are the Cascaded Neural Networks based on cascade-correlation (CCN), first published by Fahlman and Lebiere [1], a variant of the Recurrent Neural Network (RNN) [115] known as Long Short-Term Memory (LSTM) [101], and one class of reservoir computing, Echo State Networks (ESN) [116]. Next, the underlying theory is detailed, and the choice of these models is motivated.

cascade-correlation-based Cascaded Neural Networks The cascade-correlation architecture [1] is beneficial due to the following main ideas. Firstly, the network is grown iteratively and only if required, i.e. it only adds new neurons when the current existing architecture designed and constructed thus far is still unable to solve the problem satisfactorily, a process known as cascading. Cascading ensures the resultant network is parsimonious, thus saving

on computational expense. Secondly, while cascading, the network trains the new neurons one at a time. This training procedure overcomes issues associated with backpropagation (BP) [117], [118]. These issues include vanishing and exploding gradient [119]. The vanishing gradient problem is encountered in the training of artificial neural networks where the gradients used to update the weights shrink exponentially, causing the weights to stop updating and stalling learning. The exploding gradient problem is the direct opposite, where the gradients used to update the weights grow exponentially, preventing the BP algorithm from updating the weights, and making the learning process unstable [119].

The learning algorithm starts off with a base network composed of only an input layer and an output layer. At this stage, the network has zero hidden layers. Since there are no hidden neurons yet, the learning at this stage can use the simple gradient descent (GD) [120] algorithm applied individually for each output neuron (see Appendix A). During this initial learning, new neurons are added to the network one at a time. Each neuron is inserted into a new hidden layer and fully connected to the existing network, i.e. to all the already existing input and hidden neurons. At this point, all the input connections to each neuron are held constant for the remainder of network design and training.

Creating these hidden layer neurons is a two-step process where first some candidate neuron is connected fully to the network, except for this neuron's output. This process enables training only the candidate's output weight, whilst the rest of the network is held constant. The term *candidate* is used to emphasize that the unit is not yet added to the network. It will be added only if it reduces the overall error, otherwise the cascading terminates. In the second step, the candidate is activated (i.e. now completely trained) and all the input connections of the output layer neurons are now trained. This process is repeated iteratively until some termination condition is satisfied, e.g. network accuracy or maximal depth. This Fahlman and Lebiere [1] process is described in Algorithm 1.

There is a scheme where one may use a pool of candidates, as opposed to just one. Each of the weight vectors associated with said candidates can be initialized differently. Training can be simultaneous because the candidates are independent of each other and do not affect the active network. Thereafter, the best performing candidate is selected, i.e. the one with the best correlation. Fahlman and Lebiere [1] demonstrate that typically less than ten candidate units suffice for ensuring good candidate representation in every

Algorithm 1 Cascade-Correlation Neural Network Process [1].

1: Start with the required input and output units; both layers are fully connected; the number of inputs and outputs is dictated by the problem
2: Train all connections ending at an output unit with a common learning algorithm until the squared error E of the neural network no longer decreases
3: Generate a candidate unit that receives trainable input connections from all of the network's external inputs and from all pre-existing hidden units (if any exist); the output of this candidate unit is not yet connected to the active network (output)
4: Train the unit (its weight) to maximize the correlation C

$$C = \sum_{t=1}^{m} \left| \sum_{j \in J} \left(V_j - \bar{V}\right) \left(E_t - \bar{E}_t\right) \right|, \tag{3.1}$$

where j is the training sequence, V is the candidate unit's value, E_t the residual output error observed at time t, and $\bar{V}$ and $\bar{E}_t$ are the values of V and E_t averaged over all training sequences, respectively; learning takes place with an ordinary learning algorithm; training is stopped when the correlation score no longer improves
5: Connect the candidate unit with the outputs and freeze its input weights; the candidate unit now acts as an additional input unit
6: Train the input-outputs connections again, by minimizing the squared error E as outlined in step 2
7: Repeat steps 3 to 6 adding one hidden unit at a time
8: Stop training when the error E of the network falls below a pre-specified value ϵ

pool.

In Algorithm 1, an optimization routine has to be employed in steps 2 and 4. In the original presentation, Fahlman and Lebiere [1] use the Quickprop algorithm, which is similar to standard BP as it computes the derivative of E w.r.t. the weights, except that it replaces simple GD with a second order routine based on Newton's method [121].

Long Short-Term Memory Recurrent Neural Networks Recurrent Neural Networks (RNN) [122] are characterized by built-in feedback loops, and an internal state (memory) for processing sequences of inputs. RNNs can be difficult to train, due to effects from the vanishing gradient problem that are potentially worse in this architecture than vanilla feed-forward networks.

RNNs are called recurrent because they execute the same computation on every element of a sequence, in a way that the previous computations also inform the output. In this way, they synthesize memory, capturing information

about what has been calculated so far.

Because of this synthetic memory feature, one benefit is that a RNN can effectively incorporate temporal dependencies within the input data. The incorporation can be achieved by providing the network at each time step with feedback connections from previous time steps, the implementation of which is efficient due to parameter sharing between the unrolled neural networks over all time steps. Additionally, RNNs are a reasonable approach for tasks that need to model sequences with different length, as the network can be dynamically unrolled according to the length of the input sequence.

Another interesting point is that RNNs have been successfully used in probabilistic forecasting [39]. For instance, in language processing, it assigns a probability to sentences, and predicts the next word given the history of words that have come before. This makes RNNs ideal for forecasting and quantifying forecast uncertainty.

The formulae governing the computation in a RNN are as follows. Given an input sequence $\boldsymbol{x} = (x_1, \ldots, x_N)$, a vanilla RNN computes the hidden vector sequence $\boldsymbol{h} = (h_1, \ldots, h_N)$ and output vector sequence $\hat{\boldsymbol{y}} = (\hat{y_1}, \ldots, \hat{y_N})$

$$\begin{aligned} h_t &= \mathcal{F}(W_{xh}x_t + W_{hh}h_{t-1}) \\ &= \mathcal{F}\left((W_{hh}W_{xh})\,(h_{t-1}x_t)^T\right) \\ &= \mathcal{F}\left(W\,(h_{t-1}x_t)^T\right) \qquad (3.2) \\ \hat{y}_t &= W_{h\hat{y}}h_t, \qquad (3.3) \end{aligned}$$

where the $W_{xh}, W_{hh}, W_{h\hat{y}}$ matrices are weights from the input to the hidden layer, hidden layer to the next hidden layer, and last hidden layer to the output layer, respectively. $W = W_{hh}W_{xh}$ is a composite weight matrix, and $\mathcal{F}(\cdot)$ indicates the activation function of the layer.

RNNs have the capacity to incorporate an additional stored state, that incorporates aforementioned feedback loops, sometimes characterising delayed effects inherent in the data. These are known as *gated memory state cells*, the two major variants being LSTM [101] and gated recurrent units (GRUs) [123]. This research only focuses on the former, as even though they train slower, they remember longer sequences than GRUs and outperform them in instances where one has to model longer lag relations [124]. Józefowicz, Zaremba, and Sutskever [125] compared different architectural modifications to LSTMs and GRUs but failed to find a model which performs consistently better than these baseline architectures.

LSTM improves on traditional neural networks, by not assuming independence of all the inputs (and outputs) and overcomes the vanishing and exploding gradient associated with vanilla RNNs.

The LSTM state cell c_t (which captures the state of the RNN) is made up of at least one self-connected memory cell, and three additional gates that control the flow of information using multiplicative computations. The learned input and output gates control the flow of information, and the forget gate enables the cells to reset.

$$c_t = f \odot c_{t-1} + k \odot g \tag{3.4}$$

$$h_t = o \odot \tanh(c_t) \tag{3.5}$$

$$\begin{pmatrix} k \\ f \\ o \\ g \end{pmatrix} = \left(\boldsymbol{\mathcal{F}}\right) \left(W \left(h_{t-1} x_t\right)^T\right), \tag{3.6}$$

$$\tag{3.7}$$

where c_t is the cell state at time t. $\boldsymbol{\mathcal{F}}$ is composed of, respectively, three logistic sigmoid functions followed by a tanh activation function, and l, f, o, and g are respectively the input gate, forget gate, output gate and update gate. These gates detail respectively whether or not and how much to write to a cell, whether or not to erase a cell and how much to forget, how much of the cell to reveal to the outside world, and what candidate value to consider writing to the cell. All these gates have the same dimension as $\boldsymbol{h}$.

Reservoir Computing Reservoir Computing (RC) is a framework where an input signal is fed into a fixed (randomly initialized) dynamical system known as a reservoir. The reservoir then maps this input to a higher dimensional space. The reservoir should be relatively large $\mathcal{O}(10N)$ compared to the input size (if there are, say, N input units). The connections between the neurons are sparse and use a sigmoidal transfer function. These connections inside the reservoir are assigned once at random, i.e. there is no training involved at this stage. The input neurons to the reservoir are also assigned random weights. In addition to the reservoir, the artificial neural network has a readout (or output) layer, where the only training occurs. In this way, RC also circumvents the challenges associated with BP such as the vanishing gradient problem.

The main idea is that the sparse (randomly initialized) connections inside the reservoir allow the previous states to *echo* through the network even after they have passed, i.e. the activation states $\boldsymbol{u}$ of (arbitrarily assembled) RNN can be regarded as a function of historical input sequences. The output layer is trained to read the state of the reservoir and map it to the desired output. The main reservoir is fixed, and only the readout is trained.

There are two main classes of RC. These are Liquid-state Machines (LSM) [126] and Echo State Networks (ESN) [116]. LSM was developed primarily as a biologically plausible tool for use in generic cortical microcircuits, and applications of this model have mostly been limited to circuits of spiking neurons [127]. In contrast, ESN was designed for high performance at engineering tasks [127]. Because of the comparatively higher efficiency offered, this thesis focuses on the latter.

The ESN architecture consists of a fixed weight matrix W_{dr} where the subscript stands for dynamic reservoir. W_{dr} gives connections between internal units' weights and recurrent pathways. The reservoir outputs a series of activation states, which are then used to train the output connections.

If there are, say, N input units, K internal network units, and m output units, then the activations of the network units at time t are described by

$$\boldsymbol{x} = (x_1, \ldots, x_N) \quad \text{...inputs,} \tag{3.8}$$

$$\boldsymbol{u} = (u_1, \ldots, u_K) \quad \text{...internal units,} \tag{3.9}$$

$$\hat{\boldsymbol{y}} = (\hat{y_1}, \ldots, \hat{y_m}) \quad \text{...output units.} \tag{3.10}$$

Thus there are three additional weight matrices W_{in}, W_{out}, and W_{bp} giving connections between input nodes and internal units, connections from the dynamic reservoir to output units, and the backprojection weight matrix for connections that project back from the output unit to the internal. These matrices are of dimensions $K \times N$, $m \times (N + K + m)$, and $K \times m$, respectively. W_{dr} is of dimension $K \times K$.

If the output functions of the dynamic reservoir nodes are

$$\boldsymbol{f}_{dr} = (f_{dr,1}, \ldots, f_{dr,K}), \tag{3.11}$$

then the states are updated using

$$\boldsymbol{u} \leftarrow \boldsymbol{f}_{dr}\left(W_{in}\boldsymbol{x} + W_{dr}\boldsymbol{u} + W_{bp}\hat{\boldsymbol{y}}\right). \tag{3.12}$$

If the output functions of the output units are

$$\boldsymbol{f}_{out} = (f_{out,1}, \ldots, f_{out,m}),$$

and $\boldsymbol{q} = (\boldsymbol{x}, \boldsymbol{u}, \hat{\boldsymbol{y}})$ is a concatenation of the (current) input, internal, and (previous) output vectors, then the output is computed using

$$\hat{\boldsymbol{y}} \leftarrow \boldsymbol{f}_{out}\left(W_{out}\boldsymbol{c}\right). \tag{3.13}$$

Definition 3.2.1 (Echo State Property [116]) *Assume an untrained network with weights W_{in}, W_{dr} and W_{bp} is driven by teacher input $\boldsymbol{x}$ and teacher-forced*[1] *by teacher output $\boldsymbol{y}$ from compact intervals Ω and Y, if for every left-infinite input/output sequence $(\boldsymbol{x}; \boldsymbol{y})$, where $t = \ldots, -2, -1, 0$, and for all state sequences $\boldsymbol{u}$ and $\boldsymbol{u}'$ compatible with the teacher sequence, e.g. with*

$$\boldsymbol{u} \leftarrow f(W_{in}\boldsymbol{x} + W_{dr}\boldsymbol{u} + W_{bp}y\boldsymbol{y}) \tag{3.14}$$

$$\boldsymbol{u}' \leftarrow f(W_{in}\boldsymbol{x} + W_{dr}\boldsymbol{u}' + W_{bp}y\boldsymbol{y}) \tag{3.15}$$

it holds that $\boldsymbol{u} = \boldsymbol{u}'$, for all preceding time.

An intuitive explanation of the echo state property is that the reservoir will asymptotically wash out any information from initial conditions that are arbitrarily set. A sufficient condition for non-existence of the echo state property: W_{dr} has spectral radius larger than one,

$$|\lambda_{\max}| > 1. \tag{3.16}$$

This Jaeger [116] process is described in Algorithm 2.

Computational Methods for Prediction Interval Construction

For aiding in the construction of prediction intervals, Monte Carlo Dropout [18] and the quantile bootstrap [128] also known as the reverse percentile bootstrap [129] are employed. Details on implementation follow in Sections 3.2.6 and 3.2.7 respectively. Other bootstrap methods are detailed in Appendix A.

[1] Teacher forcing [115] is a strategy for training recurrent neural networks that uses model output from a prior time step as an input.

Algorithm 2 Echo State Network Reservoir Computing.

1: *Initialization:* Generate untrained $W_{dr,0}$, choose W_{in} and W_{bp} arbitrarily; to ensure the condition stated in Equation (3.16) is satisfied, some heuristics include setting (i) the size of K to be at least $\frac{N}{10}$ (this order of magnitude also helps mitigate overfitting); and (ii)

$$W_{dr,1} = W_{dr,0}\left(\frac{1}{|\lambda_{\max}|}\right),$$

i.e. normalize the initial weight matrix with its spectral radius, then set

$$W_{dr} = W_{dr,1},$$

i.e. scale such that

$$|\lambda_{\max}| => 1$$

2: *Training:* Initialize $\boldsymbol{u}$ arbitrarily; train the network using data for times $t = 1, \ldots, N$ presenting teachers input $\boldsymbol{x}_t$ and teacher-forcing output $\boldsymbol{y}_t$ by updating with Equation (3.12); use arbitrary $\boldsymbol{y}_1$ as $\boldsymbol{y}_{t-1}$ is not defined at $t = 2$ because the output is lagged; for all times after initial washout time N_0 (useful for cancelling out effects of random arbitrary initialization), collect input-reservoir-previous output state $\left(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{y}_{t-1}\right)$ (concatenated, a row in the state collecting matrix S of dimension

$$(N - N_0 + 1) \times (N + K + m);$$

for $t > N_0$, collect sigmoid-inverted teacher output $\tanh^{-1} \boldsymbol{y}_t$ row-wise into teacher collection matrix M of dimension $(N - N_0 + 1) \times m$

3: *Calibration:* Compute the output weight matrix W_{out}, of dimension $m \times (N + K + m)$ using

$$(W_{out})^T = S^{-1}M, \tag{3.17}$$

where the superscript T indicates transposition

3.2.4 Analysis

AQD

The LSTM is configured with 50 neurons in the first and only hidden recurrent layer, and one neuron in the output layer for predicting pollution. The model is trained for 25 epochs with a batch size of 72. The input shape is 24 time steps with 8 features.

The ESN is initialized with 384 input units, one output unit, and 4,096 internal network units in the dynamic reservoir ($\mathcal{O}$ (10 x # input units)). A 24 hour wash-out interval is used.

The CCN is initialized with 72 neurons in the input layer and one neuron in the output layer. The cascading is capped at 24 layers.

The structure of the deep learning architectures for AQD are given in Figure 3.3 as an illustrative example. The CCN starts off with one hidden cell that is fully connected. As additional cells are added (if required), they are connected to the input and output nodes, as well as each of the preceding hidden nodes. The ESN's reservoir is composed of recurrent cells that are sparsely connected. The deep learning architecture configurations for the other datasets are detailed below.

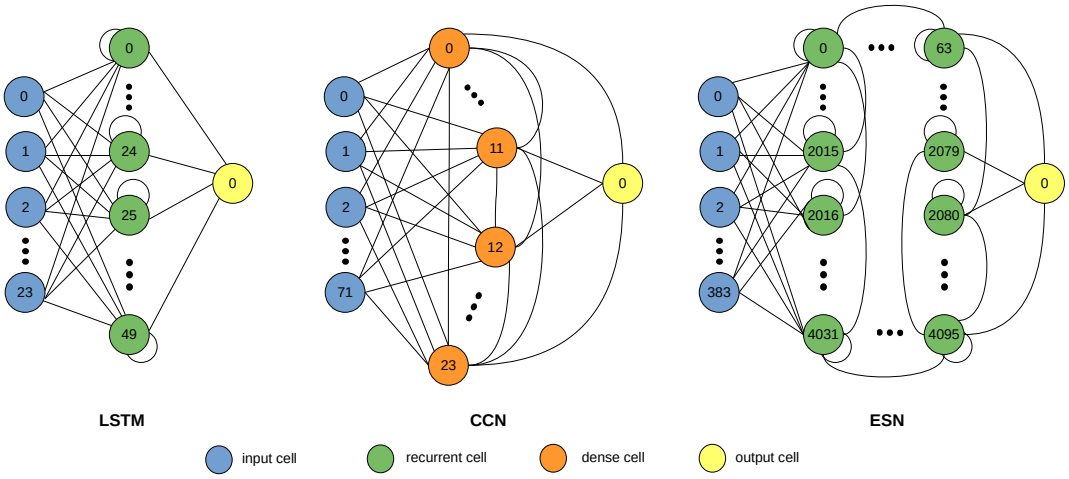


FIGURE 3.3: Flowchart Diagrams Depicting the Deep Learning Model Architecture Configurations for AQD.

HPC

The model is developed with one hidden LSTM recurrent layer with 200 neurons. The second layer is fully connected with 200 neurons. Lastly, the direct forecast is produced by an output layer that outputs a vector with seven entries, as previously described. The model is trained for 65 epochs with a batch size of 18, determined experimentally to be optimal on the validation set.

The ESN uses a 28 day wash-out interval. The CCN has seven neurons in the output layer, with cascading capped at 28 layers.

The first three calendar years are used as training and validation data and the final year for model evaluation purposes.

BSD

The LSTM is initialized with 100 neurons in the sole hidden recurrent layer, followed by an output layer that directly forecasts a vector with seven days'

worth of forecasts. The model is fit for 25 epochs with a batch size of 32.

The ESN uses a seven-day wash-out interval. The CCN has 168 neurons in the output layer, with cascading capped at 28 layers.

NBD

The LSTM is instantiated with 100 neurons in the input layer, one hidden recurrent layer with 50 neurons, and an output layer comprised of four neurons. The model is fit for 100 epochs with a batch size of 32.

The ESN is initialized with 48 input units, four output units, and $1,024$ internal network units in the dynamic reservoir. A 24 hour wash-out interval is utilized.

The CCN is composed of 100 neurons in the input layer and four neurons in the output layer. The cascading is capped at 24 layers.

YWB

The LSTM has five hidden layers with 4, 16, 48, 16, and 4 neurons, respectively. The model is fit for 45 epochs with a batch size of 24.

The ESN uses a three-day wash-out interval. The CCN has four neurons in the output layer, with cascading capped at nine layers.

Overall

In terms of AQD, holdout cross-validation is used, fitting each of the models on the first four years of data and testing with the remaining year. For all the other datasets, walk-forward validation is used to optimize the hyperparameters. For instance, with HPC, the models forecast a week ahead $\hat{y}_{t+1}$. After that, the actual observations from a week ahead y_{t+1} are presented to the models, and are used in the set of observations $\ldots y_{t-2}, y_{t-1}, y_t, y_{t+1}$ for the current iteration in the walk-forward validation process, before the subsequent week $\hat{y}_{t+2}$ is forecast.

As an additional method to mitigate computational cost, the models are evaluated statically, i.e. they are trained on the historical observations and then iteratively forecast each step of the walk-forward validation.

Each architecture uses the MAE loss function and the efficient Adam version of stochastic gradient descent [130] (see Appendix A). Each configuration is selected over various others based on empirical studies of validation RMSE reduction and configuration behaviour over time.

The input variables to VARMAX and SARIMAX are declared as exogenous.

After each model is fit, forecasts are produced, the seasonality and trend components are replaced, the scaling is inverted, and an error score is computed for each.

3.2.5 Evaluation

For code implementation details, refer to Appendix B.

3.2.6 Recurrent Layer Dropout

The forecast accuracy is compared at dropout rates of 15%, 30%, 45%, and 60% to the standard LSTM fit above (i.e. with no dropout). The dropout is applied specifically in order to transform the otherwise deterministic model into a pseudo-stochastic model. The stochasticity is then used to derive artificial prediction intervals. Each dropout experiment is repeated 35 times and the aggregate results are compared. The aggregated approach mitigates potential skewness introduced by randomness. Table 3.1, and the box and whisker plot in Figure 3.4 display the distributions of results for each configuration in the case of AQD as an illustrative example.

TABLE 3.1: Dropout RMSE Score Distributions.

	Dropout Rate				
	0.0	**0.15**	**0.3**	**0.45**	**0.6**
mean	20.7746	21.8681	22.5086	23.7820	25.4886
std	0.0000	0.1283	0.1793	0.2436	0.2812
min	20.7746	21.5665	22.2587	23.4037	24.8913
25%	20.7746	21.8193	22.4969	23.5335	25.2295
50%	20.7746	21.8543	22.5990	23.7627	25.4671
75%	20.7746	21.9773	22.7260	23.8939	25.6549
max	20.7746	22.0783	22.9699	24.3575	25.8718

From Table 3.1, the standard implementation (LSTM with no dropout) outperforms all the configurations with dropout. At 15%, the output has the tightest distribution (a desirable property for constructing prediction intervals). The 15% dropout configuration has a competitive RMSE compared to the standard implementation. Because the drop out is at such a low probability, the 15% configuration retains a lot more information than the others and is thus the configuration selected for prediction interval construction.

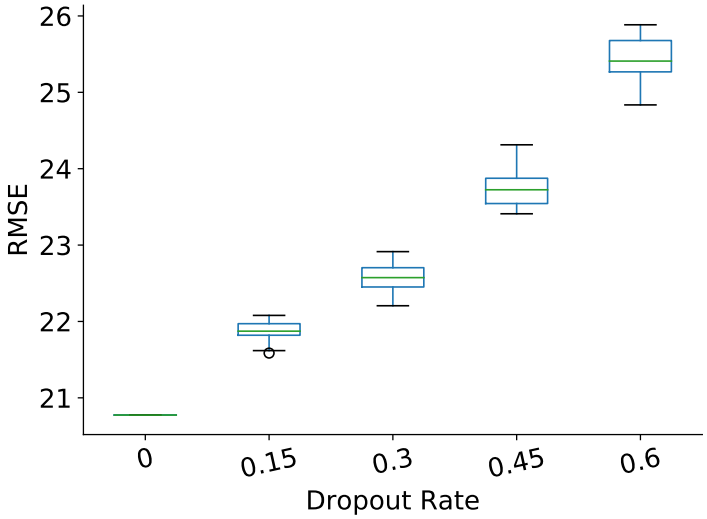


FIGURE 3.4: Dropout RMSE Score Distributions.

The chosen configuration is executed 100,000 times. After forecasting, the distribution of said forecast values is used to construct the prediction intervals. To compute the $(1-\alpha) \times 100\%$ prediction interval, the percentiles at these bounds are used

$$\left(\frac{\alpha}{2}, (1-\alpha) + \left(\frac{\alpha}{2}\right)\right).$$

3.2.7 Bootstrapping

Residuals are re-sampled in the manner detailed in Algorithm 3.

Algorithm 3 Residual Bootstrapping.

1: Once the model is fit, retain the values fitted, y'_t and residuals

$$\varepsilon_t = y_t - y'_t, t = 1, \ldots, n,$$

where $n \leq m$ the forecast horizon

2: For each tuple, $(\boldsymbol{x}_t, y_t)$, in which $\boldsymbol{x}_t$ is the multivariate explanatory variable, add the randomly re-sampled residual, ε_i, to the fitted value y'_t, thus creating synthetic response variables

$$y^*_t = y'_t + \varepsilon_i$$

where i is selected randomly from the list $(1, \ldots, n)$ for every t

3: Using the values $(\boldsymbol{x}_t, y^*_t)$, forecast $\hat{y}$, then refit the model using synthetic response variables y^*_t, again retaining the residuals and forecasts

4: Repeat steps 2 and 3, a total of 100,000 times

The distribution of the forecast values are used to construct the prediction intervals using percentiles in the same manner described above in Section 3.2.6. As in the case of the recurrent layer dropout, this synthetic prediction interval construction technique is advantageous as it also retains the explanatory variables' information.

3.2.8 Anomaly Detection

Anomalies are labelled in YWB. NBD does not contain any golden labels. Anomalies are synthesized by superimposing them on the other three datasets. To ensure soundness of the datasets after said synthesis, the following stochastic methodology is implemented. The types of anomalies are kept constant to ensure variety for general application purposes, i.e. changepoint, collective, and contextual anomalies for AQD, HPC, and BSD, respectively. The parameters that are variable are the size of contamination, the scale of the anomaly or how far they lie from normal observations, and the direction of the two-sided anomality (is an anomalous observation higher or lower than the normal observations in magnitude). The contamination size is set between one to three anomalies per 100 observations, the scale is zero to 200% the magnitude of normal observations, and the direction is uniform in nature. The consequence of conducting the stochastic data synthesis in the outlined manner is that the output data presents with small and large anomalies, that are bidirectional in nature, with contamination size in line with the contamination size of the other two readily anomalous datasets YWB and NBD.

Because anomalies are synthesized for AQD, BSD and HPC, only A1 is used from YWB. Using the other synthetic benchmarks in YWB may not yield any further insights to the analysis. As an illustrative example, the stochastic superimposition of anomalies yields a typical composition as given in Table 3.2.

TABLE 3.2: Anomaly Data Structure Summary.

Data	# Metrics	Type	Contamination
AQD	10	changepoint	0.0133
HPC	7	collective	0.0142
BSD	17	contextual	0.0209
NBD	36	changepoint	0.0191
YWB	67	contextual & collective	0.0176

If a new observation falls within the constructed prediction interval, it is classified as normal. If it falls outside the interval, then:

1. determine by which magnitude the observation falls outside the bounds. This magnitude is deducible using a variant of MIS_α defined earlier (Equation 2.6 in Section 2.3), i.e. the Interval Score (IS_α)

$$\text{IS}_\alpha = (U_t - L_t) + \frac{2}{\alpha}(L_t - y_t)\,\mathbb{1}_{\{y_t < L_t\}} + \frac{2}{\alpha}(y_t - U_t)\,\mathbb{1}_{\{y_t > U_t\}}$$

   If this measure is at least 33% larger than that of the previous observation that fell outside bounds, then
2. employ Chebyshev's inequality [131] which guarantees a useful property for a wide class of probability distributions, i.e. no more than a certain fraction of values can be more than a certain distance from the mean. A suitable threshold can be tuned to avoid many false alerts; this inequality is used here to identify that 99% of the values must lie within ten times the standard deviation.

If an out-of-sample observation falls outside the prediction interval bounds and falls beyond ten times the standard deviation of the observations (the dynamic threshold, $\leq 1\%$ of all observations), it is classified as anomalous.

The value of IS_α detailed above is chosen empirically through optimization, where the objective target is to maximize the model accuracy across all datasets considered. Including all datasets in the objective function makes the proposed anomaly detector better suited to different application domains.

The above anomaly detection procedure is summarized in Algorithm 4, and the structure of the proposed deep learning models and data flow process is presented graphically in Figure 3.5.

Algorithm 4 Anomaly Detection.

if $U_t \leq y_t \leq L_t$ **then**
  y_t is normal
else
  if $\text{IS}_\alpha(y_t) \geq 1.33 \times \text{IS}_\alpha(y_*)$ **and** $y_t > 10 \times \text{std}\{\ldots, y_{t-3}, y_{t-2}, y_{t-1}\}$ **then**
    y_t is anomalous {where y_* is the last anomalous observation}
  end if
end if

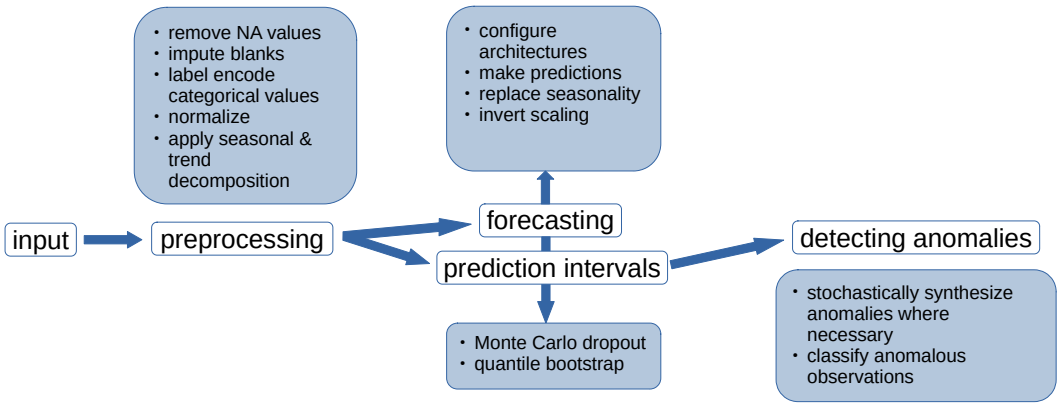


FIGURE 3.5: Flowchart Diagram Depicting the Proposed Deep Learning Model Architectures.

3.3 Results and Discussion

3.3.1 Forecasting

A persistence model [132] gives an RMSE score of 30, 465, 94, 1.8, and 301 respectively for AQD, HPC, BSD, NBD and YWB. Those from the models under consideration, truncated to four decimals, are shown in Table 3.3.

TABLE 3.3: Model Error Scores.

		Data				
		AQD	**HPC**	**BSD**	**NBD**	**YWB**
Statistical Benchmarks	MLR	24.8791	415.8602	43.8581	1.5001	278.6595
	SARIMAX	22.9377	377.8571	34.3472	1.3712	251.8902
	VARMAX	23.1093	401.3319	39.2279	1.4879	266.0144
Deep Learning	LSTM	20.1342	337.9452	31.8951	0.9588	213.8532
	ESN	21.2357	349.1773	31.0451	1.0057	209.3517
	CCN	22.9876	392.5459	35.6843	1.1421	227.1096

3.3.2 Prediction Intervals

For ease of interpretation, the MIS_α is modified to introduce a scaling factor, such that the best performing model has scaled Mean Interval Score (sMIS_α) exactly equal to one, and the rest are larger than one. The worst performing model thus has the largest sMIS_α value. For each significance level α considered

$$\text{sMIS}_{\alpha,A_i} = \frac{\text{MIS}_{\alpha,A_i}}{\min\limits_{A_i} \text{MIS}_{\alpha,A_i}}, \tag{3.18}$$

where A_i is algorithm i (from both statistical and deep learning algorithms). Furthermore, the top two performers at each significance level are highlighted.

Tables 3.4 to 3.8 detail the prediction interval results. At least one deep learning algorithm is always in the top two performers. The interpretation is that deep learning algorithms outperform the traditional statistical methods, or at the very least, are competitive.

TABLE 3.4: Prediction Interval Performance for Air Quality Data.

		$\mathbf{sMIS}_\alpha$			$\mathbf{CS}_\alpha$		
		0.1	**0.05**	**0.01**	**0.1**	**0.05**	**0.01**
Statistical Benchmarks	MLR	1.1704	1.2105	1.2323	0.9031	0.9171	0.9305
	SARIMAX	**1.0778**	1.2093	1.2167	**0.9524**	**0.9668**	**0.9797**
	VARMAX	1.4141	1.2662	1.0881	0.9449	0.9621	0.9761
Deep Learning	LSTM	**1.0000**	1.1437	**1.0000**	**0.9620**	**0.9810**	**0.9824**
	ESN	1.1859	**1.1365**	1.1367	0.9179	0.9308	0.9435
	CCN	1.2103	**1.0000**	**1.0655**	0.9094	0.9285	0.9633

TABLE 3.5: Prediction Interval Performance for Bike Sharing Data.

		$\mathbf{sMIS}_\alpha$			$\mathbf{CS}_\alpha$		
		0.1	**0.05**	**0.01**	**0.1**	**0.05**	**0.01**
Statistical Benchmarks	MLR	1.2131	1.1688	1.2072	0.9062	0.9169	0.9165
	SARIMAX	1.0364	1.2052	1.2708	**0.9493**	0.9501	**0.9553**
	VARMAX	1.1236	1.1796	1.1676	0.9075	0.9107	0.9194
Deep Learning	LSTM	1.1727	1.0872	1.1662	0.9370	**0.9519**	0.9422
	ESN	**1.0000**	**1.0000**	**1.0000**	**0.9587**	**0.9683**	**0.9772**
	CCN	**1.0186**	**1.0388**	**1.0409**	0.9468	0.9472	0.9495

TABLE 3.6: Prediction Interval Performance for Household Power Consumption Data.

		$\mathbf{sMIS}_\alpha$			$\mathbf{CS}_\alpha$		
		0.1	**0.05**	**0.01**	**0.1**	**0.05**	**0.01**
Statistical Benchmarks	MLR	1.2262	1.2123	1.1686	0.9081	0.9117	0.9144
	SARIMAX	1.1081	**1.0181**	1.1012	**0.9413**	0.9514	0.9760
	VARMAX	1.1764	1.1565	1.1148	0.9178	0.9262	0.9322
Deep Learning	LSTM	**1.0449**	1.1005	**1.0211**	**0.9507**	**0.9594**	**0.9767**
	ESN	1.0910	**1.0000**	**1.0000**	0.9372	**0.9516**	**0.9895**
	CCN	**1.0000**	1.1452	1.0708	0.9157	0.9299	0.9571

TABLE 3.7: Prediction Interval Performance for Bearings Data.

		$\mathbf{sMIS}_\alpha$			$\mathbf{CS}_\alpha$		
		0.1	**0.05**	**0.01**	**0.1**	**0.05**	**0.01**
Statistical Benchmarks	MLR	1.3562	1.2020	1.1549	0.9106	0.9134	0.9287
	SARIMAX	**1.0000**	1.1254	1.1832	0.8957	0.9009	0.9071
	VARMAX	1.1764	1.1565	1.2149	0.9075	0.9161	0.9193
Deep Learning	LSTM	**1.0541**	1.1045	**1.0299**	**0.9401**	**0.9412**	**0.9491**
	ESN	1.0910	**1.0000**	**1.0000**	0.9202	**0.9346**	**0.9398**
	CCN	1.1081	**1.0181**	1.1012	**0.9213**	0.9277	0.9301

TABLE 3.8: Prediction Interval Performance for Webscope Benchmark Data.

		$\mathbf{sMIS}_\alpha$			$\mathbf{CS}_\alpha$		
		0.1	**0.05**	**0.01**	**0.1**	**0.05**	**0.01**
Statistical Benchmarks	MLR	1.3054	1.112	1.1499	0.8878	0.8923	0.8971
	SARIMAX	1.1081	**1.0169**	1.1012	**0.9293**	0.9313	0.9360
	VARMAX	1.2262	1.2123	1.1686	0.8981	0.9108	0.9142
Deep Learning	LSTM	1.0812	**1.0000**	**1.0000**	0.9271	**0.9360**	**0.9494**
	ESN	**1.0546**	1.1050	**1.0233**	**0.9307**	**0.9394**	**0.9464**
	CCN	**1.0000**	1.1452	1.0708	0.9157	0.9299	0.9309

Said outperformance is also evidenced by the granular result for aggregated daily RMSE for HPC, as displayed in Figure 3.6. Figure 3.6 is supplementary to Table 3.6 as it gives a granular score at a daily level, which is consistent with the overall score detailed in the latter.

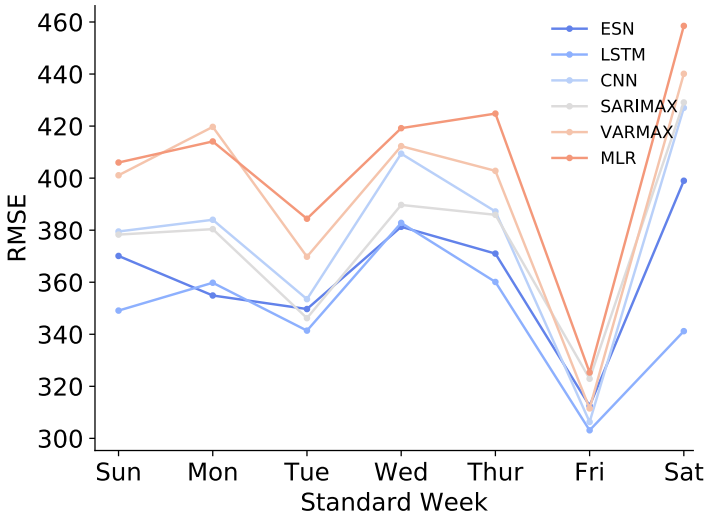


FIGURE 3.6: Overall Daily RMSE for Household Power Consumption Dataset.

Figure 3.7 illustrates the point forecasts and prediction intervals for ESN (best performer for BSD) at the last 120 hours of the forecast horizon. The

results are consistent with the findings of Makridakis, Spiliotis, and Assimakopoulos [33], for example, who noted that deep learning models produce better prediction intervals, albeit their setting was in the univariate case.

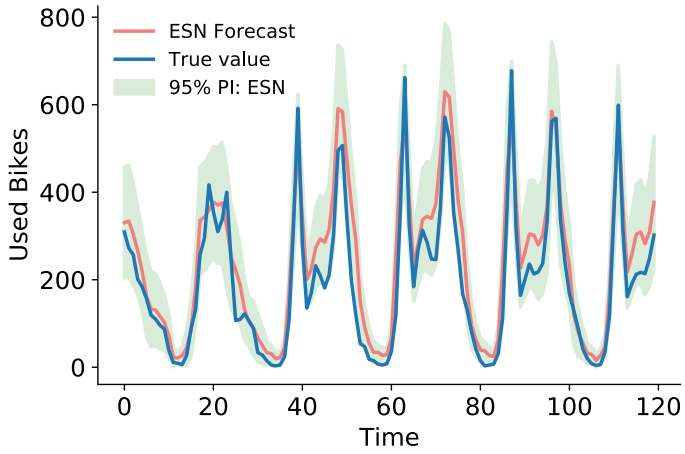


FIGURE 3.7: Point Forecasts and Constructed Prediction Intervals for Bike Sharing Dataset.

ESN is the best performing architecture for BSD at all prediction intervals considered. This result is consistent with the RMSE scores in Table 3.3.

3.3.3 Anomaly Detection

Figures 3.8 and 3.9 detail the distributions for the F_1- and *Ed*-scores, respectively, for all the models applied to each dataset. The F_1-score is varied for each dataset, but the results indicate LSTM is the best performer, followed in every instance by either ESN or CCN. The worst performer is MLR across all datasets. Since the F_1-score is the harmonic average of Precision and Recall, where one is the perfect score and zero is worst, the results indicate that deep learning models are better at learning the dynamic threshold.

Again, with the *Ed*-score, the statistical methods are outperformed by the deep learning models. This result follows as better dynamic threshold calibration leads to earlier detection of anomalies.

3.4 Conclusion

The point of departure is that (i) deep learning models have been shown to outperform traditional statistical methods at forecasting, and (ii) deep learning has been shown to produce better uncertainty quantification. In

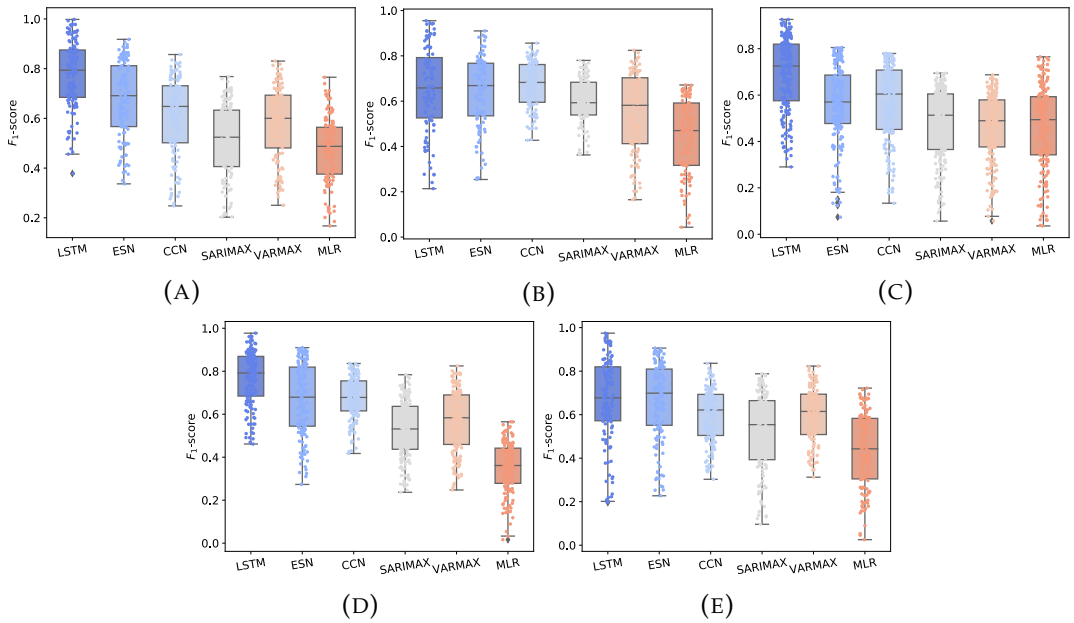


FIGURE 3.8: F_1-Score Distributions for Each Dataset and All the Models. (A) AQD. (B) BSD. (C) HPC. (D) NBD. (E) YWB.

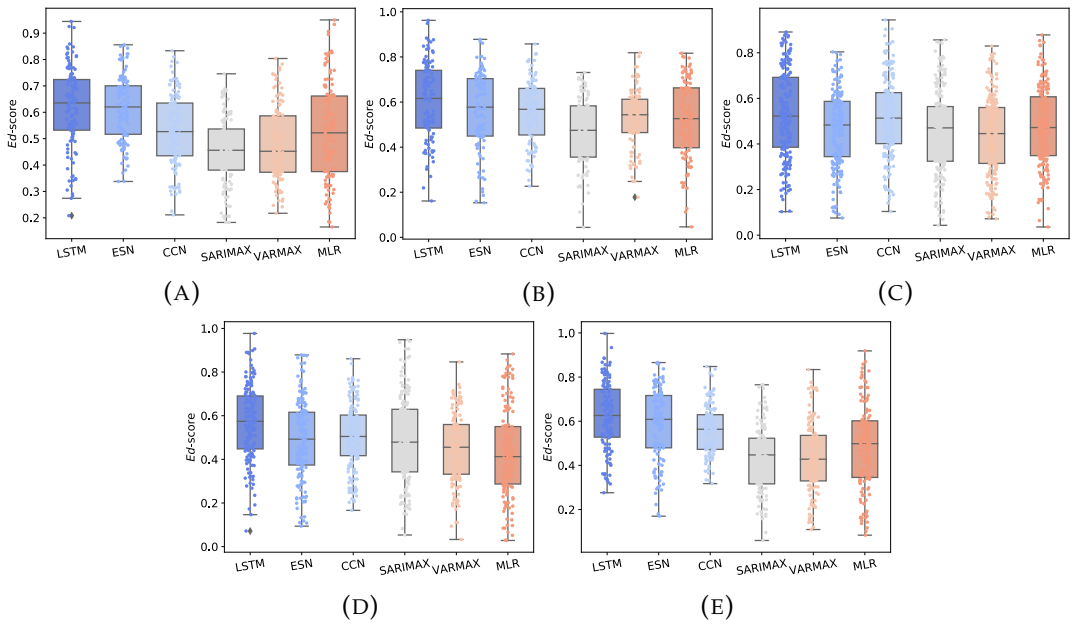


FIGURE 3.9: *Ed*-Score Distributions for Each Dataset and All the Models. (A) AQD. (B) BSD. (C) HPC. (D) NBD. (E) YWB.

this chapter, (i) point forecasts are produced, along with (ii) their associated prediction intervals, which are produced using computational means where no built-in probabilistic constructs exist, and (iii) said prediction intervals are used to detect anomalies on different multivariate datasets.

The methodology presented, indicates that better point forecast accuracy leads to the construction of better prediction intervals, which in turn leads to the main result presented: better prediction intervals lead to more accurate and quicker anomaly detection (in terms of early detection from the *Ed*-scores). The improvement is as a result of better parameterized models being capable of more accurate forecasts. These models in turn are used in the prediction interval construction process, and subsequently multivariate anomaly detection.

The goal was not to prove that deep learning is far superior to statistical methods in this regard, but to show that when proper methodology is followed, it can be competitive in the worst case. This result is important as it opens the way for the wide-scale adoption of deep learning in other applications.

The computational cost of the deep learning algorithms is not discussed in this chapter, and this could be an avenue for further exploration. The algorithmic time complexity is known for each model, with ESN being the fastest deep learning architecture. Even at that, it is still not as fast as any of the statistical methods. In terms of the deep learning architectures, future work can consider techniques for improving upon the cost of computation, so that even in this aspect they become competitive to their statistical counterparts.

Finding techniques to address this last issue in particular will enable researchers to study multivariate time series in more detail and at a grander scale. There exists a need for conducting the kind of analysis conducted here on multivariate time series at the scale of, say, the Makridakis-4 (M4) Forecasting Competition [33].

In the next chapter, the hypothesis that *merging some of the best characteristics of the two approaches (classical statistics and deep learning) yields better results* is investigated. The investigation is conducted with an application to morbidity and mortality modelling.

Chapter 4

Statistics and Deep Learning-based Hybrid Model

The work adapted and presented in this chapter appears in the following publication: Mathonsi and van Zyl [61]. T. Mathonsi participated in the conception of the research, formulated and implemented the methods, conducted the experiments and analysed the results, and was the principal participant in the writing of the manuscript. T. van Zyl participated in the conception of the research and reviewed the manuscripts.

Summary

Hybrid methods have been shown to outperform pure statistical and pure deep learning methods at forecasting tasks and quantifying the uncertainty associated with those forecasts (prediction intervals). One example is Exponential Smoothing Recurrent Neural Network (ES-RNN), a hybrid between a statistical forecasting model and a deep recurrent neural network variant. ES-RNN achieves a 9.4% improvement in absolute error in the Makridakis-4 Forecasting Competition. However, this improvement and similar outperformance from other hybrid models have primarily been demonstrated only on univariate datasets. Difficulties with applying hybrid forecast methods to multivariate data include (i) the high computational cost involved in hyperparameter tuning for models that are not parsimonious, (ii) challenges associated with modelling the auto-correlation inherent in the data, as well as (iii) complex dependency (cross-correlation) between the covariates that may be hard to capture. This chapter presents Multivariate Exponential Smoothing Long Short-Term Memory (MES-LSTM), a generalized multivariate extension to ES-RNN, that overcomes these challenges. MES-LSTM utilizes a vectorized implementation and is tested on coronavirus disease of 2019 (COVID-19) morbidity and mortality datasets. The findings are that the

proposed hybrid approach shows consistent, significant improvement over both pure statistical and pure deep learning methods at forecast accuracy and prediction interval construction.

4.1 Introduction

Lessons from the comparison of deep learning and statistical methods in Chapter 3 are taken forward and applied to the experimentation in this chapter. One such lesson is that forecast machinery that can produce accurate forecasts and build reliable prediction intervals can be adapted and applied successfully to the task of anomaly detection.

Another notable lesson is derived from the main result, that with the presented approach for anomaly detection, deep learning outperforms (or is at least competitive to) classical statistical methods. However, deep learning still suffers from severe computational cost when compared to the statistical counterparts. In this chapter, a hybrid model is proposed that merges some of the best characteristics from the two research streams (classical and deep learning-based approaches), and overcomes the problems highlighted in the above Summary. The performance of the proposed method is tested and compared to benchmarks with an application to morbidity and mortality modelling using OTD, DIC, and OWID datasets (see Section 2.2).

Morbidity and mortality modelling is crucial for planning in the global economy, national healthcare systems, and other industries such as insurance. Practitioners from statistics, machine learning, and actuarial backgrounds have invested into improving the accuracy of morbidity and mortality forecasting. Some recent advances have emerged from the fields of hybrid models, and interpretable models such as Temporal Fusion Transformers [19]. Despite these advances, the recent devastating global impact of the coronavirus disease of 2019 (COVID-19) virus has highlighted the importance of effective planning from government agencies and healthcare bodies across the globe. This kind of planning requires reliable projections into the future, and as a result there exists a need for improved forecasting techniques and methods.

Smyl [34] developed a hybrid method for generating point forecasts and quantifying the uncertainty associated with those point forecasts. The model quantifies uncertainty by producing prediction intervals at the 1%, 5% and

10% levels of significance. The hybrid method combines Exponential Smoothing (ES) [133] with Recurrent Neural Networks (RNN) [116] and the resulting scheme is referred to as ES-RNN. This name is a bit of a misnomer as Smyl's [34] methodology actually involves combining ES, with a variant of RNN called Long Short-Term Memory (LSTM) [101], [119].

Smyl [34] purports that their method produces "*...forecasts that are more accurate than those generated by either pure statistical or pure [machine learning] approaches, thus exploiting their advantages while avoiding their drawbacks.*" The hybrid method, ES-RNN, outperformed pure statistical and pure deep learning methods submitted alongside it to the M4 Forecasting Competition [33]. The M4 Competition provides to Competition participants 100,000 datasets from a cross-section of industries and applications for the entrants to apply their techniques to.

Redd, Khin, and Marini [134] extend the Smyl ES-RNN [34] hybrid technique by executing a graphical processing unit (GPU) implementation (GPU-ES-RNN). Using a vectorized and GPU-based implementation of the original ES-RNN, Redd, Khin, and Marini [134] state they achieve up to 322x increase in training speed, largely attributed to batching and parallelization. In addition to the increased efficiency, the authors report that their artificial neural network produces performance results similar to those reported in the original Smyl submission. Furthermore, the GPU implementation migrates the original C++ code to Python and PyTorch.

In both cases (ES-RNN and GPU-ES-RNN), the authors focus on univariate datasets. However, a significant number of real world forecasting involves multivariate datasets [22]. In some cases, researchers have found it useful to augment their univariate models with multivariate exogenous factors or multivariate analogues of their models to improve predictive accuracy [56]. In addition, the complexities associated with modelling morbidity and mortality often require multiple data sources in order to fit a model that better describes the real world situation [135].

This chapter extends both the ES-RNN and GPU-ES-RNN research by adapting the ES and LSTM hybrid method to make it applicable to the widely prevalent multivariate case. By incorporating exogenous factors throughout, the proposed extension departs from the classical univariate case and, as such, poses more complications associated with, for instance, auto-correlation inherent in the data and cross-correlation dependencies between covariates.

The proposed model incorporates exogenous covariates through the use of global and local variables, i.e. those derived from all the data or large

segments of it, and those derived from separate covariates. This combination allows the model to cross-learn at the same time leverage information presented at a granular level.

Essentially, there are three workstreams i.e. multivariate exponential smoothing, recurrent neural networks in general and LSTM in particular, and hybrid methods. Next, some relevant recent advances and the current state-of-the-art from all three workstreams are reviewed and their advantages and disadvantages are discussed.

4.1.1 Recent Advances and the Current State-of-the-Art

Long Short-Term Memory in Morbidity and Mortality Modelling

The urgency for developing solutions to combat the recent COVID-19 pandemic has provided a unique opportunity due to (*i*) a multitude of open datasets and (*ii*) extensive related research exploring the performance of multivariate forecasting models. In the domain of COVID-19-related research, LSTM networks have been applied by Kırbaş, Sözen, Tuncer, *et al.* [136], for instance, who focus their attention on the spread of the pandemic in countries including Denmark, Belgium, Germany, France, United Kingdom, Finland, Switzerland and Turkey. Kırbaş, Sözen, Tuncer, *et al.* [136] model using Auto-Regressive Integrated Moving Average (ARIMA) and Nonlinear Autoregression Neural Network (NARNN) [137] as benchmarks and report that LSTM is more accurate in terms of forecasting the cumulative cases of infected individuals.

Chandra, Jain, and Singh Chauhan [73] use variations of LSTM including Bidirectional LSTM (Bi-LSTM), and encoder-decoder LSTM (ed-LSTM) models for multi-step intra-country forecasting in the short-term in India. They cite challenges in the modelling process caused by difficulties in capturing, in any available COVID-19 dataset, potentially important multivariate factors such as population density, travel logistics, and other societal issues (e.g. the general standard of living). The authors purport that if such data were available, these exogenous factors could be useful in improving predictive skill in the modelling process.

Chimmula and Zhang [138] base their forecasts on training data acquired from the John Hopkins University and the Canadian Health Authority. They use a standard implementation of LSTM. They forecast the pandemic ending in June 2020 (which at the time of writing this is known to be incorrect).

Shahid, Zameer, and Muneeb [139] model the rises and declines of confirmed cases, deaths and recoveries in ten countries, including China. They rate their model performances from best to poorest as Bi-LSTM, LSTM, Gated Recurrent Units (GRU) [124], Support Vector Regressor (SVR) [140] and ARIMA. They report Bi-LSTM has better performance over the other models with the lowest MAE and RMSE values of 0.0070 and 0.0077, respectively.

As evidenced by previous related research, there are successes and failures with using LSTM to forecast COVID-19. Various factors contribute to the at times poor performance of LSTM and its variations in this regard. One such factor is that the pandemic is relatively recent, and any available dataset is a small fraction of the requisite volume of training data the data-hungry artificial neural networks require [111]. LSTM has been shown to perform well in a variety of applications with datasets that are sufficiently long [60]. In this chapter, the problem of limited data is circumvented by integrating a small, parsimonious LSTM network in the proposed model. Details about the structure of the proposed model are offered in Section 4.2.4.

Statistics and Deep Learning-based Hybrid Models

Univariate forecasting with the aid of pure machine learning has been conducted as far back as the works of Hu [9]. Techniques merging machine learning with statistical methods have since gained popularity.

Recently, the original ES-RNN hybrid technique won over hundreds of other submissions both in terms of (*i*) point forecasts as well as (*ii*) their associated prediction intervals. Both components are difficult to model, but perhaps the latter even more so. Deep learning models often do not have a mechanism for quantifying uncertainty [40]. In such cases, prediction intervals must be constructed using computational mechanisms [60].

Oreshkin, Carpov, Chapados, *et al.* [141], however, produce forecast machinery comprised of deep, fully connected layers and report that their model outperforms ES-RNN on the M4 Competition dataset. Based on Neural Basis Expansion Analysis (NBEATS), Oreshkin, Carpov, Chapados, *et al.* [141] conclude that pure deep learning is better than hybrid methods as their method outperforms the best statistical and the best hybrid method by 11% and 3%, respectively. The technique has since been extended to include exogenous factors (NBEATS-x) [142].

Multivariate Exponential Smoothing

There are several notable works concerning multivariate extensions of exponential smoothing. Jones [143] applies recursion for estimating the smoothing matrix. Enns, Machak, Spivey, *et al.* [144] consider a class of various exponential smoothing models and use them as a proxy for the univariate exponential smoothing counterpart. Trigg and Leach [145] adapt the smoothing matrix of their so-called adaptive models periodically, with the aid of maximum likelihood estimation. Harvey [146] further simplifies the class of models proposed by Enns, Machak, Spivey, *et al.* [144] and the simplifications enable the use of univariate smoothing models in forecasting tasks for correlated time series. Moreover, Harvey's [146] results are also valid in the case where the smoothing models exhibit polynomial trend and seasonal components. Pfeffermann and Allon [147] focus on structural models aimed at producing optimal forecasts. By building upon previous multivariate time series exponential smoothing research, Pfeffermann and Allon [147] also offer detailed instructions for parameter initialization and model-fitting.

Harvey's approach is most suitable for the interests of this thesis for two reasons. One, because the method offers reduced computational complexity; and two, adapting Harvey's approach yields similar results to using an adapted version of, for example, the more complex Pfeffermann and Allon [147] approach. The second reason is due to the LSTM's ability to model complex interdependency between covariates [148], [149], which negates the need to model the cross-correlation using the preprocessing layer statistical methods.

4.1.2 Motivation

Further evidence has emerged since the end of the Makridakis-5 (M5) Accuracy and Uncertainty Competitions [58], [59] that pure deep learning models may be better suited for hierarchical forecasting. However, hybrid methods are still worth investigating in the setting of multivariate forecasting with exogenous variables. Starting here as a point of departure, the research hypothesis in this chapter is that *a hybrid technique such as ES-RNN may work just as well in the higher dimensional setting by exploiting a simple LSTM's ability to model cross-correlation*.

Furthermore, extending the current ES-RNN hybrid forecasting research to the multivariate case is crucial as multivariate datasets are usually more

representative of realistic forecasting scenarios likely to be encountered in mortality and morbidity modelling, as well as other applications.

4.1.3 Contribution

The novel contribution in this chapter can be summarized as follows:

- the current research, which focuses on ES and RNN hybrid methods for univariate forecasting, is extended to a multivariate framework; thus, tests previously conducted empirically only on univariate data are extended to multivariate mortality data with exogenous variables, i.e. are hybrid methods better than pure statistical or pure deep learning methods at (*i*) forecasting tasks and (*ii*) quantifying forecast uncertainty? In particular, a natural generalised extension of Smyl's ES-RNN to higher dimensions is presented;
- the proposed forecast engine MES-LSTM is presented, which is an efficient generalization, and as such, may be applied not only to the multivariate case but also to the univariate setting; and
- whereas previous (univariate) research on forecasting hybrid methods primarily focuses on the multiplicative seasonality case, both multiplicative and additive cases are considered here, with automatic adaptation to the case most applicable to the particular dataset.

4.2 Methodology

In this chapter, GPU computation is employed by utilising Tensorflow's eager execution to transform the original Smyl [34] ES-RNN to a generalised multivariate implementation. In their GPU implementation, Redd, Khin, and Marini [134] initialise per-series variables according to the guidelines of the M4 Competition. The method proposed here initialises per-covariate parameters, and has a vectorized ES-LSTM, which is termed Multivariate ES-LSTM (MES-LSTM).

The other difference between ES-RNN and MES-LSTM is that the latter uses the most suitable between additive or multiplicative seasonality, whereas previous authors have only considered multiplicative seasonality. The most suitable seasonality structure in each case is ascertained by tracking the exponential smoothing fit on the training data.

In the remainder of this chapter, the quintessential aspects of MES-LSTM are explicitly defined. The description is analogous to that of Smyl [34],

with relevant extensions made to suit the novel multivariate presentation. The proposed model comprises two distinct layers: an exponential smoothing layer inside a preprocessing module, and an LSTM layer used for learning the dependency between the covariates. This methodology is consistent with Section 4.1.1, where the choice to expand upon Harvey's approach [146] over the Pfeffermann and Allon [147] approach for multivariate exponential smoothing is outlined.

Concisely, the proposed model learns the parameters associated with each covariate in the preprocessing step, then learns the correlation between the covariates in the deep learning layer. Parameters optimized in both steps are then used at the inference stage, where the prediction of multivariate point forecasts are produced, and the uncertainty associated with these forecasts is quantified.

4.2.1 Preprocessing layer

For each covariate in the multivariate sample space, let $\{y\}_t$ represent, for example, a weekly time series (weekly series for ease of exposition only). Assume the weekly series can be decomposed into the additive form

$$y_t = \iota_t + \tau_t + s_t + \zeta_t, \tag{4.1}$$

where ι_t is the level in week t, s_t the seasonal effect and ζ_t the noise term centered at zero with constant variance. Let $\hat{\iota}_t$ denote the estimated level in week t and $\hat{\tau}_t$ the corresponding trend estimate. Let the estimate of the seasonal effect at time t corresponding to week $t+i$, $i = 1, 2, \ldots, 52$ be denoted by $\hat{s}_{t,t+i}$. The estimates $\hat{s}_{t,t+i}$ satisfy the condition $\sum_{i=1}^{52} \hat{s}_{t,t+i} = 0$. When a new observation y_{t+1} becomes available, all estimated components, i.e. seasonality, trend, and level, are updated with the aid of three smoothing constants $0 \leq \varrho, \gamma, \delta \leq 1$ as follows,

$$\hat{\iota}_{t+1} = \varrho\left(y_{t+1} - \hat{s}_{t,t+1}\right) + (1-\varrho)\left(\hat{\iota}_t + \hat{\tau}_t\right) \tag{4.2}$$

$$\hat{\tau}_{t+1} = \gamma\left(\hat{\iota}_{t+1} - \hat{\iota}_t\right) + (1-\gamma)\,\hat{\tau}_t \tag{4.3}$$

$$\hat{s}^*_{t+1,t+1} = \delta\left(y_{t+1} - \hat{\iota}_{t+1}\right) + (1-\delta)\,\hat{s}_{t,t+1} \tag{4.4}$$

$$\sum_{i=0}^{51} \hat{s}^*_{t+1,t+1+i} = 0. \tag{4.5}$$

Equations (4.4) and (4.5) define a two-step computational procedure of the seasonal effects $\hat{s}_{t+1,\,t+1+i}$. The second step standardizes the initial values

computed in the first (with the asterisk indicating "intermediary value") so that they sum to zero.

The forecast at time t of a future out-of-sample realization ($m > i$) y_{t+m} is given by

$$\hat{y}_{t,t+m} = \hat{\imath}_t + m\hat{\tau}_t + \hat{s}_{t,t+m}. \tag{4.6}$$

For the multiplicative seasonality case, make the substitution $\{y\}_t = \ln(\{y\}_t)$. This substitution amounts to assuming the original series level changes in approximately constant rates instead of constant increments. The substitution also requires changing the condition imposed on the estimates of the multiplicative seasonal factors over the 52 weeks from summing to zero to their geometric mean equalling one. Equations (4.2) to (4.5) then become

$$\hat{\imath}_{t+1} = \varrho\left(\frac{y_{t+1}}{\hat{s}_{t,t+1}}\right) + (1-\varrho)\hat{\imath}_t\hat{\tau}_t \tag{4.7}$$

$$\hat{\tau}_{t+1} = \gamma\left(\frac{\hat{\imath}_{t+1}}{\hat{\imath}_t}\right) + (1-\gamma)\hat{\tau}_t \tag{4.8}$$

$$\hat{s}^*_{t+1,t+1} = \delta\frac{y_{t+1}}{\hat{\imath}_t\hat{\tau}_t} + (1-\delta)\hat{s}_{t,t+1} \tag{4.9}$$

$$\left(\prod_{i=0}^{51}\hat{s}^*_{t+1,t+1+i}\right)^{\frac{1}{51}} = 1, \tag{4.10}$$

and the predictive Equation (4.6) becomes

$$\hat{y}_{t,t+m} = \hat{\imath}_t\left(\hat{\tau}_t\right)^m\hat{s}_{t,t+m}. \tag{4.11}$$

In both the additive and multiplicative seasonality cases expressed above, the initial smoothing parameters are estimated as follows. The level is taken to be the overall in-sample average. The trend is initialized as the difference between the first and last in-sample observations, divided by the total number of increments. Seasonality is initialized by the deviations between the first week's observations and the level plus trend fit. More details about the dimensionality of the parameter space are given in Section 4.2.4.

Now that the formulations for both the additive and multiplicative seasonality cases are well defined, the preprocessing layer is merged with the deep learning component as follows. For simplicity, the input size and the size of the output predictions are kept equal ($N = m$).

For the univariate additive case

$$\hat{y}_{t,\ t+1,\ldots,t+m} = \hat{\imath}_t + [t+1,\ \ldots,\ t+m]^\top \odot \text{LSTM}(X_t) + \hat{s}_{t,\ t+1,\ \ldots,\ t+m}, \quad (4.12)$$

where $\odot$ is the Hadamard product, and X_t denotes a vector of de-trended and de-seasonalized observations of which a scalar element x_i is given by

$$x_i = y_i - \hat{\imath}_i - \hat{s}_i, \text{ for } i = t+1,\ \ldots,\ t+m. \quad (4.13)$$

For the multivariate additive case

$$\hat{\boldsymbol{y}}_{t,\ t+1,\ \ldots,\ t+m} = \hat{\boldsymbol{\imath}}_t + [t+1,\ldots,\ t+m]^\top \odot \text{LSTM}(\boldsymbol{X}_t) + \hat{\mathbf{s}}_{t,\ t+1,\ \ldots,\ t+m}, \quad (4.14)$$

where (if there are, say, k covariates in total) $\boldsymbol{X}_t$ is a k x m matrix of de-trended, de-seasonalized features of which a vector component (for each covariate) $\boldsymbol{x}_i$ is calculated via the equation

$$\boldsymbol{x}_i = \boldsymbol{y}_i - \hat{\boldsymbol{\imath}}_i - \hat{\mathbf{s}}_i, \text{ for } i = t+1,\ \ldots,\ t+m. \quad (4.15)$$

Here, the vectors $\hat{\boldsymbol{\imath}}_i$ and $\hat{\mathbf{s}}_i$ are the same dimension as $\boldsymbol{y}_i$.

The univariate multiplicative seasonality case is given by

$$\hat{y}_{t,\ t+1,\ \ldots,\ t+m} = \text{LSTM}(X_t) \odot \hat{\imath}_t \odot \hat{s}_{t,\ t+1,\ \ldots,\ t+m} \quad (4.16)$$

$$x_i = \frac{y_i}{\hat{\imath}_i \hat{s}_i} \text{ for } i = t+1,\ \ldots,\ t+m, \quad (4.17)$$

where the LSTM takes as input a vector of de-trended and de-seasonalized observations of which a scalar element x_i is computed using Equation (4.17).

Finally, the multivariate multiplicative case is thus expressed as

$$\hat{\boldsymbol{y}}_{t,t+1\ldots t+m} = \text{LSTM}(\boldsymbol{X}_t) \odot \hat{\boldsymbol{\imath}}_t \odot \hat{\mathbf{s}}_{t,t+1\ldots t+m} \quad (4.18)$$

$$\boldsymbol{x}_i = \boldsymbol{y}_i \odot (\hat{\boldsymbol{\imath}}_i \odot \hat{\mathbf{s}}_i)^{-1} \text{ for } i = t+1,\ \ldots,\ t+m, \quad (4.19)$$

where the LSTM takes as input a size k x m matrix $\boldsymbol{X}_t$ of de-trended, de-seasonalized observations, where each vector component $\boldsymbol{x}_i$ is computed via Equation (4.19).

4.2.2 Deep Learning Layer

The neural network data flow is presented in Figure 4.1. The input is a matrix composed of k vectors each of size m. After preprocessing, the matrix is input

to the LSTM layer. The LSTM layer output (composed of v predictands each sized n) feeds into a dense layer, then postprocessing is conducted to produce the model's final output. LSTMs employ gated connections, are good at modelling latent representations with temporal dependency, and as a result, are better than vanilla RNNs [101]. The deep learning and exponential smoothing layers are optimized consecutively, as depicted in Figure 4.1.

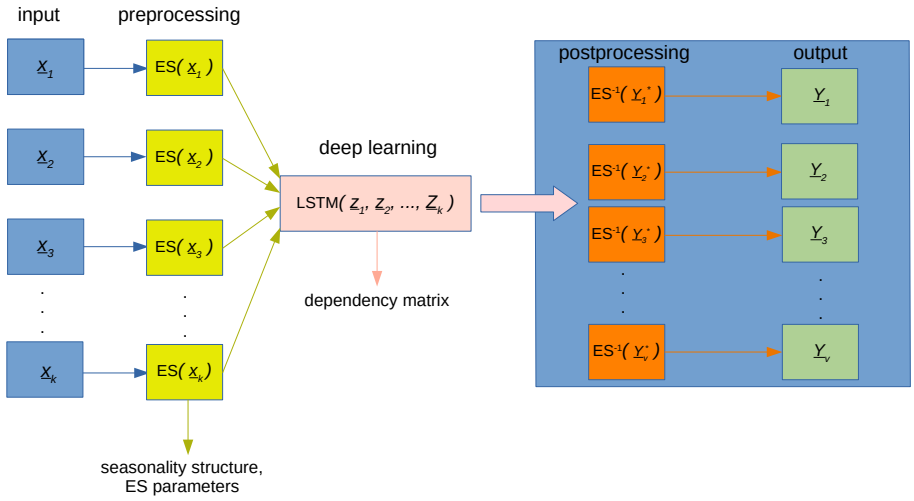


FIGURE 4.1: MES-LSTM Data Flowchart.

In order to quantify the uncertainty associated with the point forecasts, the above architecture is modified as follows. Instead of the dense layer that back-propagates and learns scalar sets of weights and biases, a densely-connected layer class with Flipout estimator [150] is employed, that learns a *distribution* of weights and biases. In this variational probabilistic setting, traditional back-propagation is replaced by Bayes-by-backprop [151].

Flipout [150], through Monte Carlo simulation, approximates a distribution of weights by integrating over the kernel and bias. By assuming the kernel and biases are drawn from distributions, the Flipout [150] is able to implement the Bayesian Variational inference dense-flipout layer, analogous to the above (traditional) dense layer. Using samples from the kernel and bias posteriors, this dense-flipout layer is able to run stochastic forward passes. Another difference between this stochastic process and the analogous densely connected layer, is the use of variational inferencing by minimizing the KL-divergence [152]. The uncertainty quantification is implemented using Tensorflow Probability [153].

In effect, forecasts can now be produced using a sample from the distribution of weights and biases. Applying Monte-Carlo simulation, each time

a forecast is produced, one can iteratively compile a distribution of forecasts through a process similar to the one described in Section 3.2.6 and 3.2.7.

Appropriate percentiles are then used to extract the desired prediction intervals from the forecast distribution. So, to compute the $(1-\alpha)\times 100\%$ prediction interval, use the percentiles at

$$\left(\frac{\alpha}{2}, (1-\alpha) + \left(\frac{\alpha}{2}\right)\right). \tag{4.20}$$

In terms of prediction interval construction, the variational inference approach described above, Flipout, is similar to dropout [18] and bootstrapping [128], [129] as they all use quantiles extracted from synthetic distributions to produce the final intervals.

The key differences in all three methods is how their respective distributions are constructed, and their relative uncertainty quantification skill. See for instance the work of Wen, Vicol, Ba, *et al.* [150], where Flipout has been shown to outperform dropout. In addition, a fortuitous side-effect of the variational inference (and dropout methods) is a reduction in epistemic uncertainty, which in turn regularizes the network and addresses any concerns that may arise from the possibility of a lack of sufficient training data, and issues associated with overfitting.

The data flow process for quantifying uncertainty is not dissimilar to Figure 4.1. The only difference is that the deep learning layer now produces probabilistic forecasts instead of deterministic ones.

To summarize, the hybrid model takes as input a matrix (sized number of training observations by number of predictors). The preprocessing layer computes all the exponential smoothing parameters as outlined in Section 4.2.1 above. The output is then fed directly to the deep learning layer, which discerns the inherent dependency within the covariates. The first step in this two-step methodology is inspired by the works of Harvey [146], who, through simplifications, enables forecasting related time series through the use of univariate smoothing models. This two-step methodology for MES is found to yield similar results to using the one-step formulation given by Pfeffermann [147], with the proposed method enjoying the added benefit of reduction in computational cost. The output from the LSTM is then fed directly to the dense layer or dense-flipout layer in the case of point forecasts and prediction interval construction, respectively. Finally, seasonality and trend estimates are added back to the intermediate output. One now has the final model forecasts and prediction intervals, respectively.

Figure 4.2 illustrates the proposed model architecture zoomed in to the deep learning layer. The input shape is (k, m) where there are k covariates each with m observations in the training data. The network's intermediate output (before postprocessing) is generated by feeding the output of the LSTM through to a dense layer with a Rectified Linear Units (ReLU) activation function [154] (see Appendix A). The size of the LSTM, S, is deduced empirically using the technique described in Section 4.2.4 below. The size of the dense layer (or dense-flipout layer for uncertainty quantification) and correspondingly the output layer is determined by the number of predictands v. The output in the schematic then goes through postprocessing to meet the required format. It is re-trended and re-seasonalized (using the parameter estimates from the exponential smoothing equations) to arrive back at the format and scale presented by the ground truth data.

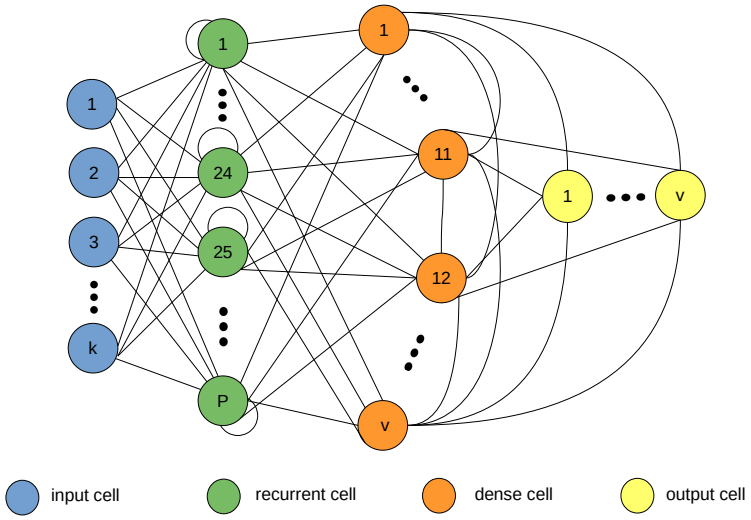


FIGURE 4.2: MES-LSTM Architecture Diagram.

4.2.3 Benchmarks

For benchmarking MES-LSTM, the statistical methods used are Multiple Linear Regression (MLR) [112], Vector Autoregression Moving-Average with Exogenous Regressors (VARMAX) [113] and Seasonal Autoregressive Integrated Moving-Average with Exogenous Regressors (SARIMAX) [114]. The deep learning benchmarks are vanilla LSTM (without direct incorporation of a preprocessing layer as described in Subsection 4.2.1), Light Gradient Boosting Machine (LGB) [155], and a deep autoregressive recurrent neural network (DeepAR) [39], which is built on multiple LSTM networks.

The choice of the pure deep learning benchmarks/baselines models is based on, for one, the architecture of the proposed model. Because the proposed model is a statistical-and-LSTM hybrid, it makes sense to focus on LSTM w.r.t. pure deep learning techniques. Secondly, the choice of LGB (and to some extent LSTM) is motivated by the findings of the M5 Competitions. Almost all of the top 50 submissions for the M5 Accuracy [53] and Uncertainty [54] competitions use a variation of LGB. In particular, considering the top five: submissions ranked first, second, fourth, and fifth from the Accuracy competition [53] incorporate LGB into their model, and the submission ranked third uses DeepAR. From the top five submissions in the M5 Uncertainty competition [54], those ranked first, second, third, and fifth incorporate LGB, while the submission ranked fourth incorporates LSTM.

4.2.4 Model Hyperparameter Tuning

Note from the formulation given in Equations (4.14) and (4.15) (multivariate model with additive seasonality), and Equations (4.18) and (4.19) (multivariate model with multiplicative seasonality), there is no need to compute the estimates for the trend component $\hat{\boldsymbol{\tau}}_t$ and this significantly reduces the hyperparameter search space of the preprocessing layer. The initial estimates of the coefficients for level and seasonality are deduced by calculating primer estimates following the classical exponential smoothing Equations (4.2) and (4.4), as well as Equations (4.7) and (4.9) for the respective additive and multiplicative seasonality cases. The level is initialized as the in-sample average, and the initial seasonality is the deviations between the level plus trend fit and the first week's observations. If there are k variables in the dataset, the model computes and stores $(k+1) \times (2+P)$ exponential smoothing parameters, where P indicates seasonality length.

The deep learning layer comprises a small model with relatively few parameters. The reasons for this model configuration are two-fold. First, there is a relatively small amount of data for training, and secondly, a large over-parameterized network might overfit and not generalize well to the test data [156].

By iterating over a subset of LSTM sizes and training epochs, one can select the model configuration with the best forecast accuracy for the validation data and fewest parameters to fit. Optimization is conducted for the batch

size and number of samples the rolling window looks into the past to forecast the next example. The hyperparameter search space for the deep learning layer is summarized in Table 4.1. The best five configurations in terms of forecasting all of the predictands are inspected (w.r.t. the error metrics detailed in Section 2.3). Then the model configuration that is most parsimonious is selected in order to ensure the model best mitigates overfitting over multiple runs. Parsimony is understood to mean the smallest ES-LSTM with adequate performance is selected, in line with how cascading is terminated in Fahlman and Lebiere's CCN [1] described in Section 3.2.3. According to this methodology, the best configuration is given by an LSTM of size 50, trained over 25 epochs, using batches sized 16, with an input window of 14 days.

TABLE 4.1: Configuration Grid for Deep Learning Layer Hyperparameter Search Space.

Hyperparameter	**Search Space**
LSTM size	50, 55, 60, . . . , 150
Epochs	15, 20, 25, . . . , 75
Batch size	8, 16, 24, . . . , 64
Input window	7, 14, 21

The order of the VARMAX model is deduced by conducting a grid search for optimization in much the same way as above for MES-LSTM. The maximum iterations for maximum likelihood search are set to 200 (four times the default in Python) to ensure convergence. The trend component is deduced through a grid search varying the trend polynomial from constant, linear, quadratic, to cubic. The predictors are auto-regressed and the predictands are declared as exogenous.

The model hyperparameter optimization process for SARIMAX is exactly the same as for VARMAX. In both cases the grid search is performed and the model with the smallest Akaike information criteria (AIC) [157] is chosen. For simplicity, invertibility is not enforced on the moving average polynomials. Relaxing the invertibility constraint also ensures more of the models are estimable. In cases where there are several suitable model configurations (equal AIC values) the model that is most parsimonious is chosen (smallest product from order elements).

The MLR is fit using Ordinary Least Squares (OLS) [112]. The optimal hyperparameters for LSTM are plugged into the DeepAR, with additional searches for the dropout rate (0.1, 0.15, 0.2), and likelihood (noise model for probabilistic forecasts, in order to discern uncertainty) between Gaussian

and student-T. The LGB model is tuned considering tree maximum depth (capped at 5), maximum leaves (capped at 30), number of estimators (capped at 125), and the learning rate (same search space as DeepAR).

The train-validation-test data split is 75%-15%-10%. The reason for this is forecasting is conducted in the short-to-medium term, so 10% test data is sufficient to evaluate the proposed model's performance. For programmatic platform and hardware details refer to Appendix B.

4.2.5 Metrics

The metrics are detailed in Section 2.3. They are listed here succinctly for ease of exposition. For determining the most suitable seasonality structure in the preprocessing layer, the Sum of Squared Errors (SSE) is employed. For training in the deep learning layer, Mean Absolute Error loss (MAE) with Adam efficient optimizer [158] is used. For performance evaluation of the point forecasts, the symmetric Mean Absolute Percent Error (sMAPE) [81] and Root Mean Squared Error (RMSE) are employed. The former is in line with the performance metrics from the M4 Competition, and is used for continuity as results published by Smyl [34] also use this metric. For evaluating the prediction intervals the Mean Interval Score (MIS_α) [85] is utilized, and the Coverage Score (CS_α) is used as a supplementary metric.

4.3 Results and Discussion

In this section the results for three independent experiments are detailed. The first is conducted on the Ontario COVID-19 Testing Dataset (OTD) [76], the second on the Deaths Involving COVID-19 (DIC) [76] dataset, and finally on the Our World in Data (OWID) coronavirus disease of 2019 (COVID-19) dataset [71], [72]. Both OTD and DIC are multidimensional in nature, but the problem framing converts them to a univariate forecasting landscape. This modelling approach also serves to test the generalizability of the proposed method with regards to applicability to both univariate and multivariate data. OWID is kept in the multivariate domain as is and the experimentation results presented are the the most in-depth of the three.

4.3.1 COVID-19 Positive Testing Percentage by Age Group

The testing results are separated for each of the five age groups, such that each time series of seven-day running average positive testing percentage

results can be treated as one of five univariate time series. This section reports on the forecast performance for experiments conducted for each age group.

The results for point forecasts each age group are tabulated in Table 4.2. MES-LSTM's performance is in the top two for all the age groups except sMAPE for age group 25-64. The best two performers in each instance are highlighted for ease of interpretation. Interestingly, overall, the worst forecast performance from MES-LSTM in terms of sMAPE is observed in the biggest age group range, which is not the same as the age group range presenting the worst performance in terms of RMSE. This variability in results could be a result of the age group splits, and a different categorization could impact the accuracy with respect to both metrics.

TABLE 4.2: Forecast Error Averaged Over All Trials for each Age Group.

	MES-LSTM		LSTM		LGB		DeepAR		VARMAX		SARIMAX		MLR	
Age Group	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**
0 - 13	**5,2**	**13,9**	8,5	19,3	7,4	**19,0**	10,2	22,5	9,2	27,5	9,4	36,5	**3,9**	45,7
14 - 17	**2,8**	**12,7**	7,5	19,6	6,9	**14,1**	**1,9**	23,8	7,9	26,9	6,3	32,2	9,8	23,6
18- 24	**3,9**	**15,0**	4,7	17,7	5,3	**8,7**	6,5	21,5	11,4	24,4	8,2	20,7	**0,4**	24,2
25 - 64	6,5	**10,3**	6,7	21,7	7,0	19,6	**4,7**	18,5	**5,3**	21,9	11,3	34,1	7,9	**11,9**
$\geq$ 65	**2,0**	**12,0**	7,8	18,7	6,8	14,2	4,9	22,7	**0,2**	28,4	7,9	24,5	11,2	**7,7**

In Table 4.3 the forecast results are averaged across the entirety of the tested population in Ontario. MES-LSTM is the best aggregate performer. Since RMSE gives the error in the same units as the original predictand, which in this case is percentages, ideally a forecast should only deviate from the out-of-sample ground truth by at most a percentage points in order to be considered useful for planning and management of the pandemic outbreak in question. That said, the error presented by MLR for example would render it impractical for use in such a critical, potentially life-saving application.

TABLE 4.3: Forecast Error Averaged Over All Trials and Aggregated for All Age Groups.

	MES-LSTM		LSTM		LGB		DeepAR		VARMAX		SARIMAX		MLR	
	sMAPE	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**
mean	4,1	12,8	7,0	19,4	6,7	15,1	5,6	21,8	6,8	25,8	8,6	29,6	6,7	22,6
std	1,8	1,8	1,5	1,4	0,8	4,4	3,0	2,0	4,3	2,7	1,9	6,7	4,4	14,8

Figure 4.3 plots the error realized from the aggregated results for all models for all trials. The best performers when taking into consideration both forecast error metrics are DeepAR, LGB, LSTM, and MES-LSTM.

It can be said that the data is of good quality, however one concern is because the data is presented as a seven-day rolling average time series. The inherent nature of the rolling averaging procedure means if there was an error in one day's observations within any calendar week, either through

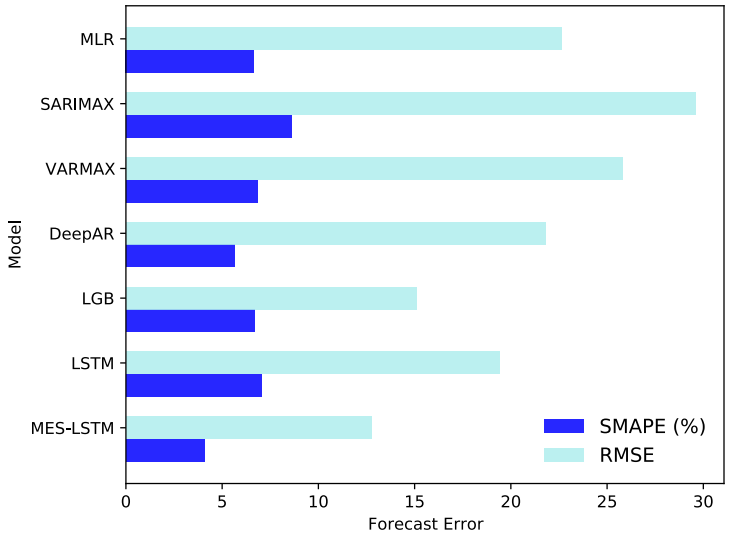


FIGURE 4.3: Forecast Accuracy Aggregated Across All Age Groups.

human error or incomplete information, then that error will propagate and impact the accuracy of the next six days' observations.

Another concern is all current COVID-19 strains do not affect children as much as teens and adults. This age-bias means there's a possibility that the smallest age group does not see significant changes to the time series data as frequently as the other age groups. This lack of variability is closer to the linear assumptions of the statistical benchmark models, i.e., VARMAX, SARIMAX and MLR. As a result, they are at times observed in the top two performers as detailed in Table 4.2.

Perhaps the biggest concern is the question of how the experimental results would be impacted if the age group splits were different. That aside, following the presented methodology, there is evidence from the results that suggests the deep learning models have better performance at this task compared to the statistical benchmarks.

4.3.2 Deaths Involving COVID-19 by Fatality Type

In a manner similar to the OTD experimentation detailed above, the three attributes of interest are treated as distinct independent univariate time series. The main difference is that DIC is composed not of seven-day rolling averages, but of totals aggregated daily.

The results for point forecasts for each fatality type are tabulated in Table 4.4. The top two performing models with respect to each error metric are

highlighted. MES-LSTM outperforms the benchmarks in less instances than the previous experimental study. In particular, the proposed model seems not to do well with fatality types where COVID-19 is confirmed as the cause or where COVID-19 is completely rules out. With the exception of VARMAX for *no covid*, the statistical benchmarks are outperformed throughout.

TABLE 4.4: Forecast Accuracy Averaged Over All Trials for each Fatality Type.

	MES-LSTM		LSTM		LGB		DeepAR		VARMAX		SARIMAX		MLR	
Fatality Type	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**
no covid	8,5	19,7	10,6	29,4	**7,1**	**16,4**	**3,5**	23,3	8,2	**19,3**	14,0	35,8	7,4	42,4
covid	10,0	**20,6**	**6,2**	29,3	**5,0**	**22,3**	7,2	26,0	11,9	33,8	17,2	32,3	8,5	29,7
covid contrib	**4,9**	**19,7**	6,8	26,8	**6,1**	**17,9**	7,7	21,8	14,1	38,3	11,8	37,0	8,4	43,2
unknown missing	**5,6**	**19,9**	11,9	27,7	6,0	**20,4**	**5,3**	23,3	7,4	29,7	10,3	21,7	9,2	36,5

Table 4.5 shows the aggregated results for all fatality types averaged across all independent trials. The tabulated results suggest that the proposed model is outperformed by LGB and DeepAR with respect to sMAPE. In terms of RMSE, LGB and MES-LSTM show comparable predictive skill.

TABLE 4.5: Forecast Accuracy Averaged Over All Trials for each Fatality Type.

	MES-LSTM		LSTM		LGB		DeepAR		VARMAX		SARIMAX		MLR	
	sMAPE	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**
mean	7,3	19,9	8,9	28,3	6,0	19,2	5,9	23,6	10,4	30,3	13,3	31,7	8,4	37,9
std	2,4	0,4	2,8	1,3	0,9	2,6	1,9	1,8	3,2	8,1	3,0	7,0	0,8	6,2

Figure 4.4 is a graphical representation of Table 4.5. The statistical benchmarks are outperformed by the deep learning models with the exception of LSTM. LGB outperforms the proposed model although there is some evidence of comparative predictive skill.

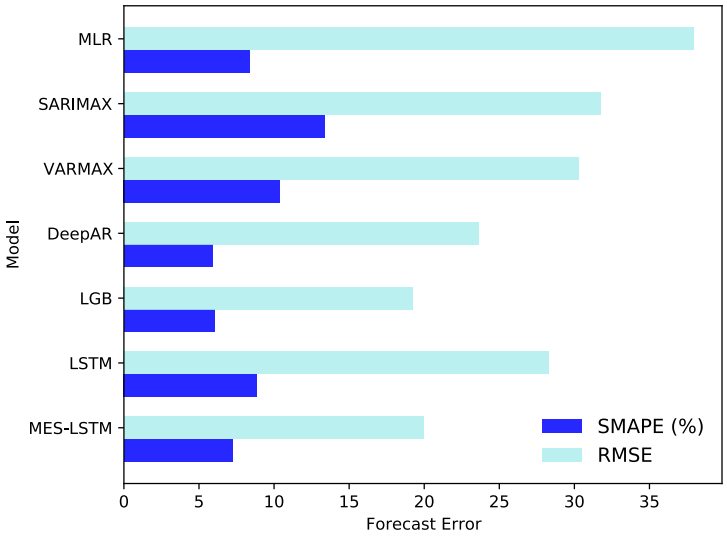


FIGURE 4.4: Forecast Accuracy Aggregated Across All Fatality Types.

Distinguishing the distinct causes of death during a pandemic is important for planning and management. One concern with the current study is that the data may change in future. One example is if a test shows *no covid* and is recorded as such, but some time later some suspicion necessitates a new test. If the new test show *covid* the data will be corrected to indicate the new result on the original tabulated date. These kind of changes, if they occur at scale, can impact the results of said experimental study.

4.3.3 COVID-19 Morbidity and Mortality within a Localized Context

The previous two studies are suggestive that the proposed model has some predictive skill, which warrants further in-depth probing, investigation, and analysis. This section details the results for multivariate point forecasts as well as the prediction intervals for the predictands *total cases* and *total deaths* for OWID. In order to mitigate the stochastic nature of some deep learning models, the experiments are repeated 35 times. The results presented are the aggregates of all the repeated independent trials.

The variables that are omitted from this study are duplicates where numerical data has been smoothed, e.g. *new cases smoothed*. The smoothed variables are removed for two reasons: OWID offers no insights on how the data is smoothed, and it is prudent to avoid any duplication as the proposed model also conducts preprocessing.

In particular, the proposed model is used to forecast and quantify the associated prediction uncertainty for the two variables of interest. The predictions and uncertainty quantification can be conducted for a single country, or easily across a multitude of countries or even averaged over a specific region. Regional or multi-country inferencing can assist policymakers to make tough decisions such as, for example, restricting inter-provincial/state travel or closing national borders completely.

Below, results for the analysis conducted for the Southern African Development Community (SADC) are presented. SADC is a regional economic community comprising 16 member states: Angola, Botswana, Comoros, Democratic Republic of Congo (DRC), Eswatini, Lesotho, Madagascar, Malawi, Mauritius, Mozambique, Namibia, Seychelles, South Africa, Tanzania, Zambia and Zimbabwe.

Forecast Performance for SADC

The results for point forecasts for SADC are tabulated in Tables 4.6 and 4.7, for each of the predictands. MES-LSTM outperforms the benchmarks for all the nations in SADC, except sMAPE for *total cases* in South Africa. The best performer in each instance is highlighted for ease of interpretation. Interestingly, overall, the worst forecast performance from MES-LSTM is in South Africa and this could be a result of the nation presenting the most accurate data. The other nations possibly do not update their data as frequently, and the model learns easier as there is not a lot of variability. Furthermore, the (possibly less accurate) data from the other nations is closer to the linear assumptions of the statistical benchmark models, i.e., VARMAX, SARIMAX and MLR.

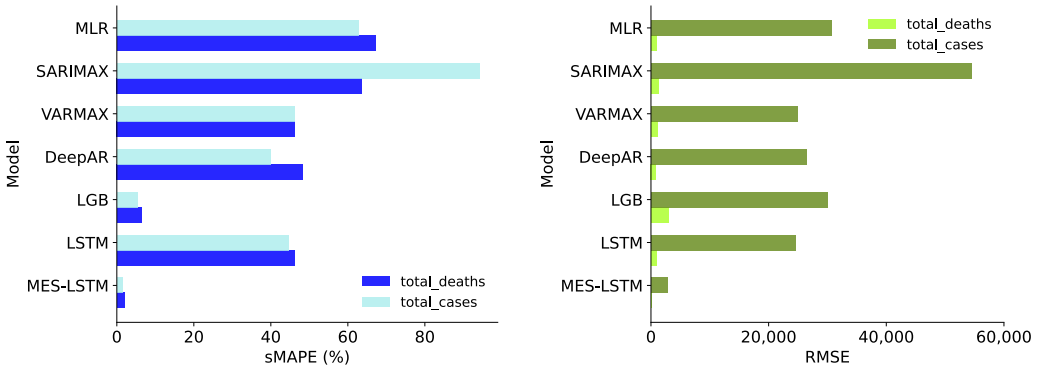
TABLE 4.6: Forecast Accuracy Averaged Over All Trials for *total cases* in the SADC Region.

	MES-LSTM		LSTM		LGB		DeepAR		VARMAX		SARIMAX		MLR	
Country	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**
Angola	**0.7**	**563.1**	68.9	28,023.4	5.6	28,524.5	59.0	23,811.6	76.4	35,442.3	107.1	59,732.6	77.6	35,830.6
Botswana	**1.6**	**5,817.7**	85.5	94,414.1	5.8	32,635.2	62.9	72,983.0	99.5	125,387.6	71.0	123,392.6	114.6	137,334.7
Comoros	**1.1**	**52.5**	16.5	623.3	5.3	32,005.1	50.1	1,443.4	6.2	313.8	23.6	1,058.6	17.6	704.2
DRC	**0.8**	**468.3**	72.6	25,987.3	5.3	29,933.9	41.9	17,302.5	12.8	7,047.8	27.6	15,527.6	83.9	34,102.8
Eswatini	**1.3**	**617.7**	59.1	18,348.9	5.7	31,572.0	61.2	17,619.4	63.6	23,417.0	37.9	16,687.6	110.6	33,055.4
Lesotho	**0.7**	**167.2**	47.2	7,324.4	5.5	28,662.4	33.9	5,484.3	35.2	6,478.7	137.8	24,942.5	42.4	7,603.3
Madagascar	**1.3**	**636.6**	35.1	12,019.8	5.5	29,387.4	45.6	13,719.5	11.1	5,495.1	8.8	3,889.1	22.3	9,014.0
Malawi	**1.3**	**817.5**	22.0	11,967.0	5.9	30,163.2	14.1	8,183.4	10.4	7,629.0	9.7	5,956.9	42.2	23,504.7
Mauritius	**3.8**	**2,056.8**	99.8	9,891.6	5.8	29,677.4	91.1	8,643.8	179.5	17,135.7	242.9	32,875.6	171.3	16,742.4
Mozambique	**1.1**	**1,685.0**	5.0	7,457.6	5.2	29,234.1	3.0	4,438.9	2.5	3,946.2	90.5	125,768.7	10.3	15,089.2
Namibia	**1.1**	**1,508.8**	5.2	7,516.3	5.6	26,149.9	2.3	2,969.0	12.0	17,579.1	25.3	32,551.7	7.7	10,010.1
Seychelles	**0.9**	**266.7**	59.9	8,908.6	5.4	27,596.0	62.1	8,596.8	11.4	2,428.1	47.4	10,115.2	99.0	14,797.0
South Africa	5.0	**26,979.2**	4.8	150,416.4	5.6	30,420.9	7.8	212,612.0	2.8	87,376.0	8.2	265,998.0	**4.0**	118,139.5
Tanzania	**0.3**	**134.6**	116.3	15,445.1	6.0	30,900.7	96.5	12,839.8	184.1	25,060.9	636.9	87,188.8	194.9	25,818.8
Zambia	**1.2**	**2,499.0**	6.4	15,567.7	5.8	28,686.9	2.0	4,333.4	7.3	16,766.9	72.1	143,490.3	2.1	6,120.9
Zimbabwe	**1.1**	**1,518.5**	8.6	10,652.2	5.4	32,284.2	5.7	7,251.2	8.4	12,296.7	6.4	9,733.6	2.6	4,164.0

TABLE 4.7: Forecast Accuracy Averaged Over All Trials for *total deaths* in the SADC Region.

	MES-LSTM		LSTM		LGB		DeepAR		VARMAX		SARIMAX		MLR	
Country	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**	**sMAPE**	**RMSE**
Angola	**0.9**	**19.8**	74.6	783.5	6.6	2,886.7	60.8	687.0	89.9	1,055.7	71.6	1,038.1	89.4	1,052.1
Botswana	**1.0**	**26.1**	86.1	1,203.9	6.7	2,909.9	77.2	1,128.8	97.6	1,576.5	15.7	363.9	120.8	1,806.0
Comoros	**1.5**	**2.4**	12.5	16.7	6.2	3,069.4	72.4	66.8	15.6	27.3	13.5	20.6	10.9	16.0
DRC	**1.0**	**12.0**	67.2	470.1	6.1	3,059.6	38.2	326.4	2.4	34.2	11.6	128.9	74.0	594.3
Eswatini	**1.1**	**14.2**	49.6	435.1	7.0	2,594.2	58.8	490.0	46.3	525.6	35.3	403.1	95.1	801.0
Lesotho	**0.8**	**5.4**	47.2	222.9	6.9	2,790.2	42.4	207.1	46.8	249.7	97.4	516.2	44.5	240.8
Madagascar	**1.2**	**12.2**	48.1	330.3	6.6	2,820.6	71.5	431.7	4.2	44.6	2.0	20.4	41.7	335.9
Malawi	**1.2**	**28.7**	27.1	518.9	6.6	3,072.6	20.4	416.1	14.5	349.6	14.2	314.9	46.2	958.7
Mauritius	**1.7**	**39.9**	103.0	109.6	6.9	3,014.0	104.4	110.3	173.1	183.1	300.2	341.7	172.7	183.1
Mozambique	**1.1**	**21.9**	5.1	96.7	6.4	3,269.7	4.5	85.6	4.9	98.7	9.2	180.1	15.6	281.4
Namibia	**1.1**	**41.9**	7.9	333.8	6.1	3,200.6	4.9	174.9	19.6	848.9	17.2	601.2	29.3	913.9
Seychelles	**1.7**	**7.0**	71.9	54.0	7.1	3,288.2	82.4	58.9	5.9	8.3	43.5	47.7	116.9	88.8
South Africa	**4.9**	**446.2**	7.8	7,902.4	5.9	2,773.4	10.4	8,515.8	10.7	10,806.6	16.1	15,438.9	6.5	6,276.2
Tanzania	**0.3**	**3.8**	114.7	425.6	6.7	3,143.2	113.7	423.5	179.3	685.9	362.3	1,379.1	193.0	712.9
Zambia	**1.3**	**47.7**	6.6	266.8	6.6	2,806.5	4.3	174.2	7.5	301.4	1.9	87.8	11.7	420.1
Zimbabwe	**1.1**	**53.4**	6.5	298.2	6.3	3,067.7	8.1	361.0	14.8	791.5	5.8	289.7	6.7	345.0

In Figure 4.5 the forecast results are averaged across the entirety of the SADC region. MES-LSTM is the best aggregate performer. This figure also illustrates the importance of choosing multiple error metrics.



(A) (B)

FIGURE 4.5: Forecast Accuracy in the SADC Region. (**A**) sMAPE. (**B**) RMSE.

In Figure 4.5A SARIMAX for instance, seems to perform fairly poorly for both predictands, but this as a stand-alone interpretation would be misleading. From Figure 4.5B it is evident that in terms of regional skill for *total deaths*, SARIMAX is actually competitive. Recall from Section 2.3 that RMSE gives the error in the same units as the original predictand. South Africa has a total population of 65 million, so any model that over- or under-forecasts *total cases* by a few thousand cases could still be useful for planning and management of the pandemic outbreak in question.

Prediction Interval Performance for SADC

The results for the prediction interval are documented in Tables 4.8 and 4.9. The best performer or best tied performers are highlighted. MES-LSTM MIS_α shows consistent improvement on performance over the other models for both predictands for almost the entire SADC region.

TABLE 4.8: Prediction Interval Accuracy Averaged Over All Trials for *total cases* in SADC ($\alpha = 0.05$).

	MES-LSTM		LSTM		LGB		DeepAR		VARMAX		SARIMAX		MLR	
Country	MIS_α	CS_α	MIS_α	CS_α	MIS_α	CS_α	MIS_α	CS_α	MIS_α	CS_α	MIS_α	CS_α	MIS_α	CS_α
Angola	**1,910.8**	**95.2**	116,739.1	0.0	55,663.7	81.6	54,875.9	0.0	902,714.3	0.0	814,594.0	0.0	660,826.0	0.0
Botswana	**11,735.6**	**85.2**	378,936.2	0.0	164,794.4	77.6	171,350.6	0.0	4,572,270.6	0.0	1,203,716.2	0.0	4,259,387.7	0.0
Comoros	**172.9**	97.4	360.0	87.6	3,908.4	84.4	5,064.1	0.0	3,085.7	40.9	1,790.1	59.1	5,461.2	**100.0**
DRC	**2,351.8**	**97.6**	102,410.9	0.0	45,538.0	65.2	68,391.5	0.0	143,127.3	0.0	137,288.5	0.0	413,486.5	0.0
Eswatini	**1,837.3**	**91.0**	86,314.2	0.0	44,987.2	79.6	44,252.9	0.0	518,283.5	14.3	140,160.1	0.0	683,626.1	0.0
Lesotho	**750.4**	**97.4**	24,269.2	0.4	20,065.3	89.0	18,693.3	0.0	218,810.3	0.0	310,359.2	0.0	24,172.3	55.8
Madagascar	**2,310.7**	83.6	25,720.5	16.9	30,953.7	83.1	19,934.0	56.3	139,138.3	0.0	5,270.5	**100.0**	48,987.3	**100.0**
Malawi	**2,503.3**	88.0	27,408.2	38.8	40,635.9	78.9	31,982.9	0.0	60,588.0	70.2	33,628.7	**100.0**	303,117.6	55.3
Mauritius	**2,460.2**	**90.1**	44,971.4	0.0	21,957.8	77.6	43,120.5	0.0	625,931.1	0.0	441,038.4	0.0	592,480.4	0.0
Mozambique	**5,249.2**	94.2	17,924.2	92.1	68,002.5	86.0	3,197.7	46.2	23,040.1	**100.0**	1,364,340.7	0.0	83,378.3	22.9
Namibia	**4,487.2**	**93.1**	18,930.5	93.9	53,947.4	79.7	1,726.9	59.2	48,9847.0	0.0	28,333.2	38.8	45,068.1	67.3
Seychelles	**728.2**	**93.1**	32,422.8	0.0	20,364.2	73.7	17,835.9	14.3	68,568.1	0.0	103,106.7	0.0	232,264.3	0.0
South Africa	**193,075.6**	80.7	348,860.7	97.7	1,418,713.2	86.6	239,009.3	**100.0**	740,131.9	**100.0**	1,061,162.0	**100.0**	828,447.2	**100.0**
Tanzania	**943.6**	**96.2**	67,427.5	0.0	32,459.5	77.3	67,350.0	0.0	990,720.2	0.0	1,279,781.1	0.0	1,018,128.2	0.0
Zambia	**8,972.0**	**89.2**	24,151.6	96.4	89,095.3	73.5	8,316.6	16.6	489,501.1	0.0	1,509,742.0	0.0	36,167.6	95.8
Zimbabwe	**4,770.4**	87.7	23,667.6	85.8	63,299.2	77.9	2,987.9	**100.0**	367,165.9	0.0	14,904.8	95.8	16,359.3	**100.0**

TABLE 4.9: Prediction Interval Accuracy Averaged Over All Trials for *total deaths* in SADC ($\alpha = 0.05$).

	MES-LSTM		LSTM		LGB		DeepAR		VARMAX		SARIMAX		MLR	
Country	MIS_α	CS_α	MIS_α	CS_α	MIS_α	CS_α	MIS_α	CS_α	MIS_α	CS_α	MIS_α	CS_α	MIS_α	CS_α
Angola	**53.2**	**94.0**	3,206.0	0.0	3,348.1	77.1	2,089.5	0.0	29,346.8	0.0	24,022.4	0.0	23,844.1	0.0
Botswana	**106.4**	**93.3**	4,803.4	0.0	4,357.0	75.1	1,706.3	10.6	53,087.1	0.0	1,079.0	48.9	51,432.7	0.0
Comoros	**5.9**	95.3	12.5	86.3	318.6	78.3	189.9	0.0	575.5	0.0	34.5	**100.0**	225.3	**100.0**
DRC	**46.1**	95.7	1,725.4	0.0	1,636.2	84.0	1,013.0	20.4	163.4	**100.0**	336.5	**100.0**	4,297.0	20.4
Eswatini	**49.0**	**90.7**	2,236.0	0.0	2,350.4	90.2	671.6	0.0	8,795.5	42.9	7,693.3	0.0	10,342.2	0.0
Lesotho	**22.8**	**97.6**	795.3	0.3	1,302.8	85.2	677.5	0.0	8,941.9	0.0	11,845.1	0.0	1408.6	23.3
Madagascar	**46.5**	85.4	733.9	11.8	1,787.3	77.7	894.2	2.1	244.4	56.3	108.1	**100.0**	1,108.4	**100.0**
Malawi	**89.4**	88.8	1,223.5	29.5	3,526.6	85.7	1,202.7	0.0	3,309.8	61.7	1,153.1	**100.0**	15,170.2	31.9
Mauritius	**75.8**	**74.0**	468.6	0.0	484.5	79.9	454.8	0.0	6,978.1	0.0	8,379.1	0.0	6,861.6	0.0
Mozambique	**66.8**	94.2	244.7	91.2	1,928.7	75.0	70.9	42.1	341.8	**100.0**	602.0	22.9	2,707.0	22.9
Namibia	134.0	91.3	587.2	90.8	3,411.9	81.7	**101.4**	89.7	21,239.2	0.0	23,742.4	0.0	10,490.6	0.0
Seychelles	**5.8**	**91.0**	188.7	0.0	258.6	76.1	143.5	0.0	33.4	77.6	694.2	0.0	1,984.9	0.0
South Africa	**5,785.9**	79.8	10,713.1	97.4	97,311.6	84.7	9,589.2	17.6	63,626.1	48.1	33,910.2	**100.0**	26,489.6	**100.0**
Tanzania	**28.4**	**94.7**	1,853.4	0.0	1,911.1	85.9	1,844.1	0.0	27,048.4	0.0	35,267.8	0.0	28,062.0	0.0
Zambia	**150.0**	89.7	429.3	94.9	3,701.5	77.6	429.3	19.1	6,940.6	6.3	458.3	**100.0**	1,731.8	77.1
Zimbabwe	169.9	87.3	810.4	85.8	5,004.7	84.6	**119.7**	**100.0**	26,898.9	0.0	746.0	95.8	4,778.9	20.8

Recall from Section 2.3 that the MIS_α penalizes large intervals, so the proposed method generally has the narrowest prediction intervals. In Figure 4.6 the results for prediction intervals are averaged across the entirety of the SADC region. MES-LSTM consistently has the narrowest aggregate intervals for both predictands at all prediction intervals, as evidenced in Figure 4.6A–C.

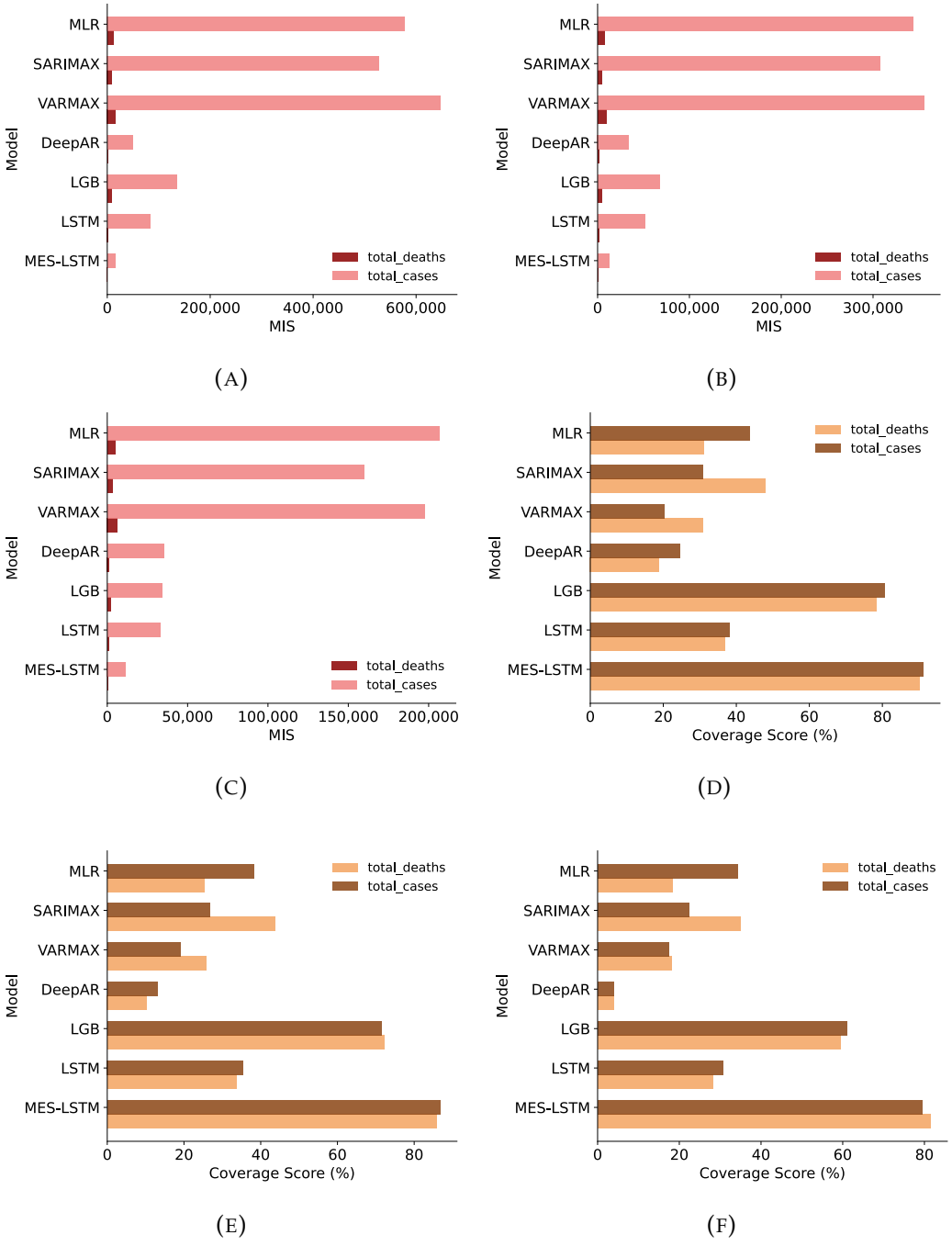


FIGURE 4.6: Prediction Interval Accuracy in the SADC Region. (**A**) MIS_α ($\alpha = 0.05$). (**B**) MIS_α ($\alpha = 0.1$). (**C**) MIS_α ($\alpha = 0.2$). (**D**) CS_α ($\alpha = 0.05$). (**E**) CS_α ($\alpha = 0.1$). (**F**) CS_α ($\alpha = 0.2$).

In terms of Coverage, the proposed model outperforms the benchmarks for the individual countries except in a few instances. In most of the exceptions, the difference is minuscule, whereas in other instances (e.g. South Africa) the difference is admittedly not negligible. What is more important however, is the aggregate performance over the entire region, where MES-LSTM scores better (as can be seen in Figure 4.6D–F).

Forecast Performance for South Africa

After noting the relatively poor performance for South Africa, there is an incentive for further investigation.

The observed error metrics distribution over the independent trials for all the models are presented for both predictands in South Africa. Tables 4.10 and 4.11 show the forecasting error for *total cases* and *total deaths* respectively. The models VARMAX, SARIMAX and MLR are all deterministic, so the output is identical for different trials. As a result, there is no variation in the accuracy of these models and the standard deviation is zero. For *total cases*, VARMAX (DeepAR) has the lowest (highest) average sMAPE, and MES-LSTM (SARIMAX) has the lowest (highest) average RMSE. For *total deaths*, MES-LSTM has the lowest average sMAPE and RMSE, while SARIMAX has the highest figures for both.

TABLE 4.10: Forecast Accuracy Distribution for All Trials for *total cases* in South Africa.

	MES-LSTM		LSTM		LGB		DeepAR		VARMAX		SARIMAX		MLR	
	sMAPE	RMSE	sMAPE	RMSE	sMAPE	RMSE	sMAPE	RMSE	sMAPE	RMSE	sMAPE	RMSE	sMAPE	RMSE
mean	5.0	26,979.2	4.8	150,416.4	5.6	30,420.9	7.8	212,612.0	2.8	87,376.0	8.2	265,998.0	4.0	118,139.5
std	1.1	5,641.0	3.8	121,306.0	1.3	12,885.4	0.1	1,192.2	0.0	0.0	0.0	0.0	0.0	0.0

TABLE 4.11: Forecast Accuracy Distribution for All Trials for *total deaths* in South Africa.

	MES-LSTM		LSTM		LGB		DeepAR		VARMAX		SARIMAX		MLR	
	sMAPE	RMSE	sMAPE	RMSE	sMAPE	RMSE	sMAPE	RMSE	sMAPE	RMSE	sMAPE	RMSE	sMAPE	RMSE
mean	4.9	446.2	7.8	7,902.4	5.9	2,773.4	10.4	8,515.8	10.7	10,806.6	16.1	15,438.9	6.5	6,276.2
std	1.2	106.0	5.3	5,578.8	2.2	1,259.7	0.1	61.2	0.0	0.0	0.0	0.0	0.0	0.0

MES-LSTM and SARIMAX results generally being on opposite sides of the spectrum is also evidenced by the box and whisker plots in Figure 4.7.

The results from the LSTM are far more variable for each trial than MES-LSTM. LSTM seems competitive to the proposed model for some trials but shows little consistency for multiple independent runs. The LSTM's relatively poor and oft-inconsistent performance with COVID-19 modelling agrees with observations by other researchers (as detailed in Section 4.1.1.)

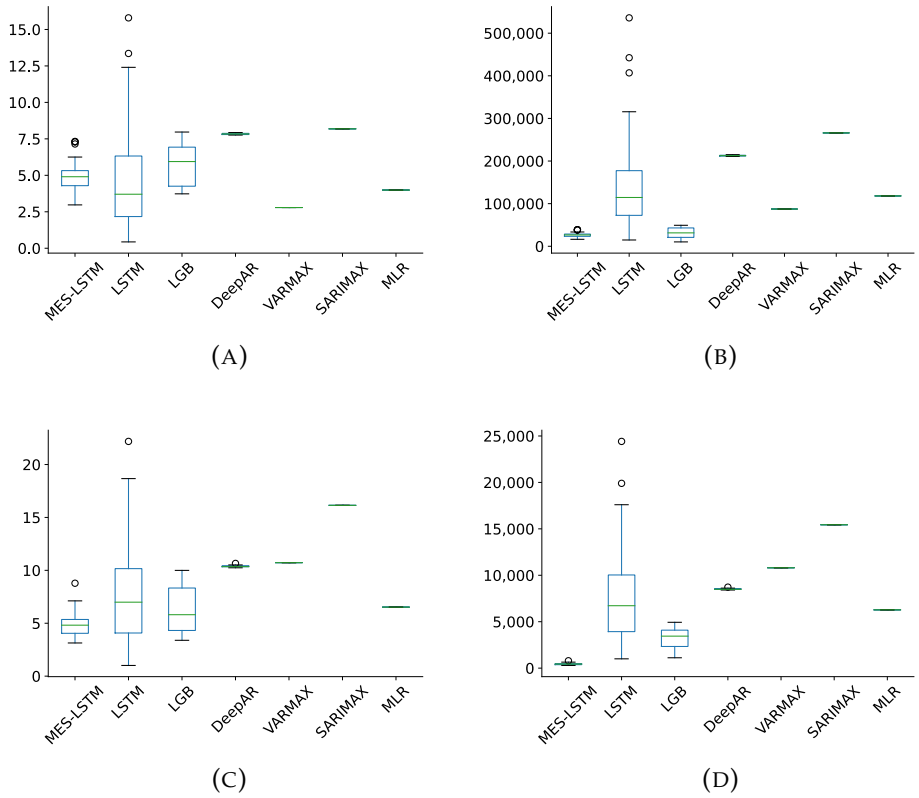


FIGURE 4.7: Forecast Accuracy Distribution Boxplots for All Trials in South Africa. (**A**) sMAPE for *total cases*. (**B**) RMSE for *total cases*. (**C**) sMAPE for *total deaths*. (**D**) RMSE for *total deaths*.

When viewed together with the bar graphs presented in Figure 4.8, one possible interpretation of all the forecast results is a ranking of the best performance for South Africa in increasing order as SARIMAX, VARMAX, LSTM, DeepAR, MLR, LGB, and MES-LSTM.

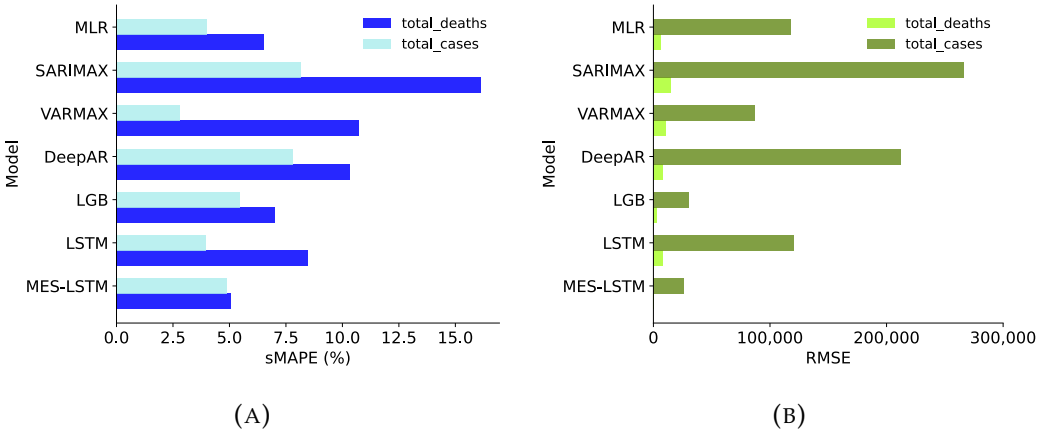


FIGURE 4.8: Forecast Accuracy in South Africa. (**A**) sMAPE. (**B**) RMSE.

MES-LSTM shows consistent outperformance over (or at least competitiveness to) the benchmarks. The proposed model also presents results with a tight distribution, second in terms of tightness to DeepAR (from the probabilistic forecasting models).Even in instances where outliers are present in the forecast distribution of the proposed model, these outliers are still very close to the core distribution. This tendency to produce a tight distribution reaffirms the robustness of the proposed model when it comes to producing consistently accurate forecasts.

Prediction Interval Performance for South Africa

Tables 4.12 and 4.13 show the prediction interval summary statistics at the $\alpha = 0.05$ level of significance for the respective predictands in South Africa. The distribution for MES-LSTM is tight and this characteristic persists throughout all significance levels.

TABLE 4.12: Prediction Interval Accuracy Distribution for All Trials for *total cases* in South Africa ($\alpha = 0.05$).

	MES-LSTM		**LSTM**		**LGB**		**DeepAR**		**VARMAX**		**SARIMAX**		**MLR**	
	MIS_α	**CS_α**	**MIS_α**	**CS_α**	**MIS_α**	**CS_α**	**MIS_α**	**CS_α**	**MIS_α**	**CS_α**	**MIS_α**	**CS_α**	**MIS_α**	**CS_α**
mean	193,075.6	80.7	348,860.7	97.7	1,418,713.2	86.6	239,009.3	100.0	740,131.9	100.0	1,061,162.0	100.0	828,447.2	100.0
std	13,881.3	11.2	76,242.6	13.7	0.0	30.2	1,674.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0

TABLE 4.13: Prediction Interval Accuracy Distribution for All Trials for *total deaths* in South Africa ($\alpha = 0.05$).

	MES-LSTM		**LSTM**		**LGB**		**DeepAR**		**VARMAX**		**SARIMAX**		**MLR**	
	MIS_α	**CS_α**	**MIS_α**	**CS_α**	**MIS_α**	**CS_α**	**MIS_α**	**CS_α**	**MIS_α**	**CS_α**	**MIS_α**	**CS_α**	**MIS_α**	**CS_α**
mean	5,785.9	79.8	10,713.1	97.4	97,311.6	84.7	9,589.2	17.6	63,626.1	48.1	33,910.2	100.0	26,489.6	100.0
std	1,714.1	15.1	3,448.0	15.6	0.0	28.5	723.9	6.5	0.0	0.0	0.0	0.0	0.0	0.0

The bar graphs in Figure 4.9 indicate that even though the proposed model's Coverage score is outperformed in some instances, it is not by a great margin. Furthermore, MES-LSTM has the benefit of the lowest MIS_α, indicating the tightest prediction intervals. The second benefit is consistency, i.e., where other models may only have decent Coverage for one predictand, the proposed model has a consistent level of Coverage across all the predictands. Moreover, there's a dependence on the significance level, with stricter levels leading to marginally higher Coverage. This Coverage-alpha dependence from MES-LSTM is important as it also speaks to the robustness of the proposed model. In contrast, DeepAR for instance, goes from almost complete Coverage ($\alpha = 0.05$, $\alpha = 0.1$) for *total cases* to none ($\alpha = 0.02$) and MLR

presents close to perfect Coverage for *total cases* throughout. This vast difference across levels does not instill much trust in the statistical method's uncertainty quantification.

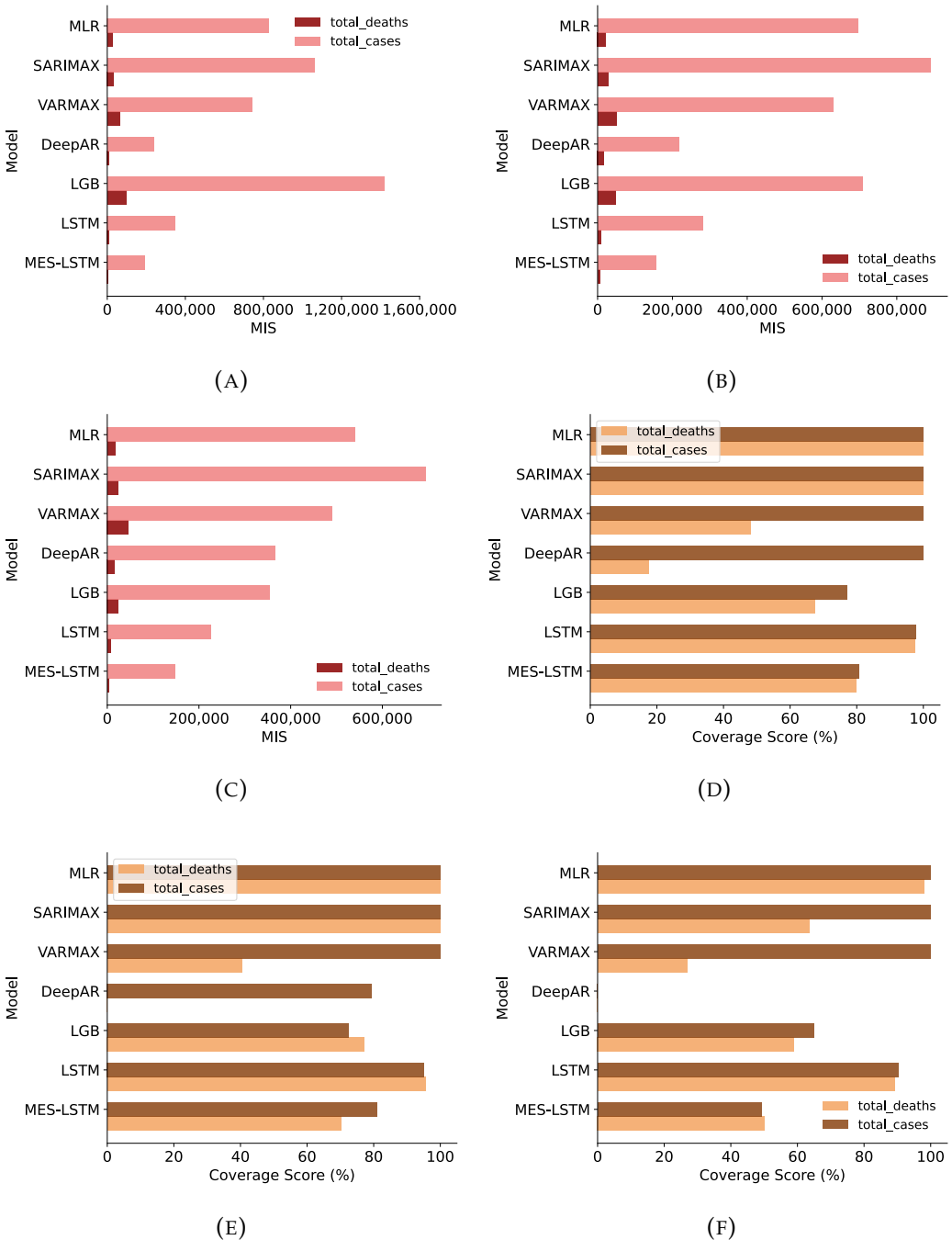


FIGURE 4.9: Prediction Interval Accuracy in South Africa. (**A**) MIS_α ($\alpha = 0.05$). (**B**) MIS_α ($\alpha = 0.1$). (**C**) MIS_α ($\alpha = 0.2$). (**D**) CS_α ($\alpha = 0.05$). (**E**) CS_α ($\alpha = 0.1$). (**F**) CS_α ($\alpha = 0.2$).

Overall, all models perform worse for the SADC region than for South Africa. The worse performance is due to more variability introduced in the

data and thus a higher level of uncertainty. This variability can be seen when comparing the Coverage scores from Figure 4.9 to those presented in Figure 4.6.

As an additional point, a one-sided t-test is performed to check whether the distribution of the proposed model's forecast and prediction interval results are significantly better than those produced by the benchmark models. Concisely, MES-LSTM is compared to each of the other models in turn. The t-test results are presented in Tables 4.14–4.17 truncated to three decimal places. The null hypothesis is H_0: The benchmark models produce forecasts that are more accurate than and prediction intervals superior to those produced by MES-LSTM.

TABLE 4.14: Student's t-test for Forecast RMSE (H_0: Benchmark Forecasts Outperform MES-LSTM).

	Total Cases						Total Deaths					
	LSTM	**LGB**	**DeepAR**	**VARMAX**	**SARIMAX**	**MLR**	**LSTM**	**LGB**	**DeepAR**	**VARMAX**	**SARIMAX**	**MLR**
statistic	−6.014	**−1.448**	−190.477	−63.342	−250.673	−95.605	−7.906	−10.891	−389.954	−578.058	−836.515	−325.283
p-value	0.000	**0.077**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

TABLE 4.15: Student's t-test for Forecast sMAPE (H_0: Benchmark Forecasts Outperform MES-LSTM).

	Total Cases						Total Deaths					
	LSTM	**LGB**	**DeepAR**	**VARMAX**	**SARIMAX**	**MLR**	**LSTM**	**LGB**	**DeepAR**	**VARMAX**	**SARIMAX**	**MLR**
statistic	**0.270**	**−1.883**	−15.455	**12.180**	−17.383	**5.616**	−3.216	−2.422	−27.328	−29.110	−56.140	−8.166
p-value	**0.606**	**0.032**	0.000	**1.000**	0.000	**1.000**	0.001	0.009	0.000	0.000	0.000	0.000

TABLE 4.16: Student's t-test for Prediction Interval MIS_α (H_0: Benchmark Prediction Intervals Outperform MES-LSTM).

	Total Cases						Total Deaths					
	LSTM	**LGB**	**DeepAR**	**VARMAX**	**SARIMAX**	**MLR**	**LSTM**	**LGB**	**DeepAR**	**VARMAX**	**SARIMAX**	**MLR**
statistic	−5.328	−25.791	−2.851	−22.095	−34.220	−25.119	−2.299	−37.924	−10.070	−40.222	−20.029	−14.501
p-value	0.000	0.000	0.004	0.000	0.000	0.000	0.013	0.000	0.000	0.000	0.000	0.000

TABLE 4.17: Student's t-test for Prediction Interval Coverage (H_0: Benchmark Prediction Intervals Outperform MES-LSTM).

	Total Cases						Total Deaths					
	LSTM	**LGB**	**DeepAR**	**VARMAX**	**SARIMAX**	**MLR**	**LSTM**	**LGB**	**DeepAR**	**VARMAX**	**SARIMAX**	**MLR**
statistic	**−1.8770**	**1.6106**	**0.2267**	**−2.7459**	**−2.7459**	**−2.7459**	**−2.8639**	**−0.2274**	8.3710	3.5521	**−3.5613**	**−3.5613**
p-value	**0.9666**	**0.0567**	**0.4110**	**0.9952**	**0.9952**	**0.9952**	**0.9968**	**0.5895**	0.0000	0.0006	**0.9994**	**0.9994**

In terms of forecasting accuracy (Tables 4.14 and 4.15) in each instance, the null hypothesis w.r.t. both predictands is rejected at the $\alpha = 0.01$ level of significance, except for LSTM, LGB, VARMAX, and MLR. This result may seem contrary to previously presented results in this chapter, but the box plots in Figure 4.7 explain this. Although the core and lower distributions

may overlap (the box and bottom whisker) for the distributions in question compared to MES-LSTM, the higher levels of inaccuracy are only reported for the benchmarks, i.e., the upper whiskers for the benchmarks peak higher than MES-LSTM.

For the prediction interval performance (Tables 4.16 and 4.17), one is able to reject the null hypothesis at the $\alpha = 0.01$ level of significance for MIS_α in all instances. Finally, one is unable to conclude that the presented model's Coverage Score is not outperformed except perhaps by DeepAR and VARMAX for *total deaths*. Again, the t-test results are consistent with the results previously discussed.

A Diebold-Mariano (DM) [89] test is conducted to compare the out-of-sample predictive skill of MES-LSTM to the benchmark models (Table 4.18). The test is configured to use Mean Absolute prediction Error (MAPE) [90] and the null hypothesis is H_0: There is no significant difference in the accuracy of the competing forecasts. The null hypothesis is rejected at the $\alpha = 0.01$ level of significance w.r.t. both predictands.

TABLE 4.18: Diebold-Mariano Test for Forecast Accuracy (H_0: Benchmark Forecasts Outperform MES-LSTM).

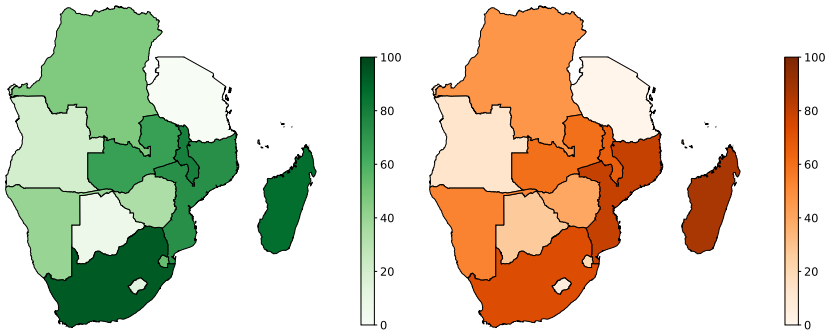
	Total Cases						Total Deaths					
	LSTM	**LGB**	**DeepAR**	**VARMAX**	**SARIMAX**	**MLR**	**LSTM**	**LGB**	**DeepAR**	**VARMAX**	**SARIMAX**	**MLR**
statistic	−19.817	−277.776	−144.279	−12.665	−16.160	−8.048	−5.567	−374.711	−112.500	−7.066	−11.817	−43.647
p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

This chapter is concluded with a look into the effects of introducing more variability on the proposed model's performance.

Effects of Variability on Model Performance

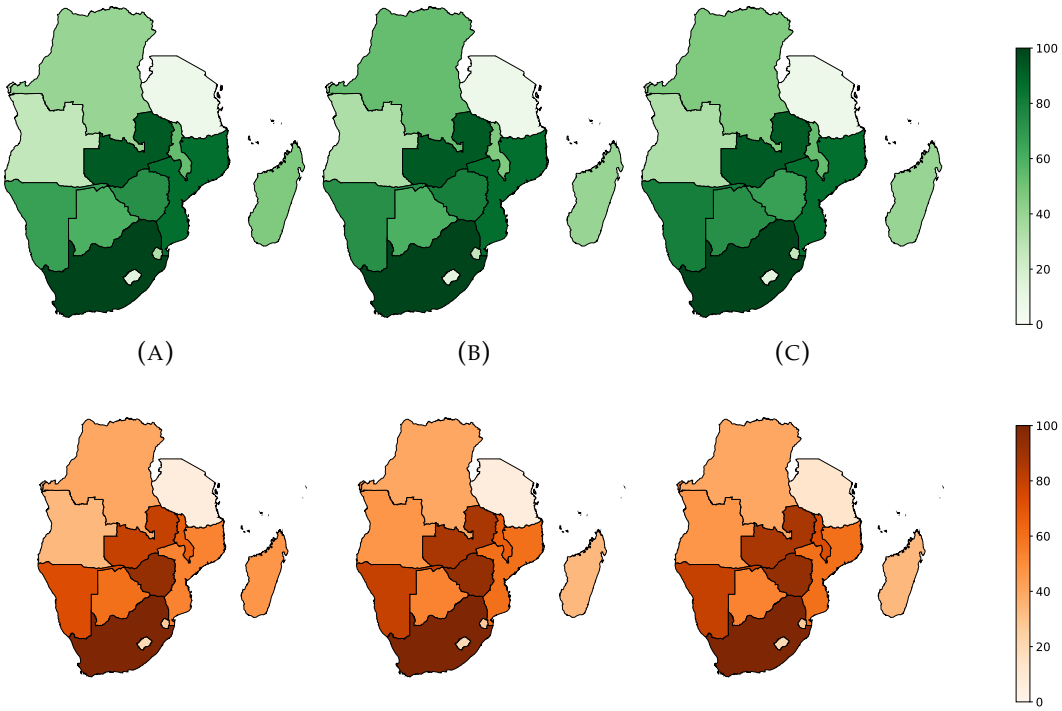
The methodology presented suggests that the introduction of more variability into the data impacts both forecasting accuracy and prediction interval construction. The focus here is specifically on MES-LSTM. Figures 4.10 and 4.11 indicate that in most instances, the countries in which the predictions are least (most) accurate are also the countries in which the prediction intervals are widest (narrowest). This direct correlation reinforces the consistency of the proposed model. The MIS_α in Figure 4.11 have been normalized at each significance level before plotting the maps for ease of interpretation.

In terms of Coverage, MES-LSTM is in instances outperformed by the benchmark methods. These instances of poor Coverage are an area where the model can be improved. However, there are some crucial areas where the model improves on the skill of its counterparts. The methodology presented



(A) (B)

FIGURE 4.10: MES-LSTM Forecast Accuracy Ranked for Each Country in the SADC Region. (**A**) sMAPE for *total cases*. (**B**) sMAPE for *total deaths*.



(A) (B) (C)

(D) (E) (F)

FIGURE 4.11: MES-LSTM Prediction Interval Accuracy (Normalized MIS_α) Ranked for Each Country in the SADC Region. (**A**) *total cases* ($\alpha = 0.05$). (**B**) *total cases* ($\alpha = 0.1$). (**C**) *total cases* ($\alpha = 0.2$). (**D**) *total deaths* ($\alpha = 0.05$). (**E**) *total deaths* ($\alpha = 0.1$). (**F**) *total deaths* ($\alpha = 0.2$).

suggests that the hybrid MES-LSTM is indeed able to outperform statistical methods and deep learning techniques at both forecasting tasks and prediction interval construction for morbidity and mortality data with exogenous factors. In terms of the overall prediction error, there is a significant improvement over the benchmark models considered. The proposed model reports forecasts consistently within a tight range for multiple independent trials. MES-LSTM also offers the narrowest prediction intervals for all the predictands for all geographical regions at all the significance levels considered.

4.4 Conclusion

An efficient generalized multivariate extension of a hybrid model is introduced, for multivariate prediction and forecast uncertainty quantification and it is applied to morbidity and mortality data with exogenous factors. The univariate counterpart, Smyl's ES-RNN [34], has been shown to perform well in the univariate case, outperforming both pure machine learning and pure statistical methods. The hypothesis in this chapter is that the presented multivariate extension also outperforms statistical and machine learning models at both forecasting tasks and constructing prediction intervals. With the methodology presented and the data considered, MES-LSTM significantly improves the skill of its classical probabilistic and pure deep learning counterparts.

MES-LSTM shows consistent outperformance with forecast accuracy and the MIS_α of the prediction intervals constructed at all significance levels considered. There remains room for improvement when it comes to the Coverage of the prediction intervals.

The major contribution of this chapter is model extension and application to the multivariate setting. Although attention is mostly limited to the multivariate setting (except in Section 4.2.1 where the univariate exposition is used as a building block before introducing the proposed model), careful framing of the problem statement in the first two experimental studies presented enabled inclusion of running and reporting on extensive MES-LSTM experimentation in the univariate setting as well. The benchmarks were applied since both SARIMAX and VARMAX can easily be modified to model univariate data if no exogenous inputs are declared, MLR is fit using OLS, and gradient boosting techniques can also be applied to univariate data.

Applying the kind of techniques discussed in this chapter to varied data with expedience is still a major limitation in related research. For example,

the M5 Competitions although arguably among the most important forecast competitions globally, only consider retail units for one chain of supermarket. Future work may also include applying the proposed model to more multivariate datasets from a broader cross-section of industries and applications beyond morbidity and mortality modelling.

In the next chapter, an adaptation and further extension of the model proposed here is presented, with a focus on another application domain, namely, multivariate anomaly detection. Furthermore, an investigation is conducted regarding what additional benefits such a model could offer if it is interpretable and explainable in terms of its model output and inference processes. The choice of benchmark models is motivated in Section 4.2.3, but in the next chapter the performance of the extension of MES-LSTM is compared against even more deep learning models such as convolutional and attention-based models.

Chapter 5

Hybrid Model for Interpretable Anomaly Detection

The work presented in this chapter is adapted from a research article submitted for presentation and publication in proceedings at the 3rd Southern African Conference for Artificial Intelligence Research (SACAIR 2022). It is currently under peer review, and a pre-print version of the manuscript is available by Mathonsi and van Zyl [62]. T. Mathonsi participated in the conception of the research, formulated and implemented the methods, conducted the experiments and analysed the results, and was the principal participant in the writing of the manuscript. T. van Zyl participated in the conception of the research and reviewed the manuscripts.

Summary

Hybrid methods have been shown to outperform pure statistical and pure deep learning methods at both forecasting tasks, and at quantifying the uncertainty associated with those forecasts (prediction intervals). One example is Multivariate Exponential Smoothing Long Short-Term Memory (MES-LSTM), a hybrid between a multivariate statistical forecasting model and a Recurrent Neural Network variant (Chapter 4). It has also been shown that a model that (i) produces accurate forecasts and (ii) is able to quantify the associated predictive uncertainty satisfactorily, can be successfully adapted to be a model suitable for anomaly detection tasks (Chapter 3). With the increasing ubiquity of multivariate data, and new application domains, there have been numerous anomaly detection methods proposed in recent years. The proposed methods have largely focused on deep learning techniques, which are prone to suffer from challenges such as (i) large sets of parameters that may be computationally intensive to tune, (ii) returning too many false positives rendering the techniques impractical for use, (iii) requiring labelled datasets

for training which are often not prevalent in real life, and (*iv*) understanding of the root causes of anomaly occurrences inhibited by the predominantly black-box nature of deep learning methods. In this chapter an extension of MES-LSTM is presented, an interpretable anomaly detection model that overcomes these challenges. With a focus on renewable energy generation as an application domain, the proposed approach is benchmarked against the state-of-the-art. The findings are that MES-LSTM anomaly detector is at least competitive to the benchmarks at anomaly detection tasks, and less prone to learning from spurious effects than the benchmarks, thus making it more reliable at root cause discovery and explanation.

5.1 Introduction

The growing abundance of time series data has motivated the increased research output with regards to time series classification for normal and anomalous data observations [159].

Interpretability and explainability are often used interchangeably. The terms are understood to mean the action of demystifying the predominantly black-box nature of deep neural networks. Gunning, Stefik, Choi, *et al.* [20] define an interpretable deep neural network as one that answers questions such as "*Why did it (or did it not) do that*?"; "*When does it succeed (or fail)?*"; or "*When can it be trusted?*" There are two broad categories of interpretable deep learning models, (*i*) model transparency and (*ii*) model functionality [160]. The former refers to understanding what the network model has learned and the reasons behind the learning. The latter explains inferences produced by the model in what is also known as post-hoc explanation generation [161].

The Clever Hans effect [162] demonstrates the importance and relevance of explainable machine learning in general, and explainable deep learning-based anomaly detection in particular. The Clever Hans effect concept is derived from a famed horse that was thought to perform accurate arithmetic, when it was in fact picking up visual cues from the master. It has been shown that anomaly detectors that model with deep learning are not immune to learning from spurious effects [163]. Learning from spurious effects is particularly dangerous when explainability is considered, as interested stakeholders may base important decisions on effects that are explained erroneously.

Time series anomaly detection is useful in applications from a variety of industries [4]. In their analytical study comparing classical and deep learning-based anomaly detection models, Munir, Chattha, Dengel, *et al.* [49] observe

that deep learning outperforms the classical approaches. In another study, Mathonsi and van Zyl [60] conclude that deep learning is at least competitive to statistical-based models. However, there is still a relative gap that exists for hybrid approaches in anomaly detection within the multivariate setting. Furthermore, root cause discovery and explainability remains an open research problem in the context of multivariate anomaly detection [38], [164].

One of the goals from the end of Chapter 3 is to merge the best attributes of classical statistical and deep learning-based models. This goal is achieved in Chapter 4 through the development and presentation of a statistics and deep learning-based hybrid model, Multivariate Exponential Smoothing Long Short-Term Memory (MES-LSTM) [61]. The developed method leverages multivariate exponential smoothing within the pre- and postprocessing modules. It also circumvents some problems associated with large sets of parameters requiring tuning, computational cost at training, and extensive inference time. These problems are circumvented by incorporating a small parsimonious recurrent deep neural network for learning, for instance, the cross-dependency structure within the covariates.

One of the goals at the end of Chapter 4 is extending MES-LSTM to the task of anomaly detection. Another, is to investigate the potential of explainability and interpretability for such a model. This chapter focuses on these goals, with an application to the renewable energy domain. Moreover, lessons from the two preceding chapters are consolidated and incorporated into a unified approach for explainable anomaly detection.

5.1.1 Recent Advances and the Current State-of-the-Art

The literature review is segmented into (i) work that has been presented in the research community relating to explainable anomaly detection, and (ii) *explainer systems* or techniques for interpreting anomaly detection models, explanation discovery, and root cause analysis.

A video can also be considered time series when the streaming images are taken as a matrix of pixel values with coordinates and a time dimension. As a result, some techniques that were traditionally applied to streaming video have also been adapted and extended to fit time series anomaly detection problems. As such, techniques from computer vision have not been excluded in the review of recent advances and the state-of-the-art.

Explainable Anomaly Detection

Some work has been done in unsupervised machine learning. Nguyen, Lim, Divakaran, *et al.* [165] for instance, consider the problem of anomaly detection in networks data, with an application to an Internet Service Provider example. The authors show their approach, using variational autoencoders, is able to effectively detect malicious attacks on a network. They use the gradients from the autoencoders for model interpretability.

Rad, Song, Jacob, *et al.* [38] consider the problem of explainable anomaly detection framed within a high dimensionality setting. The authors report competitive model performance without significant gains in computational cost.

Carletti, Masiero, Beghi, *et al.* [166] offer a technique for interpreting Isolation Forests (IF) [167], a commonly used model for anomaly detection tasks. They limit their focus to the scenario of Industry 4.0., with a particular focus on root cause analysis. Root cause analysis, as an application domain for explainable anomaly detection, emphasizes the need for robust models that are deployable in realistic industrial settings [75], [168].

Westerski, Kanagasabai, Shaham, *et al.* [169] consider explainable anomaly detection within a framework for procurement fraud identification. To this end, they consider real data from a government department in Singapore. The authors report their techniques, in real-life deployment, resulted in cost and time reduction of up to 10% compared to previously applied compliance checks. The lack in ubiquity of such studies illustrates the need for models that do not suffer from high computational cost. Great computational expense is one of the biggest criticisms of deep learning as it hinders real-time deployment in real-world applications [37].

Liznerski, Ruff, Vandermeulen, *et al.* [170] use an explainable convolutional model for one class image classification, and detail some challenges from spurious effects such as watermarks. A similar approach is followed by Pang, Ding, Shen, *et al.* [41]. A distinguishing factor is that the latter considers both training with no anomalies, and training with anomalous observations as well. Considering the problem of streaming images, Wu, Shao, Tunc, *et al.* [171] apply denoising autoencoders to surveillance video. The authors report competitive results with reduced computational time.

In terms of image classification, Ruff, Kauffmann, Vandermeulen, *et al.*

[55] offer a comprehensive review of other techniques that have been applied in the field, and systematically compare their performance on benchmark examples. Another notable review [48] looks at deep anomaly detection, and critically analyses the advantages and disadvantages of various techniques. Others, such as the works of Chalapathy and Chawla [172] and Kiran, Thomas, and Parakkal [46], only consider different classes of models applied to specific application domains. The interested reader is also directed to a discussion of explainability in health data [173], financial data [21], and object detection and recognition in image data [174], [175]. For a more general discussion of concepts related to explainability such as interpretability, and understandability, there are targeted resources such as Barredo Arrieta, Díaz-Rodríguez, Del Ser, *et al.* [176].

Applications for time series classification or anomaly detection can be discerned from the plurality of public dataset repositories, such as the UCR [177] or the UEA archive [178], for instance. Such applications, with a focus on explainable anomaly detection, include predictive maintenance at industrial sites [47], or rule extraction for unsupervised anomaly detection conducted [52].

Some models have been successful at classification and anomaly detection tasks, but due to their complex hierarchical architectures, incorporating explainability proves difficult. Some of these techniques are discussed here for completeness. One such advancement is an ensemble method, Collective of Transformation-based Ensembles (COTE) [179], where 35 models are ensembled over different time series representations based on transformations such as time warping [180] or shapelets [181], for instance.

COTE was further extended by Lines, Taylor, and Bagnall [182], using Hierarchical Voting (HIVE-COTE). HIVE-COTE performed well over the UCR and UEA archives [183]. Using a weighted probabilistic ensemble [184], this ensemble approach combines Shapelet Transform Classifier (STC) [185], Contractable Bag of Symbolic-Fourier Approximation Symbols (CBOSS) [186], Time Series Forest (TSF) [187] and Random Interval Spectral Ensemble (RISE) [182].

Explainability Systems and Methods

When categorized based on scope of the explanation, explainer systems offer either local or global explanations. Local explanations explain a single prediction result over the entire model, i.e., it explains the conditional interaction between dependent and independent variables with respect to the single

prediction. As mentioned by Ribeiro, Singh, and Guestrin [188], the explanation is required to make sense within a local setting. In the context of this thesis, this means that one explanation should be valid in some region encompassing immediate *neighbors*. The immediate neighbors are understood to be anomalous observations occurring around the same time, and of the same type.

Explainability of anomalies can also be conducted for a (potentially large) *set* of anomalies, for example, in the form of rule lists. These are called global explanations. Finding a truly global explanation, one that applies to multiple anomalous observations of different types occurring at different times is a difficult task [164]. As such, global explanations are usually aggregates of different explanations, or the most representative explanations for the entire model.

Another distinction can be drawn between model-specific and model-agnostic explainer systems. The former are applicable to certain kinds of model(s) (say for instance, strictly convolutional or strictly recurrent models, but usually not applicable to both) while the latter can be applied to multiple models.

Yet another distinction can be drawn between feature attribution, path attribution, and association rule mining techniques. Feature attribution determines the contribution of each feature towards the model's prediction for a given input example. This attribution shows the relationship between a feature and the prediction. As a result, users are able to understand which features their network relies on.

Path attribution methods explain the output of the model that is based on gradients. That means the contribution of each feature is computed by aggregating the gradients from baseline values to the current input along the path. One such method is Path Integrated Gradients (PIG) [189].

In contrast, association rule mining finds correlations and co-occurrences between features in a large dataset. They are considered as most interpretable prediction models with their simple if-then rules. A rule is essentially an if-then statement with two components: an antecedent and a consequent. The input feature with a condition is an antecedent and a prediction is its consequent. The popular techniques to extract the rules from a large dataset are Scalable Bayesian Rule Lists (SBRL) [190], Gini Regularization (GiniReg) [191] and Rule Regularization (RuleReg) [192]. Such techniques have been applied successfully to classifiers in surveillance tasks [193].

Barredo Arrieta, Díaz-Rodríguez, Del Ser, *et al.* [176] discuss transparent

models that automatically incorporate explainability such as Logistic regression, decision trees, and nearest neighbour models, as well as post-hoc models, that are explainable with the aid of an additional technique.

Explainability techniques that have been developed for or are typically used for image-based models include Deep Learning Important FeaTures (DeepLIFT) [194], Local Explanation Method using Nonlinear Approximation (LEMNA) [195], and Gradient-weighted Class Activation Mapping (Grad-CAM) [174]. These explainer systems usually output heatmaps [162] or saliency visualisations [196] that rank the feature importance of input images input to the network. An example of saliency maps is presented by Siddiqui, Mercier, Munir, *et al.* [197], for example, with application to convolutional layers.

For time series models, techniques often employed are Model Agnostic Supervised Local Explanations (MAPLE) [198], Local Interpretable Model-agnostic Explanations (LIME) [188], Local rule-based explanations (LORE) [199], Learning to explain (L2X) [200] and Shapley additive explanations (SHAP) [201]. As a cautionary note in particular for time series modelling, there is difficulty due to temporal dependence inherent in the data. As a consequence, surrogate solutions such as LIME or SHAP neglect the chronological sequence ordering of the model inputs.

LIME [188] explains model inferences by using a local interpretable sparse linear model as an approximation. Anchors [50] offers an incremental improvement on LIME by replacing the linear model used as proxy with a logical rule for explaining inferences. Anchors offers better coverage and explainability of anomalous neighbors but is not readily applicable to time series data. Other local explainer systems that rank feature importance include responsibility scores (RESP) [57] and axiomatic attribution [189].

5.1.2 Motivation

It is straightforward to motivate for deep learning as a mechanism for solving time series classification problems and anomaly detection tasks. One reason is to leverage the feature learning abilities of deep learning algorithms [202]. Deep neural networks have also performed well at other tasks requiring temporal sequence modelling (similar to time series) such as natural language processing [203] and speech recognition [204]. Lastly, deep learning has been shown to scale well with increased dimensionality [205].

However, there exists some challenges with the deep learning approach.

These include large sets of parameters that may be computationally expensive to tune, and long inference time that may be impractical in settings that require fast reliable information as feedback from models [61].

As evidenced from existing scholarly research, there is a great need for explanation discovery and interpretable anomaly detection with real-world applications such as root cause analysis [75], [167], [168]. There is a need to circumvent the computational cost and time complexity usually associated with deep learning that prevents them from being used outside of a laboratory setting and enables deployment in real-world applications such as in the compliance study conducted by Westerski, Kanagasabai, Shaham, *et al.* [169] or the streaming video study by Wu, Shao, Tunc, *et al.* [171].

In addition, learning from spurious effects can contaminate the root cause and explanation discovery leading to stakeholders making bad decisions informed by incorrectly explained model inferences. It is important to minimize the effects of learning from, say, random noise in time series data or even watermarks in image data [170].

As a final point, this research may be motivated using another factor from real world applicability. If an anomaly is identified, it might be time consuming for a domain expert or human agent to inspect all the components that may have possibly contributed to the anomalous event in order to identify the root cause. It may be more useful to the inspector if, for instance, a model is able to narrow the search space down to a reasonable fraction of components that are most probable to have contributed to the anomaly.

5.1.3 Contribution

The novel contribution in this chapter can be summarized as follows:

- a statistics and deep learning-based forecast machinery is extended to anomaly detection tasks;
- this new hybrid anomaly detection model (*i*) incorporates a dynamic threshold-setting approach, which learns and adapts the applicable threshold as new information becomes available, and (*ii*) functions within a semi-supervised framework, so no golden labels are required for training nor setting the thresholds for anomaly detection;
- the presented approach is augmented with explainability and interpretability thus enabling root cause analysis; and
- how well models avoid learning from spurious effects is assessed using a novel metric.

5.2 Methodology

Renewable energy resources, such as wind/solar farms, affect the grid in a different way compared to conventional fossil fuel generators due to their stochastic nature. In particular, the uncertain disturbances introduced by renewables may compromise operational grid safety. This scenario emphasizes the need for system operators to accurately identify disturbances in a timeous fashion. They are then able to perform corrective measures timely so as to ensure the safety of the grid.

System operators have access to streaming time-stamped measurements, from monitors such as phasor measurement units. These measurements enable system operators to answer critical questions including (i) When is an event happening? (ii) What type of event is happening? and (iii) Where is the source that caused the event? These are the research questions stated succinctly, and they would be phrased equivalently for other domains besides renewable energy regeneration.

Following the methodology of Zheng, Xu, Trinh, *et al.* [206], who first proposed the Power Systems Machine Learning dataset (PSML) [74] for use within the machine learning for decarbonized energy grids domain, the problem can be formulated as follows.

5.2.1 Problem Statement for Main Experimental Study

The streaming measurements can be denoted by $X \in \mathbb{R}^{N \times K}$, where N is the number of available observations and K is the number of measurements or covariates.

Event detection aims to answer the first question above, by identifying an oscillation occurrence when or if it happens. Answering this question involves using a model $\mathcal{H}$ to identify the oscillation occurrence given sequence X, i.e., $\mathcal{H} : X \to \{0, 1\}$. Suppose an event occurs at time t_*: an alarm should be raised when $t_m \geq t_*$, and as soon as possible (1 predicted).

Event classification answers the second question above based on streaming sensor measurements. Given the observations X, the model $\mathcal{H}$ must classify the underlying event type ξ, i.e., $\mathcal{H} : X \to \xi$. PSML presents a truly multivariate problem as ξ is more than just binary classification (i.e. normal or anomalous), but it constitutes a subset of disturbances $\mathcal{C}$ where $\mathcal{C} :=$ {branch fault, branch tripping, bus fault, bus tripping, generator tripping, forced oscillation}. This problem framing emphasizes the need to keep track

of multiple streams of data with interdependent covariates that are autocorrelated interacting within the global grid. By observing variables such as voltage from each bus in the system, the aim is to determine based on thresholds, for example, if and what kind of event is occurring.

Event localization focuses on locating events from disturbances $\mathcal{C}$, or the root cause of events (for forced oscillations) by observing measurements. The model $\mathcal{H}$ must map measurements X to the bus(es) z nearest to the events detected or the root cause of the events, i.e., $\mathcal{H} : X \rightarrow z$, where z is a subset of buses $\mathcal{Z}$ in the entire system.

5.2.2 Univariate Studies

In order to test the efficacy of the proposed model against established benchmark methods, two additional experiments are conducted, and the results reported herein. The first experimental study uses Nordpool Power Demand (NPD) [77], and the second uses the Large-scale Annotated Dataset for Energy Anomaly Detection (LEAD) [78]. Both datasets are annotated. NPD is univariate and although LEAD is multidimensional, the univariate property is enforced in a manner similar to OTD and DIC in Chapter4. The demand measurements from the more than 1,400 electricity meters are assumed independent for simplicity and are treated as separate univariate time series. The performance of the proposed model and benchmarks are then aggregated and presented.

In this univariate context, the Problem Statement detailed above can be relaxed by focusing on only the first of the three problems specified, i.e. *Event detection*, modified as follows. The streaming measurements can be denoted by $X \in \mathbb{R}^{N \times 1}$, where N is the number of available observations and 1 denotes a single variable of interest. Using a model $\mathcal{H}$, the problem involves identifying the anomalous occurrences given sequence X, i.e., $\mathcal{H} : X \rightarrow \{0, 1\}$. Suppose an event occurs at time t_*: an alarm should be raised when $t_m \geq t_*$, while minimizing $m - *$ (1 predicted).

5.2.3 Algorithms

The following benchmark models are used: InceptionTime [207], multi-channels deep convolutional neural networks (MC-DCNN) [208], Residual Network (ResNet) [43], Time series attentional prototype network (TapNet) [209], Minimal random convolutional kernel transform (MiniRocket) [210], one-Nearest neighbour with Euclidean distance (1NN) [211], independent dynamic time

warping (iDTW) [212], and dependent dynamic time warping (dDTW) [212]. The architectures of the different benchmark models are briefly described next.

Classical Models

Nearest Neighbour The first of classical model is one-Nearest neighbour with Euclidean distance (1NN). Nearest neighbour classifiers with a distance function have been among the most popular techniques for time series classification [211]. In one study, classifiers with dynamic time warping distance perform well as baselines [183]. In another, Lines and Bagnall [211] shows dynamic time warping is at least competitive to all the other distance measures considered. Interestingly, the best performers in the study are reported to be ensembling neural network classifiers combined with different distance measures.

Dynamic Time Warping Dynamic Time Warping (DTW) can be applied to time series data composed of varying length, but for simplicity, the following description is limited to the case involving series of equal length, such as presented by Bagnall, Lines, Bostrom, *et al.* [183]. The distance between two equal length series $\boldsymbol{a} = (a_1, a_2, \ldots, a_m)$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_m)$ is calculated as follows:

1. $\boldsymbol{\psi}$ is a matrix sized $m \times m$ where $\boldsymbol{\psi}_{i,r} = (a_i - b_r)^2$
2. A warping path $\rho = ((e_1, f_1), (e_2, f_2), \ldots, (e_s, f_s))$ is a contiguous set of matrix indices from $\boldsymbol{\psi}$, subject the constraints:
   - $(e_1, f_1) = (1, 1)$
   - $(e_s, f_s) = (m, m)$
   - $0 \leq e_{i+1} - e_i \leq 1 \, \forall \, i < m$
   - $0 \leq f_{i+1} - f_i \leq 1 \, \forall \, i < m$
3. Let $p_i = \boldsymbol{\psi}_{e_i, f_i}$, then the path distance is $\mathcal{D}_p = \sum_{i=1}^{m} p_i$
4. Multiple warping paths exists, but the aim is to find a path that minimizes the accumulative distance $\rho^* = \min_{p \in \rho} \mathcal{D}_p(\boldsymbol{a}, \boldsymbol{b})$
5. Solving the following relation yields the optimal distance

$$\mathrm{DTW}_{(i,r)} = \boldsymbol{\psi}_{i,r} + \min \begin{cases} \mathrm{DTW}_{(i-1,r)} \\ \mathrm{DTW}_{(i,r-1)} \\ \mathrm{DTW}_{(i-1,r-1)} \end{cases}, \tag{5.1}$$

   where the final distance is given by $\mathrm{DTW}_{(m,m)}$.

Some improvements may be applied to DTW for increased efficiency, such as constraining deviations from the diagonal but this falls beyond the scope of this thesis. Shokoohi-Yekta, Hu, Jin, *et al.* [212] defines strategies for applying DTW to multivariate setting. These are independent and dependent approaches.

The independent method, iDTW, as the name suggests, has a separate treatment for each dimension. Using a separate distance matrix for each dimension, iDTW then sums the resulting time warping distances:

$$\text{iDTW}_{i,r}(\boldsymbol{x_a}, \boldsymbol{x_b}) = \sum_{k=1}^{d} \text{DTW}(x_{a,i,k} - x_{b,r,k})^2 \tag{5.2}$$

The main idea behind Dependent dynamic time warping (dDTW) is the assumption that the accurate warping is identical for all the dimensions. Given a single time series, the matrix $\boldsymbol{\psi}_{i,r}$ is no longer considered the distance between two points, but is redefined as the Euclidean vector distance computed on the vectors that constitute a representation of the full dimensional space. The dependant strategy is more efficient as warping is simultaneous for all the dimensions, and the distance between steps i and r in terms of time resolution is given by

$$\boldsymbol{\psi}_{i,r}(\boldsymbol{x_u}, \boldsymbol{x_v}) = \sum_{k=1}^{d} (\boldsymbol{x_{u,k}}, \boldsymbol{x_{v,k}})^2. \tag{5.3}$$

There also exists an adaptive strategy [212] for selecting between independent and dependent dynamic time warping. How the distance is chosen depends on an instance-by-instance threshold deducible from the training data. Adaptive time warping falls beyond the scope of the current study.

Deep Learning-based Models

MiniRocket MiniRocket [210] is adapted from Rocket [213] which was ranked among the best performers on multiple datasets in a recent study [214]. The authors report MiniRocket is at most 75 times faster that Rocket, with comparative accuracy. Rocket combines convolution kernels that are randomly initialised using a linear classification model, typically ridge or logistic regression. The method produces feature maps where the maximum value the proportion of positive values (ppv) are extracted.

Hyperparameter tuning is restricted to the following search spaces. The length $\varsigma \in \{7, 9, 11\}$; the kernel weights $w_i \sim \mathcal{N}(0, 1)$; the dilation, d, is

sampled from the exponential distribution; and whether the series is padded is decided with equal probability.

In contrast, MiniRocket attempts to minimize the randomness characteristic of Rocket. It achieves this by pre-assigning values to a subset of the hyperparameters discussed above, or limiting the search space to a smaller grid, yet still reportedly achieving comparable accuracy. The changes are summarized in Table 5.1 [210], where $\mathcal{N}$ is the normal distribution and $\mathcal{U}$ is the uniform distribution.

TABLE 5.1: Summary of Changes from Rocket to MiniRocket [210].

Hyperparameter	**Rocket**	**MiniRocket**
Length	$\{7, 9, 11\}$	9
Weights	$\mathcal{N}(0, 1)$	$\{-1, 2\}$
Bias	$\mathcal{U}(-1, 1)$	from convolution output
Dilation	random	fixed
Padding	random	fixed
Features	ppv, max	ppv
Number of features	20,000	10,000

MC-DCNN Multi-Channel Deep Convolutional Neural Network (MC-DCNN) [208] is a modification of conventional deep convolutional neural networks. The convolutions are applied independently per covariate in the multivariate input space.

Every dimension of the multivariate input data goes through two convolutional stages with eight filters each of length five and configured with ReLU activation functions. After each convolution there is a max-pooling operation, followed by a fully connected layer. Softmax is used for the final classification.

ResNet ResNet [43] architecture has three convolutional layers within each of three residual blocks, followed by a Global Average Pooling (GAP) layer. The main idea behind ResNet is the use of residual shortcuts connecting consecutive convolutional layers. The key difference when compared with conventional convolutions from fully convolutional networks for instance, is the addition of these linear shortcuts. The shortcuts reduce the vanishing gradient effect [215], by enabling the gradient to flow directly through these connections. In a recent study [214], ResNet ranked among the best performers on multiple datasets.

InceptionTime InceptionTime incorporates ResNet [43] and Inception modules [216]. An Inception module takes as input multivariate series of size $m \times k$. By using a bottleneck layer with length and stride one, it reduces the dimensionality to $m \times k$ where $k' < k$. InceptionTime assigns random initial weights to five instances of the artificial neural network and ensembles them for greater stability [207]. One out of the five networks replaces the three blocks of the aforementioned three classical convolutional layers from ResNet with dual-blocks containing three Inception components each. However, the new blocks similarly retain residual connections, and they too lead out to two layers, GAP and softmax.

TapNet The final benchmark model considered combines classical and deep learning-based traits. Zhang, Gao, Lin, *et al.* [209] observe how deep learning methods are good at learning features of low dimension, and classical approaches such as dynamic time warping are competitive for applications involving small datasets. TapNet, combining these traits, has a network architecture composed of three components: Random Dimension Permutation, Multivariate Time Series Encoding and Attentional Prototype Learning. These modules can further be broken down into fully connected layers, three sets of convolutional layers that are one dimension each, a global pooling layer, batch normalisation, and Leaky Rectified Linear Units (LReLU) [217] (see Appendix A).

5.2.4 Anomaly Detection

The benchmark models and their hyperparameters are kept constant from the original manuscripts that the techniques were initially introduced. Hyperparameters are important because tuning each architecture to a specific task, and using the best predictor obtained, severely impacts performance. However, there are instances where tuning may not be the best approach in terms of computing resources, time constraints, and avoiding development based on task specificity with models that may not be able to generalize well [214]. Recently, authors of reviews and studies comparing the performance of the latest cutting-edge methods and oft-preferred classical architectures on forecasting and classification tasks have opted not to tune hyperparameters [183], [214], [218]. The same approach is followed in this chapter for all the benchmark models. Similarly for MES-LSTM, the model

architecture as described in Chapter 4 is retained, and Algorithm 4 from Section 3.2.8 is employed in order to adapt the forecast machinery for the task of anomaly detection.

For PSML, training time series are extracted from the millisecond transient phasor measurement units' data. Training samples are randomly selected amounting to 439 time series, and the remaining 110 time series are used for testing (20%). Each sequence has metadata associated with the event type similar to the classification use-case, i.e. *branch fault, branch trip, bus fault, bus trip,* and *gen*. Each time series has a sequence length of 960 observations, representing 4s in the system recorded at 240Hz. There are 91 dimensions for each time series, including voltage, current and power measurements across the transmission system. The experiment is repeated 35 times to mitigate the stochastic nature of the deep learning models.

5.2.5 Interpretability

The presented approach uses model-agnostic feature attribution techniques. LIME [188] is a local explainability technique suitable for local explanations. This method perturbs the input in the neighbourhood of an instance and examines the output of the model. LIME, thus, indicates the input features the model considers when making a prediction. LIME works by using a proxy based on a simple model, intrinsically interpretable, such as a linear regression model. The surrogate model is applied around each prediction between the input variable space and the corresponding outcome variables space. Explainability is discerned from the main anomaly detection model by perturbing the input variables of a multivariate observation and tracking how the predictions change.

SHAP [201] use Shapley values from cooperative game theory, which indicates what reward players can expect depending on a coalition function. To extend this approach to the explainability of artificial intelligence agents, players are considered as features and reward as the outcome of the model. Although SHAP also provides global interpretability (behaviour of the entire model), this thesis focuses on its ability to shed light on local interpretability (behaviour of a single prediction). This local focus enables the use of SHAP in conjunction with LIME and facilitates comparison. The importance rank of feature i is deduced by taking all subsets of features except feature i, $D \setminus x_i$, and computing the effect of the output predictions after adding feature i back to all the subsets previously extracted. All the contributions are

then combined to compute the marginal contribution of each feature.

The labelled PSML dataset used in the multivariate study stipulates which predictor contributed most to an anomaly. Ideally, for the interpretability component to be useful for stakeholders, the correct contributor should be identified and ranked high up in terms of feature importance. To ensure this, a novel metric is presented, that can be tuned to the appropriate task-specific sensitivity.

Definition 5.2.1 (Mean Discovery Score.) *Let β indicate the task-specific sensitivity, i.e. ideally, the principal contributing predictor κ should be ranked in the highest β features in terms of importance. Let the i^{th} out-of-sample observation that is in fact anomalous be denoted by $y_i \in \{a\}$, where $\{a\}$ is the set of all anomalies. Then, by aggregating how many times this ranking occurs at the specific sensitivity, the mean discovery score (MDS_β) is given by*

$$MDS_\beta = \frac{1}{\textbf{card}\left(\{a\}_D^{A_i}\right)} \sum_{t=1}^{m} \mathbb{1}_{y_i \in \{a\} \cap \{\mathcal{R}(\kappa_i) \leq \beta\}}, \tag{5.4}$$

where the rank of a variable's feature importance is denoted by $\mathcal{R}$, and $\textbf{card}\left(\{a\}_D^{A_i}\right)$ *is how many times an algorithm A_i is able to detect anomalies in dataset D.*

MDS_β is useful for scoring the explainability of accurate anomaly detection models, and would not be suitable for use if the set of anomalies correctly detected $\{a\}$ by algorithm A_i is small in comparison to the set of overall anomalous events.

5.2.6 Analysis

For programmatic platform and hardware details refer to Appendix B. The code implementations in this Chapter's studies retain the default settings detailed by the respective authors in their original manuscripts. Some algorithms have modules embedded that perform hyperparameter tuning. In cases where this is applicable, the hyperparameter optimization modules are kept as is, but no additional tuning is conducted. Below, the configurations for each algorithm is detailed.

For DTW the full warping window is used. MiniRocket is configured with a ridge regression classifier and 10,000 kernels. TapNet uses defaults set to 500 trees, 3,000 epochs, a learning rate of 10^{-4}, weight decay of 10^{-2}, stop threshold of 10^{-8}, number of filters given by 256, 256, and 128 respectively,

kernels by 8, 5, and 3 respectively, while dilation is one with no dropout. ResNet has 1,500 epochs, a batch size of 16, learns at a rate of 10^{-2} which, if no improvement is observed for 50 epochs, is set to 5^{-2}. ResNet is configured with three residual blocks composed of three convolutional layers each, where the sizes of the kernels are 8, 5, and 3 respectively, and 64, 128, and 128 filters respectively per convolutional layer for each block. InceptionTime runs for 1,500 epochs with a batch size of 64. InceptionTime is configured with dual residual blocks, each composed of three Inception modules where the sizes of the kernels are sizes 10, 20, and 40 respectively per module, and learns at a rate of 10^{-2}, which, if no improvement is observed for 50 epochs, is set to 5^{-2}.

5.3 Results and Discussion

This section analyses the results of the performance of the proposed method compared to the state of the art. Both the anomaly detection and the interpretability tasks are analysed and discussed. The discussion begins with the univariate studies and concludes with the main, multivariate experimental study.

5.3.1 Univariate Modelling and Anomaly Detection

Table 5.2 gives the aggregated results for the area under the ROC (auROC) curve with regards to the experimental study conducted on NPD. ResNet shows the highest aggregate accuracy achieved on this detection task, whereas the 1NN shows the lowest. The noticeable variability for the non-deterministic models with respect to each independent trial speaks to the matter of reliability. Again in this variance instance ResNet has the tightest distribution of performance scores.

TABLE 5.2: Area Under Receiver Operating Characteristic Curve for All Models Over All Trials.

	1NN	dDTW	iDTW	InceptionTime	MC-DCNN	MES-LSTM	MiniRocket	ResNet	TapNet
mean	0,2701	0,3646	0,4100	0,6979	0,6615	0,6236	0,6388	0,7136	0,5595
std	0,0000	0,0000	0,0000	0,1337	0,0455	0,0670	0,0742	0,0473	0,0830

The box and whisker plot in Figure 5.1 indicates InceptionTime has achieves the highest median score, whereas TapNet achieves the lowest from all non-deterministic models. The proposed model has the second to worst median score, although achieving the third best minimum score.

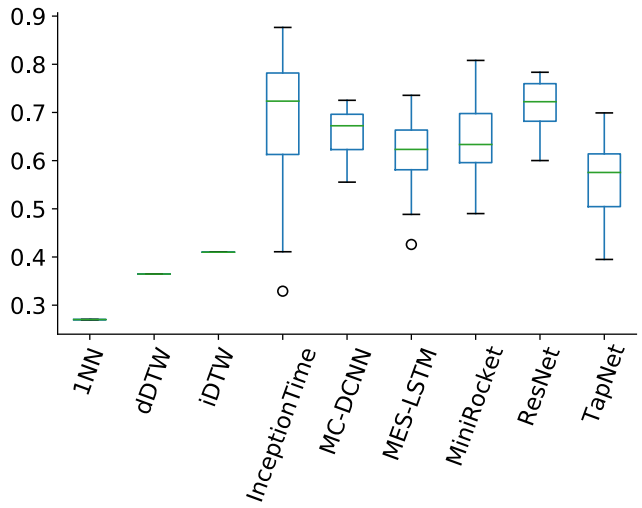


FIGURE 5.1: Area Under the ROC Curve Distribution Boxplot for All Trials.

The deterministic models performed poorly at this task, whereas the other benchmarks achieved comparative skill and sometimes outperform the proposed model. In order of increasing detection accuracy, the deterministic models can be ranked as 1NN, dDTW, and iDTW. One of the best performers is InceptionTime, but the box and whisker plot in Figure 5.1 shows a big spread of performance scores. There is even an outlier minimum value that is comparative the the time-warping methods.

Attention is now turned to the experimental study conducted on LEAD. Examining the area under the Precision-Recall (auPR) curve in Table 5.3, MC-DCNN achieves the best aggregate class separability score. MC-DCNN also has the lowest variance when compared to the non-deterministic models.

TABLE 5.3: Area Under Precision-Recall Curve for All Models Over All Trials.

	1NN	dDTW	iDTW	InceptionTime	MC-DCNN	MES-LSTM	MiniRocket	ResNet	TapNet
mean	0,4935	0,4100	0,5001	0,7186	0,8596	0,8389	0,7353	0,8106	0,5712
std	0,0000	0,0000	0,0000	0,0901	0,0281	0,0590	0,0497	0,0165	0,0591

The box and whisker plot in Figure 5.2 further shows the maximum median score is achieved by MC-DCNN, with small variability as evidenced by the tight band for the classification trials (not accounting for outliers). The proposed method shows competitive performance skill, with the highest maximum detection score if disregarding anomalies, and the second highest with anomalies taken into account.

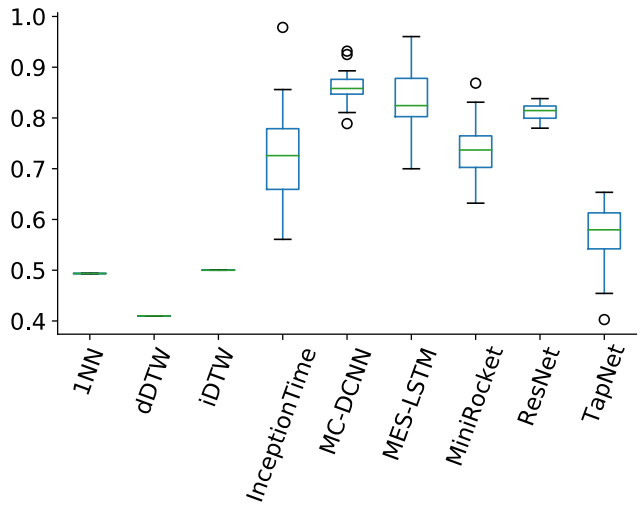


FIGURE 5.2: Area Under the PR Curve Distribution Boxplot for All Trials.

The proposed model is at the very least competitive, showing small variability with performance results from multiple independent trials. The model also shows high maximal scores, and high minimal scores. The main concern for the two above experiments is that they are univariate in nature. This is of particular concern here because of the chosen application domain. Electricity demand is perhaps better understood with exogenous factors, such as weather for instance. Nonetheless, following the presented methodology, the results offer enough evidence of anomaly detective skill to warrant further probing the efficacy of the model proposed model with a deeper exploration. The in-depth study and analysis is conducted and discussed in the next section.

5.3.2 Multivariate Anomaly Detection

One shortcoming with the studies detailed in the previous section, is because they are univariate in nature, feature attribution is hard to apply. As a consequence, it is hard to add a layer of explainability to the aforementioned analytical results, an extension that could add value for practitioners and domain experts who may be interested in *why* a specific day of the week sees more pronounced spikes in demand than all the other days, for instance. As such, the experiments conducted in this chapter are multivariate, and are thus able to take the discussion further than anomaly detection but touches on model strength and reliability due to explainability as well.

Table 5.4 details the aggregated results for the auROC curve. When observing the standard deviation, the deterministic models all have no variability in their results, i.e. 1NN, and the dynamic time warping models. InceptionTime shows the most variability and this property may be somewhat undesirable within the context of assisting domain experts. A good anomaly detection model should have results that are not only accurate but are also consistent over multiple trials. In this regard of variability, ResNet has the most consistent results from the non-deterministic models.

TABLE 5.4: Area Under Receiver Operating Characteristic Curve for All Models Over All Trials.

	1NN	dDTW	iDTW	InceptionTime	MC-DCNN	MES-LSTM	MiniRocket	ResNet	TapNet
mean	0.3301	0.4146	0.4500	0.5822	0.7547	0.7376	0.5675	0.7012	0.4804
std	0.0000	0.0000	0.0000	0.1025	0.0500	0.0736	0.0925	0.0358	0.0812

Table 5.4 also indicates that MC-DCNN is the best performing anomaly detection model, followed closely by MES-LSTM, ResNet, InceptionTime and MiniRocket. TapNet is not much better than the deterministic distance-based models.

The box and whisker plot in Figure 5.3 indicates MES-LSTM achieves the highest performance score on a single trial (highest whisker end-point), although MC-DCNN has the highest overall mean aggregated over all trials. Inception Time and MiniRocket have the highest variability in terms of performance results, whilst ResNet is the most consistent model.

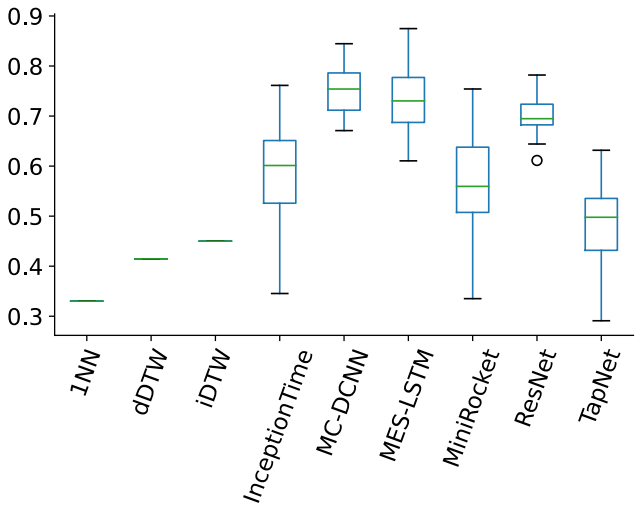


FIGURE 5.3: Area Under the ROC Curve Distribution Boxplot for All Trials.

Examining the auPR curve in Table 5.5 reaffirms the above discussion. The main difference is in the range of performance scores. The range is higher across all the models as the auPR metric takes into account the class imbalance inherent in the data.

TABLE 5.5: Area Under Precision-Recall Curve for All Models Over All Trials.

	1NN	dDTW	iDTW	InceptionTime	MC-DCNN	MES-LSTM	MiniRocket	ResNet	TapNet
mean	0.4225	0.4803	0.5500	0.7332	0.8470	0.8421	0.6359	0.8117	0.5604
std	0.0000	0.0000	0.0000	0.1087	0.0328	0.0549	0.0594	0.0170	0.0471

The box and whisker plot in Figure 5.4 further shows the maximum auPR is achieved by InceptionTime, at the cost of the aforementioned variability (which also adversely impacts the overall performance mean). With regards to highest overall performance mean, the top five models are, in order from best, MC-DCNN, MES-LSTM, ResNet, InceptionTime, and MiniRocket.

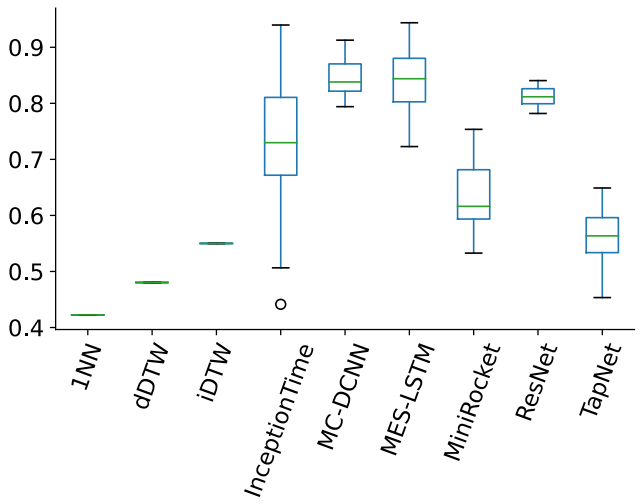


FIGURE 5.4: Area Under the PR Curve Distribution Boxplot for All Trials.

A Student's t-test for statistical significance is conducted at the $\alpha = 0.01$ level of significance, for the performance results in terms of anomaly detection. The null hypothesis is H_0: the benchmark models outperform MES-LSTM. From Tables 5.6 and Table 5.7, the only instance where one is unable to reject the null hypothesis at the $\alpha = 0.01$ level of significance is for MC-DCNN.

TABLE 5.6: Student's t-test for Testing Significance of auROC Performance Results (H_0: Benchmark Models Outperform MES-LSTM).

	1NN	dDTW	iDTW	InceptionTime	MC-DCNN	MiniRocket	ResNet	TapNet
statistic	32.7568	25.9649	23.1195	7.2838	-1.1331	8.5118	2.6362	13.8820
p-value	0.0000	0.0000	0.0000	0.0000	0.8692	0.0000	0.0056	0.0000

TABLE 5.7: Student's t-test for Testing Significance of auPR Performance Results (H_0: Benchmark Models Outperform MES-LSTM).

	1NN	dDTW	iDTW	InceptionTime	MC-DCNN	MiniRocket	ResNet	TapNet
statistic	45.2511	39.0184	31.5026	5.2917	-0.4495	15.0871	3.1367	23.0563
p-value	0.0000	0.0000	0.0000	0.0000	0.6726	0.0000	0.0016	0.0000

5.3.3 Interpretabilty and Explainability

Since there are 96 covariates in total, a starting point would be to consider what a domain expert might consider useful inference from a machine learning tool. In case of power trip, for instance, it would be time consuming to check all 96 possible contributors. However, checking a reasonable subset would be more feasible. Below, β is set to elements in the range $\{5, 10, 15\}$, although this range can be determined by requirements specific to a particular use-case scenario. The rationale behind the chosen range is ideally, a good explainer system should rank the chief contributor in the first five highest ranked features (saving the domain expert the most amount of time); an explainer with moderate skill would rank the chief contributor in the five to first ten highest ranked features, while the worst would rank the chief contributor in the first ten to 15 highest ranked features and beyond.

Below, only the top four performing models are considered. Anything that offers less than 50% in accuracy for anomaly detection can be argued to be no better than random guessing. TapNet, although above 50% in performance score, does not report anomaly detection performance skill much higher than the distance-based methods and is also not included in the discussion that follows.

Table 5.8 shows MES-LSTM has the highest correct attribution at $\beta = 5$ features considered, followed by MC-DCNN, InceptionTime and ResNet. At $\beta = 10$ and at $\beta = 15$ features considered, MES-LSTM and InceptionTime are in the top two. ResNet peaks at around 80% correct attribution at $\beta = 15$ features considered, while InceptionTime has the highest overall score at 94.61%. Not one of the models reaches 100%, indicating that there are still

some missing key contributing factors not accounted for even after 15 covariates are explored in terms of feature importance.

TABLE 5.8: Mean Discovery Score for LIME Applied to Top Four Anomaly Detectors.

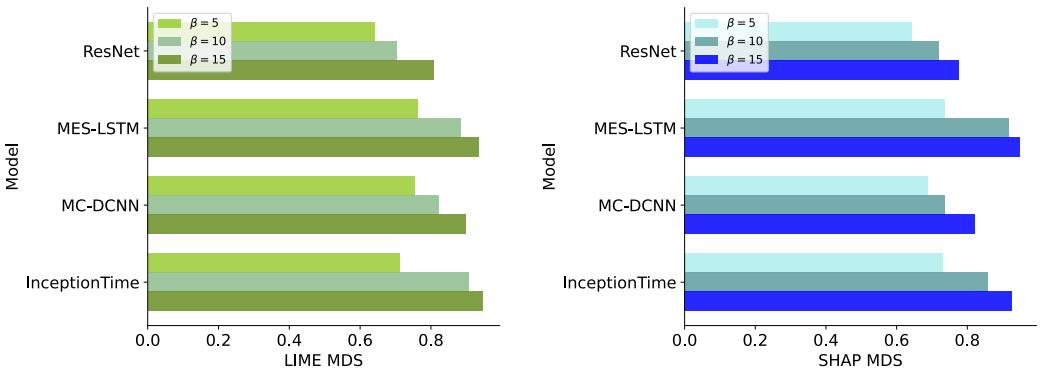
β	**InceptionTime**	**MC-DCNN**	**MES-LSTM**	**ResNet**
5	0.7113	0.7554	0.7634	0.6419
10	0.9059	0.8222	0.8856	0.7025
15	0.9461	0.8985	0.9346	0.8077

From Table 5.9 it is deducible that at $\beta = 5$ features considered, at most only 73% explainability is accounted for. The maximum score is achieved by MES-LSTM at $\beta = 15$ features considered.

TABLE 5.9: Mean Discovery Score for SHAP Applied to Top Four Anomaly Detectors.

β	**InceptionTime**	**MC-DCNN**	**MES-LSTM**	**ResNet**
5	0.7311	0.6871	0.7376	0.6437
10	0.8570	0.7380	0.9179	0.7205
15	0.9263	0.8219	0.9481	0.7754

The discovery scores are illustrated graphically in Figure 5.5. MES-LSTM has the highest correct attribution at all levels of features considered for both LIME and SHAP, except for LIME at $\beta = 10$ (where MES-LSTM is outperformed by InceptionTime).



(A) (B)

FIGURE 5.5: Mean Discovery Score for Explainer Techniques Applied to Top Four Anomaly Detectors. (**A**) LIME. (**B**) SHAP.

5.4 Conclusion

There are two main objectives in this chapter. One is to build an anomaly detection machinery that is a hybrid composed of statistical and deep learning techniques. The second is to incorporate interpretability to such a technique. The desired outcomes related to these objectives are one, that the anomaly detection skill is at least competitive to the state-of-the-art; and two, that the explainability component has a good level of correct attribution.

The proposed method is outperformed in some instances with regards to anomaly detection by, for example, MC-DCNN. MC-DCNN is able to model the spatial correlations well, and this could be a contributing factor to the outperformance. MES-LSTM is outperformed marginally and although competitive with regards to anomaly detection, the overall performance is an area for improvement.

However, when it comes to correctly attributing important features for the anomalies detected by each of the top four performing models, MES-LSTM is the overall highest achiever. The high discovery scores are as a result of the architecture's good modelling of temporal dependence. Accurate attribution is important as it ensures the model is not learning from spurious effects. It also reinforces trust for manual inspectors of, say, mechanical systems when an anomaly within the system is detected and possible causes are reported.

The voltage measurements and current measurements are governed by both time evolution from external oscillation events, as well as spatial dependency from the inherent network connectivity over the entire grid. The problem framing is of multivariate spatio-temporal anomaly detection and interpretability. Future work may involve graph neural networks, which show significant promise in the tasks underpinned by similar settings, such as in climate modelling [219].

It is possible that through additional engineering of the benchmark algorithms and tuning of their hyperparameters, better overall performance can be realized. However, the idea was to test the anomaly detectors based on the configuration recommendations suggested by the respective authors in their original manuscripts. Although not discussed in detail in this current research, the approach using original configurations allows an additional layer of comparison between the models related to how well they generalize to new problems.

In this study, a novel metric is proposed for assessing usability of a model with regards to usefulness of the model's interpretability to a domain expert.

There are many metrics for tasks such as forecasting and anomaly detection, but research centered around explainability is lacking in terms of metrics for ease of comparison among multiple models. Adding more metrics to measure the level of correct attribution is also an avenue for future research.

Chapter 6

Conclusion

The aim of this research was to develop a hybrid model that is parsimonious, thus computationally inexpensive to train with real-time inference. Such a model was developed by extending existing methods from classical statistical techniques and combining them with deep learning. Said novel anomaly detection model was developed systematically through empirically evaluating it at each phase of the proposed development pipeline. Although each of the chapters relevant to a topic concludes with a summary of findings and prospects for future work, this chapter serves to tie together the main ideas presented in this thesis, contextualize the findings, and highlight some key points.

6.1 Findings and Research Output

In Chapter 3, a novel method of regression-based anomaly detection was introduced, and it was applied to both deep learning and classical statistics-based models. The method involves dynamic threshold setting and does not require golden labels for training. Using computational techniques to synthesize better prediction intervals lead to more accurate and quicker anomaly detection. Applying the novel anomaly detection method to both classes of techniques also illuminated their respective advantages and drawbacks. This chapter was adapted from work that was presented at the 7th International Conference on Soft Computing and Machine Intelligence, 2020. The publication in conference proceedings [40] and subsequently the extended, full journal article [60] both contributed to this chapter.

In Chapter 4, a novel statistics and deep learning-based hybrid model for forecasting and uncertainty quantification is introduced. The hybrid model was applied to morbidity and mortality data with exogenous factors. Its performance was compared to pure deep learning and pure statistical models, and the novel model was shown to significantly improve upon the predictive

skill of its classical pure probabilistic and pure deep learning counterparts in the best case. It was shown to be at least comparative in the worst case. The model also drew on insights from Chapter 3 to avoid the drawbacks of methods and leverage the advantages from models in the two classes of techniques, namely classical and deep learning. This chapter was adapted from work presented in another journal publication [61].

In Chapter 5, the methodologies and results from the two preceding chapters are consolidated and combined and culminate to a novel interpretable anomaly detection model. It was applied to energy demand applications as well renewable energy generation in decarbonized grids. The anomaly detection skill of the proposed model was at least competitive to the state-of-the-art. In addition, the explainability results indicated the novel approach has a good level of correct attribution and is not prone to learning from spurious effects. Using a novel metric for root cause discovery performance, the results show the proposed approach would be beneficial for root cause discovery assisting human agents in, say, industrial applications. This chapter constitutes recent results submitted to and currently under peer review for presentation and proceedings publication at the 3rd Southern African Conference for Artificial Intelligence Research (SACAIR 2022). A pre-print version of the manuscript is available [62].

All the experimental studies from the above streams of research were conducted through the use of datasets detailed in Chapter 2. The datasets were selected from a diverse array of application domains, ranging from climate to space aviation. These datasets were specifically selected to ensure they are distinctly different in terms of size, frequency, levels of class-imbalance, spatial and temporal resolution, and dimensionality. The diversification requirement within the data sourcing methodology was useful in mitigating biases in model performance.

6.2 Discussion

Forecasting and quantifying uncertainty is still an important research topic. With the recent COVID-19 pandemic, for instance, there has been unprecedented impact on healthcare. A lot of the negative impact can be alleviated with multidisciplinary approaches that drive accurate forecasting to better inform decisions and planning. With the rise of access to toolkits, the democratization of data science, increasing availability and ease of access to open datasets, it is becoming increasingly important to provide useful insights to

assist in minimizing the strain on global healthcare systems. As an example, predicting if an individual who has an upper respiratory tract viral infection might worsen to more severe complications is difficult, and can be considered as future work. The usefulness can be demonstrated as patients who progress to the severe conditions are often admitted to intensive care with heightened risk of not recovering.

The above benefit goes beyond clinical applications and the scope exceeds forecasting and uncertainty quantification. It extends to multiple application domains where there is also applicability and utility from techniques including anomaly detection and explainability. With the aid of multivariate pools of big data, one can be better equipped to develop, implement, test, and optimize the decision-making pathways of stakeholders in many different applications. This expansive applicability is demonstrated through the multiple datasets from different industries and application domains that have been utilized in the experimental studies reported in this research.

Research progress has been made since the introduction of classical statistical models such as linear regression, which can be thought of as a precursor of supervised machine learning. One leap into the right direction is the incorporation of flexibility in exploring unknown associations, such as has been applied with the models proposed herein through, for instance, semi-supervised learning. The benefits of such modelling can be demonstrated in the context of global pandemic events, for example.

One of the research questions addressed here has to do with how one can merge the best characteristics of deep learning and classical models to form an interpretable hybrid model for multivariate forecasting, uncertainty quantification, and anomaly detection, that inherits the least possible negative traits from each. Classical methods are still very useful in research and industry alike. For example, linear regression models such as the Acute Physiology and Chronic Health Evaluation (APACHE) score is still used for patience in intensive care [220]. Another example is the Model of End-Stage Liver Disease (MELD), which is actively used all over the world [221].

On the other hand, deep neural networks are able to learn correlations between covariates but have been criticized for their inherent difficulty associated with interrogating the learnt latent features, for example. Additionally, there currently is no mathematical proof that previously unseen data entering a system has appropriate representation within the internal structure of the model, and cross-validation is not necessarily the best indicator of this [222]. There is also no mathematical proof that given sufficiently large

data, a deep neural network after implicitly learning a representation of the training data can be applied out-of-sample [222]. Regression models in contrast, approximate predictive performance using closed-form solutions.

In this thesis, rigorous mathematical derivations have been provided as part of the novel theory presented. There is however more work still to do in terms of addressing the criticism of deep neural networks being black-box in nature. One example where this black-box feature can lead to catastrophic results is where the hidden mechanics may provide outcomes in health application that physicians consider useful but actually have limited-to-no biologically relevant causal effects.

Despite the poor performance of MLR throughout the studies conducted in this research, one can still advocate for its use in other applications which may be more appropriate as it can provide useful insights. Take for instance MLR, the coefficients in this classical regressor can be translated literally. Large positive correlation can be understood as high importance, and vice versa. One application domain would then be precision or individualized medical treatment where clinicians can tailor their decision rules based on personal factors unique to each patient.

So, citing criticisms of each approach, classical and deep learning, it is straight forward to advocate for the presented approach as it combines the two streams in a way that attempts to select the best characteristics of each, and neglect the worst. This kind of selective hybridising is applied throughout the entire presented research and development pipeline, i.e. forecasting, uncertainty quantification, anomaly detection and explainability.

6.3 Future Work and Closing Remarks

Although theme-specific avenues for future work were highlighted in the concluding remarks each of the three preceding chapters, the thesis is concluded here with overarching ideas and suggestions. The presented anomaly detection method is semi-supervised and does not require labelled data for training. However, scoring correct attribution does require labelled metadata pointing to the root cause. Explanation discovery is mostly possible through exploiting metadata such as information containing exactly which predictors contribute most to an anomaly. Possible future directions for research could involve producing or sourcing, cataloguing, and availing such data to the research community for conducting similar experimentation and advancing the field.

One of the key attributes of MES-LSTM is the preprocessing layer that is equipped to deal with, for instance, missing data in a prudent manner. It may be worthwhile investigating how missing data could have potentially been anomalous, and what impact this would have on model performance.

Hybrid methods show significant promise, but so do other techniques, such as hierarchical ensemble forecasters and anomaly detectors. There exists a vacuum in terms of interpretability of the latter because of their complex non-linear architectures. Explainability of classical methods was also not considered in detail in this research, but future work could take this avenue. Bias in inferences made by deep machine learning models also remains largely an unsolved issue. Future work can also investigate the effects of bias from all phases of model development, including training and validation.

Ordinary least squares, for example, is efficient, fast to train, and easily interpretable. Bayesian techniques can be used to insert prior domain knowledge into hierarchical regressors [223]. Over and above correlation, causality can also be explored using classical statistical methods through the implementation of directed acyclic graphs. This causality approach can be particularly useful in application domains where knowledge of the state of the world as described in the study is more important than improved accuracy performance.

Using the techniques presented herein as building blocks, explainability of hierarchical models can also be performed as future research, with applicability to ethical artificial intelligence, for example. Given the methodology followed in this research, the results discussed, the final findings interrogated, and the avenues proposed for future work, the future advancement of this and related research, and their applications, looks promising.

Appendix A

Supplementary Algorithms and Theory

A.1 Introduction

This appendix serves to describe the general training rules, supplementary theory and algorithms referenced in the main text of this thesis. First, the perceptron training [224] rule is considered. Then an exposition is offered of the delta rule [225], gradient descent (GD) [120], and its non-deterministic variation known as stochastic gradient descent (SGD) [226], [227]. There's a discussion of backpropagation (BP) [117], [118], which can be used to obtain the gradients according to the defined error or cost function [228]. Rectified Linear Units (ReLU) [154] and a variation known as leaky ReLUs (LReLU) [217] are also detailed, along with a description of three different bootstrapping techniques.

A.2 The Perceptron Training Rule

A perceptron is a unit that inputs a $\mathbb{R}$-valued vector, computes a linear combination of these inputs, then outputs a 1 if the result is greater than some threshold and -1 otherwise, i.e. the output $\hat{y}\left(x_1, \ldots x_N\right)$ given inputs $\left(x_1, \ldots, x_N\right)$ is given by

$$\hat{y}(x_1, \ldots, x_N) = \begin{cases} 1 & \text{if } w_1 x_1 + \cdots + w_N x_N > 0 \\ -1 & \text{otherwise,} \end{cases} \tag{A.1}$$

or in vector notation

$$\hat{y}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} > 0 \\ -1 & \text{otherwise,} \end{cases} \tag{A.2}$$

where each $\boldsymbol{w} = (w_1, w_2, \ldots, w_N)$ is a $\mathbb{R}$-valued constant vector of weights that determines the contribution of each component input from $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$ to the perceptron output. Here $x_1 = 1$. The aforementioned threshold is in this case given by $(-w_1)$, such that the perceptron will output a 1 if the weighted combination of inputs $w_1 x_1 + \cdots + w_N x_N$ exceeds $-w_1$. Equation (A.2) can be written as

$$\hat{y}(\boldsymbol{x}) = \operatorname{sgn}(\boldsymbol{w} \cdot \boldsymbol{x}), \tag{A.3}$$

where

$$\operatorname{sgn}(z) = \begin{cases} 1 & \text{if } z > 0, \quad z \in \mathbb{R} \\ -1 & \text{otherwise.} \end{cases} \tag{A.4}$$

A perceptron is clearly based on a threshold function that is discontinuous in nature. What follows is an exposition of learning the weights for a single perceptron, i.e. determining a weight vector $\boldsymbol{w}$ that causes the perceptron to produce the correct ± 1 output for each of the N given training examples. Several algorithms exist that solve this problem. Here, only the perceptron rule and the delta rule are considered, both important as they provide the basis for learning networks of many units, and more complicated architectures build upon this.

A straightforward manner in which $\boldsymbol{w}$ may be learned is starting with random weights, then applying the perceptron in an iterative fashion to each of the N training example, by accordingly changing the weights whenever an example is misclassified. Repeatedly, this process is applied through the training examples as many times as required until the perceptron correctly classifies all N training examples. $\boldsymbol{w}$ Is modified at each step according to this perceptron training rule, each time adjusting the w_i associated with input x_i for $i = 1, \ldots, N$, according to the rule

$$w_i \leftarrow w_i + \Delta w_i, \tag{A.5}$$

where

$$\Delta w_i = \eta(y - \hat{y}) x_i, \tag{A.6}$$

with $y = d(\boldsymbol{x})$ symbolizing the target value for some unknown nonlinear function d, and η is a small constant (e.g. 0.01) known as the *learning rate*.

The perceptron rule may fail to converge and find an acceptable $\boldsymbol{w}$ in

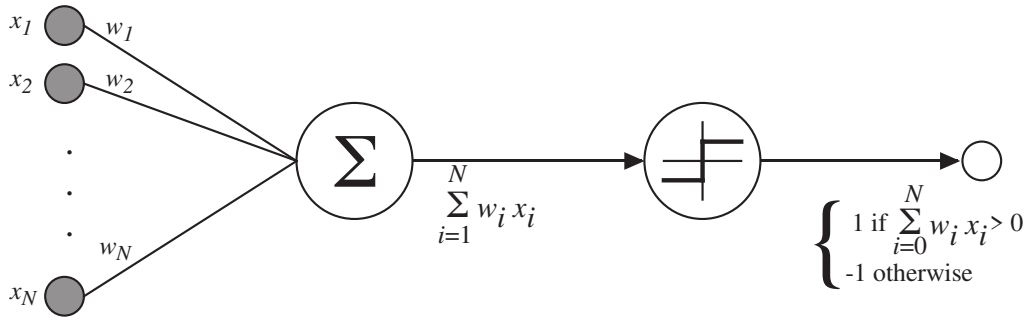


FIGURE A.1: A Perceptron.

the case that the training examples are not linearly separable. This possible failure is because convergence is not assured [229]. The delta rule overcomes this problem.

A.3 The Delta Training Rule and Gradient Descent

In the case where the training examples are not linearly separable, the delta rule converges toward a best-fit approximation to the desired output.

The delta rule's key idea is that it uses GD to search the space of all possible $\boldsymbol{w}$ to find the weights that best fit the training examples. In addition, GD provides the basis for the back propagation algorithm, which can learn networks with multiple interconnected units.

The delta rule is described by considering an example, where an unthresholded perceptron is trained, i.e. a linear unit corresponding to the first stage of a perceptron, without the threshold,

$$\hat{y}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}. \tag{A.7}$$

It essentially aims to to learn the weights that minimize, for instance, what is termed the half-squared error

$$E[\boldsymbol{w}] \equiv \frac{1}{2} \sum_{j \in J} (y_j - \hat{y}_j)^2, \tag{A.8}$$

where J is set of training examples. This specification of an error function has been selected here for convenience although there are many other possibilities. The error is a function of both the weights and the training examples,

but the latter is kept constant throughout training, so the dependence is not expressed in this formulation.

One can visualize the space of possible weight vectors and their associated E values as in Figure A.2, where a simple linear unit has 2 weights w_0 and w_1, and the vertical axis is indicative of the error with respect to a fixed set of training examples. In this simple $2-$dimensional case, because of how E is defined, the error surface takes a parabolic shape (dependant on the particular set of training examples) with a global minimum that may not be unique, i.e. there may be at least one global minimum, and no guarantee that one such global minimum may be found. In general, the problem has to be sufficiently well-posed so that finding at least a local minimum solves the problem.

GD search determines $\boldsymbol{w}$ that minimizes E. As previously stated, begin with an arbitrary initial weight vector, e.g. $\boldsymbol{w} = \mathbf{0}$, then take small steps for modification. At each of these steps, $\boldsymbol{w}$ is altered in the direction of steepest descent along the error surface, as illustrated in Figure A.2, up until the minimum error is reached.

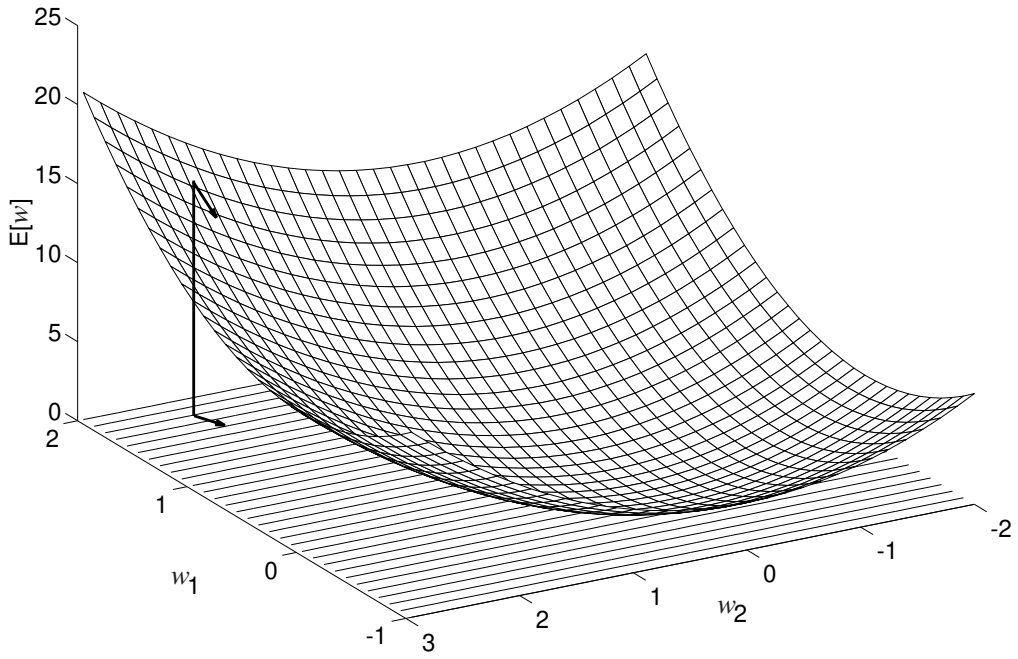


FIGURE A.2: Error Surface for Linear Unit with Two Weights.
The Vertical Line Indicates the Direction of Steepest Descent.

To be precise, the GD search determines $\boldsymbol{w}$ that minimizes E by arbitrarily initializing $\boldsymbol{w}$, then modifying it in small, repeated steps. At each step, $\boldsymbol{w}$ is altered in the direction producing the steepest descent along the error

surface. This process is repeated until at least some local minimum error is reached.

The direction is determined by computing the gradient of E with respect to $\boldsymbol{w}$

$$\nabla E\left[\boldsymbol{w}\right] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots, \frac{\partial E}{\partial w_N}\right]. \tag{A.9}$$

Equation (A.9), however, specifies the direction that produces the steepest increase in E. Hence, the concern is with $-\nabla E[\boldsymbol{w}]$, indicated by the arrow in Figure A.2. The delta rule is then given by

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \Delta \boldsymbol{w}, \tag{A.10}$$

where

$$\Delta \boldsymbol{w} = -\eta \nabla E[\boldsymbol{w}], \tag{A.11}$$

or in component form, substitute into Equation (A.5)

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}. \tag{A.12}$$

Note that differentiating E w.r.t. the weight w_i yields

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_j (y_j - \hat{y}_j)^2 \\
&= \frac{1}{2} \sum_j \frac{\partial}{\partial w_i} (y_j - \hat{y}_j)^2 \\
&= \frac{1}{2} \sum_j 2(y_j - \hat{y}_j) \frac{\partial}{\partial w_i} (y_j - \hat{y}_j) \\
&= \sum_j (y_j - \hat{y}_j) \frac{\partial}{\partial w_i} (y_j - \boldsymbol{w} \cdot \boldsymbol{x_j}) \\
\therefore \quad \frac{\partial E}{\partial w_i} &= \sum_j (y_j - \hat{y}_j)(-x_{i,j}),
\end{aligned} \tag{A.13}$$

where $x_{i,j}$ is the component input x_i for the training example j. $\frac{\partial E}{\partial w_i}$ Is now expressed in terms of the linear unit inputs $x_{i,j}$, outputs $\hat{y}_j$, and target values y_j that are associated with each of the training examples. Substituting Equation (A.13) into Equation (A.12) yields the delta weight update rule, or

Equation (A.5) with

$$\Delta w_i = \eta \sum_{j \in J} (y_j - \hat{y}_j) x_{i,j}. \tag{A.14}$$

One has to employ a learning rate η that is sufficiently small, otherwise the GD search may overstep the minimum in the error surface.

Implementation of gradient descent has some challenges [230]:

- convergence to a local minimum can be slow, sometimes requiring many thousands of GD steps,
- in the case where there are multiple local minima in the error surface, then convergence to the global minimum is not guaranteed, and
- this method is applicable only in instances where the error function is differentiable w.r.t. the weights.

A variation that circumvents these challenges is SGD.

A.4 Stochastic Gradient Descent

The difference here is instead of computing weight updates subsequent to summing over all the training examples in J, the SGD approximates this GD search by incrementally updating $\boldsymbol{w}$, after calculating the error for all of the individual examples. The modified training rule is such that as iterating through the training examples, $\boldsymbol{w}$ is updated using

$$\Delta w_i = \eta (y - \hat{y}) x_i, \tag{A.15}$$

where y, $\hat{y}$, and x_i respectively give the target value, unit output, and i^{th} input for the training example in question as previously defined. For implementation its simpler to use a different error function for each of the training examples j

$$E_j[\boldsymbol{w}] \equiv \frac{1}{2}(y_j - \hat{y}_j)^2. \tag{A.16}$$

As the SGD iterates over each training examples $j \in J$, it alters $\boldsymbol{w}$ in accordance with the gradient w.r.t. $\boldsymbol{w}$, $\nabla E_j[\boldsymbol{w}]$. An approximation to descending the gradient w.r.t. the original error function $E[\boldsymbol{w}]$ is then given by the sequence of $\boldsymbol{w}$ updates, when iterated over all the training examples. So by choosing a sufficiently small η, SGD can be made to approximate true GD arbitrarily closely [226], [227].

A.5 Backpropagation

The GD approach scales to the case of multilayer networks. What is needed for such multilayer networks is a unit that produces output that is a nonlinear function of its inputs, as well as differentiable w.r.t. said output. One solution is the sigmoid unit (Figure A.3). The sigmoid is similar to the perceptron but has the desirable property of being a smooth and differentiable threshold function. The output of this sigmoid is given by

$$\hat{y}(\boldsymbol{x}) = S\left(\boldsymbol{w} \cdot \boldsymbol{x}\right), \tag{A.17}$$

where

$$S(z) = \frac{1}{1+e^{-z}}, \quad z \in \mathbb{R}. \tag{A.18}$$

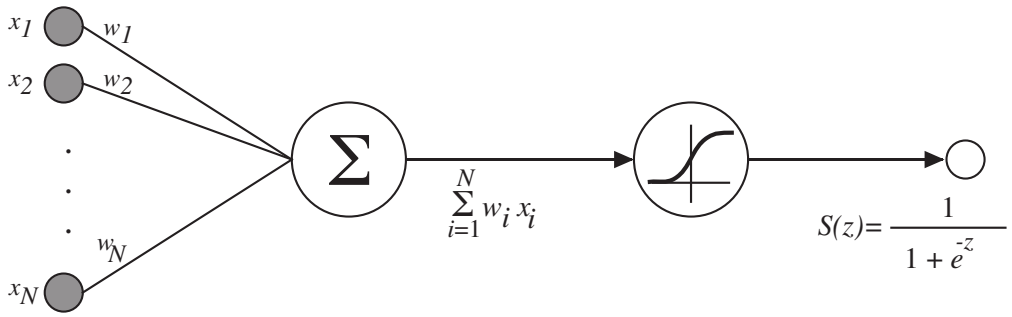


FIGURE A.3: A Sigmoidal Threshold Unit.

The sigmoid thus maps a very large input domain to a small range of outputs, hence it is also referred to as the squashing function of the unit, a property among the difficulties of using neural networks to detect anomalies. The sigmoid also has the useful property that its derivative is easily expressed in terms of its output,

$$\frac{dS(z)}{dz} = S(z)(1 - S(z)). \tag{A.19}$$

This derivative is used in the GD learning rule. The gradient decent rules can be derived to train a single sigmoid unit in a manner analogous to the perceptron case above, or multilayer networks of sigmoid units. The latter is known as BP.

Another notable non-linear function that has grown in popularity and is most prevalently applied to the output unit is the ReLU,

$$ReLU(z) = \max\{0, z\}. \tag{A.20}$$

The rectified units have been shown to significantly reduce training time, and speed up model convergence [231]. Furthermore, they are particularly useful for anomaly detection with deep learning as sigmoid activations saturate easier, i.e. once a sigmoid reaches either the left or right plateau, the derivative is very close to zero. On the other hand, ReLUs only saturate when the input is less than 0. And even this saturation can be eliminated by using a variation known as leaky ReLUs (LReLU) [217]. For very deep networks, saturation hampers learning so ReLUs are beneficial in this regard.

It is also useful to consider the effects of these different classes of activation functions for multivariate time series forecasting (TSF) and AD. For instance, the squashing function effect of sigmoidal activations may not be best suited for detecting anomalies, as they take even the furthest outliers and outputs them close to the bulk of observations. In contrast, ReLUs can grow arbitrarily large on the right.

A.6 Bootstrapping

Say there is a sample $\boldsymbol{x} = (x_1, \ldots, x_N)$ drawn from an unknown population F, and one is facing the problem of estimating a population statistic $\mathcal{S}$ using a sample statistic $\mathcal{S}_N$ as an approximation,

$$\mathcal{S}_N(\boldsymbol{x}) \sim \mathcal{S}(F). \tag{A.21}$$

Determining the confidence interval requires knowledge of the sampling distribution $\mathcal{G}_N$ of $\mathcal{S}_N(\boldsymbol{x}) - \mathcal{S}(F)$, i.e.

$$\mathcal{G}_N(\alpha) = \mathbb{P}\left(\mathcal{S}_N(\boldsymbol{x}) - \mathcal{S}(F) \leq \alpha\right) \forall \alpha. \tag{A.22}$$

An example is how the sample mean $\bar{x}_N = \frac{1}{N}\sum_{i=1}^{N} x_i$ may be a good approximate of the true population mean μ. Deriving the confidence interval for μ requires $\bar{x}_N - \mu$. In practice this is hardly ever known, hence Efron [232] proposed the bootstrap method.

To compute the confidence interval for μ, one would

- compute a bootstrap sample $\boldsymbol{x}^* = (x_1^*, \ldots, x_N^*)$ using the original sample, using sampling with replacement
- compute $h_j = \sqrt{N}\frac{\left(\boldsymbol{x}^{*(j)} - \bar{x}\right)}{\sigma_N}$ for $j = 1, \ldots, N$ bootstrap samples
- arrange the elements of $\{h\}_i$ in increasing order, and rename this permutation $\{\vartheta\}_i$ where $\vartheta_1 < \vartheta_2 < \cdots < \vartheta_N$ for $i = 1, \ldots, N$
- read off interval from the cumulative distribution function, e.g. the 95% confidence interval is such that $\bar{x} - \vartheta_a \frac{\sigma_N}{\sqrt{N}} \leq \mu \leq \bar{x} + \vartheta_b \frac{\sigma_N}{\sqrt{N}}$ where $\vartheta_a = 0.5N$ and $\vartheta_b = 0.95N$.

Repeating the above process a very large number of times (conventionally in the tens or hundreds of thousands), each time computing $\mathcal{S}_N(\boldsymbol{x}^*)$ an approximate distribution of the estimator $\mathcal{S}_N(\boldsymbol{x})$ is obtained.

A.6.1 Nonoverlapping Block Bootstrap

The above algorithm does not, however, preserve the sometimes-prevalent dependence structure between the observations. Resample methods for dependent data have been developed, most of which define blocks using segments of the data, such that the dependence structure within each block is retained. Versions are differentiated by how the blocks are constructed.

For example, the original time series $\boldsymbol{x} = (x_1, \ldots, x_N)$ may be divided into $q > 1$ blocks of consecutive observations of length $\mathcal{L}$, where the blocks are denoted by

$$B_i = \left(x_{(i-1)\mathcal{L}+1}, x_{(i-1)\mathcal{L}+2}, \ldots, x_{i\mathcal{L}}\right), i = 1, \ldots, q. \tag{A.23}$$

A random sample of $r \geq 1$ blocks, $B_1^*, \ldots, B_r^*$ is selected with replacement from $\{B_1, \ldots, B_q\}$. The $N = r \times l$ observations are joined, yielding the bootstrapped sample $\boldsymbol{x}^* = (x_1^*, \ldots, x_{\mathcal{L}}^*, \ldots, x_{(k-1)\mathcal{L}+1}^*, \ldots, x_N^*)$. This process is the Nonoverlapping Block Bootstrap, of Carlstein [233].

Other methods that improve upon the Efron [232] bootstrap include the Moving Block Bootstrap [234], the Circular Block Bootstrap [235] as well as the Stationary Block Bootstrap [236]. For a theoretical comparison see, for instance, Lahiri [237].

A.6.2 The Sieve Bootstrap

The sieve bootstrap [238] is different in that it fits a model first, then resamples from the residuals. Instead of fitting a fixed-dimensional model, one is able to approximate an infinite-dimensional non-parametric model using a

sequence of finite-dimensional parametric models. The bootstrapped sample is (conditionally) stationary.

Algorithm 5 The Sieve Bootstrap.

1: Begin by sampling $x_1, \ldots, x_N$ from the process $\{x_t\}$
2: Fit an autoregressive process, of increasing order Q as the sample size N increases, i.e. $Q = Q(N) = \mathcal{O}(N)$
3: Estimate the, say, p coefficients of Equation (A.28) $\hat{\Phi}_{1,N}, \ldots, \hat{\Phi}_{p,N}$ using, for instance Yale-Walker estimates; this step involves first subtracting the sample mean $\bar{x}$, yielding residuals

$$\hat{\zeta}_{t,N} = \sum_{i=0}^{P} \Phi_{i,N} \left(x_{t-i} - \bar{x} \right), \hat{\Phi}_{0,N} = 1, t = Q+1, \ldots, N \tag{A.24}$$

4: Resample based on this autoregressive approximation, and centre the residuals

$$\tilde{\zeta}_{t,N} = \hat{\zeta}_{t,N} - (N-Q)^{-1} \sum_{t=Q+1}^{N} \hat{\zeta}_{t,N}, t = Q+1, \ldots, N \tag{A.25}$$

5: The empirical cumulative distribution function of $\left\{ \tilde{\zeta}_{t,N} \right\}_{t=Q+1}^{N}$ is denoted by $\hat{F}_{\zeta,N}(\cdot) = (N-Q)^{-1} \sum_{t=Q+1}^{N} \mathbb{1}_{\left[\tilde{\zeta}_{t,N} \leq \cdot\right]}$, where $\mathbb{1}$ is the indicator function
6: Resample for any $t \in \mathbb{Z}$ the independently and identically distributed residual $\zeta_t^* \sim \hat{F}_{\zeta,N}$, and define the bootstrapped sample $\{x_t^*\}$ by the recursion

$$\sum_{i=0}^{Q} \Phi_{i,N} \left(x_{t-i}^* - \bar{x} \right) = \zeta_t^*, \hat{\Phi}_{0,N} = 1 \tag{A.26}$$

Let $\{x_t\}_{t \in \mathbb{Z}}$ be a real valued stationary process, with constant mean $\mathbb{E}(x_t) = \mu$. If $\{x_t\}$ is purely stochastic then $\{x_t - \mu\}$ can be written as a one-sided infinite order moving average process,

$$x_t - \mu = \sum_{j=0}^{\infty} \Psi \zeta_{t-j}, \Psi_0 = 1, \quad t \in \mathbb{Z}, \tag{A.27}$$

where $\{\zeta_t\}$ is a sequence of uncorrelated variables, with centred mean $\mathbb{E}(\zeta_t) = 0$, and $\sum_{j=0}^{\infty} \Psi_j^2 < \infty$. Under the assumption of invertibility of this process, $\{x_t\}$ can be represented as a one-sided infinite order autoregressive process,

$$\sum_{j=0}^{\infty} \Phi_j \left(x_{t-j} - \mu \right) = \zeta_t, \quad \Phi_0 = 1, \quad t \in \mathbb{Z}, \tag{A.28}$$

with $\sum_{j=0}^{\infty} \Phi_j^2 < \infty$. This autoregressive representation motivates an autoregressive approximation as a sieve for the stochastic process $\{x_t\}$. This Bühlmann [238] process is described in Algorithm 5 above. Equation (A.27) in Algorithm 5 employs the following theorem.

Theorem A.6.1 (Wold's Decomposition Theorem) *Every covariance-stationary time series can be written as a sum of two series, one deterministic and the other stochastic.*

Appendix B

Code Implementation Details

B.1 Introduction

This appendix serves to describe the software and hardware components utilized for implementation of the code related to the various experiments presented in this thesis. The motivation for not including such information in the main body of the thesis is because of technological advances. After a significant amount of time passes after writing this, hardware will be better, and software may likely be outdated. So to that end, this appendix serves as a self-contained reference for implementation details for the research studies presented herein. The general implementation details applicable to all the experiments include using a random seed of zero for training. Both the deterministic and non-deterministic models are trained once and then inferencing is conducted 35 times for the latter to ensure mitigation for their stochastic nature and variability in independent trial outputs. The details unique to each experiment are described in the sections that follow.

B.2 Deep Learning and Statistical-based Models for Anomaly Detection

The LSTM is implemented using the LSTM and Dense class of the Keras API v2.1.5 [239] for Python 3.7 using TensorFlow v1.15.0 framework as backend [240]. The code implementation for the CCN is written from scratch using Python 3.7, and ESN is implemented using PyTorch v0.3 [241] for Python 3.7. The naïve or benchmark methods are implemented using the Statsmodels API [242] for Python 3.7. All models are trained on an Intel Core i7-9750H processor with an Nvidia GeForce GTX 1650 GPU.

B.3 Statistics and Deep Learning-based Hybrid Model

All the models including the proposed model and the benchmarks are trained on a cluster of Intel Core i7-9750H processors each with an Nvidia GeForce GTX 1650 GPU and 4GB RAM. The uncertainty quantification is implemented using Tensorflow Probability [153].

B.4 Hybrid Model for Interpretable Anomaly Detection

InceptionTime, MC-DCNN and ResNet are implemented in Tensorflow, TapNet and MiniRocket in Pytorch, and the DTW techniques from scratch.

Bibliography

[1] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems 2*, Morgan Kaufmann, 1990, pp. 524–532. DOI: 10.1007/978-1-4899-7687-1_33.

[2] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, University of Cambridge, 2016.

[3] F. Y. Edgeworth, "Xli. on discordant observations," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 23, no. 143, pp. 364–375, 1957. DOI: 10.1080/14786448708628471.

[4] A. S. Alla S., *Practical Use Cases of Anomaly Detection.* Apress, Berkeley, CA., 2019. DOI: 10.1007/978-1-4842-5177-5_8.

[5] C. Chaparro and W. Eberle, "Detecting anomalies in mobile telecommunication networks using a graph based approach," in *The Twenty-Eighth International FLAIRS Conference*, 2015. [Online]. Available: https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS15/paper/view/10377/10278.

[6] C. Spence, L. Parra, and P. Sajda, "Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model," in *IEEE Workshop on mathematical methods in biomedical image analysis*, 2001, pp. 3–10. DOI: 10.1109/MMBIA.2001.991693.

[7] H. Bello-Salau, A. Aibinu, A. Onumanyi, E. Onwuka, J. Dukiya, and H. Ohize, "New road anomaly detection and characterization algorithm for autonomous vehicles," *Applied Computing and Informatics*, vol. 16, pp. 223–239, 2018. DOI: 10.1016/j.aci.2018.05.002.

[8] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, 15:1–15:58, Jul. 2009, ISSN: 0360-0300. DOI: 10.1145/1541880.1541882.

[9] M. J. C. Hu, "Application of the adaline system to weather forecasting," M.S. thesis, Technical Report 6775-1, Stanford Electronic Laboratories, Stanford CA, 1964.

[10] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35 –62, 1998, ISSN: 0169-2070. DOI: 10.1016/S0169-2070(97)00044-7.

[11] C. Hamzaçebi, D. Akay, and F. Kutay, "Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting," *Expert Systems with Applications*, vol. 36, no. 2, Part 2, pp. 3839 –3844, 2009, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2008.02.042.

[12] C. Robinson, B. Dilkina, J. Hubbs, *et al.*, "Machine learning approaches for estimating commercial building energy consumption," *Applied Energy*, vol. 208, pp. 889 –904, 2017, ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2017.09.060.

[13] C. Voyant, G. Notton, S. Kalogirou, *et al.*, "Machine learning methods for solar radiation forecasting: A review," *Renewable Energy*, vol. 105, pp. 569 –582, 2017, ISSN: 0960-1481. DOI: 10.1016/j.renene.2016.12.095.

[14] M. Fičura, "Forecasting stock market realized variance with echo state neural networks," *European Financial and Accounting Journal*, vol. 2017, no. 3, pp. 145–156, 2017. DOI: 10.18267/j.efaj.193.

[15] A. B. Kock and T. Teräsvirta, "Forecasting macroeconomic variables using neural network models and three automated model selection techniques," *Econometric Reviews*, vol. 35, no. 8-10, pp. 1753–1779, 2016. DOI: 10.1080/07474938.2015.1035163.

[16] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward," *PLOS ONE*, vol. 13, no. 3, pp. 1–26, 2018. DOI: 10.1371/journal.pone.0194889.

[17] M. Abdar, F. Pourpanah, S. Hussain, *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, vol. 76, pp. 243–297, 2021, ISSN: 1566-2535. DOI: 10.1016/j.inffus.2021.05.008.

[18] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, ser. ICML'16, vol. 48, New York, NY, USA: JMLR.org, 2016,

pp. 1050–1059. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045390.3045502.

[19] B. Lim, S. Ö. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021, ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2021.03.012.

[20] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, and G.-Z. Yang, "XAI—explainable artificial intelligence," *Science Robotics*, vol. 4, no. 37, 2019. DOI: 10.1126/scirobotics.aay7120.

[21] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Comput. Surv.*, vol. 51, no. 5, Aug. 2018, ISSN: 0360-0300. DOI: 10.1145/3236009.

[22] S. R. Beeram and S. Kuchibhotla, "Time series analysis on univariate and multivariate variables: A comprehensive survey," in *Communication Software and Networks*, S. C. Satapathy, V. Bhateja, M. Ramakrishna Murty, N. Gia Nhu, and J. Kotti, Eds., Singapore: Springer Singapore, 2021, pp. 119–126, ISBN: 978-981-15-5397-4. DOI: 10.1007/978-981-15-5397-4_13.

[23] H. Nguyen, K. Tran, S. Thomassey, and M. Hamad, "Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management," *International Journal of Information Management*, vol. 57, p. 102 282, 2021, ISSN: 0268-4012. DOI: 10.1016/j.ijinfomgt.2020.102282.

[24] U. Sivarajah, M. M. Kamal, Z. Irani, and V. Weerakkody, "Critical analysis of big data challenges and analytical methods," *Journal of Business Research*, vol. 70, pp. 263–286, 2017, ISSN: 0148-2963. DOI: 10.1016/j.jbusres.2016.08.001.

[25] F. Liu, S. Xue, J. Wu, *et al.*, "Deep learning for community detection: Progress, challenges and opportunities," English, in *Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI 2020*, C. Bessiere, Ed., ser. IJCAI International Joint Conference on Artificial Intelligence, 29th International Joint Conference on Artificial Intelligence, IJCAI 2020 ; Conference date: 01-01-2021, International Joint Conferences on Artificial Intelligence, 2020, pp. 4981–4987. DOI: 10.24963/ijcai.2020/693.

[26] D. Hawkins, *Identification of outliers*, ser. Monographs on applied probability and statistics. Chapman and Hall, 1980, ISBN: 041221900X. DOI: 10.1002/bimj.4710290215.

[27] H. Sun, Q. He, K. Liao, *et al.*, "Fast anomaly detection in multiple multi-dimensional data streams," in *2019 IEEE International Conference on Big Data (Big Data)*, Los Alamitos, CA, USA: IEEE Computer Society, Dec. 2019, pp. 1218–1223. DOI: 10.1109/BigData47090.2019.9006354.

[28] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS'99, Denver, CO: MIT Press, 1999, pp. 582–588.

[29] A. Carreńo, I. Inza, and J. Lozano, "Analyzing rare event, anomaly, novelty and outlier detection terms under the supervised classification framework," *Artificial Intelligence Review*, vol. 53, pp. 3575–3594, Jun. 2020. DOI: 10.1007/s10462-019-09771-y.

[30] A. Theofilatos, G. Yannis, P. Kopelias, and F. Papadimitriou, "Predicting road accidents: A rare-events modeling approach," *Transportation Research Procedia*, vol. 14, pp. 3399–3405, 2016, Transport Research Arena TRA2016, ISSN: 2352-1465. DOI: 10.1016/j.trpro.2016.05.293.

[31] A. Carreño, I. Inza, and J. A. Lozano, "Analyzing rare event, anomaly, novelty and outlier detection terms under the supervised classification framework," *Artificial Intelligence Review*, vol. 53, no. 5, pp. 3575–3594, 2020. DOI: 10.1007/s10462-019-09771-y.

[32] C. C. Aggarwal, "Probabilistic and statistical models for outlier detection," in *Outlier Analysis*, New York, NY: Springer, 2013, ch. 2, pp. 41–74. DOI: 10.1007/978-1-4614-6396-2_2.

[33] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 Competition: 100,000 time series and 61 forecasting methods," *International Journal of Forecasting*, vol. 36, no. 1, pp. 54–74, 2020. DOI: 10.1016/j.ijforecast.2019.04.014.

[34] S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," *International Journal of Forecasting*, vol. 36, no. 1, pp. 75–85, 2020, ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2019.03.017.

[35] J. Li, W. Pedrycz, and I. Jamal, "Multivariate time series anomaly detection: A framework of hidden markov models," *Applied Soft Computing*, vol. 60, pp. 229 –240, 2017, ISSN: 1568-4946. DOI: 10.1016/j.asoc.2017.06.035.

[36] X. Su, S. Xue, F. Liu, *et al.*, *A comprehensive survey on community detection with deep learning*, 2021. arXiv: 2105.12584 [cs.SI].

[37] H. Brink, J. Richards, and M. Fetherolf, "Scaling machine-learning workflows," in *Real-World Machine Learning*, New York, NY: Manning Publications Co., 2016, ch. 9. [Online]. Available: https://livebook.manning.com/book/real-world-machine-learning/chapter-9/8.

[38] B. Rad, F. Song, V. Jacob, and Y. Diao, "Explainable anomaly detection on high-dimensional time series data," in *Proceedings of the 15th ACM International Conference on Distributed and Event-Based Systems*, ser. DEBS '21, Virtual Event, Italy: Association for Computing Machinery, 2021, 2–14, ISBN: 9781450385558. DOI: 10.1145/3465480.3468292.

[39] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020, ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2019.07.001.

[40] T. Mathonsi and T. L. van Zyl, "Prediction interval construction for multivariate point forecasts using deep learning," in *2020 7th International Conference on Soft Computing Machine Intelligence (ISCMI)*, 2020, pp. 88–95. DOI: 10.1109/ISCMI51676.2020.9311603.

[41] G. Pang, C. Ding, C. Shen, and A. van den Hengel, *Explainable deep few-shot anomaly detection with deviation networks*, 2021. arXiv: 2108.00462 [cs.CV].

[42] Y. Himeur, K. Ghanem, A. Alsalemi, F. Bensaali, and A. Amira, "Artificial intelligence based anomaly detection of energy consumption in buildings: A review, current trends and new perspectives," *Applied Energy*, vol. 287, p. 116 601, 2021, ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2021.116601.

[43] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International joint conference on neural networks (IJCNN)*, IEEE, 2017, pp. 1578–1585. DOI: 10.1109/IJCNN.2017.7966039.

[44] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," *Neural Networks*, vol. 116, pp. 237–245, 2019, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2019.04.014.

[45] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," in *PloS One*, 2015. DOI: doi.org/10.1371/journal.pone.0118432.

[46] B. R. Kiran, D. M. Thomas, and R. Parakkal, "An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos," *Journal of Imaging*, vol. 4, no. 2, 2018, ISSN: 2313-433X. DOI: 10.3390/jimaging4020036.

[47] O. Serradilla, E. Zugasti, J. Ramirez de Okariz, J. Rodriguez, and U. Zurutuza, "Adaptable and explainable predictive maintenance: Semi-supervised deep learning for anomaly detection and diagnosis in press machine data," *Applied Sciences*, vol. 11, no. 16, 2021, ISSN: 2076-3417. DOI: 10.3390/app11167376.

[48] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection," *ACM Computing Surveys*, vol. 54, no. 2, 1–38, Apr. 2021, ISSN: 1557-7341. DOI: 10.1145/3439950.

[49] M. Munir, M. A. Chattha, A. Dengel, and S. Ahmed, "A comparative analysis of traditional and deep learning-based anomaly detection methods for streaming data.," in *ICMLA*, M. A. Wani, T. M. Khoshgoftaar, D. Wang, H. Wang, and N. Seliya, Eds., IEEE, 2019, pp. 561–566. DOI: 10.1109/ICMLA.2019.00105.

[50] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11491.

[51] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017, Online Real-Time Learning Strategies for Data Streams, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.04.070.

[52] A. Barbado, O. Corcho, and R. Benjamins, "Rule extraction in unsupervised anomaly detection for model explainability: Application to oneclass svm," *Expert Systems with Applications*, vol. 189, p. 116 100, 2022, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2021.116100.

[53] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, *The m5 accuracy competition: Results, findings and conclusions*, 2020. [Online]. Available: https://www.researchgate.net/publication/344487258_The_M5_Accuracy_competition_Results_findings_and_conclusions.

[54] S. Makridakis, E. Spiliotis, V. Assimakopoulos, *et al.*, *The m5 uncertainty competition: Results, findings and conclusions*, 2020. [Online]. Available: https://www.researchgate.net/publication/346493740_The_M5_Uncertainty_competition_Results_findings_and_conclusions.

[55] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, *et al.*, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, 2021. DOI: 10.1109/JPROC.2021.3052449.

[56] C. Bharathi Priya and N. Arulanand, "Univariate and multivariate models for short-term wind speed forecasting," *Materials Today: Proceedings*, 2021, ISSN: 2214-7853. DOI: 10.1016/j.matpr.2020.12.1090.

[57] L. Bertossi, J. Li, M. Schleich, D. Suciu, and Z. Vagena, "Causality-based explanation of classification outcomes," in *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, ser. DEEM'20, New York, NY, USA: Association for Computing Machinery, 2020, ISBN: 9781450380232. DOI: 10.1145/3399579.3399865.

[58] S. Makridakis and E. Spiliotis, "The M5 Competition and the Future of Human Expertise in Forecasting," *Foresight: The International Journal of Applied Forecasting*, no. 60, pp. 33–37, 2021. [Online]. Available: https://ideas.repec.org/a/for/ijafaa/y2021i60p33-37.html.

[59] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m5 competition: Background, organization, and implementation," *International Journal of Forecasting*, 2021, ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2021.07.007.

[60] T. Mathonsi and T. L. van Zyl, "Multivariate anomaly detection based on prediction intervals constructed using deep learning," *Neural Computing and Applications*, pp. 1–15, 2022. DOI: 10.1007/s00521-021-06697-x.

[61] T. Mathonsi and T. L. van Zyl, "A statistics and deep learning hybrid method for multivariate time series forecasting and mortality modeling," *Forecasting*, vol. 4, no. 1, pp. 1–25, 2022, ISSN: 2571-9394. DOI: 10.3390/forecast4010001.

[62] T. Mathonsi and T. L. van Zyl, *Statistics and deep learning-based hybrid model for interpretable anomaly detection*, 2022. arXiv: 2202.12720 [cs.LG].

[63] S. X. Chen, *UCI machine learning repository Beijing PM2.5 data set*, http://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data, [Online; accessed 28-February-2022], 2014.

[64] G. Hebrail and A. Berard, *UCI machine learning repository individual household electric power consumption data set*, https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption, [Online; accessed 28-February-2022], 2012.

[65] E. Watanabe, R. Stephan, and M. Aredes, "New concepts of instantaneous active and reactive powers in electrical systems with generic loads," *IEEE Transactions on Power Delivery*, vol. 8, no. 2, pp. 697–703, 1993. DOI: 10.1109/61.216877.

[66] H. Fanaee-T and J. Gama, *UCI machine learning repository bike sharing data set*, https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset, [Online; accessed 05-June-2020], 2012.

[67] J Lee, H Qiu, G Yu, J Lin, and Rexnord Technical Services, *NASA Ames prognostics data repository*, https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/#bearing, [Online; accessed 28-February-2022], 2007.

[68] A. Prosvirin, M. M. M. Islam, C. Kim, and J. Kim, "Fault prediction of rolling element bearings using one class least squares SVM," in *Proceedings of the Engineering and Arts Society in Korea*, vol. 15, Dec. 2017, pp. 93–96. [Online]. Available: https://www.researchgate.net/publication/321906715_Fault_Prediction_of_Rolling_Element_Bearings_Using_One_Class_Least_Squares_SVM.

[69] N Laptev and S Amizadeh, *Yahoo webscope benchmark dataset*, http://webscope.sandbox.yahoo.com/catalog.php, [Online; accessed 28-February-2022], 2015.

[70] M. Thill, W. Konen, and T. Bäck, "Online anomaly detection on the webscope s5 dataset: A comparative study," in *2017 Evolving and Adaptive Intelligent Systems (EAIS)*, 2017, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/7954844.

[71] E. Mathieu, H. Ritchie, E. Ortiz-Ospina, *et al.*, "A global database of COVID-19 vaccinations," *medRxiv*, 2021. DOI: 10.1101/2021.03.22.21254100.

[72] J. Hasell, E. Mathieu, D. Beltekian, *et al.*, "A cross-country database of covid-19 testing," *Scientific Data*, vol. 7, pp. 345 –347, Oct. 2020. DOI: 10.1038/s41597-020-00688-8.

[73] R. Chandra, A. Jain, and D. Singh Chauhan, "Deep learning via lstm models for covid-19 infection forecasting in india," *PloS one*, vol. 17, no. 1, e0262708, 2022. DOI: 10.1371/journal.pone.0262708.

[74] X. Zheng, N. Xu, D. Wu, *et al.*, *PSML: A Multi-scale Time-series Dataset for Machine Learning in Decarbonized Energy Grids (Dataset)*, version 0.2, Aug. 2021. DOI: 10.5281/zenodo.5130612.

[75] M. Ma, Z. Yin, S. Zhang, *et al.*, "Diagnosing root causes of intermittent slow queries in cloud databases," *Proceedings of the VLDB Endowment*, vol. 13, no. 8, pp. 1176–1189, 2020. [Online]. Available: http://nkcs.iops.ai/wp-content/uploads/2020/04/paper-VLDB20-iSQUAD.pdf.

[76] Ontario Agency for Health Protection and Promotion (Public Health Ontario), *Ontario COVID-19 data tool*, version 1.0, Aug. 2021. [Online]. Available: https://www.publichealthontario.ca/en/data-and-analysis/infectious-disease/covid-19-data-surveillance/covid-19-data-tool?tab=summary.

[77] Nordpool Group, *Nordpool Power Systems Market Data*, version 2.1, Aug. 2021. [Online]. Available: https://www.nordpoolgroup.com/en/Market-data1/Power-system-data/Consumption1/Consumption-prognosis/ALL/Hourly/?view=table.

[78] M. Gulati and P. Arjunan, "LEAD 1.0: A Large-scale Annotated Dataset for Energy Anomaly Detection in Commercial Buildings," *arXiv preprint arXiv:2203.17256*, 2022. DOI: 10.48550/ARXIV.2203.17256.

[79] C. Miller, P. Arjunan, A. Kathirgamanathan, *et al.*, "The ASHRAE Great Energy Predictor III competition: Overview and results," *Science and Technology for the Built Environment*, vol. 26, no. 10, pp. 1427–1447, 2020. DOI: 10.1080/23744731.2020.1795514.

[80] C. Miller, A. Kathirgamanathan, B. Picchetti, *et al.*, "The building data genome project 2, energy meter data from the ASHRAE Great Energy Predictor III competition," *Scientific Data*, vol. 7, no. 368, 2020. DOI: 10.1038/s41597-020-00712-x.

[81] S. Makridakis and M. Hibon, "The m3-competition: Results, conclusions and implications," English, *International Journal of Forecasting*, vol. 16, no. 4, pp. 451–476, 2000.

[82] A. Koehler, "Commentaries on the m3-competition," *International Journal of Forecasting*, vol. 17, no. 4, pp. 537–584, 2001, ISSN: 0169-2070. DOI: 10.1016/S0169-2070(01)00119-4.

[83] P. Goodwin and R. Lawton, "On the asymmetry of the symmetric mape," *International Journal of Forecasting*, vol. 15, no. 4, pp. 405 –408, 1999, ISSN: 0169-2070. DOI: 10.1016/S0169-2070(99)00007-2.

[84] "Mean absolute error," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2010, pp. 652–652, ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_525.

[85] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," English, *Journal of the American Statistical Association*, vol. 102, no. 477, pp. 359–378, 2007.

[86] T. S. Buda, H. Assem, and L. Xu, "Ade: An ensemble approach for early anomaly detection," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, pp. 442–448. [Online]. Available: https://www.researchgate.net/publication/318664756_ADE_An_ensemble_approach_for_early_Anomaly_Detection.

[87] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997, ISSN: 0031-3203. DOI: 10.1016/S0031-3203(96)00142-2.

[88] B. STUDENT, "The probable error of a mean.," *Biometrika*, vol. 6, no. 1, pp. 1–25, 1908. [Online]. Available: https://bayes.wustl.edu/Manual/Student.pdf.

[89] F. Diebold and R. Mariano, "Comparing predictive accuracy," *Journal of Business and Economic Statistics*, vol. 13, no. 3, pp. 253–63, 1995. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.454.4490&rep=rep1&type=pdf.

[90] "Mean absolute percentage error," in *Encyclopedia of Production and Manufacturing Management*, P. M. Swamidass, Ed. Boston, MA: Springer US, 2000, pp. 462–462, ISBN: 978-1-4020-0612-8. DOI: 10.1007/1-4020-0612-8_580.

[91] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2015, pp. 89–94. [Online]. Available: https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2015-56.pdf.

[92] S. Chauhan and L. Vig, "Anomaly detection in ecg time signals via deep long short-term memory networks," Oct. 2015, pp. 1–7. DOI: 10.1109/DSAA.2015.7344872.

[93] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *CoRR*, vol. abs/1607.00148, 2016. [Online]. Available: http://arxiv.org/abs/1607.00148.

[94] J. P. Assendorp, "Deep learning for anomaly detection in multivariate time series data," ger, M.S. thesis, HAW Hamburg, Berliner Tor 5, 20099 Hamburg, 2017.

[95] M. Sölch, J. Bayer, M. Ludersdorfer, and P. van der Smagt, "Variational inference for on-line anomaly detection in high-dimensional time series.," *CoRR*, vol. abs/1602.07109, 2016.

[96] D. P. Kingma and M. Welling, "Auto-encoding variational bayes.," *CoRR*, vol. abs/1312.6114, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1312.html#KingmaW13.

[97] J. Bayer and C. Osendorfer, "Learning stochastic recurrent networks," in *Workshop on Advances in Variational Inference, Neural Information Processing Systems 2014*, 2014. [Online]. Available: http://arxiv.org/abs/1411.7610.

[98] N. N. Thi, V. L. Cao, and N. Le-Khac, "One-class collective anomaly detection based on long short-term memory recurrent neural networks," pp. 73–85, 2017. DOI: 10.1007/978-3-662-56266-6_4.

[99] L. Zhu and N. Laptev, "Deep and confident prediction for time series at uber," in *2017 IEEE International Conference on Data Mining Workshops ICDMW*, 2017, pp. 103–110. DOI: 10.1109/ICDMW.2017.19.

[100] V. Mullachery, A. Khera, and A. Husain, "Bayesian neural networks," *CoRR*, vol. abs/1801.07710, 2018. [Online]. Available: http://arxiv.org/abs/1801.07710.

[101] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[102] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," pp. 146–157, 2017. DOI: 10.1007/978-3-319-59050-9_12.

[103] X. Ma, J. Wu, S. Xue, *et al.*, "A comprehensive survey on graph anomaly detection with deep learning," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021. DOI: 10.1109/TKDE.2021.3118815.

[104] S. Makridakis, R. J. Hyndman, and F. Petropoulos, "Forecasting in social settings: The state of the art," *International Journal of Forecasting*, vol. 36, no. 1, pp. 15 –28, 2020, ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2019.05.011.

[105] D. Kwon, H. Kim, J. Kim, S. Suh, I. Kim, and K. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, vol. 22, Jan. 2019. DOI: 10.1007/s10586-017-1117-8.

[106] S. Thudumu, P. Branch, J. Jin, and J. Singh, "A comprehensive survey of anomaly detection techniques for high dimensional big data," *Journal of Big Data*, vol. 42, pp. 7–37, 2020. DOI: 10.1186/s40537-020-00320-x.

[107] A. O. Adewumi and A. A. Akinyelu, "A survey of machine-learning and nature-inspired based credit card fraud detection techniques, international journal of system assurance engineering and management," *International Journal of System Assurance Engineering and Management*, International Journal of System Assurance Engineering and Management, vol. 8, no. 2, pp. 937–953, 2016, ISSN: 0976-4348, 0975-6809. DOI: 10.1007/s13198-016-0551-y.

[108] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research (JMLR)*, vol. 12, pp. 2825–2830, 2011.

[109] O. Claveria, E. Monte, and S. Torra, "Effects of removing the trend and the seasonal component on the forecasting performance of artificial neural network techniques," Research Institute of Applied Economics, Spain, Working Paper 2015/031/16, 2015. DOI: 10.13140/RG.2.2.27306.21447.

[110] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "STL: A seasonal-trend decomposition procedure based on LOESS (with discussion)," *Journal of Official Statistics*, vol. 6, pp. 3–73, 1990.

[111] X. Zhu, C. Vondrick, C. C. Fowlkes, and D. Ramanan, "Do we need more training data?" *International Journal of Computer Vision*, vol. 119, no. 1, pp. 76–92, 2016. DOI: 10.1007/s11263-015-0812-2.

[112] D. E. Matthews, "Multiple linear regression," in *Encyclopedia of Biostatistics*. American Cancer Society, 2005, ch. 5, pp. 119–133. DOI: 10.1002/0470011815.b2a09033.

[113] E. J. Hannan and M. Deistler, "The statistical theory of linear systems," *Econometric Theory*, vol. 8, no. 1, pp. 135–143, 1992. DOI: 10.1017/S026646660001080X.

[114] N. Arunraj, D. Ahrens, and M. Fernandes, "Application of SARIMAX model to forecast daily sales in food retail industry," *International Journal of Operations Research and Information Systems*, vol. 7, pp. 1–21, Apr. 2016. DOI: 10.4018/IJORIS.2016040101.

[115] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989, ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.2.270.

[116] H. Jaeger, "The "Echo State" approach to analysing and training recurrent neural networks," GMD - German National Research Institute for Computer Science, GMD Report 148, 2001. [Online]. Available: http://www.faculty.jacobs-university.de/hjaeger/pubs/EchoStatesTechRep.pdf.

[117] S. Linnainmaa, "The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors," M.S. thesis, Univ. Helsinki, 1970.

[118] ——, "Taylor expansion of the accumulated rounding error," *BIT Numerical Mathematics*, vol. 16, no. 2, pp. 146–160, 1976.

[119] S. Hochreiter, Y. Bengio, and P. Frasconi, "Gradient flow in recurrent nets: The difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Neural Networks*, K. S. and K. J., Eds., IEEE Press, 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.7321&rep=rep1&type=pdf.

[120] L. A. Cauchy, "Methode générale pour la résolution des systemes d'éqations simultanées," *Comptes Rendus*, vol. 2, no. 25, p. 536, 1847.

[121] S. E. Fahlman, "Faster-learning variations on back-propagation: An empirical study," in *Proceedings of the 1988 Connectionist Models Summer School*, D. S. Touretzky, G. E. Hinton, and T. J. Sejnowski, Eds., San Francisco, CA: Morgan Kaufmann, 1989, pp. 38–51.

[122] L. C. Jain and L. R. Medsker, *Recurrent Neural Networks: Design and Applications*, 1st. Boca Raton, FL, USA: CRC Press, Inc., 1999, ISBN: 0849371813.

[123] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8), 2014*, 2014".

[124] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Deep Learning and Representation Learning Workshop*, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555.

[125] R. Józefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, Lille, France: JMLR.org, 2015, pp. 2342–2350.

[126] W. Maass, T. Natschläger, and H. Markram, "Real-time Computing without Stable States: A New Framework for Neural Computation Based on Perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[127] J. G. Ulloa, *Applied Biomedical Engineering Using Artificial Intelligence and Cognitive Models*. Elsevier Science & Technology, 2021.

[128] A. C. Davison and D. V. Hinkley, *Bootstrap Methods and Their Application*. USA: Cambridge University Press, 2013, ISBN: 0511802846. [Online]. Available: https://dl.acm.org/doi/book/10.5555/2556084.

[129] T. Hesterberg, *What teachers should know about the bootstrap: Resampling in the undergraduate statistics curriculum*, 2015. DOI: 10.1080/00031305.2015.1089789.

[130] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 3rd International Conference for Learning Representations, San Diego, 2015, 2014.

[131] W. J. Dixon and F. J. Massey, *Introduction to statistical analysis*, English, 3d ed. McGraw-Hill New York, 1968, vii, 638 p. [Online]. Available: https://nla.gov.au/nla.cat-vn645066.

[132] C. F. Coimbra and H. T. Pedro, "Chapter 15 - Stochastic-learning Methods," in *Solar Energy Forecasting and Resource Assessment*, J. Kleissl, Ed., Boston: Academic Press, 2013, pp. 383–406, ISBN: 978-0-12-397177-7. DOI: 10.1016/B978-0-12-397177-7.00015-2.

[133] R. J. Hyndman, A. B. Koehler, R. D. Snyder, and S. Grose, "A state space framework for automatic forecasting using exponential smoothing methods," *International Journal of Forecasting*, vol. 18, no. 3, pp. 439 –454, 2002, ISSN: 0169-2070. DOI: 10.1016/S0169-2070(01)00110-8.

[134] A. Redd, K. Khin, and A. Marini, "Fast ES-RNN: A GPU implementation of the ES-RNN algorithm," *CoRR*, vol. abs/1907.03329, 2019. [Online]. Available: http://arxiv.org/abs/1907.03329.

[135] I. Olkin and A. Sampson, "Multivariate analysis: Overview," in *International Encyclopedia of the Social and Behavioral Sciences*, N. J. Smelser and P. B. Baltes, Eds., Oxford: Pergamon, 2001, pp. 10 240–10 247, ISBN: 978-0-08-043076-8. DOI: 10.1016/B0-08-043076-7/00472-1.

[136] I. Kırbaş, A. Sözen, A. D. Tuncer, and F. Ş. Kazancioğlu, "Comparative analysis and forecasting of covid-19 cases in various european countries with arima, narnn and lstm approaches," *Chaos, Solitons and Fractals*, vol. 138, p. 110 015, 2020, ISSN: 0960-0779. DOI: 10.1016/j.chaos.2020.110015.

[137] M. Ibrahim, S. Jemei, G. Wimmer, and D. Hissel, "Nonlinear autoregressive neural network in an energy management strategy for battery or ultra-capacitor hybrid electrical vehicles," *Electric Power Systems Research*, vol. 136, pp. 262–269, 2016, ISSN: 0378-7796. DOI: 10.1016/j.epsr.2016.03.005.

[138] V. K. R. Chimmula and L. Zhang, "Time series forecasting of covid-19 transmission in canada using lstm networks," *Chaos, Solitons and Fractals*, vol. 135, p. 109 864, 2020, ISSN: 0960-0779. DOI: 10.1016/j.chaos.2020.109864.

[139] F. Shahid, A. Zameer, and M. Muneeb, "Predictions for covid-19 with deep learning models of lstm, gru and bi-lstm," *Chaos, Solitons and Fractals*, vol. 140, p. 110 212, 2020, ISSN: 0960-0779. DOI: 10.1016/j.chaos.2020.110212.

[140] M. Awad and R. Khanna, "Support vector regression," in *Efficient learning machines*, Springer, 2015, pp. 67–80. DOI: 10.1007/978-1-4302-5990-9_4.

[141] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, *N-beats: Neural basis expansion analysis for interpretable time series forecasting*, 2020. [Online]. Available: https://arxiv.org/pdf/1905.10437.pdf.

[142] K. G. Olivares, C. Challu, G. Marcjasz, R. Weron, and A. Dubrawski, "Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with nbeatsx," *CoRR*, vol. abs/2104.05522, 2021. [Online]. Available: https://arxiv.org/abs/2104.05522.

[143] R. H. Jones, "Exponential smoothing for multivariate time series," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 28, no. 1, pp. 241–251, 1966, ISSN: 00359246. [Online]. Available: http://www.jstor.org/stable/2984290.

[144] P. G. Enns, J. A. Machak, W. A. Spivey, and W. J. Wrobleski, "Forecasting applications of an adaptive multiple exponential smoothing model," *Manage. Sci.*, vol. 28, no. 9, 1035–1044, Sep. 1982, ISSN: 0025-1909. DOI: 10.1287/mnsc.28.9.1035.

[145] D. W. Trigg and A. G. Leach, "Exponential smoothing with an adaptive response rate," *OR*, vol. 18, no. 1, pp. 53–59, 1967, ISSN: 14732858. DOI: 10.2307/3010768.

[146] A. C. Harvey, "Analysis and generalisation of a multivariate exponential smoothing model," *Manage. Sci.*, vol. 32, no. 3, 374–380, Mar. 1986, ISSN: 0025-1909. DOI: 10.1287/mnsc.32.3.374.

[147] D. Pfeffermann and J. Allon, "Multivariate exponential smoothing: Method and practice," *International Journal of Forecasting*, vol. 5, no. 1, pp. 83–98, 1989, ISSN: 0169-2070. DOI: 10.1016/0169-2070(89)90066-6.

[148] F. Tan, "Regression analysis and prediction using LSTM model and machine learning methods," *Journal of Physics: Conference Series*, vol. 1982, no. 1, pp. 012–013, Jul. 2021. DOI: 10.1088/1742-6596/1982/1/012013.

[149] Y. Hu, F. O'Donncha, P. Palmes, M. Burke, R. Filgueira, and J. Grant, *A spatio-temporal lstm model to forecast across multiple temporal and spatial scales*, 2021. [Online]. Available: https://arxiv.org/pdf/2108.11875.pdf.

[150] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse, "Flipout: Efficient pseudo-independent weight perturbations on mini-batches," in *International Conference on Learning Representations*, 2018.

[151] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, Lille, France: JMLR.org, 2015, 1613–1622.

[152] J. M. Joyce, "Kullback-leibler divergence," in *International Encyclopedia of Statistical Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 720–722, ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_327.

[153] J. V. Dillon, I. Langmore, D. Tran, *et al.*, "Tensorflow distributions," *ArXiv*, vol. abs/1711.10604, 2017. [Online]. Available: https://arxiv.org/pdf/1711.10604.pdf.

[154] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, pp. 947–951, 2000.

[155] G. Ke, Q. Meng, T. Finley, *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, pp. 3146–3154, 2017.

[156] J. Lever, M. Krzywinski, and N. Altman, "Points of significance: Model selection and overfitting," English (US), *Nature Methods*, vol. 13, no. 9, pp. 703–704, Aug. 2016, ISSN: 1548-7091. DOI: 10.1038/nmeth.3968.

[157] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Second International Symposium on Information Theory*, B. N. Petrov and F. Csaki, Eds., Budapest: Akadémiai Kiado, 1973, pp. 267–281.

[158] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980.

[159] D. F. Silva, R. Giusti, E. Keogh, and G. E. Batista, "Speeding up similarity search under dynamic time warping by pruning unpromising alignments," *Data Mining and Knowledge Discovery*, vol. 32, no. 4, pp. 988–1016, 2018. DOI: 10.1007/s10618-018-0557-y.

[160] S. Chakraborty, R. Tomsett, R. Raghavendra, *et al.*, "Interpretability of deep learning models: A survey of results," in *2017 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, Internet of people and smart city innovation*, IEEE, 2017, pp. 1–6. DOI: 10.1109/UIC-ATC.2017.8397411.

[161] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.," *Queue*, vol. 16, no. 3, pp. 31–57, 2018. [Online]. Available: https://queue.acm.org/detail.cfm?id=3241340.

[162] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller, "Unmasking clever hans predictors and assessing what machines really learn," *Nature Communications*, vol. 10, p. 1096, 2019. DOI: 10.1038/s41467-019-08987-4.

[163] J. R. Kauffmann, L. Ruff, G. Montavon, and K. Müller, "The clever hans effect in anomaly detection," *CoRR*, vol. abs/2006.10609, 2020. [Online]. Available: https://arxiv.org/abs/2006.10609.

[164] V. Jacob, F. Song, A. Stiegler, B. Rad, Y. Diao, and N. Tatbul, "Exathlon: A Benchmark for Explainable Anomaly Detection over Time Series," *Proceedings of the VLDB Endowment (PVLDB)*, Jul. 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03381732.

[165] Q. P. Nguyen, K. W. Lim, D. M. Divakaran, K. H. Low, and M. C. Chan, "Gee: A gradient-based explainable variational autoencoder for network anomaly detection," in *2019 IEEE Conference on Communications and Network Security (CNS)*, IEEE, 2019, pp. 91–99. DOI: 10.1109/CNS.2019.8802833.

[166] M. Carletti, C. Masiero, A. Beghi, and G. A. Susto, "Explainable machine learning in industry 4.0: Evaluating feature importance in anomaly detection to enable root cause analysis," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, IEEE, 2019, pp. 21–26. DOI: 10.1109/SMC.2019.8913901.

[167] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *2008 Eighth IEEE International Conference on Data Mining*, IEEE, 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[168] V. Jeyakumar, O. Madani, A. Parandeh, A. Kulshreshtha, W. Zeng, and N. Yadav, "Explainit! – a declarative root-cause analysis engine for time series data," in *Proceedings of the 2019 International Conference on Management of Data*, ser. SIGMOD '19, New York, NY, USA: Association for Computing Machinery, 2019, 333–348. DOI: 10.1145/3299869.3314048.

[169] A. Westerski, R. Kanagasabai, E. Shaham, A. Narayanan, J. Wong, and M. Singh, "Explainable anomaly detection for procurement fraud identification—lessons from practical deployments," *International Transactions in Operational Research*, vol. 28, no. 6, pp. 3276–3302, 2021. DOI: 10.1111/itor.12968.

[170] P. Liznerski, L. Ruff, R. A. Vandermeulen, B. J. Franks, M. Kloft, and K.-R. Müller, "Explainable deep one-class classification," *arXiv preprint arXiv:2007.01760*, 2021. [Online]. Available: https://openreview.net/pdf?id=A5VV3UyIQz.

[171] C. Wu, S. Shao, C. Tunc, P. Satam, and S. Hariri, "An explainable and efficient deep learning framework for video anomaly detection," *Cluster Computing*, 2021. DOI: 10.1007/s10586-021-03439-5.

[172] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019. [Online]. Available: https://arxiv.org/pdf/1901.03407.pdf.

[173] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (xai): Toward medical xai," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 11, pp. 4793–4813, 2021. DOI: 10.1109/TNNLS.2020.3027314.

[174] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2017/papers/Selvaraju_Grad-CAM_Visual_Explanations_ICCV_2017_paper.pdf.

[175] M. Mitchell, S. Wu, A. Zaldivar, *et al.*, "Model cards for model reporting," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, ser. FAT* '19, New York, NY, USA: Association for Computing Machinery, 2019, 220–229, ISBN: 9781450361255. DOI: 10.1145/3287560.3287596.

[176] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020, ISSN: 1566-2535. DOI: 10.1016/j.inffus.2019.12.012.

[177] H. A. Dau, A. Bagnall, K. Kamgar, *et al.*, "The ucr time series archive," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, 2019. DOI: 10.1109/JAS.2019.1911747.

[178] A. J. Bagnall, H. A. Dau, J. Lines, *et al.*, "The UEA multivariate time series classification archive, 2018," *CoRR*, vol. abs/1811.00075, 2018. [Online]. Available: http://arxiv.org/abs/1811.00075.

[179] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-series classification with cote: The collective of transformation-based ensembles," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, 2016, pp. 1548–1549. DOI: 10.1109/ICDE.2016.7498418.

[180] R. J. Kate, "Using dynamic time warping distances as features for improved time series classification," *Data Mining and Knowledge Discovery*, vol. 30, no. 2, pp. 283–312, 2016. DOI: 10.1007/s10618-015-0418-x.

[181] A. Bostrom and A. Bagnall, "Binary shapelet transform for multiclass time series classification," in *Transactions on Large Scale Data and Knowledge Centered Systems XXXII*, Springer, 2017, pp. 24–46. DOI: 10.1007/978-3-662-55608-5_2.

[182] J. Lines, S. Taylor, and A. Bagnall, "Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles," *ACM Transactions on Knowledge Discovery from Data*, vol. 12, no. 5, 2018. DOI: 10.1145/3182382.

[183] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data mining and knowledge discovery*, vol. 31, no. 3, pp. 606–660, 2017. DOI: 10.1007/s10618-016-0483-9.

[184] J. Large, J. Lines, and A. Bagnall, "A probabilistic classifier ensemble weighting scheme based on cross-validated accuracy estimates," *Data mining and knowledge discovery*, vol. 33, no. 6, pp. 1674–1709, 2019. DOI: 10.1007/s10618-019-00638-y.

[185] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data mining and knowledge discovery*, vol. 28, no. 4, pp. 851–881, 2014. DOI: 10.1007/s10618-013-0322-1.

[186] M. Middlehurst, W. Vickers, and A. Bagnall, "Scalable dictionary classifiers for time series classification," in *International Conference on Intelligent Data Engineering and Automated Learning*, Springer, 2019, pp. 11–19. DOI: 10.1007/978-3-030-33607-3_2.

[187] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Information Sciences*, vol. 239, pp. 142–153, 2013, ISSN: 0020-0255. DOI: 10.1016/j.ins.2013.02.030.

[188] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, New York, NY, USA: Association for Computing Machinery, 2016, 1135–1144, ISBN: 9781450342322. DOI: 10.1145/2939672.2939778.

[189] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Aug. 2017, pp. 3319–3328.

[190] H. Yang, C. Rudin, and M. Seltzer, "Scalable Bayesian rule lists," in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Aug. 2017, pp. 3921–3930. [Online]. Available: https://proceedings.mlr.press/v70/yang17h.html.

[191] N. Burkart, P. M. Faller, E. Peinsipp, and M. F. Huber, "Batch-wise regularization of deep neural networks for interpretability," in *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2020, pp. 216–222. DOI: 10.1109/MFI49285.2020.9235209.

[192] N. Burkart, M. Huber, and P. Faller, "Forcing interpretability for deep neural networks through rule-based regularization," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019, pp. 700–705. DOI: 10.1109/ICMLA.2019.00126.

[193] M. Veerappa, M. Anneken, and N. Burkart, "Evaluation of interpretable association rule mining methods on time-series in the maritime domain," in *International Conference on Pattern Recognition*, Springer, 2021, pp. 204–218. DOI: 10.1007/978-3-030-68796-0_15.

[194] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International Conference on Machine Learning*, PMLR, 2017, pp. 3145–3153. [Online]. Available: http://proceedings.mlr.press/v70/shrikumar17a/shrikumar17a.pdf.

[195] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing, "Lemna: Explaining deep learning based security applications," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18, New York, NY, USA: Association for Computing Machinery, 2018, 364–379, ISBN: 9781450356930. DOI: 10.1145/3243734.3243792.

[196] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[197] S. A. Siddiqui, D. Mercier, M. Munir, A. Dengel, and S. Ahmed, "Tsviz: Demystification of deep learning models for time-series analysis," *IEEE Access*, vol. 7, pp. 67 027–67 040, 2019. DOI: 10.1109/ACCESS.2019.2912823.

[198] G. Plumb, D. Molitor, and A. S. Talwalkar, "Model agnostic supervised local explanations," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/b495ce63ede0f4efc9eec62cb947c162-Paper.pdf.

[199] D. Pedreschi, F. Giannotti, R. Guidotti, A. Monreale, S. Ruggieri, and F. Turini, "Meaningful explanations of black box ai decision systems," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 9780–9784. DOI: 10.1609/aaai.v33i01.33019780.

[200] J. Chen, L. Song, M. Wainwright, and M. Jordan, "Learning to explain: An information-theoretic perspective on model interpretation," in *Proceedings of the 35th International Conference on Machine Learning*,

J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, Jul. 2018, pp. 883–892. [Online]. Available: https://proceedings.mlr.press/v80/chen18j.html.

[201] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st international conference on neural information processing systems*, 2017, pp. 4768–4777. [Online]. Available: https://proceedings.neurips.cc//paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[202] R. Neamtu, R. Ahsan, E. A. Rundensteiner, *et al.*, "Generalized dynamic time warping: Unleashing the warping power hidden in pointwise distances," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, 2018, pp. 521–532. DOI: 10.1109/ICDE.2018.00054.

[203] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *International Conference on Learning Representations*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.0473.

[204] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, *et al.*, "Improvements to deep convolutional neural networks for lvcsr," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013, pp. 315–320. DOI: 10.1109/ASRU.2013.6707749.

[205] E. Keogh and A. Mueen, "Curse of dimensionality," in *Encyclopedia of Machine Learning and Data Mining*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2017, pp. 314–315, ISBN: 978-1-4899-7687-1. DOI: 10.1007/978-1-4899-7687-1_192.

[206] X. Zheng, N. Xu, L. Trinh, *et al.*, "Psml: A multi-scale time-series dataset for machine learning in decarbonized energy grids," *arXiv preprint arXiv:2110.06324*, 2021. [Online]. Available: https://arxiv.org/pdf/2110.06324.pdf.

[207] H. I. Fawaz, B. Lucas, G. Forestier, *et al.*, "InceptionTime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020. DOI: 10.1007/s10618-020-00710-y.

[208] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *International conference on web-age information management*, Springer, 2014, pp. 298–310. DOI: 10.1007/978-3-319-08010-9_33.

[209] X. Zhang, Y. Gao, J. Lin, and C.-T. Lu, "TapNet: Multivariate time series classification with attentional prototypical network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 6845–6852. DOI: 10.1609/aaai.v34i04.6165.

[210] A. Dempster, D. F. Schmidt, and G. I. Webb, "MiniRocket: A very fast (almost) deterministic transform for time series classification," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2021, 248–257, ISBN: 9781450383325. DOI: 10.1145/3447548.3467231.

[211] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 565–592, 2015. DOI: 10.1007/s10618-014-0361-2.

[212] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, and E. Keogh, "Generalizing dtw to the multi-dimensional case requires an adaptive approach," *Data mining and knowledge discovery*, vol. 31, no. 1, pp. 1–31, 2017. DOI: 10.1007/s10618-016-0455-0.

[213] A. Dempster, F. Petitjean, and G. I. Webb, "ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels," *Data Mining and Knowledge Discovery*, vol. 34, no. 5, pp. 1454–1495, 2020. DOI: 10.1007/s10618-020-00701-z.

[214] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. Bagnall, "The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 35, no. 2, pp. 401–449, 2021. [Online]. Available: https://ueaeprints.uea.ac.uk/id/eprint/77815/7/Ruiz2020_Article_TheGreatMultivariateTimeSeries.pdf.

[215] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. [Online]. Available: https://www.cs.princeton.edu/courses/archive/spring16/cos598F/msra-deepnet.pdf.

[216] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.

[217] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[218] A. Kigerl, Z. Hamilton, M. Kowalski, and X. Mei, "The great methods bake-off: Comparing performance of machine learning algorithms," *Journal of Criminal Justice*, vol. 82, p. 101 946, 2022, ISSN: 0047-2352. DOI: 10.1016/j.jcrimjus.2022.101946.

[219] S. R. Cachay, E. Erickson, A. F. C. Bucker, *et al.*, *The world as a graph: Improving el Niño forecasts with graph neural networks*, 2021. arXiv: 2104.05089 [cs.LG].

[220] W. A. Knaus, E. A. Draper, D. P. Wagner, and J. E. Zimmerman, "Apache ii: A severity of disease classification system.," *Critical care medicine*, vol. 13, no. 10, pp. 818–829, 1985. [Online]. Available: https://cir.nii.ac.jp/crid/1570009750973141504.

[221] P. S. Kamath, R. H. Wiesner, M. Malinchoc, *et al.*, "A model to predict survival in patients with end-stage liver disease," *Hepatology*, vol. 33, no. 2, pp. 464–470, 2001, ISSN: 0270-9139. DOI: https://doi.org/10.1053/jhep.2001.22172.

[222] T. K. Burch, "Data, models, theory and reality: The structure of demographic knowledge," in *Model-Based Demography: Essays on Integrating Data, Technique and Theory*. Cham: Springer International Publishing, 2018, pp. 21–42, ISBN: 978-3-319-65433-1. DOI: 10.1007/978-3-319-65433-1_2.

[223] J. Pearl, "Causality: Models, reasoning and inference," *Econometric Theory*, vol. 19, no. 10, pp. 675–685, 2003. [Online]. Available: https://cir.nii.ac.jp/crid/1570009750973141504.

[224] F. Rosenblatt, "The perceptron, a perceiving and recognizing automaton : (project para)," *Cornell Aeronautical Laboratory report*, vol. 85-460-1, pp. 1–29, 1957.

[225] B. Widrow and M. E. Hoff, "Adaptive switching circuits," in *1960 IRE WESCON Convention Record, Part 4*, IRE, 1960, pp. 96–104.

[226] H. Robbins and S. Monro, "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[227] J. Kiefer and J. Wolfowitz, "Stochastic Estimation of the Maximum of a Regression Function," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, 1952.

[228] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986. DOI: 10.1038/323533a0.

[229] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969.

[230] T. M. Mitchell, *Machine Learning*, 1st. McGraw-Hill, Inc., 1997, ISBN: 0070428077.

[231] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks.," in *AISTATS*, G. J. Gordon, D. B. Dunson, and M. Dudík, Eds., ser. JMLR Proceedings, vol. 15, JMLR.org, 2011, pp. 315–323. [Online]. Available: http://dblp.uni-trier.de/db/journals/jmlr/jmlrp15.html#GlorotBB11.

[232] B. Efron, "Bootstrap methods: Another look at the jackknife," *The Annals of Statistics*, vol. 7, no. 1, pp. 1–26, 1979. DOI: 10.1214/aos/1176344552.

[233] E. Carlstein, "Resampling techniques for stationary time-series: Some recent developments," *New Directions in time series Analysis*, pp. 75–85, 1992.

[234] H. R. Künsch, "The jackknife and the bootstrap for general stationary observations," *The Annals of Statistics*, vol. 17, no. 3, pp. 1217–1241, 1989. DOI: 10.1214/aos/1176347265.

[235] D. N. Politis and J. P. Romano, "A circular block-resampling procedure for stationary data," in *Exploring the Limits of Bootstrap*, Wiley, 1992, 263—270.

[236] ——, "The stationary bootstrap," *Journal of the American Statistical Association*, vol. 89, no. 428, pp. 1303–1313, 1994. DOI: 10.1080/01621459.1994.10476870.

[237] S. Lahiri, "Comparison of block bootstrap methods," in *Resampling Methods for Dependent Data*, Springer Series in Statistics, 2003, pp. 115–144, ISBN: 978-1-4419-1848-2. DOI: 10.1007/978-1-4757-3803-2_5.

[238] P. Bühlmann, "Sieve bootstrap for time series," *Bernoulli*, vol. 3, no. 2, pp. 123–148, 1997.

[239] F. Chollet *et al.*, *Keras*, https://keras.io, 2015.

[240] M. Abadi, P. Barham, J. Chen, *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16, Savannah, GA, USA: USENIX Association, 2016, pp. 265–283, ISBN: 978-1-931971-33-1. [Online]. Available: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[241] A. Paszke, S. Gross, S. Chintala, *et al.*, "Automatic differentiation in PyTorch," in *NIPS 2017 Workshop Autodiff*, 2017. [Online]. Available: https://openreview.net/pdf?id=BJJsrmfCZ.

[242] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with Python," in *9th Python in Science Conference*, 2010. [Online]. Available: https://conference.scipy.org/proceedings/scipy2010/pdfs/seabold.pdf.