Generating realistic eye-tracking data with Transformers

Arnau Naval Ruiz

Thesis submitted for the degree of
Master in Applied Computer and Information Technology - ACIT
(Artificial Intelligence)
30 credits

Department of Computer Science
Faculty of Technology, Art and Design

Oslo Metropolitan University — OsloMet

Spring 2023

Generating realistic eye-tracking data with Transformers

Arnau Naval Ruiz

Generating realistic eye-tracking data with Transformers

http://www.oslomet.no/

Printed: Oslo Metropolitan University — OsloMet

Abstract

Eye-gaze forecasting is a field with a significant number of applications, such as User Interface analysis or improving self-driving cars. Despite its importance, this type of data can be hard to come by due to laws protecting users' data. Therefore, we use a vanilla Transformer and Informer, a transformer-based model focused on time series forecasting, to generate realistic artificial data that can be used for further research, trained from eye-tracking data recorded at OsloMet. In order to validate the quality of the results we generate histograms for the distribution of the velocities for the positions and angles between points, as well as the autocorrelation, this analysis is compared against the results of a simple linear model and a Markov model. This study, conducted with limited data which can affect the generalization capabilities of the larger models, finds that the well-established mathematical model significantly outperforms the Deep Learning models. Such results indicate that the transformer-based models utilized may not be adequate for such a task.

Acknowledgements

I would like to express my gratitude to those who have provided support and assistance throughout my thesis project. In particular my supervisor Pedro Lencastre for the support, guidance, and motivation throughout the whole project. In addition, I would also like to thank my cosupervisor Prof. Pedro Lind for all the very valuable feedback. Lastly, I would like to thank my family, friends, and the people at Völur for all the encouragement and support.

Arnau Naval Ruiz,
11.05.2023

Contents

Abstract **i**

Acknowledgements **iii**

1 Introduction **1**

2 Background and state of the art **7**
2.1 Time-series and eye-gaze forecasting 7
2.2 Statistics and data analysis 10
2.3 Markov models 11
2.3.1 Basic concepts 12
2.3.2 Non-parametric Markov models 12
2.4 Deep learning models for time-series forecasting 13
2.4.1 Generative Adversarial Networks 13
2.4.2 Recurrent Neural Networks 15
2.4.3 Long Short-Term Memory 16
2.4.4 Transformers and architecture 18
2.5 Evaluation Metrics 21
2.6 Previous works & state of the art 22

3 Methods and data description and processing **25**
3.1 Data description 25
3.2 Datasets 27
3.2.1 Eye-tracking dataset 27
3.2.2 Gaussian Noise dataset 28
3.2.3 Empirical dataset: oil temperature 29
3.3 Processing the eye-tracking data 30
3.3.1 Histograms of the main properties 30
3.3.2 Autocorrelation of distributions 32
3.4 Methods 33
3.4.1 Transformer 33
3.4.2 Informer 35
3.4.3 NLinear 35
3.4.4 Markov model 36
3.5 Experiments 37

4 Implementation details **39**

5 Results **43**
5.1 Prediction of eye-tracking velocities 43
5.2 Prediction of θ and φ 48
5.3 Prediction of Gaussian Noise 49
5.4 Prediction of θ and φ 53
5.5 Oil Temperature 54
5.5.1 Prediction 54
5.5.2 Data distribution 55
5.6 Past, Present, and Future correlations 56

6 Discussion and Conclusions **57**
6.1 Findings and observations 58
6.2 Conclusions and future perspectives 60

A Appendix **73**
A.1 Code 73

Chapter 1

Introduction

Eye tracking data is information gathered by observing a person's gaze, generally through the use of a sensor that monitors eye movement. Numerous research applications, such as cognitive psychology [1], human-computer interaction [2], and marketing research [3], can be investigated using this data. Eye tracking, for instance, has the potential to analyze how readers scan a page or how people interpret visual data in various scenario types. Eye tracking data can be used to examine how individuals interact with technology, such as how they use mobile devices or traverse websites, in the field of human-computer interaction. This can assist designers to create interfaces that are simpler for users to utilize. Normally, eye-tracking data alone is not regarded as Personal Identifiable Information (PII). Even so, acquiring such information frequently entails taking pictures of people's faces or other information that could be deemed PII. Additionally, the eye-tracking data could be linked to other PII-containing data types like name and address. When gathering this kind of data, it's crucial to take into account the setting in which it's being gathered and if it may be possible to connect it to a specific person.

Research in the health and biomedicine fields often involves the use of PII which is very sensitive to work with, as individual personal information is protected under the law, in the case of Europe, the General Data Protection Regulation (GDPR). Furthermore, in many cases, this type of sensitive data is owned by private hospitals which tend to be hesitant to share information for research outside their scope. The use of PII involves the signature of a consent letter from the user that it was retrieved from, as well as the processing of the data to avoid traceability back to the user. This processing involves preparation, anonymization, and a data handling plan which must be followed, failing to do so may result in very substantial fines. This process makes the use of PII to be expensive and time-consuming.

A potential solution to such an issue is biobanks which aim to provide biological big data. Biobanks are repositories that store biological samples for research use. In Ref.$\sim$ [4] the authors explain how biobanks are a powerful tool for health and biomedicine research as they allow therapies and clinical trials, which may ultimately lead to new cures. Despite that,

there are some barriers to overcome to make biobanks a widespread and feasible solution including transparency, security, and privacy as well as the opacity of international laws, data ownership, and public ignorance. Another solution includes using Artificial Intelligence (AI) to generate new data, also known as Data Augmentation (DA). This has many advantages, such as allowing the creation of as much data as needed as well as no need for processing, anonymization, and consent, as the data would not be tied to an individual human being. Thanks to the advances in computation systems and increasing accessibility to big data this latter solution is of growing interest.

With the rise in popularity of non-parametric models and Artificial Neural Networks (ANN), there has also been a rise in the need for data. Both non-parametric and ANN models can be used in an extensive range of applications as they allow the representation of different complex problems without necessarily needing to understand them or introducing simplifications. On one hand, examples of non-parametric models include clustering with mixture models, nonlinear regressions, and Hidden Markov Models, among others [5]. These have diverse applications such as outlier detection [6] or traffic flow forecasting [7]. On the other hand, ANNs have a variety of structures that incorporate Feed Forward Neural Networks, Convolutional Neural Networks, and Recurrent Neural Networks, among others. Examples of these structures include respectively indoor climate control for energy saving [8], prediction of aerodynamic flow fields [9] and electrical load forecasting [10]. Forecasting is one of the most interesting problems in modern-day research, as the convince of predicting future events could drastically improve planning, scheduling, and resource allocation and overall bring an immense competitive advantage compared to not estimating the future [11].

Forecasting is the process of making predictions about future events or conditions. It involves analyzing historical data, identifying patterns and trends, and using that information to make informed estimates about what will happen in the future [12]. Forecasting can be applied to almost all fields. AI methods have the advantage of being capable of processing large amounts of data, which can lead to the generation of forecasting models with high accuracy. Traditional processes, such as time series analysis, can become time-consuming and cumbersome for large datasets. AI methods have the ability to identify patterns in the data that may be difficult for humans to spot. ANN and non-parametric models have different advantages and weaknesses to approach forecasting. ANN are better suited for tasks that involve pattern recognition and non-linear relationship, although they usually require large amounts of data to reach high accuracies. Non-parametric models make minimal assumptions about the underlying distribution of the data which makes them more flexible for different data distributions and better at handling noisy data, but they usually also require large amounts of data to avoid overfitting.

Both methods have one common problem, needing big amounts of data. This can be especially troublesome if the data needed is PII. One common approach to counter the lack of data is the aforementioned data

augmentation (DA). DA is a technique used to artificially create samples of data to increase the dataset size as well as its diversity, which can help improve AI models as more data helps reduce the probability of overfitting [13]. These augmented samples can be simple modifications of the original data. For instance, in image data augmentation, a rotation of the original image is considered DA, more complex DA can involve adding noise or random crops.

Therefore, the primary focus of this initiative will be the generation of synthetic PII, in particular eye tracking data. To do this, Transformers, a type of Deep Learning Neural Network, will be used. To achieve real-like data augmentation, the Transformers will be utilized to forecast eye gaze sequences and provide unique data. Transformers have become more well-known in many fields, such as Natural Language Processing and Time-Series Prediction [14, 15]. Their encoder-decoder architecture, which can handle sequential data and comprehend long-term dependencies in the incoming data, makes them highly suitable for such tasks.

The topic of eye-tracking has seen a lot of recent research, some of which compares the attentional mechanisms used by humans and machines to comprehend the words from a text as well as eye-tracking prediction patterns. On the one hand, Ref.∼[16] examines how, despite the fact that both human reading and NLP use the word attention in Transformers, these two uses seem to have quite distinct meanings when viewed alone. They offer a comparison of the correlations between the two processes, demonstrating the strong similarities between the initial layers of the Transformer model BERT and human attention as determined by eye gazing. Despite seeing encouraging results, they argue that further research is necessary before eye-gaze models are used to replace the Transformers' attention systems. On the other hand, Ref.∼[17] which conducts research on the CMCL 2021 Shared Task of Predicting Reading Behaviour, offers the following question: *Is it possible to predict eye-tracking behavior given the reading material?* They suggest this question as a regression problem and create a number of models that manage to predict eye-gaze patterns with low error.

Both studies give us the impression that it is both highly possible and accurate to conduct research utilizing eye-tracking data and Transformer models. Second, it informs us that eye-gaze prediction is possible and that it has been a subject of investigation. We believe that utilizing more cutting-edge Transformer architectures created specifically for time-series forecasting can significantly enhance eye-gaze forecasting. Even though studying human reading behavior is not our goal, our research may still be useful for such studies by helping them by supplying more on-demand data for additional training. As they have also been designed to discover patterns in data and predict time series, other architectures like Temporal Convolutional Networks (TCN) or Recurrent Neural Networks (RNN) could also be taken into consideration for this purpose. Although we explore more into the reasons for our pursuit of the employment of Transformers in the sections that follow, we are confident that cutting-edge Transformers architectures will outperform other Deep Learning models.

Eye-tracking data is significant in many domains, as briefly discussed in the previous section. Following are a few examples: researchers can use eye-tracking data to examine how people visually scan and process information in order to better understand how the human visual systems work and how people pay attention to various aspects of their environment. Additionally, the use of mobile devices and website navigation can be studied using eye-tracking data. This can assist designers in producing interfaces that are more user-friendly and enhance the user experience as a whole [18]. Moreover, in the field of neuroscience, eye-tracking data can be used to study the neural basis of visual perception and attention, which can help researchers understand the brain mechanisms underlying these processes [19]. Numerous fields, such as scientific and academic research, market research, neuroscience and psychology research, medical research, usability research, packaging research, gaming research, and human factors research, use eye-tracking data [20]. Additionally, studies show that eye-tracking data can reveal information from individuals such as gender, age, and race [21], detecting when someone is lying [22], mental health monitoring [23] and alcohol consumption [24] among others.

In the field of AI, eye-tracking data can be used in computer vision tasks to help Convolutional Neural Networks (CNN) perform better [25, 26]. The human brain is extremely efficient at absorbing information and recognizing patterns through the eyes therefore, using eye-tracking data to understand the visual mechanisms of humans and using that information to train CNNs could result in a boost in performance. Furthermore, eye-tracking data can also be used as additional input to CNNs. In an image classification task, the information on the position of the eyes can provide valuable information on where the network should focus. Eye-tracking data can be useful for more than only image processing tasks; it can also be useful for Natural Language Processing (NLP) models. NLP models can enhance their comprehension and processing of the texts they are presented with by better understanding how people absorb information and concentrate on words thanks to the use of eye-tracking data. Currently, gathering high-quality eye-tracking data can only be done by using expensive specialized machinery which can be hard to acquire, thus highlighting the importance of being able to generate faithful data.

The Artificial Intelligence (AI) field can be traced back to the 1950s with the interest of creating machines that could mimic human intelligence and perform tasks such as learning, problem-solving, and decision-making. This happened at the Dartmouth conference where AI got its name [27]. Since then, AI has been facing different trends of increased and decreased popularity, which were motivated, in part, by the advances in, and lack of, computing resources, respectively. In more recent years, AI has seen some of its biggest achievements and it has been introduced to many aspects of people's daily lives as well as in a variety of industries. Some examples are smartphones that use AI for speech recognition for personal assistants or enhanced capabilities for the camera. Additionally, industries such as the automotive industry are introducing self-driving cars, finance uses AI for analysis and fraud detection, commerce can use AI to provide tailored

recommendations to its users, and even healthcare can use AI for image recognition and treatment recommendation, among others.

Some of these industries use AI for time-series forecasting, which consists of creating a model that will accurately predict future values for a time series using previous data. The origins of time-series forecasting with AI date as far back as the 1970s, when the statisticians George Box and Gwilym Jenkins developed the Box–Jenkins method, also known as Autoregressive Integrated Moving Average (ARIMA) [28]. Nowadays, one of the most popular approaches to time-series forecasting is using Machine Learning (ML). The development of algorithms and statistical models that allow a system to learn from data and make predictions or decisions without being explicitly programmed is known as ML, which is a subset of AI. ML, in particular ANNs, have demonstrated to excel at capturing complex patterns from the data, which can later be used for a diversity of tasks, including time-series forecasting. Because it enables models to discover patterns and relationships in the data that may not be immediately visible to humans, ML is well suited for time series forecasting. Time series data frequently contains complex seasonality, trends, and patterns that are challenging to model using conventional statistical techniques. On the other hand, machine learning algorithms can recognize these patterns and make precise predictions about future values using them. Utilizing machine learning for time series forecasting has several benefits, one of which is its capacity for handling large amounts of data. Large datasets may not be suitable to traditional statistical methods for time series forecasting due to not scaling well, but machine learning algorithms can be taught on large datasets, which can result in more accurate predictions. Additionally, machine learning models can adjust to the data's evolving patterns, which is frequently the case with time series data. For instance, if the stock market significantly shifts, a model that was trained on historical stock values may not perform well. Some examples of ML algorithms that have been used for such task are Recurrent Neural Networks (RNN), Long Short-Term Memory networks (LSTM), Gated Recurrent Units (GRU), and most recently Transformers.

ML proves its effectiveness in time-series forecasting across a range of applications. These industries include finance, where machine learning is used to forecast stock prices, commerce, where it can forecast sales for retail stores and online marketplaces, climate change, where it can forecast weather patterns and the effects of climate change on natural systems, and energy, where it can be used to forecast energy demand and prices.

As previously mentioned, eye-tracking data can be a form of PII which is extremely sensitive as it can be used to identify or locate an individual. With regard to eye tracking, the data can be utilized to infer details about the subject's IQ [29], personality traits [30], drug consumption [31], age, gender, and ethnicity [21], as well as conditions like ADHD and autism [32]. A long list of additional information that can be inferred from this type of data [33], some of which falls under the category of special category data, which according to GDPR, it requires additional protection.

Devices that can record eye-tracking data may potentially capture

considerably more information than what the user intends to divulge due to eye-tracking data being able to expose so much information. It would not come as a surprise to learn that additional eye-tracking studies could probably yield even more alarming information about the people, information that we do not even know could be inferred from this type of data to this day. It is important to note that interpreting information from eye-tracking can be challenging and is not always accurate. An intensive gaze fixation on another person's face, for instance, could mean a variety of things, including liking, repulsion, perplexity, and recognition [34]. Even though it is possible to some extent consciously control where we look, many parts of the ocular behavior, especially at a micro level, are impossible to control [35], these parts include pupil dilatation or spontaneous blinking, thus making it almost impossible to voluntarily avoid leaking personal information. Although some micro behavior can be controlled, this quickly becomes physically and cognitively exhausting, making it impossible to maintain for long periods of time, furthermore, it tends to generate patterns that make such behavior detectable [36]. Therefore, we consider that creating realistic eye-tracking data that is not tied to an individual can be the best process to further research this field without endangering any person.

Some attempts have been made to protect the eye-tracking data by adding random noise to the signal. Since it was not possible to re-identify the person from the data, these showed promising results, and the applications trained with the data still performed well [37].

We hypothesize that it is possible to train time-series-focused Machine Learning models in order to capture the human-gaze properties as well as its time dependencies and replicate the data in a manner that is indistinguishable from real human data. This can have multiple applications such as training ML models, conducting psychological research, enhancing virtual and augmented reality experiences, and improving accessibility in user interfaces, among others. In this thesis, we will be implementing and comparing a variety of ML models, including the state-of-the-art Transformers for time-series data, as well as some more traditional RNNs (and/or LSTMs) to discover whether any of these architectures are capable of replicating realistic eye-gazes trajectories. Therefore, we will study which of the models is better at capturing the time dependencies expressed in the eye-tracking data and, furthermore, which of the models is better at replicating the data to appear more realistic. This thesis addresses the following research question: Which Machine Learning model is better at replicating eye-gazes trajectories of humans searching for information?

The thesis is structured as follows: Chapter 2 presents all the background related to eye-tracking, data utilized as well a background in AI relevant to this work, including the evaluation metrics. Chapter 3 presents the methodology which includes a presentation of the specific models used, experiments conducted, and their evaluation. Chapter 4 showcases the results of all the experiments conducted. Chapter 5 presents a discussion of the results obtained. Finally, in Chapter 6 we present the conclusion of the project as well as future works related to this research question.

Chapter 2

Background and state of the art

2.1 Time-series and eye-gaze forecasting

The idea of causes and sequences related to weather dates back to Aristotle and the Ancient Greeks, which remained until the Renaissance. At that time scientists started collecting data related to the weather with the help of inventions such as the barometer. Robert FitzRoy, responsible for recording and publishing weather-related data for sailors in the 1850s, is considered a pioneer in weather forecasting, he named this data "weather forecast" [38]. From the same book, in the field of Medicine, an early known application is Ref.∼ [39] where in 1887, the author presented an electrocardiogram (ECG) recording which can record the electrical signals passing through the heart. Since the ECG machine was not invented until 1901, the author used a mercury capillary electrometer and measured the displacement of the mercury. Other fields such as economics and astronomy also used time series data around that period of time.

As we know it now, a time series is a sequence of data points gathered at regular time intervals. In 1970 George Box, from the Box-Jenkins method (ARIMA), was a pioneering statistician who presented this method in the statistics textbook "Time Series Analysis: Forecasting and Control" which is regarded as a major accomplishment in time series analysis. Therefore, we can consider that the discipline of time series analysis is still quite young. From there, it evolved as computer technology did. The advancements in the computing field allowed for larger datasets to be utilized and better tools to process them. More recent uses of time series analysis in ML can be found in the 1980s in different applications such as anomaly detection in computer security as well as in the invention of RNNs, in 1986. Nowadays, time series analysis is a rapidly growing field with a variety of active areas, which include, but are not limited to, economic forecasting, inventory management, weather forecasting, quality control, medical research, speech recognition, NLP, cybersecurity, and robotics.

With the advances in computer systems, a plethora of tools has been developed to efficiently work with time series data like Excel, Python, R, MATLAB, and TensorFlow among others. Following the ML path, different algorithms have also been created to generate more accurate predictions

such as the ARIMA method from George Box. Other methods include Random Forest, Gradient Boosting, Support Vector Machines, and the ones that we will be looking at in this thesis, Recurrent Neural Networks, Long Short-Term Memory networks, and Time Transformers. Additionally, these methods may be better suited for different types of time series data. Different types of time series data appear to the different methods of collection as well as the different characteristics and properties the data can have. For example, whereas irregular time series data lacks consistent spacing in time, regular time series data do have it. Other types include univariate and multivariate data, wherein in the univariate case, the data consists of only one variable, contrary to multivariate with multiple variables. Many variables can provide better context and more information but they may also be more complex to process. Similarly, stationary and non-stationary data consist of time series data that does not change over time, or that it does, respectively. Time series data can also include seasonality, where the same pattern repeats over time, or it can contain trends, where the data showcases an increasing or decreasing pattern over time. Other types of time series data exist such as data with noise or with multiple seasons, among others.

Eye tracking data is information collected by recording the movement of a person's eyes, this helps understand visual attention as it can determine where users are looking at a particular moment in time, how long they are looking at something, and the path their eyes take using eye tracking. The first attempts to gather eye-tracking data date back to the late 1800s and some of the devices utilized were at least unpleasant for the person being recorded [40]. One of these devices includes lenses with only a small opening and a pointer attached to it to be able to identify where the user was looking at. These first studies were mostly to understand the complex relationship between the brain and the visual system. In the 1940s, less intrusive systems had been created to record eye-tracking data, these used film to record the movements. One of the first usability studies was in 1947 when Paul Fitts and his colleagues used cameras to record the eyes of pilots when landing a plane to improve the design of the cockpit. In the late 1990s though, the more recent eye-tracking technologies still used to this day were finally released, with the addition of hardware and software developments allowing the technology to be more widespread outside academia.

With the current systems, it is possible to retrieve different information from a user such as the location of where the user is looking at a particular time. This is also known as a fixation, and they typically last between 100 to 600 milliseconds, during a fixation the eyes are focused on a particular x,y coordinate in the visual field. Fixations are thought to happen when the individual is processing the visual information at that location, despite that, it can also be that the eyes are resting at that particular coordinate and the attention of the user is elsewhere, which does not provide useful information. The duration of the fixations can also be measured, although there are many reasons for the difference in durations of fixations, it can help understand if the user is paying attention to that particular

element. Additionally, the movement of the eye can also be recorded, also known as saccades. Saccades are extremely rapid eye movements from the fixation point to the next fixation point and they allow the eyes to quickly scan the information of the visual environment. They take around 30 to 50 milliseconds and they cover a distance of around 0.5 to 20 degrees. Saccades can be used to understand how a user interprets a visual stimulus.

Eye-tracking data can be considered multivariate irregular time series data as the intervals between the data points are not evenly spaced and due to containing information about the x and y-axis. The data usually contains a sequence of gazes or fixations recorded over time and can be considered continuous as it usually contains 30-60 samples per second. Additionally, since eye-tracking data has a temporal order, it is considered to be sequential. Time series data, speech signals, and text are examples of sequential data. These types have in common that the order of the samples is relevant and directly affects the patterns and dependencies. Eye tracking data has a wide variety of applications, in the medical field, this type of data is used to study the cognitive functions of patients with brain injuries [41] or diseases like strokes or dementia [42, 43]. It can also provide valuable information about mental health and pain, by measuring the changes in visual attention after a painful stimulus. Eye-tracking data can also be used to develop bio-inspired systems. Bio-inspired systems are artificial systems that mimic or replicate the behavior and functionality of biological systems. Thanks to eye-tracking data, a visual system that mimics the way the human visual system processes and interprets visual information can be developed. Another bio-inspired system that can be developed with the use of eye-tracking data is a system that can interact with humans in natural and intuitive ways, as the data can provide crucial information about how people interact with technology.

According to research [44], different people inspect images differently. The movements between inspections will be remarkably comparable when someone is given a scene to examine and then given an identical scene to examine again a few days later. Particularly, more similar than comparing eye movements made by several people when seeing the same scene. Additionally, when individuals are left to examine the same complex picture over a long period of time, they tend to show cycles of inspection behavior over the same parts. These cycles are also found when an individual is left to examine the portrait of a person, where the cycles happen over the key parts of the face, including the mouth and nose, with a strong preference for the eyes. Figure 2.1 shows one of the most important contributions found in Ref.∼[44]. This figure shows the result of showing the same scene, The Visitor, to an individual seven times and asking them to perform different tasks. As Yarbus observed, "Depending on the task in which a person is engaged, ie, depending on the character of the information which he must obtain, the distribution of the points of fixation on an object will vary correspondingly because different items of information are usually localized in different parts of an object". Such findings show how the thought process of the individual affects the movements of the eyes. Furthermore, it shows how the focus

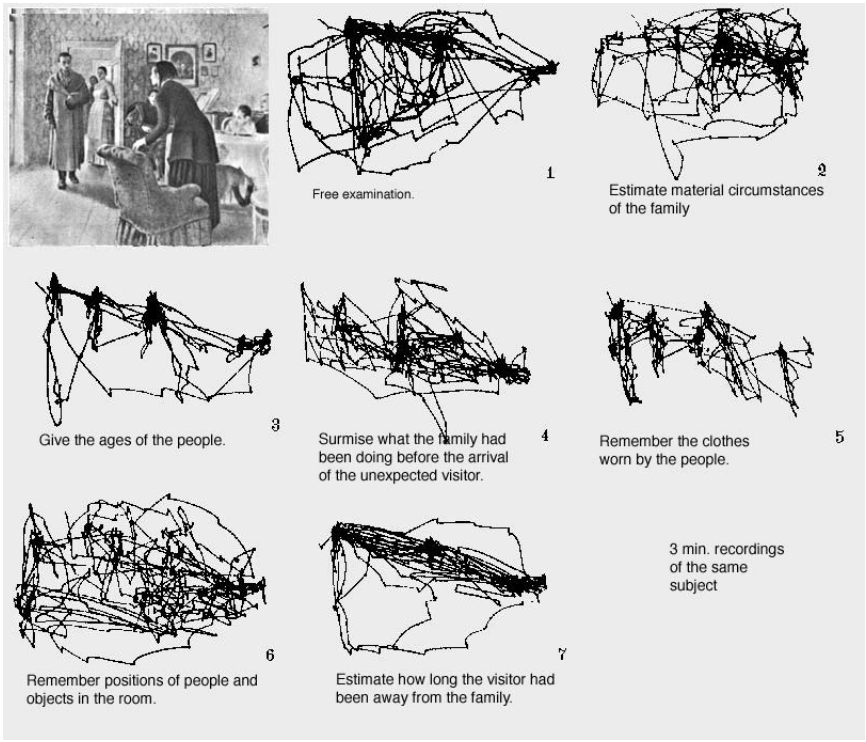


Figure 2.1: Differences of eye movements depending on the given task, from Ref.∼[44]. Picture name The Visitor.

of the individual is on parts of the scene that do not necessarily give important information, but according to the individual's opinion, they may do. This experiment showcases different patterns the eye movements present. When the task was related to the people present on the scene, such as figuring out their age or estimating how long the visitor had been gone. For this, the eye patterns are extremely focused on the people's faces, as they are the best source of information to estimate their ages or try to gain relevant information that may prove to be useful to answer the questions correctly. These eye movements appear to be mainly straight lines between the faces, while in other tasks, such as free examination or remembering the position of the people and objects in the room, the movements appear to be more random and scattered across the scene, most likely trying to obtain the most overall understanding of the scene in a quick time.

2.2 Statistics and data analysis

Humans are visual animals, in fact, according to the author of Ref.∼[45] "the human brain processes images 60,000 times faster than text, and 90 percent of information transmitted to the brain is visual.". Therefore, it is not surprising that humans have developed several techniques to be able to analyze data using visual information rather than text. One of the most important statistical tools to analyze data was introduced by Karl Pearson in 1895, where in his paper "On the Theory of Skew

Correlation" he introduced the first histogram [46]. Histograms are graphical representations of the distribution of numerical data. By segmenting the data into a number of intervals (often referred to as "bins") and counting the number of observations that fall inside each bin, they are able to produce a visual representation of the frequency distribution of a dataset. Histograms are commonly used to explore and summarize large datasets, as they allow for quickly identifying patterns and trends in them. It is worth noting that the bins of the y-axis are a frequency measurement of the data they represent. Therefore, to fit distributions to histograms, it is recommended to use measurements with a probabilistic interpretation, such as Kullback-Leibler (KL). KL is a natural choice to fit distributions to datasets, as opposed to others like Least Squared Errors (LSE), because it is invariant to the choice of histogram bin width. Another statistical tool is autocorrelation, which measures the linear relationship between a data point in a time series and its previous values, often referred to as "lags" [47]. It is commonly used to understand and model the temporal dependence or patterns in data, like identifying trends and seasonality. The autocorrelation is usually measured using the autocorrelation function (ACF) and is typically plotted as a function of the lag, with the lag on the x-axis and the autocorrelation coefficient on the y-axis. An autocorrelation of +1 indicates a strong linear relationship between a point and its lagged values, and an autocorrelation of -1 indicates a strong linear relationship in the opposite direction between a point and its lagged values, while an autocorrelation of 0 indicates no linear relationship at all. The autocorrelation cannot be confused with the similarly-named statistical measure: the correlation. The correlation also measures the relationship between variables but it provides a different interpretation. The correlation measures the linear association between two variables [48], instead of the same variable with different lags. Similarly to autocorrelation, a correlation of +1 indicates that both variables increase and decrease in a perfect linear fashion, while a correlation of -1 indicates that both variables opposedly increase and decrease in a perfect linear fashion, while a correlation of 0 indicates no relationship between the two variables. All these statistical tools are of high importance in time series analysis as they provide useful information about the characteristics and relationships of the data, which allows for understanding the patterns and relationships in the data, which can guide subsequent analysis and forecasting.

2.3 Markov models

In 1906, the Russian mathematician Andrei Markov produced the first theoretical results of a stochastic process using the term chain. Markov's initial work was motivated by applications in linguistics, where he used Markov chains to model the probability distribution of letters in Russian text. These Markov chains were later generalized to countable infinite states by Andrey Kolmogorov in 1931 [49]. Since then, Markov models have been widespread in many fields and applications, including pattern

recognition, speech recognition, and computational biology [50–52].

2.3.1 Basic concepts

Markov models are a class of statistical models used to analyze and forecast time series data. They are based on a stochastic process, known as the Markov Process that follows a set of rules known as the Markov property. The Markov property states that the future state of a system depends only on its current state, and not on any previous states. In a Markov model, the time series data is represented as a sequence of states. A state in the context of Markov models is a set of variables or parameters that can change over time, of the system being modeled. Therefore, a Markov chain is a sequence of states in which the probability of transitioning from one state to another depends only on the current state and not on any past states. These Markov chains are represented by a transition matrix, which shows the probabilities of moving from each state to every other state.

2.3.2 Non-parametric Markov models

The thesis project described was the foundation for the paper Ref.$\sim$ [53]. This paper presents the findings in the thesis and further extends them by comparing the performance of the GAN models against a Markov model for time-series prediction. Markov models are a type of statistical model that is used to model systems that change over time. They were initially introduced in 1906. Only the most basic type of VAR data and actual eye-tracking data were used for their comparisons in this instance. The GAN models' shortcomings in the previous phase, when they were having trouble modeling the distributions, were already apparent, especially in the extreme values. For the rest of the metrics, SIGCWGAN was outperforming the rest of the GANs. In the case of the Markov model, however, it did outperform all the GANs while measuring extreme values. Only SIGCWGAN achieved scores at the same level as the Markov model for the rest of the metrics. It is worth noting that the VAR data is, by construction, a Markov process, which may influence why the Markov model performs the best. For the real eye-tracking data none of the GAN models were struggling with almost all the analyzed metrics while also not being able to capture complex relationships between the different variables. Markov models, on the other hand, are capable of outperforming all GANs as they achieve to replicate the distribution of the model. In this case, the gazes are not expected to be Markovian thus eliminating the possibility that the Markov model was performing better than GANs just due to the VAR data being Markovian. Despite the simplifications introduced by the Markov model, it was significantly better at modeling both data distributions. These results are particularly impressive in light of how much simpler Markov models are compared to GANs, which can have millions of parameters. Furthermore, whereas the GAN models cannot be understood by computing the transitions due to the AI neural network's black-box nature, the Markov model can be understood by doing so.

It is surprising that a mathematical model over 100 years old such as the Markov model is better at replicating eye gazes better than GANs, which have shown great success in many areas, including Data Augmentation.

2.4 Deep learning models for time-series forecasting

Although time series forecasting has always been an interest, only recently DL methods have started to gain popularity as traditional methods focused on using parametric models such as autoregressive [54] or exponential smoothing [55, 56]. Thanks to the increase in computing power and big data, DL methods have become one of the primary tools for such tasks. DL extends from ML by adding additional layers in the artificial neural networks, the idea is that the initial layers will be able to learn a representation of the data with multiple levels of abstraction in order to solve complex problems [57]. One of the main drivers of utilizing DL for time series forecasting was the financial sector. There has always been an interest in being able to predict the prices of the stock market or the demand for products and in the last years DL has emerged as the most successful tool to perform such predictions [58]. It is to no surprise though, as DL has shown high performances in handling complex non-linear relationships, thanks to being trained with large amounts of data, and being able to detect patterns and dependencies in the sequential data that could have otherwise been missed by traditional methods.

Some of the most popular architectures for DL, include GANs, RNNs, and Transformers.

2.4.1 Generative Adversarial Networks

Generative Adversarial Networks (GAN) such as the ones utilized in the previous works are a type of neural network used to generate new data previously unseen. GANs have an adversarial architecture where a generator creates new data from noise and a discriminator tries to differentiate between the real data and the fake generated data. Both parts of the model are built as neural networks and are trained in parallel [59]. GANs have seen a lot of success in computer vision tasks including image generation [60], image-to-image translation [61], superresolution [62] and style transfer [63], some of this GANs becoming widely popular such as Nvidia's StyleGAN [64]. Another moderately less successful area of application for GANs has been time-series generation. Time series GANs, or Time-GANs for short, have seen a variety of uses in areas such as weather forecasting [65], financial forecasting [66] or even generating music [67].

As mentioned, GANs are built from two main components: the generator and the discriminator. The generator uses noise as an input to generate samples that resemble as much as possible the training data, while the discriminator is fed both real and the generated data and tries to distinguish them apart. The principle of GANs is that the

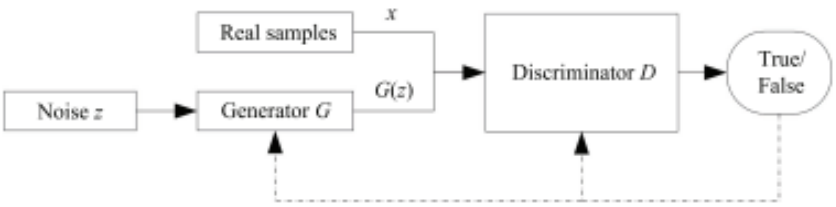


Figure 2.2: General structure of GANs, from Ref.∼[68].

two main components are in a zero-sum game, as a positive reward for one of the components results in a negative reward for the other [68]. Both components are trained simultaneously following this minimax optimization game. Typically both components are Deep ANNs even if the discriminator usually only needs to perform a binary classification task. The goal of a GAN is to reach Nash equilibrium, in which is considered that the generator has learned the representation of the data and it is capable of performing the best transformations [69]. GANs learn in by alternating the optimization of the generator and discriminator, thus when the generator is being optimized, the discriminator will be fixed, and vice versa. Overall, the discriminator will try to maximize the success of detecting fake data while the generator will try to minimize the success of the discriminator.

Figure 2.2 describes the common structure of a GAN and a Wasserstein GAN (WGAN).

Sequential GANs (SeqGAN) are an extension of traditional GANs designed for sequential data. SeqGANs have some key differences in their structure as well as in their training methodology that allow them to perform better in tasks involving sequential data such as NLP. Unlike traditional GANs, SeqGANs generate their data token by token instead of a sample of a fixed size. The discriminator evaluates each token after it is generated and gives feedback to the generator to improve future sequences, therefore instead of training the generator after a full sample has been created, in SeqGANs this happens per token. Additionally, in SeqGANs both the generator and discriminator are typically built from RNNs. A type of GAN that can be considered an extension of SeqGANs are Time-GANs, where the main difference is that instead of focusing on generating sequences of data in general for SeqGANs, Time-GANs focus on generating specifically time series data.

GANs offer the possibility of generating realistic synthetic time series data as well as a trained discriminator network that can discriminate between real and fake data, which can be an advantage for systems trying to detect fraudulent information. In some instances, Time-GANs have been able to produce realistic, high-quality time series data that are hard to tell apart from actual data [70]. In other instances, however, the created data might not be of high enough quality or might not accurately reflect the complexity of the original data [53]. Although GANs and Time-GANs can be successful, there are some weaknesses to both architectures that may have had a negative impact when trying to generate eye-tracking data.

Stability tends to be a common issue when developing a GAN, as the generator can start producing only limited variations of data, also known as mode collapse. Also, the complexity of the data can play a role in how long the model will take to converge, if ever. This may also have an impact on the quality of the generated data, which may not be optimal, especially with sequential data and its complexity. Sequential data can have missing values which need to be handled, this can break patterns and add extra complexity to training GANs if not done carefully. In addition to the computational requirements, which are not low since GANs require big amounts of data and resources to train two neural networks at the same time, GANs tend to be trained for one specific task and may struggle to generalize and be adapted for other tasks. This implies that even though the Time-GANs used to forecast eye-tracking data were built to be used with such sequential data, their structure and pre-trained weights may have played a role in not properly capturing the long-term dependencies of the eye-tracking data.

Despite this, it is important to keep in mind that Time-GANs are still a relatively new method that is actively being investigated. It is therefore likely that cutting-edge methods and models will solve these problems and offer superior performance for our eye-tracking task, some of which have already been improved upon. Wasserstein GANs is an alternative to a traditional loss function for GANs that improves stability, convergence as well as the quality of samples [71], thus justifying why the best results achieved were usually from Recurrent Conditional Wasserstein GAN (RCWGAN). These weaknesses justify that in order to achieve more positive results with the use of Time-GANs, and as noted in the limitations section, more time would have been needed to further train the models. The results achieved with Time-GANs in Ref.$\sim$ [53] motivate us to further investigate this task and adopt new implementations.

2.4.2 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are a type of deep-learning artificial neural network designed to process sequential data. Their recurrent structure allows for the data to feed back into the neurons creating a memory of past data. Such characteristic has allowed RNNs to perform tasks such as NLP [72], time-series forecasting [73], music generation [74], and anomaly detection [75] among others. The first application of an RNN dates back to 1986 for speech recognition tasks [76, 77]. RNNs were developed as an improvement on traditional feed-forward networks that were used for speech recognition but were underperforming due to the temporal dependencies. Since then, many architectures expanding from vanilla RNNs have emerged, some of them include Long Short-Term Memory networks (LSTM), Gated Recurrent Unit (GRU), and Bidirectional RNNs (BRNN).

The key concept behind RNNs is their recurrent structure. A recurrent unit of an RNN is a node that has a feedback connection from its output back into its own input, allowing information from the previous time

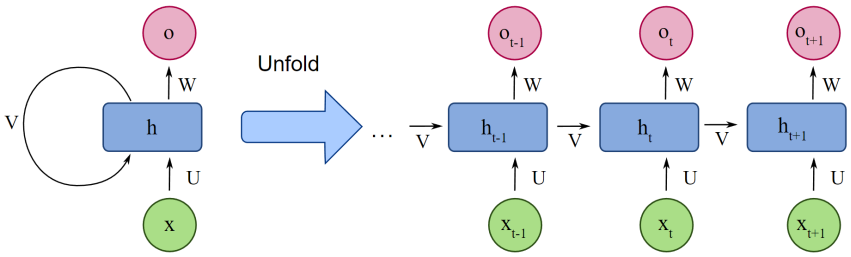


Figure 2.3: Vanilla RNN unfolded. From Ref.~[78].

step to be fed back in the next time step. These recurrent units have an internal state that is updated at each time step based on the input and the previous state. This internal state allows the network to preserve information from previous time steps and to capture dependencies and context in the sequence.

As it can be seen in figure 2.3, when the unit is unfolded, it can be seen how the output at time step h-1 is fed back into the same unit at time step h along with the input. Such architecture attributes RNNs with many advantages over traditional ANNs. Being able to process sequential data and store past dependencies as memory in their internal states allows them to be successful in sequential data processing. Moreover, RNNs are flexible and can be adapted to handle variable-length sequences. Despite their success, RNNs also includes a handful of drawbacks that have been the drivers for new recurrent architectures to be developed. Similarly to other ANNs, RNNs are affected by vanishing/exploding gradients, in which the gradients of the network become so small it slow downs the training significantly, or where they become so large it does not allow for convergence, respectively. Additionally, RNNs can be expensive to train as their recurrent nature and sequential processing do not allow for out-of-the-box parallelization, slowing down the process. Since the internal states of the recurrent units are hidden, RNNs have a very low explainability, and it can be hard to understand why it behaves in a particular way. Lastly, they are data-hungry, as they need large amounts of data to not overfit. A small amount of data can lead to wrong predictions as the network can easily remember the training data. Vanilla RNNs in particular, despite performing well with short-term dependencies, have been quite unsuccessful at detecting long-term dependencies.

With these drawbacks in mind, further research was done to develop RNNs that could address these issues and, in particular, perform better with long-term dependencies.

2.4.3 Long Short-Term Memory

Long Short-Term Memory networks (LSTM) were introduced in 1997 by Sepp Hochreiter and Jürgen Schmidhuber in their paper Ref.~[79]. LSTMs extend RNNs by adding an internal memory controlled by multiplicative

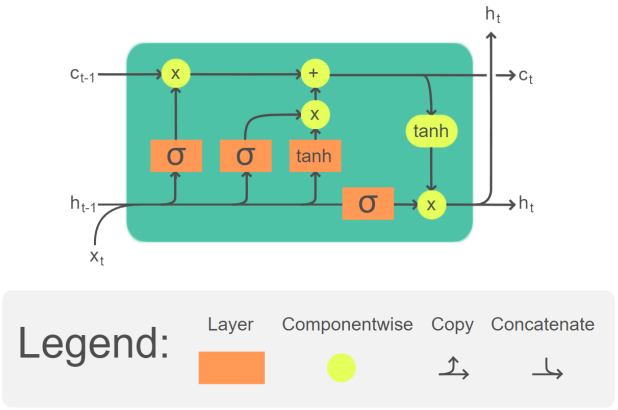


Figure 2.4: LSTM memory cell. By Guillaume Chevalier. From Ref.∼[85].

gates which makes them adequate tools for tasks such as NLP [80], time series forecasting [81], and video analysis [82], among others. Popular tools used by many individuals are built, currently or in the past, with LSTMs such as Google Translate, Siri, and Alexa [83]. LSTMs have been especially successful since 2015, likely due to the work in Ref.∼ [84] which proposes a clear approach to understanding LSTMs. Figure 2.4 shows the internal structure of an LSTM memory cell. The idea behind them is that the vector at the top, from C_{t-1} to C_t contains the cell state, which is the memory of the network capable of feeding back into the network information of the past states. The information in the cell states is controlled through three gates that regulate which information can alter the state and which cannot. The leftmost part of the cell is called the forget gate, and it utilizes a sigmoid layer to do so. The middle part is the input gate which uses a combination of a sigmoid layer and a tanh layer to decide which new information will be added to the cell state. The combination of these two steps controls the information that is being stored in the cell state. The last part of the cell, at the right, decides which will be the output of the cell, also known as the output gate. The output of the cell will be a filtered version of the cell state which is decided by the sigmoid layer of the input and a tanh layer of the current cell state combined to generate the final output. All the gates produce a value between 0 and 1, which decides the amount of information they let through, not necessarily all of it or any of it.

Thanks to these memory cells, LSTMs improve over RNNs in many of their weaknesses. For instance, the control of the cell state and the longer preservation of information allows LSTMs to have a much better handling of exploding/vanishing gradients problems. Additionally, one of the main issues of traditional RNNs is the inability to model long-term dependencies, LSTMs were designed to successfully do such things. Finally, LSTMs generalize better than traditional RNNs, which allows them to be used in a variety of problems and perform well with previously unseen data. Although the success of LSTMs is big and has been proven to outperform traditional RNNs in many applications [86], they do suffer

from a few drawbacks. The key component of LSTMs, their memory cell, is more complex than in traditional RNNs, this makes them more expensive to train to require more computational resources and data. A small dataset can make an LSTM network easily overfit due to its long-term memory, whereas with big datasets it can be complex to fine-tune the hyperparameters. The popularity of LSTMs has been backed by the releases of variants such as Bidirectional LSTM (BiLSTM) [87], Stacked LSTMs [88], Peephole LSTMs [89], and a popular simplification that allows faster training Gated Recurrent Units (GRUs) [90].

2.4.4 Transformers and architecture

Transformers are a relatively new kind of Neural Network that follows the encoder-decoder type of structure [91]. Although they share this type of architecture with Autoencoders, Transformers models are not a subtype of Autoencoders as they perform substantially different tasks. Opposite to GANs, Transformers have seen most of their success in Natural Language Processing tasks (NLP) and, at the time of writing, are considered state-of-the-art. Some popular Transformer models include BERT [92] which can perform a wide range of NLP tasks such as sentiment analysis and question answering, GPT-3 [93] which, among other tasks, is capable of text summarizing and machine translation, and T5 (Text-to-Text Transfer Transformer) [94] also capable of text generation and natural language understanding. Additionally, Transformers have also been used for time-series forecasting tasks. Although they may not be as popular as RNNs or LSTMs in this field, some architectures such as Transformer-XL[95] designed to handle long-term dependencies in time series data, and LogTrans [96] which tries to improve upon classical Transformers issues. These are some examples of Transformer models that have been used to model long-term temporal dependencies and have achieved high performance.

Transformers have three key components that help them improve upon other ANNs in sequential tasks, Positional Encdoing, Self-Attention Mechanism, and Multi-Head Attention. At the start of the encoder and decoder, there is an input embedding layer that maps the input into a high-dimensional vector space. Despite that, the model has no knowledge of the position of the input elements. In order to counter that, Positional Encoding is added to the sequential data to be able to keep track of its order. Positional Encoding consists of a set of values that encode the position of each element in the data and are added to the representation of the respective elements. The sine and cosine functions of various frequencies are typically used to encode the position since they can generate a unique encoding for each point. The relative positions of the elements are captured by the encoding values, which also reflect the sequence's original order. In other architectures, the sequence of the input was encoded at the initial layers of the network, thus allowing it to learn to capture the order of the sequence. Providing such information already in the data allows the transformers to focus their attention on other parts of the data. Next,

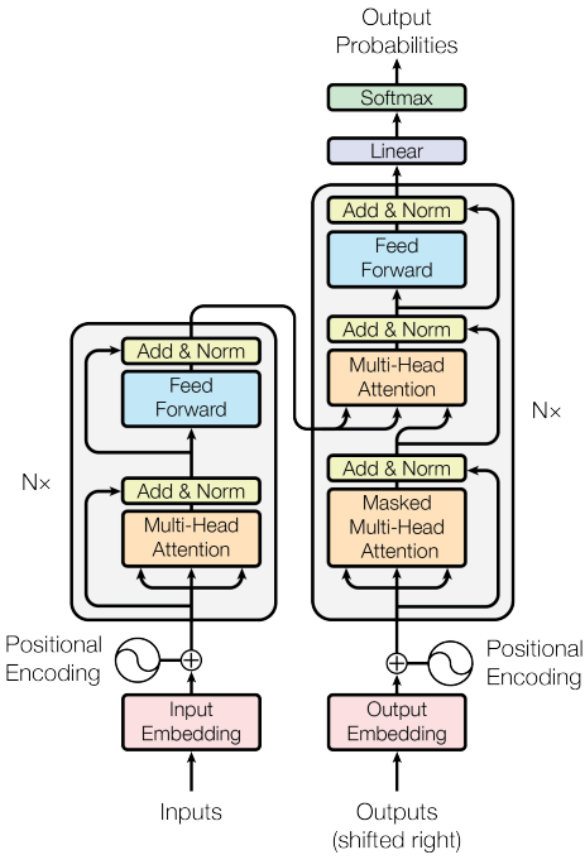


Figure 2.5: Transformer architecture from Ref.∼[91].

the Self-Attention Mechanism is one of the most relevant improvements over RNNs. In traditional RNNs, the state of the cell is dependent on the previous state and the current input, which limits how much the network can look back in its inner memory. This mechanism allows Transformers to virtually attend to all the sequences at the same time when making a prediction, allowing it to weigh the importance of each element in the input sequence for each prediction, successfully capturing long-term dependencies and making predictions based on the global context of the input. Such attention is performed with the use of the query, key, and value vectors. The key and value vectors are representations of every element in the input sequence, whereas the query vector is a representation of the element that is now being processed. The dot-product between the query and key vectors is used to compute a weight for each value vector, representing the importance of each element in the input sequence for the current prediction. The final attention-weighted representation of the input sequence is created by combining these weights with the value vectors. Lastly, the Multi-Head Attention Mechanism allows performing the Self-Attention operations just described, at the same time, for multiple heads. Each head computes its own attention-weighted representation of the

input sequence through its own unique collection of query, key, and value vectors. The final output of the Multi-Head Attention mechanism is the concatenation of the outputs from each head, which is then passed through a final linear layer to generate the Multi-Head Attention mechanism's final output. Multiple aspects of the input can be attended and different dependencies can be captured at the same time thanks to the use of multiple heads. In the decoder part of the model, the first Multi-Head Attention layer utilizes a mask in order to protect future information from influencing the prediction of the current element. For instance, in the example sentence "I am fine", when processing the word "am", the Multi-Head Attention layer should not know about the word "fine".

Transformers have many advantages which make them suitable for sequential tasks. The most notable one is their Self-attention mechanism, this allows the model to give a different weight to different parts of the input allowing it to better understand the importance of the data rather than treating all of it equally. As an extension, this can be done in parallel rather than sequentially, also known as Mulit-head attention. This allows the model to be trained much faster and also to further improve the understanding of the input data by attending to multiple parts at the same time. As mentioned, Transformers are very good at handling long-term dependencies which allows them to achieve high performance in NLP and time-series forecasting tasks, as they can capture dependencies between distant time steps as well as disentangle complex structures. Transformer models are also capable of handling variable-length inputs, which is an advantage over RNN and LSTM, an example of this being beneficial is, for instance, in machine translation when the input and the output can be of different lengths. Additionally, since Transformers do not use the recurrent structures which are present in RNNs and LSTMs, they are capable of being pre-trained on an extremely large dataset, and later fine-tuned and further trained for a more specific task, this also allows them to achieve high-performance scores faster.

Although the great success of Transformers, there has been some research trying to disprove their efficiency in time-series forecasting. Ref.∼ [97] conducts research where Transformer models are compared against very simple one-layer linear models. In their work, they show how these simple models outperform the highly complex Transformers. They question the validity of Transformers in long-term forecasting tasks as their mechanism of self-attention leads to an inevitable loss of temporal information. Despite their conclusions, it is important to note that they only focus on univariate regression and multivariate regression, which uses the same features for the input and output, or single univariate target regression that takes several inputs and produces a single distinct output. No study focuses on multivariate regression, which has both many inputs and numerous distinct outputs [98].

Therefore, we still believe that Transformer models will be able to successfully capture the time dependencies present in the eye-tracking data and improve upon the previous research conducted with GANs, allowing for the generation of realistic, previously unseen, eye-tracking data.

2.5 Evaluation Metrics

When creating eye-tracking data it is important that it is as realistic as possible. In order to measure such a thing, it can be compared with the real samples the network was trained on. The main components that need to be evaluated are: (i) that the generated data comes from a similar distribution to the real data. (ii) that the generated data reproduces the time dependencies and long correlations present in the original data. In order to compare the similarity between distributions we will use the Jensen-Shannon divergence (JSD) [99]. The JSD is a symmetric and smoothed version of the Kullback-Liebler (KL) divergence and it is used to compare the similarity between two probability distributions. It is defined as

$$JSD(P||Q) = \frac{1}{2}KL(P||\frac{(P+Q)}{2}) + \frac{1}{2}KL(Q||\frac{(P+Q)}{2}), \tag{2.1}$$

where Kullback-Liebler (KL) is defined as:

$$KL(P||Q) = \sum_{x} P(x)\log(\frac{P(x)}{Q(x)}), \tag{2.2}$$

where P and Q are two probability distributions.

In addition, commonly used performance measurement metrics will also be used, these include both Mean Squared Error (MSE), which is defined as:

$$MSE = \sum_{i=1}^{N}(x_i - y_i)^2, \tag{2.3}$$

and Mean Absolute Error (MAE), defined as:

$$MAE = \sum_{i=1}^{N}|x_i - y_i| \tag{2.4}$$

where x, are N the real values and y, are N the predicted values.

Both metrics provide a measure of how well the model is performing in terms of minimizing the error between predicted values and actual values. MSE penalizes large errors more severely than minor errors by averaging the squared disparities between the expected and actual values. All errors are treated equally by MAE, which computes the mean of the absolute disparities between the predicted and actual values.

Three elements of the data can be measured in order to find out how similar the generated and the real data are. Firstly, gaze velocity measures the rate of changes in gaze direction over time and helps study the gaze behavior, speed, and smoothness of gaze movements. Secondly, the gaze direction is the orientation of a person's eyes relative to some reference frame, at a particular moment in time. It is typically measured by calculating the angle of the direction of the gaze according to the x-axis. It helps researchers study how people allocate their attention as well as how attention is influenced by other factors such as motivation and task demands. Finally, the angle between two consecutive fixations

is measured, which adds to the information that can be studied about the user's gaze behavior as well as the cognitive and motor processes. These measurements have their own distribution that can be compared to the generated data. In order to do so the Euclidean distance, also known as L2 distance, for each of them will be computed and compared to the generated data distribution for the corresponding element. The L2 distance is a widely used measurement in machine learning as a similarity metric and it is usually preferred over the L1 distance as the squared differences between values are more meaningful than the absolute differences. The Euclidean distance (L2) is defined as:

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}, \tag{2.5}$$

where q and d are two sequences of points in an n-dimensional space.

2.6 Previous works & state of the art

This thesis is an extension of another thesis project named *Can GANs replicate eye-gaze trajectories?* by Marit Øye Gjersdal. In the project, instead of using Transformers to generate eye-tracking data, they used Generative Adversarial Networks (GAN). The architectures of GANs used were those more appropriate for time-series data processing, these include TimeGAN [100], Recurrent Conditional GAN (RCGAN) [101], Conditional Sign-Wasserstein GAN (SigCWGAN) and Recurrent Conditional Wasserstein GAN (RCWGAN) [102]. It is particularly appropriate for DA that these GAN architectures were created to replicate data sequences that were similar to the ones they had been trained on.

All the previously described GANs were implemented, trained on progressively more complicated eye-tracking data, and contrasted against one another to explore the limitations of each one of them. Similar to our effort, they aimed to produce realistic eye-gaze trajectories of the eyes. Initially, they generated synthetic data using Vector Autoregressive (VAR) with one and two dimensions. With just knowledge of recent past motions, the one-dimension VAR data is generated step-by-step and has short-temporal dependencies. The technique is made slightly more difficult by the two-dimension data's incorporation of the same temporal components as well as a feature dependency across the two dimensions. Later, they employed intermittent process data, which is thought to be more complex than VAR-generated data since it contains two different processes, fixations, and saccades that change periodically. The intermittent processes' temporal dependencies are longer than those of the previous data, which further increases complexity. Finally, they also used real eye-gaze trajectories, these, although similar to the intermittent process data, are thought to be even more complex than all previous data and are also the only non-synthetic data used. There were a total of four experiments, one for each different dataset respectively, and all models

were used in all the experiments. The models were trained to predict the following 50 data points given the first 50 initial data points. The models were tested in the first two tests. 50 beginning points were likewise provided to the models for the remaining two trials, however, in the case of the conditional GANs, 100 future steps were predicted.

SigCWGAN performed the best in the first experiment with an L2 score of 0.163 ± .009, RCGAN performed the best in the second experiment with an L2 score of 0.09 ± .02, RCWGAN performed the best in the third experiment with an L2 score of 0.55 ± .03, and RCWGAN finally performed the best in the fourth experiment with a distance intensity of 0.5 ± .2. These results appear to be positive by themselves, however, they also performed the Kolmogorov-Smirnov test (KS), which examines the likelihood that a sample is taken from the same distribution. This measurement has an acceptance rate of 1.3, in the best scenario for all experiments and all models were achieved by RCGAN with the 2-dimensional VAR data with a score of 24 ± 12, much higher than the acceptance rate. Such results imply that there is a low probability that the data generated by the GANs are from the same distribution as what they had been trained on.

This research demonstrates how challenging this task is even though the findings are not uniformly favorable. Additionally, promising results with the VAR and intermittent processes suggest that more accurate eye-gaze trajectories could be produced with additional investigation. Additionally, several constraints, such as time limitations for additional testing and training and the different possible representations of the data, which might be continuous or discrete, could have led to less-than-desirable results. We hope that Transformer models will be able to better capture the temporal dependencies of the data and consequently create more realistic eye-gaze trajectories.

In addition to the different research described in the previous sections, a variety of research shows different applications of Transformers in the field of eye-tracking data. Ref.∼ [103] uses Transformers as a complementary tool to aid Convolutional Neural Networks (CNN) in a gaze estimation task, which consists of determining a person's gaze direction or location using information about their eye movements. Another task suitable for Transformers consists of image classification using eye movement data. Ref.∼ [104] combines eye movement data and a Vision Transformer to more efficiently scan through the image and focus on the most important parts to accurately classify the objects in it. Attention prediction is also a task that can be tackled thanks to the combination of Transformers and eye-tracking data. Ref.∼ [105] used Transformers and CNNs to predict where drivers focus their attention while driving and used eye-tracking data gathered in the same scenarios to validate the efficacy of their model. Visual saliency is another example of a task in which Transformers can improve upon older research consisting mainly of CNNs, as the use of Transformers can model long-term dependencies. Visual saliency is the quality of specific portions in an image or video that draw the viewer's eye and stand out from the background, and in Ref.∼ [106] they create a Visual Saliency Transformer (VST) to do so. In Ref.∼ [107]

they demonstrate a cutting-edge human-robot interface that can detect and realize the user's manipulation intention purely through sight. To achieve such a thing they use a Transformer model paired with real-time eye-tracking data, which is used to merge visual information and human attention to improve grasp detection. This consists of identifying and localizing regions in an image or point cloud data where an object can be picked up by a robotic hand. Finally, the authors of Ref.~[108] propose a transformer model named GazeTransformer for predicting egocentric gaze points in virtual environments on head-mounted displays (HMD). The model focuses on data modalities provided by the eye-tracker or HMD and allows forecasting multiple types of eye movements. Their results outperform current state-of-the-art approaches in forecasting egocentric gaze points in virtual environments.

It is likely that additional research backs up the fact that Transformers and eye-tracking data work well together, it is our task to further validate this claim and successfully generate realistic eye-tracking data using this state-of-the-art architecture.

Chapter 3

Methods and data description and processing

3.1 Data description

The eye-tracking data used for this thesis was collected at OsloMet using the Eye-link Duo, a cutting-edge eye-tracking system with a maximum frequency of 2000 Hz and a precision of 0.1 degrees of visual angle. The data was collected with approval and consent, as stipulated by Skit (reference number: 129768). This data is the same used in the paper released after the thesis our work is based on [53]. A frequency of 200Hz has been selected for sampling the data, in other words, the data is collected 200 times per second. The data was collected in a task where the participants tried to find pre-selected targets, in this case, the targets being objects and individuals from the popular book "Where is Waldo?". The data is presented in two dimensions representing the pixels of the screen used to perform the task. The task was performed with eight different images of the book and the participants had two minutes per image to find the pre-defined targets. Although it is unlikely that two minutes was enough time to find all targets, such a short time ensured the participants remained engaged throughout the experiment.

Three important measurements are collected from this data and later used to compare with the generated data. Firstly, the velocity magnitude (v), secondly the angle of a given velocity with the horizontal axis (θ), and finally the angle between two consecutive increments (ψ). Therefore, the distributions presented by these three measurements will be compared against the respective distributions of the generated data, which will allow us to measure the likelihood of the generated data appearing realistic. Additionally, we also measure the displacement of the horizontal and vertical axis, which gives us further information on the similarity of the data.

Figure 3.1 shows an illustration of the main properties that will be analyzed when comparing the real data with the generated using different AI approaches: P1, P2, and P3 are three different consecutive points from the dataset. Between two points, P1 and P2 for example, the velocity ϕ can

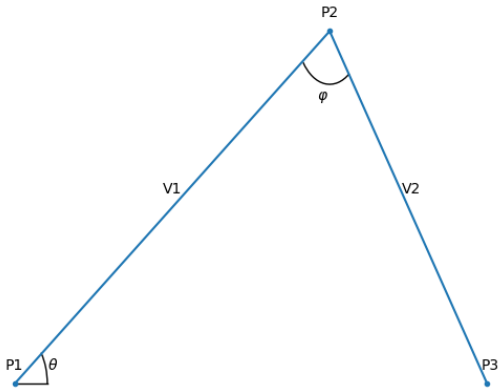
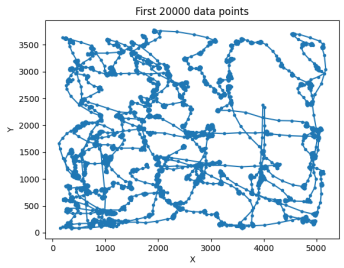
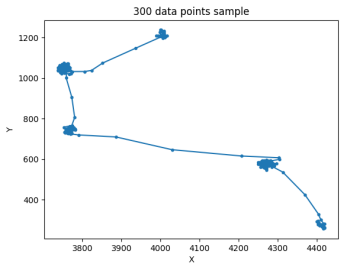


Figure 3.1: Illustration of the main properties being analyzed.



(a) First 20.000 data points of our dataset.



(b) 300 points example of eye-tracking recordings.

Figure 3.2: Eye-tracking positions.

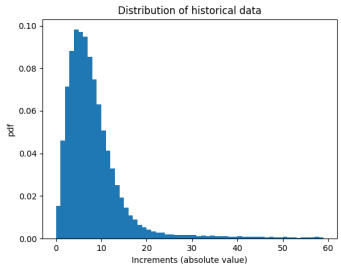
be measured as the absolute difference between their coordinates:

$$\phi = |P2 - P1|, \tag{3.1}$$

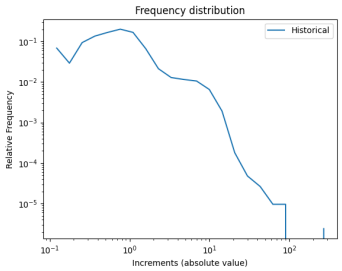
where P indicates the position of a point. Additionally, for every point, the angle between the line going to the next point and the X-axis will be measured θ, and finally, the angle between the lines connecting a point with the previous and the next will also be measured φ. Moreover, the horizontal and vertical displacements will be computed to aid the comparison between real and generated data, allowing a more exhaustive analysis of the models.

For our data, we expect the eye movements to be similar to the free examination presented in section 2.1 figure 2.1. As the task is to find predefined targets, not only do we expect to find movements trying to gain as much information as possible, but we also expect movements trying to focus on precise points such as the ones in the figuring the people's ages task.

Figure 3.2(a) shows the first 20.000 data points of our dataset. X and Y represent the horizontal and vertical axis of the screen used to show the



(a) Data distribution for the velocities of Eye-tracking.



(b) Frequency distribution for the velocities of Gaussian Noise.

Figure 3.3: All distribution representations for Eye-tracking.

picture to the participants, respectively. As it can be observed, most of the points are scattered around the canvas, also some clusters of points appear to be forming, which may indicate some areas of interest to the participant and may contain one of the indicated targets.

3.2 Datasets

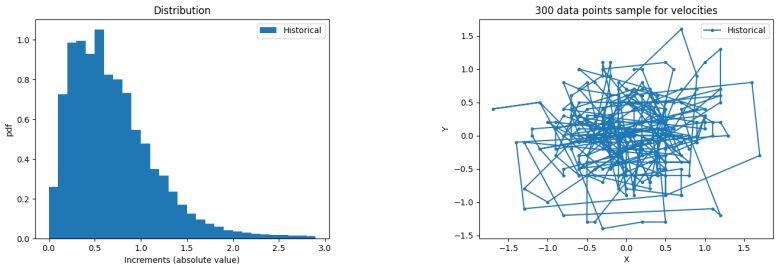
3.2.1 Eye-tracking dataset

The main dataset used is formed by real eye-tracking data. The dataset used consists of 421.124 recordings in a time span of approximately 13 minutes. As mentioned, we will compute different measurements which will allow us to compare the similarity between the real and the generated data.

Showing all the >400.000 points in the same plot does not provide many insights as in such a long time, the recordings are basically scattered all over the canvas. Therefore, we choose to show the distribution of the whole dataset, and examples of 300 points at a time, which can give a much better sense of where the eyes are fixating, and the saccades between fixation points.

Figure 3.2(b) shows the plot of the eye-tracking recordings of 300 consecutive points. As it can be seen, there are 5 fixation points that are interconnected with saccades. It can be easily counted that there are 14 points forming the saccades, therefore, the rest of the 286 points are forming the fixations. This clearly justifies how the distribution of 3.3(a) is a right-skewed long-tail distribution. Since most of the recordings are part of fixations, it does make sense that most of the data are in lower increments. The remaining points, the saccades, form the tail of the distribution.

In 3.3(a) we can see how the distribution of the training data resembles a long-tail distribution. In order to plot the distribution we have computed the standard deviation defined as $\sqrt{X^2 + Y^2}$ for all the points. This plot shows how dispersed the data is in relation to the mean. It can be seen how most of the values are between 0 and 10 increments, which demonstrates that most of the consecutive points are close to each other. In lesser



(a) Data distribution for the velocities (ϕ).

(b) 300 points example of the velocities of eye-tracking recordings.

Figure 3.4: Eye-tracking velocities.

quantities, there are also big increments that represent big movements.

For training, testing, and validation though, we will be using the velocities of the eye-tracking data. To compute the velocity we simply need to subtract the current position for the following one. This is done in order to slightly facilitate the training process. When talking about eye positions, it is very unlikely that any exact position will be repeated at any time, therefore we argue that training on the positions may be very hard to replicate.

Figures 3.4 (a,b) show the distribution and an example of the initial 300 points respectively of the velocities dataset that will be used for training, testing, and validation, instead of the positions dataset seen in 3.2(b). Since we will be predicting the distribution of the velocities, 3.4(a) is effectively the distribution of ϕ defined in 3.1.

3.2.2 Gaussian Noise dataset

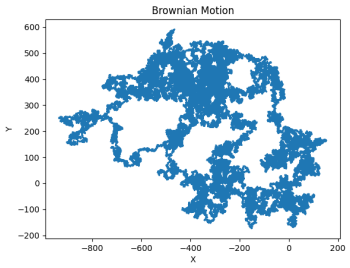
The second dataset that will be used consists of random data generated with the Brownian Motion. The Brownian Motion (BM) is a type of random motion exhibited by particles suspended in a fluid, such as air or water, which shows the random collisions that happen between particles and molecules of fluid. It tends to exhibit a zigzag erratic path with no predefined direction or pattern. By computing the differences between the positions we are obtaining Gaussian Noise data, therefore we will refer to this data as Gaussian Noise dataset for the rest of the thesis.

Such data can be generated using Python and NumPy by:

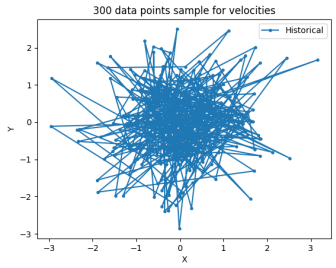
```
np.diff(np.cumsum(np.random.normal(0,1,N), np.random.normal(0,1,N)))
```

where N indicates the total number of points to generate. This dataset is created by generating two different normal distributions for both X and Y, in order to resemble as much as possible the eye-tracking dataset.

Similarly to the eye-tracking dataset, we created the same number of points to model this distribution, 421.124 points in order to make the two tests as comparable as possible. The same attributes from 3.1 will be used

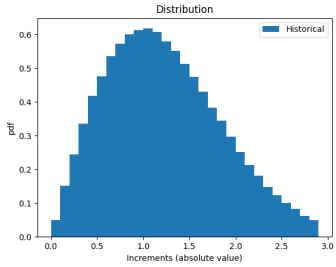


(a) Original Brownian Motion used.

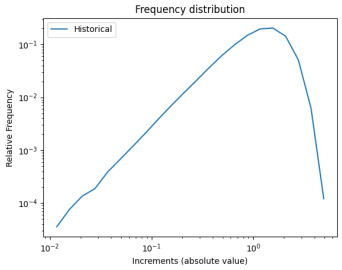


(b) 300 points example of the velocities of Gaussian noise.

Figure 3.5: Examples of Brownian Motion and its respective Gaussian Noise velocities.



(a) Data distribution for the velocities of Gaussian noise. (ϕ)



(b) Frequency distribution for the velocities of Gaussian Noise.

Figure 3.6: All distribution representations for Gaussian Noise.

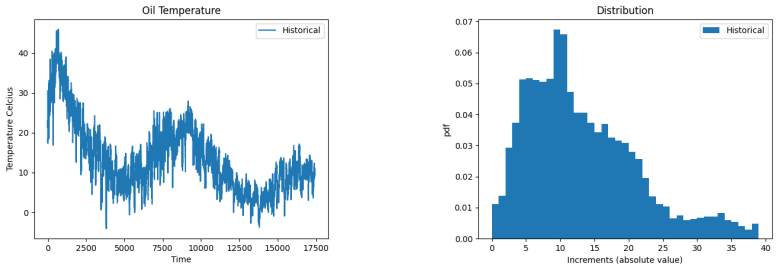
to assess the quality of the forecasting methods to model such random distribution.

Figure 3.6(a) indeed shows how the distribution of the dataset is very similar to that of a Gaussian distribution. The Gaussian distribution describes the motion of different natural phenomena, which include the particles described by BM.

Following the same process as previously described, the velocities between consecutive points will be obtained and used for training, testing, and validation. Figure 3.5(b) shows the initial 300 points of the Gaussian Noise velocities and, since it follows a Gaussian distribution, by using the velocities we are actually training, testing, and validating on Gaussian Noise. Again, since we will replicate the distribution of the velocities, 3.6(a) is effectively the distribution of ϕ defined in 3.1.

3.2.3 Empirical dataset: oil temperature

An additional dataset with no relation to the eye-tracking task will be used to further validate the performance of the models used. The dataset, in particular, is the Electricity Transformer Temperature dataset. This dataset contains information about the electric power long-term deployment and it was gathered in two separate counties in China. The variant we will



(a) Oil Temperature values. (b) Oil Temperature distribution.

Figure 3.7: All distribution representations for Gaussian Noise.

be using is the ETTh1 and it has 17.420 rows divided into seven columns. The column we will be focusing on will be Oil Temperature (OT), and the task for this dataset will be to predict this variable using the other columns, which are power load features, as indicators. This dataset was chosen since it is commonly used for time series forecasting baseline tasks and because the Transformer, Informer, and NLinear also utilized it in their respective publications.

Since this dataset is used for univariate forecasting, the evaluation of the performance of the models will be done by simply comparing the predicted values to the originals as well as the histogram of the distribution.

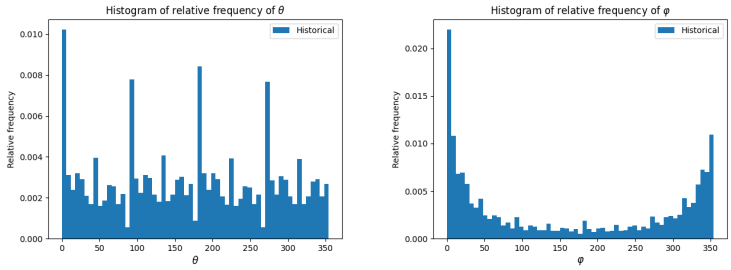
Figure 3.7(a) shows all the values of the oil temperature across all time measurements. As mentioned, we will try to forecast part of this sequence. Figure 3.7(b) shows the distribution of the oil temperature, which we will use to compare the distribution of the oil temperature forecasted.

3.3 Processing the eye-tracking data

For the two initial datasets, eye-tracking and Gaussian Noise, more in-depth testing and validation will be carried on. This will be done in order to ensure the quality of the forecasting models for the task at hand, which is forecasting eye-tracking velocities. Therefore, two separate measurements will be done in order to assess such quality. Firstly, the distribution will be presented in the form of histograms for each of the properties described in 3.1. Secondly, the autocorrelation of the horizontal, vertical, and time displacements will be presented in the form of autocorrelation plots.

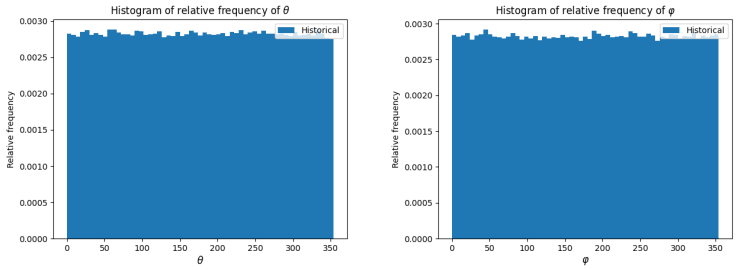
3.3.1 Histograms of the main properties

The properties illustrated in 3.1 will be measured in the form of three different histograms, one for each of them. These histograms show every single angle between 0 and 360 on the X-axis and relative frequency measurement on the Y-axis. This indicates how often a particular degree appeared in the respective measurement of the consecutive points. The histograms for the three properties will be presented and compared



(a) Angles between a point and x-axis (θ).

(b) Angles between consecutive points (φ).



(c) Angles between a point and x-axis (θ).

(d) Angles between consecutive points (φ).

Figure 3.8: θ & φ properties described in 3.1 for the Eye-tracking dataset (a,b) and Gaussian Noise dataset (c,d).

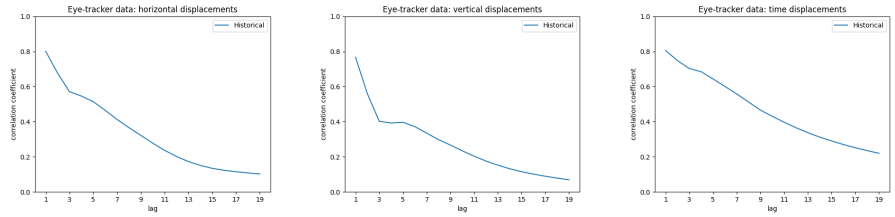
between the ones from the original data and the ones from the forecasting models used.

Eye-tracking dataset

Figure 3.8(a,b) shows the two of main attributes that will be analyzed to determine the similarity between the original distribution and the predicted distribution by the different forecasting models, for the Eye-tracking dataset. Figure 3.8(a) shows the number of points per angle between the point and the X-axis. This plot shows a repeating pattern every 90 degrees which gives insights into the behavior of human vision. Figure 3.8(b) shows the number of points for each angle between consecutive points. In this case, extreme values can be observed at 0 and 360 degrees while the rest show a more moderate behavior.

Gaussian Noise dataset

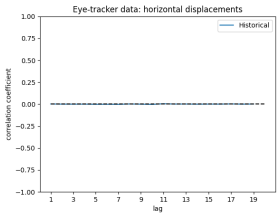
Unlike the eye-tracking data, figure 3.8(c,d) for Gaussian Noise shows no patterns in the data, which is expected in a random distribution. A static behavior can be observed, with no significant increments or decrements apart from the ones generated by the random nature of the distribution itself. Since the patterns in this dataset appear to be simpler, we expect the models to perform significantly better with this distribution rather than with the eye-tracking distribution.



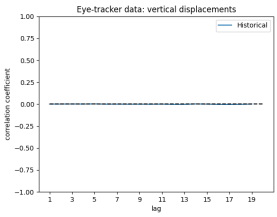
(a) Autocorrelation of horizontal displacements.

(b) Autocorrelation of vertical displacements.

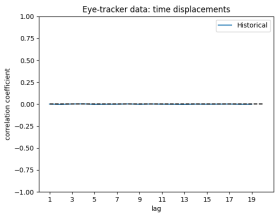
(c) Autocorrelation of time displacements.



(d) Autocorrelation of horizontal displacements.



(e) Autocorrelation of vertical displacements.



(f) Autocorrelation of time displacements.

Figure 3.9: Autocorrelations for the Eye-tracking dataset (a,b,c) and Gaussian Noise dataset (d,e,f).

3.3.2 Autocorrelation of distributions

Autocorrelation is used in order to measure the similarity between observations as a function of the time lag between them. It measures the degree of correlation between a variable current value and its past values. High autocorrelation stipulates that a current value is dependent on its past values, which can be useful in predicting future values. A low autocorrelation, therefore, indicates the opposite. Similar to the histograms, the autocorrelation plots will be presented and compared between the ones from the original data and the ones from the forecasting models used.

Eye-tracking dataset

Figure 3.9(a,b,c) shows strong autocorrelations with the lesser lags, this autocorrelation fades away as we introduce more. This indicates that the current values are strongly influenced by the most recent past ones. The further we move away from the current value, the relation to the other points decreases.

Such plots indicate that the initial lags are more significant. Although the Deep Learning models do not explicitly learn the autocorrelation, they may implicitly learn it by simply training with the data.

Gaussian Noise dataset

Opposed to the eye-tracking data, the Gaussian noise shows absolutely no autocorrelation in all three measurements. Figure 3.9 (d,e,f) illustrate

how little autocorrelation the Gaussian noise has at different lags. It is expected since the data is random. Despite the slightly simpler distribution to replicate, the absence of autocorrelation in the data will increase the complexity of the models in learning such distribution.

Therefore, in order to give the models the best chance to learn the distribution of the Gaussian noise, no lags will be introduced in the data.

Apart from the autocorrelation figures, a table with the correlation coefficients between the original horizontal velocities and the predicted horizontal velocities, for all models and datasets will be provided. The correlation coefficient is a statistical measure that indicates the degree of association between two variables, and if they move in the same or opposite directions. This table will provide a comparison of the predicted output with the real data at timestep -1 (past), with the real data at the same timestep (present), and with the real data at timestep +1 (future). This will allow us to investigate how much the forecasted data is correlated with the real data.

The past correlation is defined as:

$$\sigma(Y^{(data)}(t-1), \hat{Y}^{(data)}_{(alg)}(t)), \tag{3.2}$$

while the present correlation is defined as:

$$\sigma(Y^{(data)}(t), \hat{Y}^{(data)}_{(alg)}(t)), \tag{3.3}$$

and the future correlation is defined as:

$$\sigma(Y^{(data)}(t+1), \hat{Y}^{(data)}_{(alg)}(t)). \tag{3.4}$$

For all three definitions, σ denotes the correlation function, Y denotes the original targets of the dataset defined as $data$, at timestep t. $\hat{Y}$ denotes the output of the model used, defined as alg for the dataset $data$, at timestep t.

3.4 Methods

In order to try to fit the distributions of the previously described datasets to be able to forecast future values, we will use a variety of Deep Learning models. These models include a vanilla implementation of a Transformer [91], a more specialized Transformer model for time-series forecasting called Informer [109], a simple linear Neural Network called NLinear [97] and a Markov model used in Ref .[53].

3.4.1 Transformer

The Transformer architecture was first published by Google in 2017 [91]. Since then, Transformers have had impressive success in NLP tasks and have become state-of-the-art in that field. Transformers have an encoder-decoder architecture and present a key innovation that makes them perform better than other Deep Learning architectures, the self-attention mechanism.

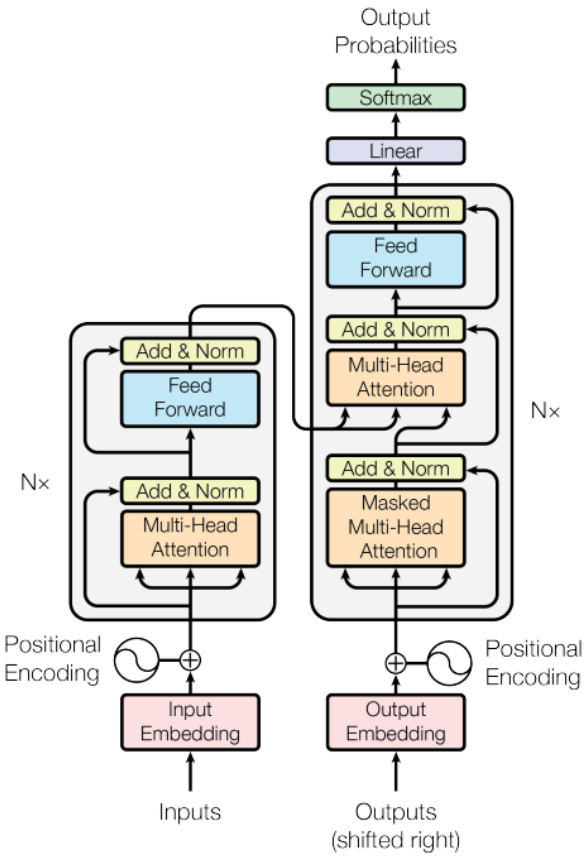


Figure 3.10: Vanilla Transformer architecture taken from Ref.∼[91].

The self-attention mechanism allows the model to weigh the importance of different parts of the input when making predictions. This is done by creating three vectors from the input sequence, the query, key, and value vectors, which are computed by multiplying the input sequence with a respective matrix with learned weights for each of the vectors. Later, the dot product of all the queries and keys vectors is computed and scaled, to compute a similarity score between vectors. The similarity scores are normalized using a softmax function. These scores are used as weights for the value vectors, which are then added up producing a weighted sum. This output of the self-attention mechanism is then combined with the original input sequence thanks to a residual connection.

One of the key parts of this self-attention mechanism is that it does not involve recurrent connections, unlike RNNs, which allow for Transfer Learning and parallelism. A Multi-Head Attention is when N self-attention layers are run in parallel and the outputs are combined in order to attend to the input sequence in the most sensible way possible. The transformer model combines such a self-attention mechanism as well as normalization, linear and softmax layers. Our Transformer, with the best hyperparameters found during optimization, has a total of 5.272.578 trainable parameters.

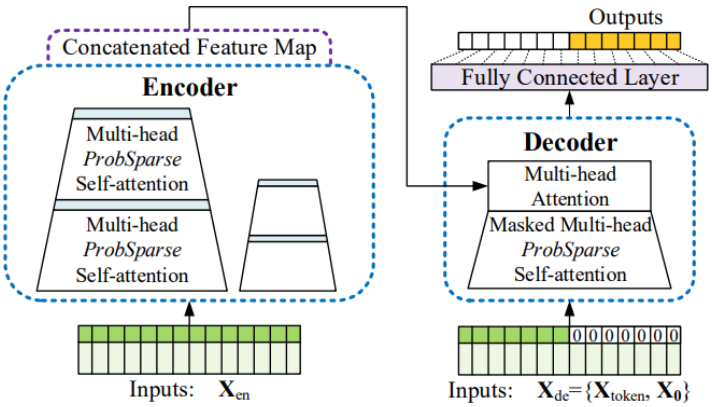


Figure 3.11: Informer architecture taken from Ref.∼ [109].

3.4.2 Informer

The Informer [109] model improves the original Transformer architecture with two main contributions, the ProbSparse self-attention mechanism and the generative inference.

The ProbSparse self-attention mechanism allows for each key vector to only attend to the top queries, which makes it more computationally efficient than the traditional self-attention. The ProbSparse self-attention mechanism returns redundant combinations of value vectors. Therefore, the authors add self-attention distilling as a way to allow the model to focus on the superior value vectors which contain dominating features. By creating a focused self-attention feature map the input's time dimension is greatly reduced, as well as the memory usage.

The generative inference is found in the decoder part of the Informer model. Instead of using a traditional start token as input to the decoder, the authors use a previous N-long sequence allowing the model to do prediction as a single forward procedure, rather than using the "dynamic decoding" in the conventional encoder-decoder architecture.

Our Informer, with the best hyperparameters found during optimization, which are the same as Transformer hyperparameters, has a total of 5.272.578 trainable parameters.

3.4.3 NLinear

The authors of Ref . [97] question the validity of Transformers in long-term time series forecasting. They argue that "While employing positional encoding and using tokens to embed sub-series in Transformers facilitate preserving some ordering information, the nature of the permutation-invariant self-attention mechanism inevitably results in temporal information loss". In order to corroborate their claims, they created three very simple linear models with very few trainable parameters and compared them against the Transformer model, as well as time series-focused Trans-

formers such as Informer, Autoformer [110], and FedFormer [111].

The model of choice from this paper is the one they named NLinear. This model simply consists of a simple subtraction of the input by the last value of the sequence, this is fed into a single linear layer, and the output is added back to the subtracted value.

The NLinear is the best-performing model of the three they propose. All the models outperform the complex Transformers despite their simplicity.

Our NLinear model, with the best hyperparameters found during optimization, has a total of 12 trainable parameters.

3.4.4 Markov model

Markov models were first introduced by the Russian mathematician Andrey Markov in the early 20th century [49]. A Markov model is a stochastic model used to describe systems that change over time.

Markov models try to predict what will happen next based on what is happening now. In other words, it assumes that the future state depends exclusively on the current state. Variations of the Markov model can have a memory of past states, but in our case, we use a simple version with no memory.

Despite their simplicity and age, Markov models are still popular and have shown impressive performance in our task, as described in Ref . [53]. Furthermore, they describe that the Markov model is the one to perform the best when compared to multiple DL models, due to its strong mathematical principles as well as the simplicity the model presents, especially when compared to the "black box" nature DL models tend to have.

Markov models do not have trainable parameters like the ones we can find in Neural Networks. Instead, they have a transition matrix. Our transition matrix has a total shape of 4096x4096 for the eye-tracking data and 1156x1156 for the Gaussian Noise data.

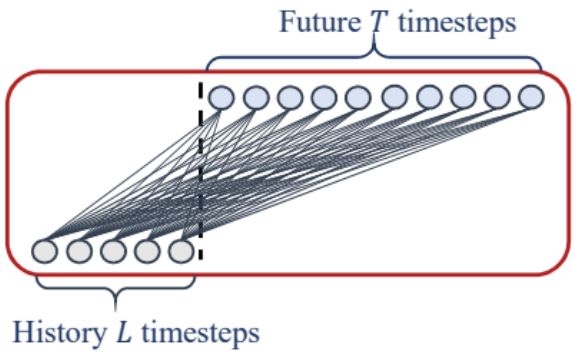


Figure 3.12: Basic linear model taken from Ref.∼[97].

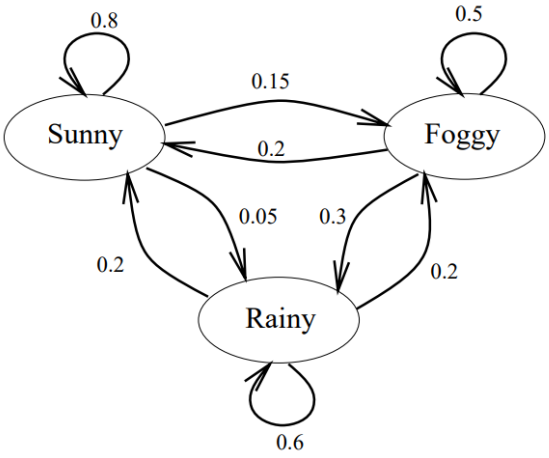


Figure 3.13: Illustration of a Markov chain. Taken from Ref.∼ [112].

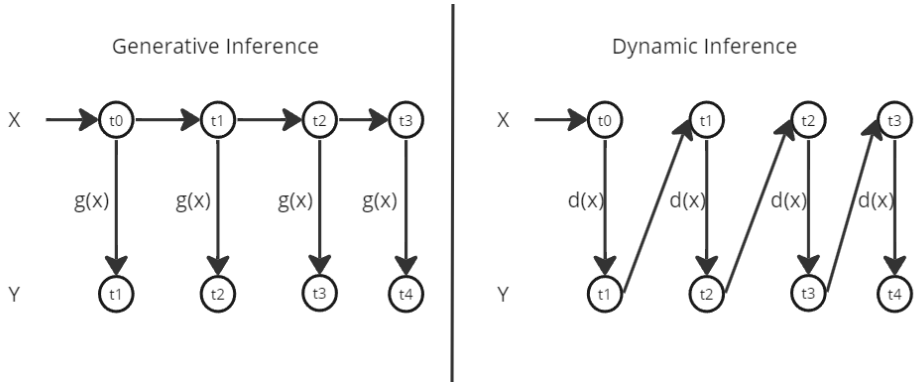


Figure 3.14: Illustration of the GI and DI experiments.

3.5 Experiments

The experiments for this thesis will consist of forecasting the final 10% of points using the previous 90% as training data. For this, we will use all three of the previously described datasets with all four of the previously described forecasting models.

There will be two types of tests, first one, we call Generative Inference when we use as input to the model the previous real value for all timesteps of the forecast. Secondly, we call Dynamic Inference when we use the output of the model at the previous timestep, as input for the current timestep, for all the predictions.

Figure 3.14 shows an illustration of both the Generative Inference and the Dynamic Inference tests. In both, X denotes the input to the model, where $g(X)$ stands for the model being used in the GI experiment, and $d(X)$ denotes the model being used in the DI experiment. Therefore, Y denotes the output of the model given an input at a timestep t. The difference between the tests is the following: GI will use the input of the original dataset at every timestep, while the DI will use as input the output of the

		Generative Inference					
Dataset	Metric	Transformer	Informer	NLinear	Markov	Previous Value	Average Value
Eye-tracking	MSE	0.00008	0.0001	0.00004	1.4807	**0.00002**	0.00008
	MAE	0.00667	0.00646	0.00402	0.77192	**0.00284**	0.00441
	JSD	0.00001	0.00001	0.000002	0.04554	**0.000001**	0.000005
Gaussian Noise	MSE	0.04591	0.04591	0.06905	2.04570	0.09133	**0.0459**
	MAE	0.17091	0.17091	0.20973	1.1418	0.24113	**0.1708**
	JSD	**0.0028**	**0.0028**	0.00421	0.08158	0.05577	**0.0028**
ETT	MSE	0.12959	0.0763	0.03028	N/A	**0.01834**	0.0808
	MAE	0.2606	0.19857	0.11121	N/A	**0.08520**	0.2134
	JSD	0.01252	0.00575	0.002306	N/A	**0.001383**	0.0056
		Dynamic Inferecne					
Dataset	Metric	Transformer	Informer	NLinear	Markov	Previous Value	Average Value
Eye-tracking	MSE	**0.0001**	0.0001	0.00032	1.76025	N/A	N/A
	MAE	**0.00599**	0.00646	0.01586	0.77051	N/A	N/A
	JSD	**0.000006**	0.00001	0.000005	0.04821	N/A	N/A
Gaussian Noise	MSE	**0.04591**	**0.04591**	0.04873	1.08737	N/A	N/A
	MAE	**0.17091**	**0.17091**	0.17627	0.83299	N/A	N/A
	JSD	**0.0028**	**0.0028**	0.028	0.045194	N/A	N/A
ETT	MSE	0.1314	**0.10253**	1328.67	N/A	N/A	N/A
	MAE	0.26272	**0.23965**	14.9782	N/A	N/A	N/A
	JSD	0.012698	**0.00727**	inf	N/A	N/A	N/A

Table 3.1: Results of all the models for all datasets.

model at the previous timestep, except for the first prediction, when we do not have an output yet.

On one hand, the three initial models, Transformer, Informer, and NLinear were implemented by their authors using Generative Inference (GI), therefore we expect better performance using this type of test, as it was their original objective. Additionally, since Generative Inference always uses as input the correct previous point of the time series, we avoid error accumulation.

On the other hand, for our specific problem, it does make sense to try to create new data, and in order to do so, we need to dynamically feed the output of the model back into the network, thus justifying the need for the Dynamic Inference (DI) test. Since the output of the model may not be correct for a variety of reasons due to the higher complexity, this test is expected to be much harder than the previous one, as that hypothetically incorrect output would be used as input in the following iteration.

The Markov model will not be used with the ETT dataset, that is due to the limited flexibility the model provides. It was specifically designed to work with 2D data such as eye velocities, therefore not compatible with the multivariate to univariate prediction the ETT dataset requires. This is an example of the flexibility DL models offer compared to mathematical models. Additionally, for the GI experiment, we will show the performance metrics results for outputting the previous value at each time step, as well as outputting the cumulative average. These will only be done in the GI test due to doing it for the DI would not bring much value, as we would only be outputting the initial value all the time.

Chapter 4

Implementation details

This chapter aims to describe the modifications done to the original code presented by the papers which we are implementing in this thesis. The code used for the whole thesis can be found in Appendix A.1. The paper Ref.∼ [97] provides a GitHub repository if the implementation of the Transformer and Informer models used [91, 109], and we chose to use their code as a template for our own implementation and experiments.

The main modification we introduced in the code was in the testing and inference implementations. When training a transformer-based model, the input to the encoder is the input tokens, in our case two-dimensional data representing the x and y velocities, while the input to the decoder are the targets of processing the input, shifted to the right. Such a thing is done to enable the decoder to learn how to generate the correct output sequence for a given input sequence, allowing the model to learn the mapping between the input and output sequences.

Figure 4.1 shows a visual representation of the aforementioned training procedure, where the targets of translating the sentence "You are welcome" are fed into the decoder.

For testing and inference, while the inputs to the encoder remain the same, the targets are no longer fed into the decoder as they are not available. This ensures that the Transformer is generating the output sequence based solely on the input sequence. In these scenarios, the input to the decoder will be a special start token first, and in the following iterations, the output of the decoder will be fed back to itself, a process called auto-regressive decoding. This process allows Transformers to handle variable-length output sequences as well as to capture dependencies between the output tokens.

Figure 4.2 shows a visual representation of the aforementioned testing and inference procedures. In this case, it can be seen how the output of the decoder is fed back into itself for the next time step.

In the original code from the papers, we quickly realized that the implementation of the testing and inference procedures was no different than the training implementation. Thus using the targets when seemingly predicting unseen data. This generated impossibly accurate results. Our modifications to the original code involved replacing the test and inference

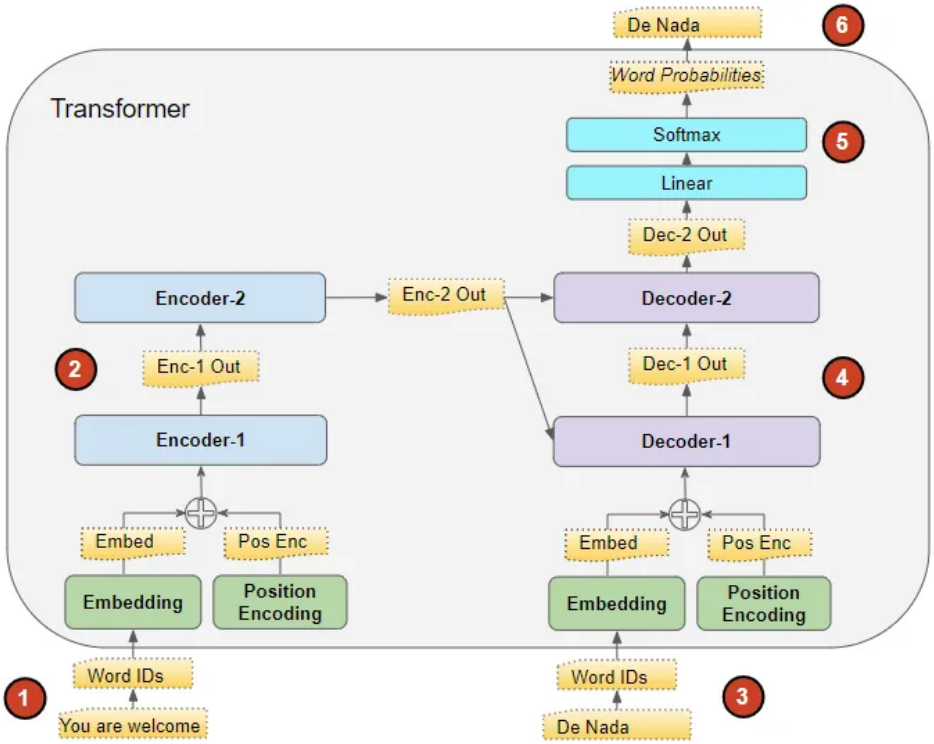


Figure 4.1: Training procedure of a Transformer. From Ref.∼[113].

procedure, and implementing it again following the 4.2 flow.

In order to compare the original implementation with our modified one, we used the Transformer to train on our datasets and predict previously unseen data. The dataset is a simple random Brownian Motion. The models will have a short training time and no hyperparameter optimization, therefore we expect a bad performance.

As shown in figure 4.3(a) the prediction of the original code is extremely accurate with the real data, a feat that is truly hard. This is because the BM data is generated with a pseudorandom algorithm, it would mean that the Transformer has managed to learn the inner works of a random algorithm, for a specific seed, using deterministic computations. Although a pseudorandom algorithm is not a truly random generator, neural networks cannot learn it because randomness is not deterministic.

Although the data generated with the modified code 4.3(b) is not nearly as impressive, we argue that it is the result of a well-implemented algorithm that uses the correct testing and inference process. Therefore, for the rest of the thesis, the modified implementation will be used.

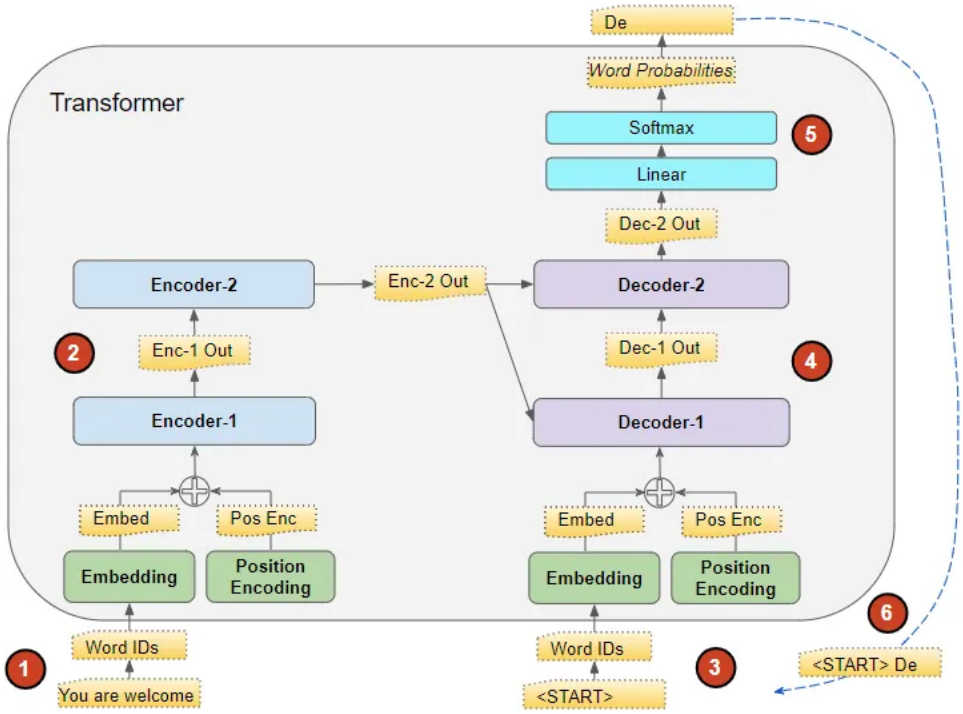
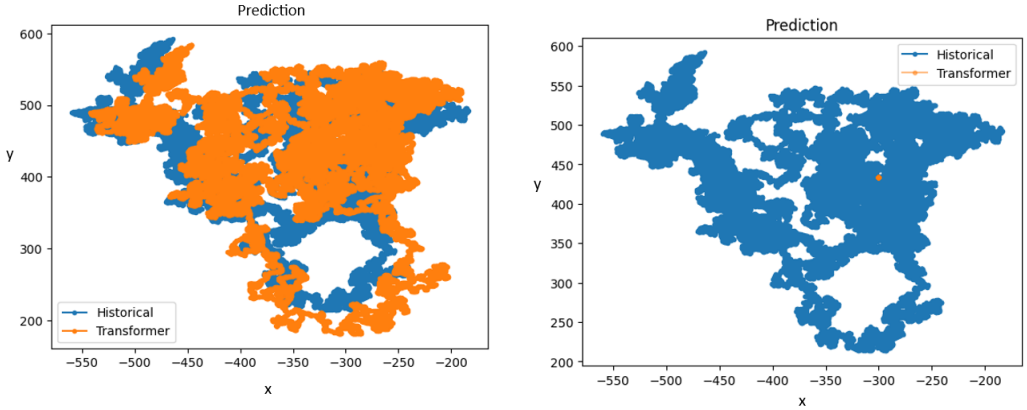


Figure 4.2: Test and Inference procedure of a Transformer. From Ref.~[113].



(a) Prediction with original code.

(b) Prediction with modified code.

Figure 4.3: Predictions of BM with original and modified code.

Chapter 5

Results

In this chapter, the results of both the Generative Inference test and the Dynamic Inference test will be provided. Extensive hyperparameter optimization was done in order to achieve the best performance for all the models implemented. In particular, for Transformer and Informer, the number of heads, size of the model, number of encoders and decoders, window size of moving average, input and output sequence length, and dropout rate were optimized. In addition, and also for the NLinear model, the loss function, number of epochs and early stopping patience were also optimized.

5.1 Prediction of eye-tracking velocities

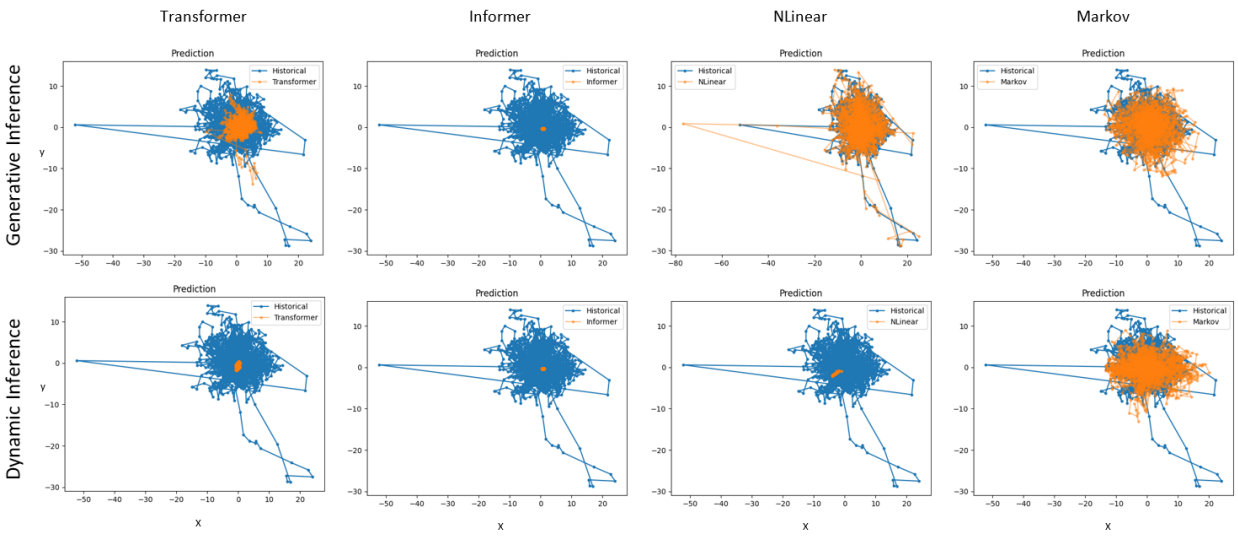


Figure 5.1: Model's predictions for Eye-tracking.

Figure 5.1 shows the predictions for all four models in both Generative and Dynamic inferences. The X-axis represents the X position while the Y-axis represents the Y position. For the GI, we can see most models

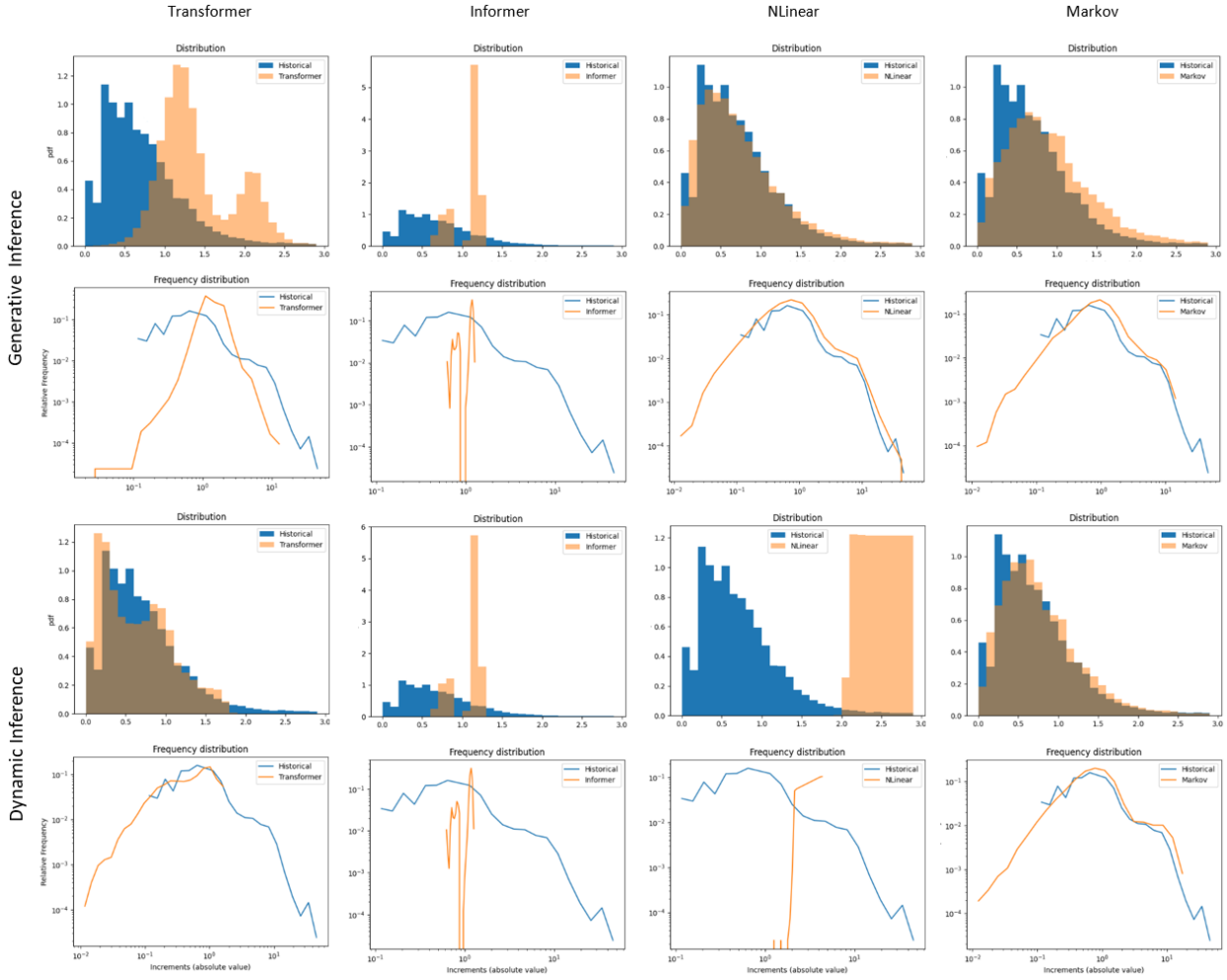


Figure 5.2: Distributions of the predictions for Eye-tracking velocities (ϕ).

performing decently, with the exception of the Informer. Both NLinear and Markov seem to generate data that is quite similar to the ground truth, especially NLinear. The Transformer seems to be able to capture some of the relations across points and be able to generate a somewhat sensible data representation. In the DI, all models except the Markov seem to be suffering from error accumulation. Furthermore, the Markov model performance in the DI could potentially be better than in GI, although it is hard to say only from this figure.

Figure 5.2 shows the distributions and frequency distributions for all models, the first two rows correspond to the GI test, while the last two rows correspond to the DI test. The X-axis represents the increments and the Y-axis the frequency. For the histograms, the data is in the time domain while for the line plots the data is in the frequency domain. The distributions of the predictions follow, for the most part, the quality of the data generated is seen in the previous figure. Informer performs poorly in both experiments, NLinear and Markov perform quite well in the GI, especially NLinear, but Markov performs much better in the DI. One

significant exception is that the Transformer performs somewhat decently in the GI, but seems to perform remarkably well in the DI, despite having generated a poor representation of the data. It seems like the distribution of the data was quite spot on. Despite that, the frequency distribution reveals that even though the increments were quite reasonable, they happened in much lower frequencies. The frequency distribution for the Transformer, NLinear, and Markov in the GI was rather spot on, while for the DI, the Markov was closest to replicating the historical frequency distribution.

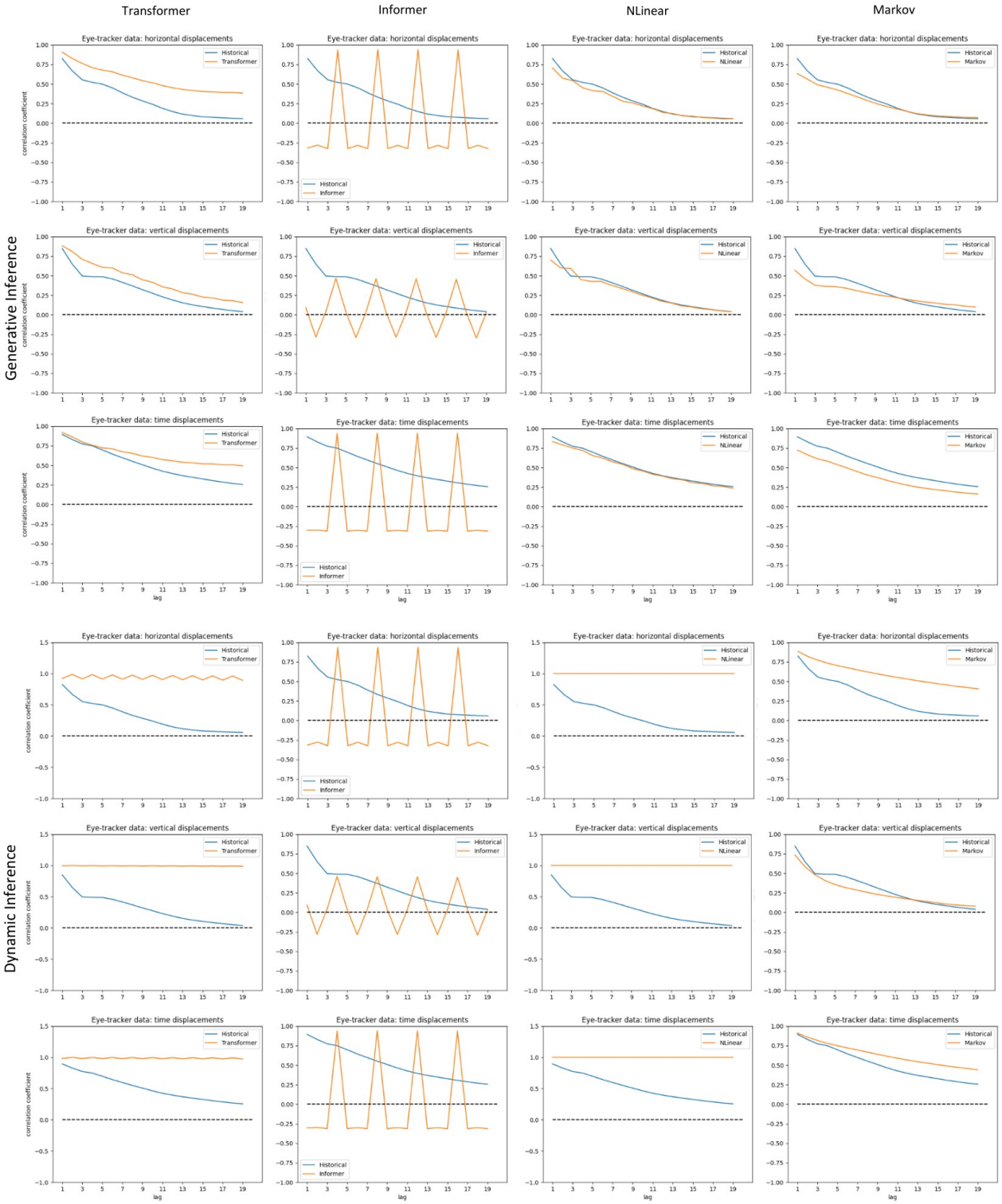


Figure 5.3: Autocorrelations for velocities of eye-tracking.

Figure 5.3 shows the autocorrelations of the horizontal, vertical, and

time displacements. The X-axis represents the number of lags while the Y-axis represents the correlation coefficient. Similarly to the previous figure, the first three rows correspond to the GI test, horizontal, vertical, and time displacements respectively, while the three last rows represent the DI test for the same measurements. Following the GI test trend, the Informer model shows the poor performance it has been exhibiting before, while, in order of best to worse, the NLinear, Markov, and Transformer perform remarkably well. These three models are able to capture the three types of displacement trends with significant accuracy. In the case of the DI test, none of the models except Markov seem to be able to replicate the trend of any of the displacements measured. The two top-performing models, excluding Markov, Transformer, and NLinear, show a constant trend for the most part, at different correlation coefficients, for Transformer and NLinear at 1, and for Markov at 0. The Informer model shows a repetitive trend, although the Transformer also seems to have such a trend, it is at a much closer coefficient than the Informer when compared to the real data. The Markov model, on the other hand, shows exceptional results. It seems like the results are better in the GI than the DI, but it is still extremely impressive that the model does not seem to suffer from any type of loss when all the others perform much worse in the DI, and sometimes even showing better results than in GI.

5.2 Prediction of θ and φ

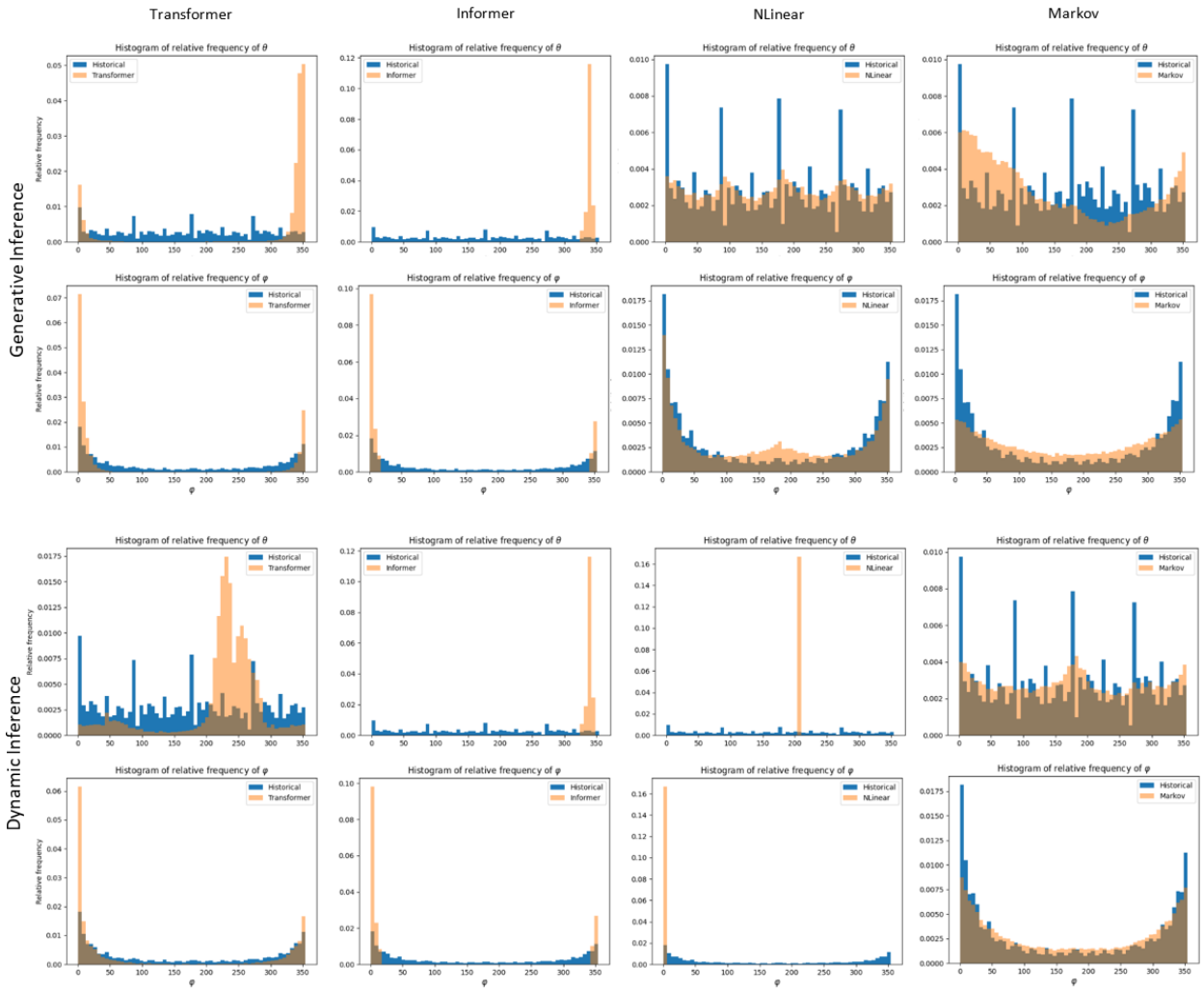


Figure 5.4: Histograms of angles of Eye-tracking.

Figure 5.4 shows the histograms of the three main quality measurements, the first three rows show the θ, φ, and ϕ respectively, for the GI test. The last three rows show the same measurements for the DI test, in the same order. The X-axis corresponds to the angle (0, 360) of the corresponding metric while the Y-axis represents the frequency. In the GI test, it can be seen how both the Transformer and Informer perform poorly, both of them are able to depict that both φ and ϕ have higher values on the edges, but completely miss on the values in between. On the other hand, the NLinear and Markov models are able to better represent such distributions. The NLinear is much better at representing θ as well as the high peaks in φ and ϕ. While Markov struggles more with the extreme values, it is significantly better are representing the smoothness found in the middle values of φ and ϕ. In the DI test, Informer shows a very similar performance to the GI test.

NLinear performs significantly worse than before, and even worse than Informer, not being able to find the high values in both extremes for φ and ϕ. The Transformer and Markov perform much better than both Informer and NLinear. Moreover, despite the poor prediction, Transformer seems to be able to depict a decent representation of θ for all angles, while Markov keeps a much more realistic frequency per angle in all measurements, and even performs better than in the GI experiment.

5.3 Prediction of Gaussian Noise

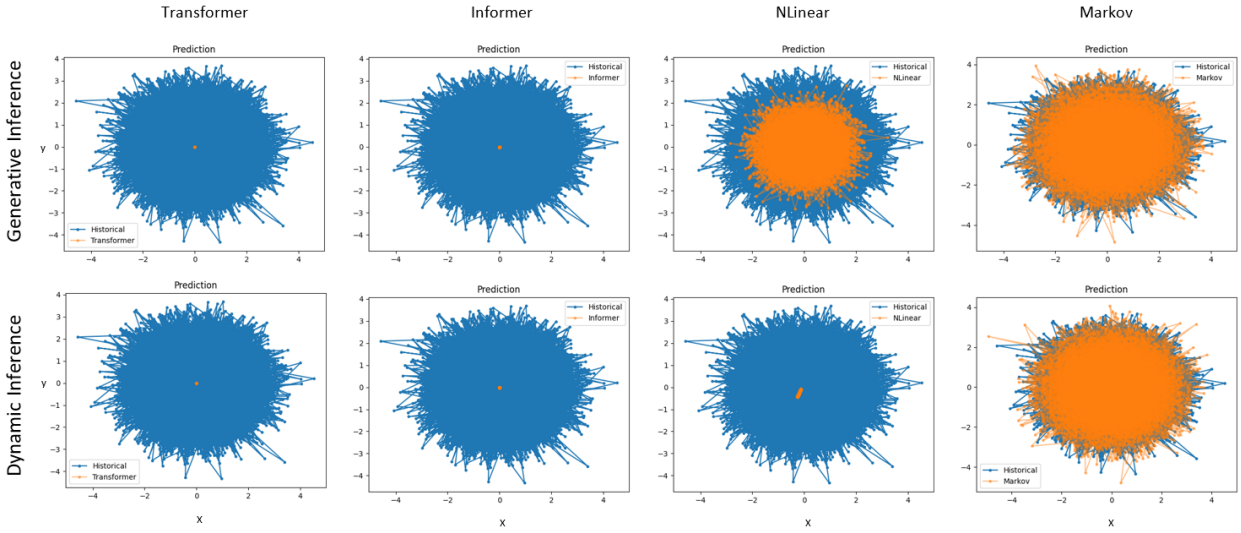


Figure 5.5: Model's predictions for Gaussian Noise.

Figure 5.5 shows the predictions for all models, both in GI and DI, for the Gaussian Noise data. This time, for the GI we can only see the NLinear and Markov performing well. Although both models seem to create a faithful representation of the original data, it seems like the Markov model is capable of doing so on a more fitting scale. On the other hand, neither the Transformer nor the Informer achieves any significant results. For the DI, both Transformer and Informer show the same result, not being able to capture the data spread at all. NLinear seems to show the same issue, with a very slightly better result. The Markov model seems to be able to forecast a very accurate data representation of the model. The scale of the forecasted data is matching the original almost perfectly, showing an impressive result for the harder test.

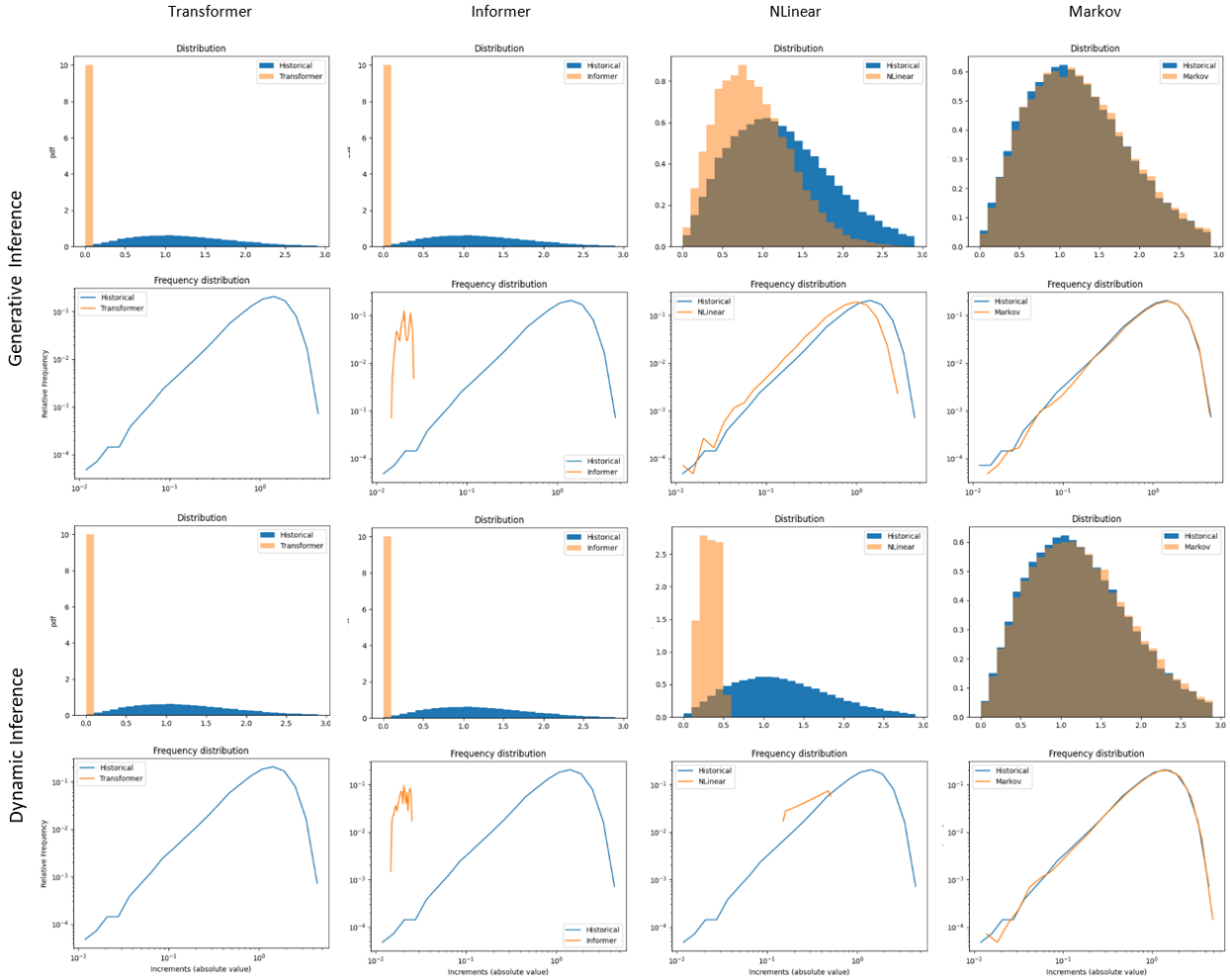


Figure 5.6: Distributions of the predictions for Gaussian Noise (ϕ).

Figure 5.6 shows the distributions and frequency distributions for all models, the first two rows correspond to the GI test, while the last two rows correspond to the DI test. For the GI, the Transformer and Informer continue to underperform. In fact, the frequency distribution of the Transformer could not be computed, therefore only the historical data frequency distribution is shown. The NLinear and Markov continue with outstanding results, especially the Markov, where both the distribution and frequency distribution are almost identical. For the DI, we see the same issues with both Transformer and Informer. Again, we see how the NLinear is incapable of achieving high accuracies, despite still being better than the Transformer and Informer. The Markov also shows impressive results for the DI test, with an almost identical distribution and frequency distribution.

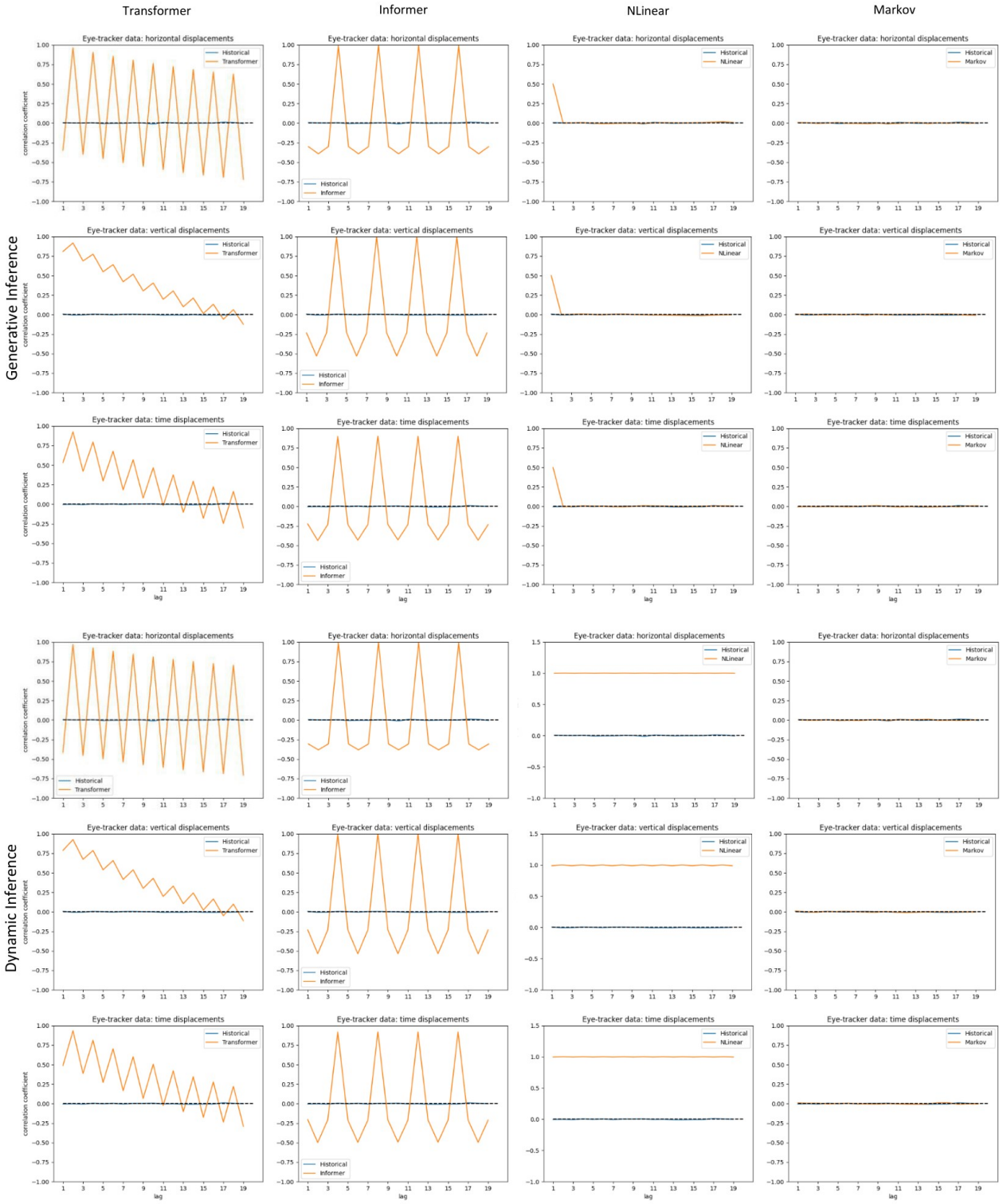


Figure 5.7: Autocorrelations for velocities of Gaussian Noise.

Figure 5.7 shows the autocorrelations of the horizontal, vertical, and

time displacements. Similarly to the previous figure, for the GI, both Transformer and Informer show poor performance, both of them experience repetitive results that do not align correctly with the historical data. On the other hand, the NLinear and Markov do show autocorrelations that fit extremely well with the historical data. The NLinear shows 0.5 of autocorrelation at lag 1 for the three displacements measured, despite that, and for the rest of the lags, it does show the expected behavior. On the contrary, the Markov model shows the expected behavior across all lags. For the DI, both the Transformer and Informer show the same erroneous behavior. Again, the Markov model depicts the correct displacements, and, the NLinear also shows the correct displacements but at a wrong coefficient, 1 instead of 0. This behavior could simply be the case of a repeating pattern such as the ones from Transformer and Informer, but with much less complexity.

5.4 Prediction of θ and φ

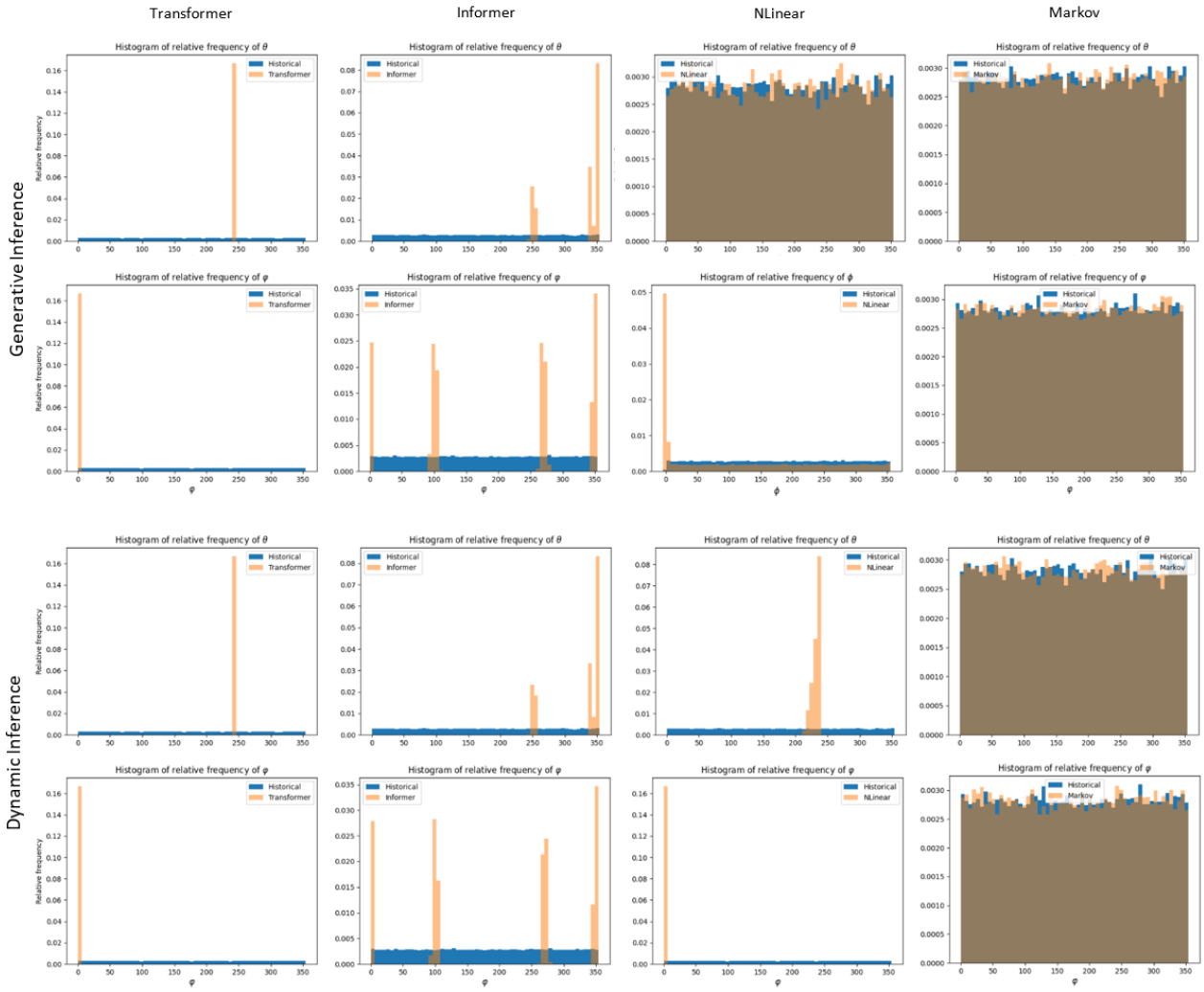


Figure 5.8: Histograms of angles of Gaussian Noise.

Figure 5.8 shows the histograms of the three main quality measurements, the first three rows show the θ, φ, and ϕ respectively, for the GI test. The last three rows show the same measurements for the DI test, in the same order. For the GI test, it seems like the Informer is performing slightly better than the Transformer, although they are both performing very badly overall. The NLinear, although it may seem like it is not performing too well, it seems like there is a high frequency on angle 0 that changes the scale on the whole plot. Despite that, it can still be seen how NLinear quite successfully manages to capture the frequency of angles for the rest of the possible values, especially for θ, where the initial high-frequency value is not present. The Markov model achieves extremely accurate depictions of the three measurements, without extreme values present in NLinear. For the DI test, Transformer, Informer, and NLinear show bad performance,

with Informer slightly surpassing the others thanks to being able to show some frequency values at more than one angle. Again, the Markov model achieves high accuracy for all three measurements.

5.5 Oil Temperature

5.5.1 Prediction

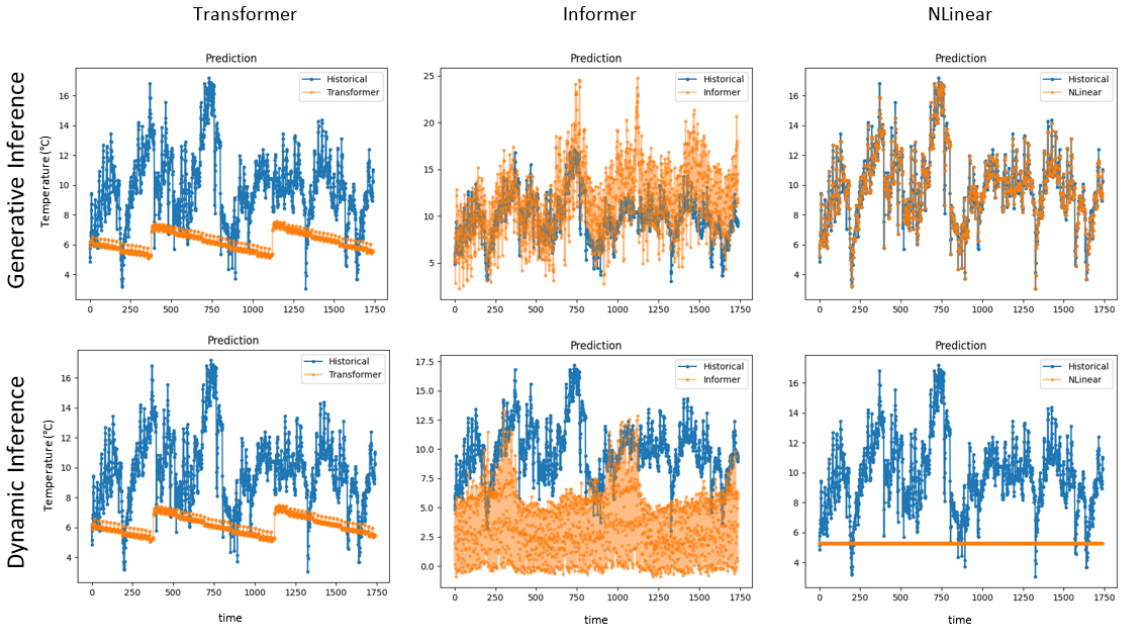


Figure 5.9: Model's predictions for ETT.

Figure 5.9 shows the predictions for all models, except Markov, both in GI and DI, for the Electrical Transformer Temperature dataset. For this type of data, the X-axis represents the time step, while the Y-axis represents the temperature in degrees Celcius. In the GI test, it can be seen how both Informer and NLinear are performing substantially better than Transformer. NLinear achieves fitting the data with high precision, while Informer also fits the data, but it does seem to be overshooting the predictions by using larger steps. The Transformer shows a repetitive pattern which indicates the incapability of fitting the data. For the DI test, it can be seen now how the Transformer shows the same repetitive behavior, which is not far from the flat line prediction of the NLinear model. Surprisingly, the Informer model shows the best performance in this test. Despite being far from perfect, it shows some intent of trying to replicate the historical data.

5.5.2 Data distribution

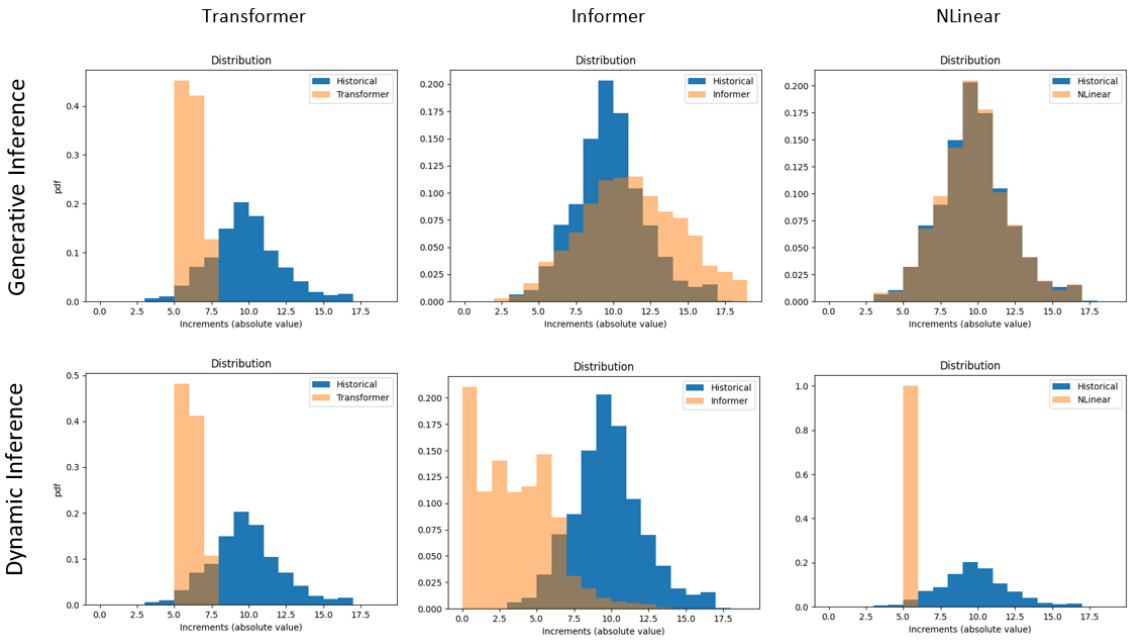


Figure 5.10: Distributions of the predictions for ETT.

Figure 5.10 shows the distributions for all models, except Markov. Similarly to the previous figure for the GI test, it can be seen how the NLinear model replicates the original distribution with outstanding precision. Although the Informer does not achieve such performance, it still manages to replicate the distribution somewhat decently, struggling to achieve the high frequency of some increments, especially in the center. The Transformer model, on the other hand, does not seem to replicate the distribution whatsoever. For the DI test, as expected the Transformer shows the same behavior, while the NLinear simply shows one increment for all points, which corresponds to the flat line prediction seen in the previous figure. The Informer model does not quite manage to fit the distribution expected but still shows much better performance than the other two models.

5.6 Past, Present, and Future correlations

			Generative Inference					
Dataset	Type	Baseline	Transformer	Informer	NLinear	Markov	Previous Value	Average Value
Eye-tracking	Past	0.824	0.404	-0.0009	0.509	0.505	**1**	0.032
	Present	1	0.676	-0.003	**0.898**	0.481	0.824	0.028
	Future	0.824	0.533	-0.0002	**0.693**	0.54	0.553	0.015
Gaussian Noise	Past	0.005	-0.013	0.003	0.004	0.003	**1**	0.017
	Present	1	0.021	-0.009	**0.358**	0.007	0.005	0.017
	Future	0.005	-0.013	**0.008**	-0.0004	-0.001	0.001	-0.009
ETT	Past	0.961	0.109	0.323	0.852	N/A	**1**	0.138
	Present	1	0.115	0.412	**0.979**	N/A	0.911	0.103
	Future	0.961	0.114	0.376	**0.939**	N/A	0.605	0.009
			Dynamic Inference					
Dataset	Type	Baseline	Transformer	Informer	NLinear	Markov	Previous Value	Average Value
Eye-tracking	Past	0.824	0.019	-0.001	**0.03**	0.003	N/A	N/A
	Present	1	0.018	-0.003	**0.03**	0.001	N/A	N/A
	Future	0.824	0.018	-0.0006	**0.03**	0.002	N/A	N/A
Gaussian Noise	Past	0.005	-0.012	-0.009	**0.003**	-0.004	N/A	N/A
	Present	1	0.005	0.002	**0.002**	0.0008	N/A	N/A
	Future	0.005	-0.012	0.007	**0.003**	-0.01	N/A	N/A
ETT	Past	0.961	**0.076**	0.016	nan	N/A	N/A	N/A
	Present	1	**0.085**	0.014	nan	N/A	N/A	N/A
	Future	0.961	**0.082**	0.02	nan	N/A	N/A	N/A

Table 5.1: All-time correlations for all tests and models.

Table 5.1 shows the correlation coefficients for the horizontal velocities for all experiments and models. The past, present, and future correlation coefficients can be observed, which indicates how dependent is the present predicted data point from the previous real point, the current, and the next one. The baseline indicates the correlation of the data with itself, therefore, how dependent the current real value with the past real value, the current, and the next one. The results of the table align with the results observed in the previous figures, where, for the most part, the best-performing models show higher correlation coefficients. The definition of these correlations can be found at 3.2, 3.3, and 3.4, for the past, present, and future correlations respectively. Similarly to Table 3.1, for the GI experiment, we will show the correlation metrics results for outputting the previous value at each time step, as well as outputting the cumulative average. These will only be done in the GI test due to doing it for the DI would not bring much value, as we would only be outputting the initial value all the time.

Chapter 6

Discussion and Conclusions

For this thesis, we hypothesized that it would be possible to train time transformers and utilize their state-of-the-art self-attention mechanism to capture the complex and advanced properties of the human gaze and utilize them to generate new, synthetic, and realistic datasets. Being capable of generating big-quality datasets would provide immense help in different fields that need Personal Identifiable Information to advance their research. We also hypothesized that being able to replicate human eye behavior, could be the start of a number of AI models that could potentially replicate other complex human behaviors that also suffer from limited availability in their datasets.

In order to validate or disprove our hypothesis, we designed three tests, with two sub-tests for each one of them. The three main experiments consist of three different datasets we used to evaluate our models' performance. Therefore, we refer to these tests by the name of their respective datasets, Eye-tracking, Gaussian Noise, and ETT. For each test, we created two sub-tests, which correspond to utilizing the real input at each time step and utilizing the previous output of the model as input at each time step. We labeled these tests Generative Inference and Dynamic Inference respectively. The Eye-tracking test was done first as it was the most important one for the thesis. The main goal to achieve was the generation of realistic human gaze trajectories, therefore it made sense to spend the most time with it. The Gaussian Noise was done in order to explore the capabilities and limitations of the models. Since the Gaussian Noise data was created following the same structure as the eye-tracking data, we presumed that a model that can perform well in one dataset would also perform well in the other. Furthermore, Gaussian Noise is perhaps the most simple type of random data that can be provided and has very simple properties, thus the models should succeed more easily with this type of data. Not only is simple to replicate, but also to interpret, which can give insights into the algorithms, as they may not be able to perform in a non-deterministic setting, despite the simplicity of it. The ETT test was conducted as a baseline, to compare the results of our models to the ones found in their respective papers. Since this dataset had been used for the Informer and NLinear papers [97, 109], we expected to see similar results,

validating that our implementations were correct.

For the first two tests, and their respective sub-tests, we tested three deep learning models: Transformer, Informer, NLinear, and the mathematical optimization model: Markov. For the last test and its sub-test, only the deep learning models were used. On the one hand, the deep learning models worked by giving the model an input of length n, and it would generate an output of the same length n, which could be done in a loop to generate a sequence of the desired length. On the other hand, the Markov model would be given an initial point and the number of points to generate. We trained the Transformer and NLinear models to use an input of length 2 and to forecast an output of length 2, the Informer model would use an input of length 4, and forecast an output of length 4. Three data distribution evaluation metrics were used in order to evaluate the performances of the models.

Initially, we expected the Informer model to perform better than the others, as it was the most sophisticated model specially developed for time-series forecasting. We also expected that the Transformer would perform well based on the results from Ref.∼[109], while the NLinear would perform drastically worse due to it being an extremely small model, despite the results in Ref.∼[97]. We already knew that the Markov model would perform well thanks to the results from Ref.∼[53]. Despite the differences in the datasets used for the original papers and ours, we expected that the models would be able to learn the dependencies and relationships of the data in a similar fashion, and show similar performances.

6.1 Findings and observations

For the first experiment, we used eye-tracking data, which consists of a two-dimensional input and output. We found out that for the GI test, the NLinear model yielded the best MSE, MAE, and JSD measurements. The figures for that experiment also corroborate the NLinear as the best-performing model. However, the Markov model, being the second-best-performing model according to the figures, yielded the worst MSE, MAE, and JSD measurements. The Transformer model performed third best according to the figures, and second best according to the measurement metrics, while the Informer performed worse overall. These results are somewhat in alignment with Ref.∼[97] where the NLinear outperforms all the self-attention-based models, although we would expect the Informer to perform better than the Transformer. From the results in Ref.∼[53], we would expect the Markov model to outperform all deep learning models, but it does not seem to be able to outperform the NLinear, either on the figures or on the measurement metrics. On the other hand, for the DI test, the Markov model results on the figures do align with the expected results, where Markov is capable of forecasting a better long sequence with no real values as an input. Despite that, the model still shows the worse MSE, MAE, and JSD across all models. For this test, the best metrics belong to the Transformer, whose prediction is significantly worse than the Markov, but

the distribution of it is significantly better, which justifies why the metrics results. For this test, Informer and NLinear do not give any significant results.

For the second experiment, we used Gaussian Noise data, which had the same shape as the eye-tracking, with two-dimensional inputs and outputs. For the GI test, we can see how both Transformer and Informer simply predict a single point, which gives them the same MSE, MAE, and JSD scores. These are the lowest across all models and yet they have the worse predictions and distributions. For this, Markov shows better figures, surpassing even the NLinear model, which shows better scores than Markov. In this case, the NLinear outperforms both self-attention models, where the Informer does slightly outperform the Transformer, which does align with Ref.∼[97]. Moreover, the Markov outperforms all deep learning models, which also aligns with Ref.∼[53]. For the DI test, we see similar results, despite the big loss of the NLinear model, it still shows a slightly better distribution than the self-attention models, where the Informer is also slightly better than the Transformer. The Markov model though manages to fully keep its accuracy and shows outstanding results for the random noise data. Despite that, the metrics show that the Transformer and Informer are tied with the best scores, with the NLinear and Markov coming after, which could not be further away from what the plots describe. For this experiment, it seems like the deterministic nature of the self-attention models plays a big role in them not being able to replicate the random data. Although the NLinear model is also deterministic, it has 12 trainable parameters instead of >5M, which may allow it to show a more random behavior rather than returning the same point given the same input, also using the same amount of training data. From this experiment, we saw that transformers are ill-equipped to deal with stochastic data and that even the NLinear model still has trouble replicating this type of data through generative inference. Moreover, this issue can also be, in part, due to the models optimizing to minimize the error.

For the third and final test, we used the Electrical Transformer Temperature data, which consists of a seven-dimensional input and a one-dimensional output. In the GI test, it can be seen how the NLinear outperforms both the Informer and Transformer, both on the measurement metrics and on the distribution figures. For this test though, the Informer vastly outperforms the Transformer both in the figures and the measurement metrics. Which aligns with Ref.∼[97]. In the DI test, as per usual, the NLinear model struggles to fully depict the distribution but, in this case, the Informer manages to still perform more adequately than the two models by quite a bit, showing better results both in the metrics and the figures, which aligns with the results in Ref.∼[109].

Although we expected the models to be able to perform similarly to our data than with the original data from their papers, that only seems to be applicable to the NLinear model, which showed good performances for the GI test with all the different datasets. Although the Transformer showed promising results for the Eye-tracking GI, its results for the Gaussian Noise and ETT were subpar, similarly, the Informer yielded good results

for the ETT, with subpar results for Eye-tracking and Gaussian Noise, which indicates that the Informer model may be best suited for univariate forecasting.

One may question the impressive performance of the Markov model. It is worth noting, that the original use of the Markov model would be the equivalent of our Dynamic Inference test. Contrary to the DL models, the Markov was developed to generate additional values without the real points as input, therefore, it is not surprising that sometimes, the Markov results of the harder test appear to be better than the GI test. Additionally, we noted the performance of the NLinear, especially for the Eye-tracking test, was very similar to the actual data. In section 5.6, table 5.1, we can see how the NLinear, Eye-tracking Present for the GI has a value of 0.898, which is high but not 1. This indicates that the values are highly correlated, which may result from the model simply returning a value similar to the input with small variations. Although we do want a high correlation for the present, having a correlation coefficient of 1 would indicate that the model returns the exact same data as it is given as input, which defeats the purpose of creating these models. For the GI, the results of the self-attention-based models are consistently worse than the NLinear and Markov. Even if the NLinear model generates a simple linear combination of the input, why can't the more complex self-attention models learn to do the same? The NLinear model has 12 trainable weights in comparison to the well over 5M for both self-attention models, therefore we cannot expect the NLinear to do much more than simply learn to return a small perturbation of the input. One may argue that the higher complex transformer-based models would need more training to learn that returning the input is highly successful, nevertheless, all the models were trained using an adaptable learning rate and early stopping. This indicates that the transformer-based models did not learn this simple policy because, after a certain number of epochs, they stopped optimizing. The authors of the NLinear model [97] argue that "While employing positional encoding and using tokens to embed sub-series in Transformers facilitate preserving some ordering information, the nature of the permutation-invariant self-attention mechanism inevitably results in temporal information loss.". This arguably indicates that the models are not capable of learning a simple policy like returning the raw input due to the temporal loss of the self-attention mechanism.

6.2 Conclusions and future perspectives

In the course of this thesis, we have found out that the time-series transformer-based models did not manage to generate realistic eye-tracking data. It is possible that these models are not well suited for this kind of data, although some acceptable results may indicate otherwise, it is possible that with further adjustments, they may be able to show good results. Moreover, we further corroborated the dominance of the Markov model for this kind of data, as it outperformed the rest. We also

discovered that a simple linear model could achieve impressive results, which may indicate a possible investigation direction to use Deep Learning for this problem. Even though the more complex models fell short of generating realistic eye-tracking data, we believe that simply the choice of time-series forecasting transformers was not adequate for this problem. Additionally, we consider the results found to be significant and relevant, given how ubiquitous the usage of these transformer-based models has become recently.

The main limitation identified for this project was found in the limited time, which affected different parts: Although the self-attention models were tuned and downsized to accommodate the limited data, these still were substantially big. A reason why a small model like NLinear can perform better may simply be because it has more than enough data to learn the distribution of the data, while transformer-based models are known to be data-hungry. Using bigger datasets could have potentially enabled the models to better learn the data distributions, which would have increased the training times as well as potentially the size of the models. Another considerable limitation was found in the measurement metrics used (MSE, MAE, and JSD). These metrics were used due to being able to limit the difference between the predicted data and the real data, thus implicitly learning the distribution. The results of the table 3.1 show how sometimes, and especially for the Gaussian Noise test, the models that yield the best predictions are punished with worse scores. Since these metrics were also used as loss functions, they could have drastically impacted the performance of the models. Moreover, we differentiate that replicating data and predicting data are two different tasks. Arguably, when replicating data the loss functions should be conservative, to avoid large deviations from the original data, which is what MSE, MAE, and JSD achieve. In our forecasting task, these losses could have affected the risk the models were willing to take to try to predict data more accurately, which does not necessarily need to be a replication of the true outputs, but of their properties. Despite that, some models still managed to achieve good results for some experiments, which may argue that the choice is correct. The limited time to develop this master's thesis made it hard to dedicate a considerable amount of time to investigate other loss functions/performance metrics.

We argue that additional work can be done to further understand how transformer-based models handle eye-tracking data. Since these models are big, complex, and data-hungry, it would be of interest to repeat the experiments with larger datasets. Although our results for these models are not extremely positive, they do show some results that indicate that further training, could yield positive results. Moreover, we believe that the choice of a transformer-based model could drastically affect the results. Thus we argue that the following models could be tested: Firstly, a combination of GAN and Transformer such as TTS-GAN [114]. Basic Transformers seem limited by their deterministic nature when the task involves generating new data, especially in the stochastic random Gaussian Noise experiment. Therefore, adding the non-determinism of

GANs to the self-attention mechanism could yield very promising results. Secondly, Temporal Diffusion Transformers such as [115]. The authors of Ref.∼[116] argue that the Temporal Fusion Transformer (TFT) outperforms all prominent Deep Learning models for time series forecasting. These TFT models leverage self-attention to capture the complex temporal dynamics of multiple time sequences. This can potentially overcome the issues described by Ref.∼[97] and yield better results than the ones covered by this thesis. Finally, one other possible future work is focusing on a completely different type of loss function. Instead of using them to minimize the error, they can be used to replicate the path signature of the data. The path signature is a function that describes completely and uniquely a random process. Thus, it would be adequate for applying in RNNs that want to replicate a process with a random component to it [117].

Bibliography

[1] Kilian Semmelmann and Sarah Weigelt. 'Online webcam-based eye tracking in cognitive science: A first look'. In: *Behavior Research Methods* 50.2 (2018), pp. 451–465.

[2] Robert JK Jacob and Keith S Karn. 'Eye tracking in human-computer interaction and usability research: Ready to deliver the promises'. In: *The mind's eye*. Elsevier, 2003, pp. 573–605.

[3] Michel Wedel and Rik Pieters. 'A review of eye-tracking research in marketing'. In: *Review of marketing research* (2017), pp. 123–147.

[4] Mary Mallappallil et al. 'A review of big data and medical research'. In: *SAGE open medicine* 8 (2020), p. 2050312120934839.

[5] Peter Orbanz and Yee Whye Teh. 'Bayesian Nonparametric Models.' In: *Encyclopedia of machine learning* 1 (2010).

[6] Sharmila Subramaniam et al. 'Online outlier detection in sensor data using non-parametric models'. In: *Proceedings of the 32nd international conference on Very large data bases*. 2006, pp. 187–198.

[7] Brian L Smith, Billy M Williams and R Keith Oswald. 'Comparison of parametric and nonparametric models for traffic flow forecasting'. In: *Transportation Research Part C: Emerging Technologies* 10.4 (2002), pp. 303–321.

[8] Tanaya Chaudhuri et al. 'A feedforward neural network based indoor-climate control framework for thermal comfort and energy saving in buildings'. In: *Applied energy* 248 (2019), pp. 44–53.

[9] Saakaar Bhatnagar et al. 'Prediction of aerodynamic flow fields using convolutional neural networks'. In: *Computational Mechanics* 64.2 (2019), pp. 525–545.

[10] Gopal Chitalia et al. 'Robust short-term electrical load forecasting framework for commercial buildings using deep recurrent neural networks'. In: *Applied Energy* 278 (2020), p. 115410.

[11] Xiaoxin Zhu, Guanghai Zhang and Baiqing Sun. 'A comprehensive literature review of the demand forecasting methods of emergency resources from the perspective of artificial intelligence'. In: *Natural Hazards* 97.1 (2019), pp. 65–82.

[12] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[13] David A Van Dyk and Xiao-Li Meng. 'The art of data augmentation'. In: *Journal of Computational and Graphical Statistics* 10.1 (2001), pp. 1–50.

[14] Tianyang Lin et al. 'A survey of transformers'. In: *AI Open* (2022).

[15] Qingsong Wen et al. 'Transformers in time series: A survey'. In: *arXiv preprint arXiv:2202.07125* (2022).

[16] Joshua Bensemann et al. 'Eye Gaze and Self-attention: How Humans and Transformers Attend Words in Sentences'. In: *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*. 2022, pp. 75–87.

[17] Peter Vickers et al. 'CogNLP-Sheffield at CMCL 2021 Shared Task: Blending cognitively inspired features with transformer-based language models for predicting eye tracking patterns'. In: *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2021)*. Association for Computational Linguistics (ACL). 2021.

[18] Jennifer Romano Bergstrom and Andrew Schall. *Eye tracking in user experience design*. Elsevier, 2014.

[19] J Doležal and V Fabian. '41. Application of eye tracking in neuroscience'. In: *Clinical Neurophysiology* 126.3 (2015), e44.

[20] Pramodini A Punde, Mukti E Jadhav and Ramesh R Manza. 'A study of eye tracking technology and its applications'. In: *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)*. IEEE. 2017, pp. 86–90.

[21] Joanna N Lahey and Douglas R Oxley. 'Discrimination at the Intersection of Age, Race, and Gender: Evidence from an Eye-Tracking Experiment'. In: *Journal of Policy Analysis and Management* 40.4 (2021), pp. 1083–1119.

[22] Edwin AJ Van Hooft and Marise Ph Born. 'Intentional response distortion on personality tests: Using eye-tracking to understand response processes when faking.' In: *Journal of Applied Psychology* 97.2 (2012), p. 301.

[23] Mélodie Vidal et al. 'Wearable eye tracking for mental health monitoring'. In: *Computer Communications* 35.11 (2012), pp. 1306–1311.

[24] Pierre Maurage et al. 'Eye tracking correlates of acute alcohol consumption: A systematic and critical review'. In: *Neuroscience & Biobehavioral Reviews* 108 (2020), pp. 400–422.

[25] Xin Wang, Nicolas Thome and Matthieu Cord. 'Gaze latent support vector machine for image classification'. In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2016, pp. 236–240.

[26] Khaled Saab et al. 'Observational supervision for medical image classification using gaze data'. In: *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part II 24*. Springer. 2021, pp. 603–614.

[27] Bruce G Buchanan. 'A (very) brief history of artificial intelligence'. In: *Ai Magazine* 26.4 (2005), pp. 53–53.

[28] George EP Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[29] Enkelejda Kasneci et al. 'TüEyeQ, a rich IQ test performance data set with eye movement, educational and socio-demographic information'. In: *Scientific Data* 8.1 (2021), pp. 1–14.

[30] Sabrina Hoppe et al. 'Eye movements during everyday behavior predict personality traits'. In: *Frontiers in human neuroscience* (2018), p. 105.

[31] Jin H Yoon et al. 'Assessing attentional bias and inhibitory control in cannabis use disorder using an eye-tracking paradigm with personalized stimuli.' In: *Experimental and Clinical Psychopharmacology* 27.6 (2019), p. 578.

[32] Coralie Chevallier et al. 'Measuring social attention and motivation in autism spectrum disorder using eye-tracking: Stimulus type matters'. In: *Autism Research* 8.5 (2015), pp. 620–628.

[33] Jacob Leon Kröger, Otto Hans-Martin Lutz and Florian Müller. 'What does your gaze reveal about you? On the privacy implications of eye tracking'. In: *IFIP International Summer School on Privacy and Identity Management*. Springer. 2020, pp. 226–241.

[34] Jakob Lemos. *System and method for determining human emotion by analyzing eye properties*. US Patent App. 11/522,476. Mar. 2007.

[35] Maria K Eckstein et al. 'Beyond eye gaze: What else can eyetracking reveal about cognition and cognitive development?' In: *Developmental cognitive neuroscience* 25 (2017), pp. 69–91.

[36] Daniel J Liebling and Sören Preibusch. 'Privacy considerations for a pervasive eye tracking world'. In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. 2014, pp. 1169–1177.

[37] Julian Steil et al. 'Privacy-aware eye tracking using differential privacy'. In: *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*. 2019, pp. 1–9.

[38] Aileen Nielsen. *Practical time series analysis: Prediction with statistics and machine learning*. O'Reilly Media, 2019.

[39] Augustus D Waller. 'A demonstration on man of electromotive changes accompanying the heart's beat'. In: *The Journal of physiology* 8.5 (1887), p. 229.

[40] Andrew Schall and Jennifer Romano Bergstrom. 'Introduction to eye tracking'. In: *Eye tracking in user experience design*. Elsevier, 2014, pp. 3–26.

[41] Uzma Samadani et al. 'Eye tracking detects disconjugate eye movements associated with structural traumatic brain injury and concussion'. In: *Journal of neurotrauma* 32.8 (2015), pp. 548–556.

[42] Margarete Delazer et al. 'Eye-tracking provides a sensitive measure of exploration deficits after acute right MCA stroke'. In: *Frontiers in neurology* 9 (2018), p. 359.

[43] Michael D Crutcher et al. 'Eye tracking during a visual paired comparison task as a predictor of early dementia'. In: *American Journal of Alzheimer's Disease & Other Dementias®* 24.3 (2009), pp. 258–266.

[44] Alfred L Yarbus and Alfred L Yarbus. 'Eye movements during perception of complex objects'. In: *Eye movements and vision* (1967), pp. 171–211.

[45] Sept. 2014. URL: https://www.t-sciences.com/news/humans-process-visual-data-better.

[46] Yannis Ioannidis. 'The history of histograms (abridged)'. In: *Proceedings 2003 VLDB Conference*. Elsevier. 2003, pp. 19–30.

[47] André Fabio Kohn. 'Autocorrelation and Cross-Correlation Methods'. In: *Wiley Encyclopedia of Biomedical Engineering* (2006).

[48] Keith R Godfrey. 'Correlation methods'. In: *Automatica* 16.5 (1980), pp. 527–534.

[49] Guy Leonard Kouemou and Dr Przemyslaw Dymarski. 'History and theoretical basics of hidden Markov models'. In: *Hidden Markov models, theory and applications* 1 (2011).

[50] Gernot A Fink. *Markov models for pattern recognition: from theory to applications*. Springer Science & Business Media, 2014.

[51] Lawrence R Rabiner. 'A tutorial on hidden Markov models and selected applications in speech recognition'. In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286.

[52] Khar Heng Choo, Joo Chuan Tong and Louxin Zhang. 'Recent applications of hidden Markov models in computational biology'. In: *Genomics, proteomics & bioinformatics* 2.2 (2004), pp. 84–96.

[53] Pedro Lind et al. 'Modern Ai Versus Century-Old Mathematical Models: How Far Can We Go with Generative Adversarial Networks to Reproduce Stochastic Processes?' In: *Available at SSRN 4305494* ().

[54] Philippe C Besse, Hervé Cardot and David B Stephenson. 'Autoregressive forecasting of some functional climatic variations'. In: *Scandinavian Journal of Statistics* 27.4 (2000), pp. 673–687.

[55] Baki Billah et al. 'Exponential smoothing model selection for forecasting'. In: *International journal of forecasting* 22.2 (2006), pp. 239–247.

[56] Bryan Lim and Stefan Zohren. 'Time-series forecasting with deep learning: a survey'. In: *Philosophical Transactions of the Royal Society A* 379.2194 (2021), p. 20200209.

[57] Yann LeCun, Yoshua Bengio and Geoffrey Hinton. 'Deep learning'. In: *nature* 521.7553 (2015), pp. 436–444.

[58] Omer Berat Sezer, Mehmet Ugur Gudelek and Ahmet Murat Ozbayoglu. 'Financial time series forecasting with deep learning: A systematic literature review: 2005–2019'. In: *Applied soft computing* 90 (2020), p. 106181.

[59] Antonia Creswell et al. 'Generative adversarial networks: An overview'. In: *IEEE signal processing magazine* 35.1 (2018), pp. 53–65.

[60] Changhee Han et al. 'GAN-based synthetic brain MR image generation'. In: *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*. IEEE. 2018, pp. 734–738.

[61] Hajar Emami et al. 'Spa-gan: Spatial attention gan for image-to-image translation'. In: *IEEE Transactions on Multimedia* 23 (2020), pp. 391–401.

[62] Uddeshya Upadhyay and Suyash P Awate. 'Robust super-resolution GAN, with manifold-based and perception loss'. In: *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE. 2019, pp. 1372–1376.

[63] Wenju Xu et al. 'Drb-gan: A dynamic resblock generative adversarial network for artistic style transfer'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6383–6392.

[64] Tero Karras, Samuli Laine and Timo Aila. 'A style-based generator architecture for generative adversarial networks'. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4401–4410.

[65] Qing Li et al. 'A Multi-step ahead photovoltaic power forecasting model based on TimeGAN, Soft DTW-based K-medoids clustering, and a CNN-GRU hybrid neural network'. In: *Energy Reports* 8 (2022), pp. 10346–10362.

[66] Domenico Santoro and Luca Grilli. 'Generative Adversarial Network to evaluate quantity of information in financial markets'. In: *Neural Computing and Applications* (2022), pp. 1–18.

[67] Yanjing Li and Xinyuan Liu. 'An Intelligent Music Production Technology Based on Generation Confrontation Mechanism'. In: *Computational Intelligence and Neuroscience* 2022 (2022).

[68] Kunfeng Wang et al. 'Generative adversarial networks: introduction and outlook'. In: *IEEE/CAA Journal of Automatica Sinica* 4.4 (2017), pp. 588–598.

[69] Lillian J Ratliff, Samuel A Burden and S Shankar Sastry. 'Characterization and computation of local Nash equilibria in continuous games'. In: *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2013, pp. 917–924.

[70] Saloni Dash et al. 'Medical time-series data generation using generative adversarial networks'. In: *International Conference on Artificial Intelligence in Medicine*. Springer. 2020, pp. 382–391.

[71] Lilian Weng. 'From gan to wgan'. In: *arXiv preprint arXiv:1904.08994* (2019).

[72] Kanchan M Tarwani and Swathi Edem. 'Survey on recurrent neural network in natural language processing'. In: *Int. J. Eng. Trends Technol* 48.6 (2017), pp. 301–304.

[73] Gábor Petneházi. 'Recurrent neural networks for time series forecasting'. In: *arXiv preprint arXiv:1901.00069* (2019).

[74] Aran Nayebi and Matt Vitelli. 'Gruv: Algorithmic music generation using recurrent neural networks'. In: *Course CS224D: Deep Learning for Natural Language Processing (Stanford)* (2015), p. 52.

[75] Jonathan Goh et al. 'Anomaly detection in cyber physical systems using recurrent neural networks'. In: *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*. IEEE. 2017, pp. 140–145.

[76] Larry R Medsker and LC Jain. 'Recurrent neural networks'. In: *Design and Applications* 5 (2001), pp. 64–67.

[77] James L McClelland, David E Rumelhart, PDP Research Group et al. *Parallel Distributed Processing, Volume 2: Explorations in the Microstructure of Cognition: Psychological and Biological Models*. Vol. 2. MIT press, 1987.

[78] June 2017. URL: https://commons.wikimedia.org/w/index.php?curid=60109157.

[79] Sepp Hochreiter and Jürgen Schmidhuber. 'Long short-term memory'. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[80] Lirong Yao and Yazhuo Guan. 'An improved LSTM structure for natural language processing'. In: *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*. IEEE. 2018, pp. 565–569.

[81] Steven Elsworth and Stefan Güttel. 'Time series forecasting using LSTM networks: A symbolic approach'. In: *arXiv preprint arXiv:2003.05672* (2020).

[82] Lianli Gao et al. 'Video captioning with attention-based LSTM and semantic consistency'. In: *IEEE Transactions on Multimedia* 19.9 (2017), pp. 2045–2055.

[83] Kamilya Smagulova and Alex Pappachen James. 'A survey on LSTM memristive neural network architectures and applications'. In: *The European Physical Journal Special Topics* 228.10 (2019), pp. 2313–2324.

[84] Christopher Olah. 'Understanding lstm networks'. In: (2015).

[85] May 2018. URL: https://commons.wikimedia.org/w/index.php?curid=109362147.

[86] Khandu Om et al. 'Modelling email traffic workloads with RNN and LSTM models'. In: *Human-centric Computing and Information Sciences* 10.1 (2020), pp. 1–16.

[87] Alex Graves and Jürgen Schmidhuber. 'Framewise phoneme classification with bidirectional LSTM and other neural network architectures'. In: *Neural networks* 18.5-6 (2005), pp. 602–610.

[88] Zhiyong Cui et al. 'Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values'. In: *Transportation Research Part C: Emerging Technologies* 118 (2020), p. 102674.

[89] Felix A Gers and Jürgen Schmidhuber. 'Recurrent nets that time and count'. In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. Vol. 3. IEEE. 2000, pp. 189–194.

[90] Kyunghyun Cho et al. 'Learning phrase representations using RNN encoder-decoder for statistical machine translation'. In: *arXiv preprint arXiv:1406.1078* (2014).

[91] Ashish Vaswani et al. 'Attention is all you need'. In: *Advances in neural information processing systems* 30 (2017).

[92] Jacob Devlin et al. 'Bert: Pre-training of deep bidirectional transformers for language understanding'. In: *arXiv preprint arXiv:1810.04805* (2018).

[93] Tom Brown et al. 'Language models are few-shot learners'. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[94] Colin Raffel et al. 'Exploring the limits of transfer learning with a unified text-to-text transformer.' In: *J. Mach. Learn. Res.* 21.140 (2020), pp. 1–67.

[95] Zihang Dai et al. 'Transformer-xl: Language modeling with longer-term dependency'. In: (2018).

[96] Shiyang Li et al. 'Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting'. In: *Advances in neural information processing systems* 32 (2019).

[97] Ailing Zeng et al. 'Are Transformers Effective for Time Series Forecasting?' In: *arXiv preprint arXiv:2205.13504* (2022).

[98] Alexander Christian DeRieux. 'Transformer Networks for Smart Cities: Framework and Application to Makassar Smart Garden Alleys'. PhD thesis. Virginia Tech, 2022.

[99] ML Menéndez et al. 'The jensen-shannon divergence'. In: *Journal of the Franklin Institute* 334.2 (1997), pp. 307–318.

[100] Jinsung Yoon, Daniel Jarrett and Mihaela Van der Schaar. 'Time-series generative adversarial networks'. In: *Advances in neural information processing systems* 32 (2019).

[101] Cristóbal Esteban, Stephanie L Hyland and Gunnar Rätsch. 'Real-valued (medical) time series generation with recurrent conditional gans'. In: *arXiv preprint arXiv:1706.02633* (2017).

[102] Hao Ni et al. 'Conditional sig-wasserstein gans for time series generation'. In: *arXiv preprint arXiv:2006.05421* (2020).

[103] Yihua Cheng and Feng Lu. 'Gaze estimation using transformer'. In: *arXiv preprint arXiv:2105.14424* (2021).

[104] Aditya Jonnalagadda et al. 'Foveater: Foveated transformer for image classification'. In: *arXiv preprint arXiv:2105.14173* (2021).

[105] Chao Gou, Yuchen Zhou and Dan Li. 'Driver attention prediction based on convolution and transformers'. In: *The Journal of Supercomputing* 78.6 (2022), pp. 8268–8284.

[106] Nian Liu et al. 'Visual saliency transformer'. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 4722–4732.

[107] Shaochen Wang et al. 'What You See is What You Grasp: User-Friendly Grasping Guided by Near-eye-tracking'. In: *arXiv preprint arXiv:2209.06122* (2022).

[108] Tim Rolff et al. 'GazeTransformer: Gaze Forecasting for Virtual Reality Using Transformer Networks'. In: *Pattern Recognition: 44th DAGM German Conference, DAGM GCPR 2022, Konstanz, Germany, September 27–30, 2022, Proceedings*. Springer. 2022, pp. 577–593.

[109] Haoyi Zhou et al. 'Informer: Beyond efficient transformer for long sequence time-series forecasting'. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 12. 2021, pp. 11106–11115.

[110] Haixu Wu et al. 'Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting'. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 22419–22430.

[111] Tian Zhou et al. 'Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting'. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 27268–27286.

[112] Eric Fosler-Lussier. 'Markov models and hidden Markov models: A brief tutorial'. In: *International Computer Science Institute* (1998).

[113] Ketan Doshi. *Transformers Explained Visually (Part 1): Overview of Functionality*. June 2021. URL: https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452.

[114] Xiaomin Li et al. 'Tts-gan: A transformer-based time-series generative adversarial network'. In: *Artificial Intelligence in Medicine: 20th International Conference on Artificial Intelligence in Medicine, AIME 2022, Halifax, NS, Canada, June 14–17, 2022, Proceedings*. Springer. 2022, pp. 133–143.

[115] Bryan Lim et al. 'Temporal fusion transformers for interpretable multi-horizon time series forecasting'. In: *International Journal of Forecasting* 37.4 (2021), pp. 1748–1764.

[116] Shereen Elsayed et al. 'Do we really need deep learning models for time series forecasting?' In: *arXiv preprint arXiv:2101.02118* (2021).

[117] Andrey Kormilitzin. *What is the signature method?* URL: https://kormilitzin.github.io/.

Appendix A

Appendix

A.1 Code

The code can be found as a zip file in the delivery of the thesis. It can also be found in the GitHub repository: https://github.com/ArnauNaval/MasterThesis_EyeTrackingForecast. Since the repository is private, permissions need to be requested.