Chair of Cyber Physical Systems

Master's Thesis

Deep Neural Energy Price Forecasting for the Hydrogen Industry

Klemens Lechner, BSc

February 2024

MONTANUNIVERSITÄT LEOBEN
www.unileoben.ac.at

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich diese Arbeit selbstständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt, den Einsatz von generativen Methoden und Modellen der künstlichen Intelligenz vollständig und wahrheitsgetreu ausgewiesen habe, und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe.

Ich erkläre, dass ich den Satzungsteil „Gute wissenschaftliche Praxis" der Montanuniversität Leoben gelesen, verstanden und befolgt habe.

Weiters erkläre ich, dass die elektronische und gedruckte Version der eingereichten wissenschaftlichen Abschlussarbeit formal und inhaltlich identisch sind.

Datum 05.02.2024

K. Lechner

Unterschrift Verfasser/in
Klemens Lechner

ACKNOWLEDGEMENTS

This work is dedicated to my family and friends.
Starting with my parents, who not only always supported me during my studies, but also gave me the best starting conditions a prospective academic could wish for. I would also like to thank my brother and sister-in-law, who have always been there to help and advise me. I would also like to take this opportunity to thank my partner, as you offered me a haven of peace in stressful times.
I would also like to take this opportunity to thank my good friends Dr. Stefan Leitner and (soon to be) Dr. Daniel Rathmaier, who have often made me rethink my theories more carefully with their perspicacity. Of course, this applies equally to my brother.
Finally, I would like to thank my supervisors from CPS and HyCentA for their constant adherence to deadlines and solution-oriented discourse.

DANKSAGUNGEN

Diese Arbeit ist meiner Familie und meinen Freunden gewidmet.
Angefangen bei meinen Eltern, die mich nicht nur während meines Studiums immer unterstützt haben, sondern mir auch die besten Startbedingungen gegeben haben, die man sich als angehender Akademiker nur wünschen kann. Ebenso möchte ich mich bei meinem Bruder und meiner Schwägerin bedanken, die mir immer mit Rat und Tat zur Seite standen. Auch meiner Partnerin möchte ich an dieser Stelle danken, da du mir in stressigen Zeiten einen Ruhepol geboten hast.
An dieser Stelle möchte ich mich auch bei meinen guten Freunden Dr. Stefan Leitner und (baldigem) Dr. Daniel Rathmaier bedanken, die mich durch ihren Scharfsinn oft dazu gebracht haben, meine Theorien nocheinmal genauer zu überdenken. Dies gilt natürlich in gleichem Maße für meinen Bruder.
Zuletzt möchte ich mich bei meinen Betreuern von CPS und HyCentA für die stete Termintreue und den lösungsorientierten Diskurs bedanken.

ABSTRACT

The HyCentA Research GmbH analyses, among other things, the technical and economic design of hydrogen plants. Since hydrogen plants are almost exclusively operated electrically, the price of electricity plays a major role in operating costs. Based on transformer models, the electricity price should be predicted for various scenarios. These scenarios consist of the electricity mix (what percentage of the electricity comes from which source), own generation and the gas price. The gas price was added to the electricity generation data because it has a major influence on the electricity price due to the merit order system. This could be observed, among other things within the Ukraine crisis (2022).
In addition, these transformer models were used to identify electricity price trends depending on the type of electricity generation. As the energy system in Europe is moving towards more renewable energies, the electricity mix is also changing towards these. A strong positive trend towards lower electricity prices was observed for wind energy and biomass in particular. The opposite trend was observed for solar power generation: Electricity prices rose as solar power generation increased. However, it was observed that even small amounts of solar power in the electricity mix reduce the price of electricity. This means that the electricity price initially starts at a lower price and only rises later. The later increase could be explained by the increased need for balancing energy due to the volatile nature of solar power generation. However, this effect still is subject to further investigation.

KURZFASSUNG

Die HyCentA Research GmbH beschäftigt sich unter anderem mit der technisch-wirtschaftlichen Auslegung von Wasserstoffanlagen. Da Wasserstoffanlagen fast ausschließlich elektrisch betrieben werden, spielt der Strompreis eine wesentliche Rolle bei den Betriebskosten. Mit Hilfe von Transformermodellen soll der Strompreis für verschiedene Szenarien vorausgesagt werden. Diese Szenarien bestehen aus dem Strommix (welcher Anteil des Stroms kommt aus welcher Quelle), der Eigenerzeugung und dem Gaspreis. Der Gaspreis wurde den Stromerzeugungsdaten hinzugefügt, da er aufgrund des Merit-Order-Systems einen großen Einfluss auf den Strompreis hat. Dies konnte unter anderem im Rahmen der Ukraine-Krise (2022) beobachtet werden.
Darüber hinaus wurden diese Transformermodelle verwendet, um Strompreistrends in Abhängigkeit von der Art der Stromerzeugung zu identifizieren. Da sich das Energiesystem in Europa hin zu mehr erneuerbaren Energien bewegt, verändert sich auch der Strommix hin zu diesen. Ein starker positiver Trend zu niedrigeren Strompreisen wurde insbesondere bei Windenergie und Biomasse beobachtet. Bei der solaren Stromerzeugung war der umgekehrte Trend zu beobachten: Die Strompreise stiegen mit zunehmender Solarstromerzeugung. Es wurde jedoch beobachtet, dass bereits geringe Mengen an Solarstrom im Strommix den Strompreis senken. Das bedeutet, dass der Strompreis zunächst auf einem niedrigeren Preisniveau startet und erst später ansteigt. Der spätere Anstieg könnte durch den erhöhten Bedarf an Regelenergie aufgrund der volatilen Natur der Solarstromerzeugung erklärt werden. Dieser Effekt muss jedoch noch genauer untersucht werden.

Contents

List of Figures

List of Tables

1 Introduction

Contents

This thesis was realised in co-operation between the Institute for Cyber Physical Systems at the Montanuniversität Leoben and Hycenta Research GmbH.

1.1 Motivation

HyCenta Research GmbH deals with many topics related to the hydrogen industry. One of these is the technical and economic design of hydrogen plants. Because hydrogen plants are almost always operated electrically, the price of electricity plays a major role in their operating costs. In order to enable a more precise technical and economic design, a tool was to be developed that estimates the electricity price for various scenarios. In order to estimate the electricity price, a data basis was first sought. Data from ENTSO-E was used as the most important data basis. The ENTSO-E is the European Network of Transmission System Operators for Electricity, that serves as a organization responsible for offering extensive data related to the electricity market. This includes the electricity price and various data on generation and consumption. However, the interaction of this data is often difficult to interpret. For this reason, suitable statistical and technical options were sought to analyse this interaction. As it has been shown many times that algorithms based on artificial intelligence are good at dealing with large amounts of data (Kulkarni and Bairagi, 2018), these were selected as the analysis tool.

1.2 Research Questions

The research objectives are articulated as follows:

1. How do different determinants such as the electricity mix (the proportion of energy from various generation sources), in-country generation, and gas prices, influence the cost of electricity?
2. Which machine learning approaches/algorithms are most suitable for accurately predicting future electricity price trends, particularly in Austria or other European countries?
3. How does the model's sensitivity to input variables, such as solar and wind energy, impact its accuracy and reliability in forecasting electricity prices?

In order to increase user-friendliness later on, a tool should also be developed so that several scenarios can be tested quickly. The tool should have the inputs stated above and output the electricity price curve for one year for this scenario, as shown in Figure 1.

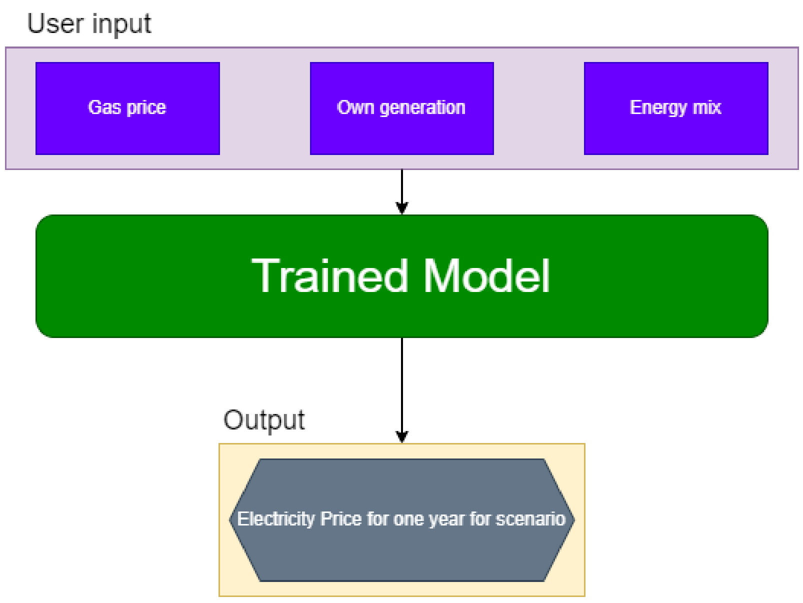


Figure 1: *Tool for scenario analysis*

1.3 Related Work

The exploration of electricity markets and methodologies for price prediction draws upon a diverse and rich array of scholarly contributions, each providing unique insights into market dynamics, data collection and preparation, and the application of predictive modeling techniques.

Geissmann and Obrist delve into the fundamental price drivers in "Continental European Day-Ahead Power Markets" (Geissmann and Obrist, 2018), offering critical insights into how generation mixes and fossil fuel prices impact electricity prices, emphasizing the need for renewable energy promotion and capacity remuneration mechanisms.

Meeus's comprehensive detailing of the evolution of electricity markets in Europe (Meeus, 2020) provides an in-depth look at the market's evolution post-liberalization, focusing on the integration challenges of renewable energies and the regulatory adaptations that have emerged.

Géron's book (Géron, 2019) serves as an essential guide for researchers entering the field of machine learning, offering practical applications with Scikit-Learn, Keras, and TensorFlow, thereby laying the groundwork for predictive modeling.

The introduction of Long Short-Term Memory (LSTM) networks by Hochreiter and Schmidhuber (Hochreiter and Schmidhuber, 1997) marked a significant advancement, enabling models to effectively learn long-term dependencies. Olah's explanation (Olah, 2015) of LSTMs makes this complex mechanism more accessible for various applications, including electricity price forecasting.

Vaswani et al.'s introduction of the Transformer model (Vaswani et al., 2017) has revolutionized sequence modeling, emphasizing the efficiency of attention mechanisms.

The reliability of data from the ENTSO-E Transparency Platform is critically assessed by Hirth, Mühlenpfordt, and Bulkeley, 2018, highlighting the importance of data cleansing and analysis for robust modeling.

Li et al.'s study (Li and Becker, 2021) showcases the capabilities of LSTM models in processing complex time-series data and uncovering market dynamics, essential for forecasting accuracy.

The efficiency and adaptability of Transformer models in financial markets are explored by Yunsi, 2022, demonstrating their superiority in capturing market trends.

Comparative analyses between LSTM and Transformer models (Zhao, Crane, and Bezbradica, 2022, Muhammad et al., 2022) highlight their potential in financial forecasting, with an emphasis on sentiment analysis.

"Transformers in Time-Series Analysis: A Tutorial" (Ahmed et al., 2023) and "Natural Language Processing with Transformers" (Tunstall, Werra, and Wolf, 2021) provide insights into leveraging Transformer models for time-series analysis and data preparation, emphasizing positional encoding.

"Deep Learning with PyTorch Step-by-Step" (Godoy, 2022) familiarizes researchers with LSTM network data preparation, enhancing model application in forecasting.

Adding to this foundation, "Electricity Price Forecasting on the Day-Ahead Market Using Machine Learning" by Tschora (Tschora et al., 2022) discusses various approaches to forecasting electricity prices, offering an initial overview of methods.

1.4 Thesis Outlook

In order to provide a roadmap for the reader the structure of the thesis is presented. The thesis will be organized as follows:
Firstly, a theoretical background is provided to clarify the methods used. Then the data basis, the foundation of every data research project, is presented. First the data generation and then the clean-up and analyses that contribute to the understanding of this data is shown. Then the usable model architectures are worked out one after the other. Initially, rudimentary models are tested, which gradually become more complex. The chosen approach is then presented. This consists of an ensemble of 5 transformer models. Then the results of this ensemble are shown.

2 Theoretical Background

Contents

This section sets out the theoretical foundations of the work. These are divided into two sections. First, the European electricity grid and electricity pricing are explained. Then Long-short term memory networks and Transformers are explained.

2.1 Electricity market in Europe

The energy market in Europe is characterized by European cooperation. There are five synchronous zones in Europe (Figure 2). A synchronous zone is characterized by the fact that the grid frequency and the phase position in the zone are identical.

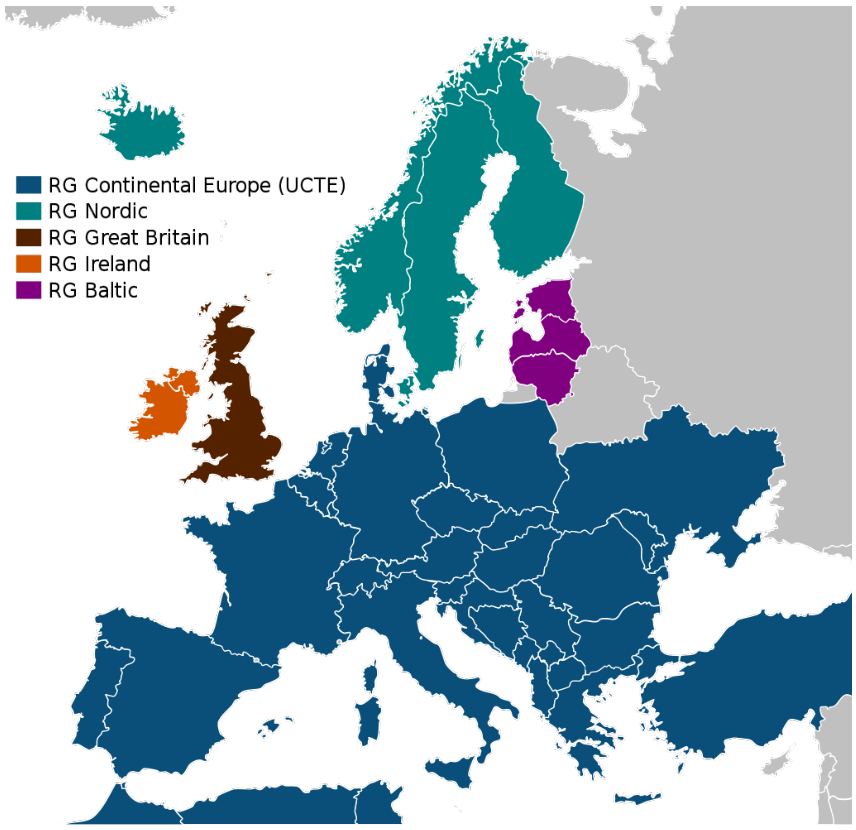


Figure 2: *Synchronous Zones in Europe. Source: Kimdime, 2006*

These synchronous zones are connected with high-voltage direct current (HVDC) transmission. Therefore, no frequency regulation can take place between the zones. One major reason why direct current is used is because it proofs to be more suitable for undersea transmission (Long and Litzenberger, 2012).

Combining several countries into a synchronous zone makes the electricity grid more stable overall. For example, if a power plant in France breaks down, Austria can provide the energy difference by switching on pumped storage power plants. As this can prevent grid failures, grid stability is improved.

These transmission grids also make it possible for different countries to trade electricity with each other. This trade was first regulated by the EU directive "Liberalization of the electricity market" in 1997 and was followed by the complete opening of the electricity markets in 2007 (Leuschner, 2023).

2.1.1 Pricing for electricity

The electricity price is made up of 3 components:

- Price for the energy
- Price for the use of the grid
- Taxes

The fee for using the grid, as well as taxes, are set by the respective country. The fees and taxed are influenced by factors such as environmental policies, economic considerations, and national energy strategies.(Harding, Martini, and Thomas, 2016) This means that the electricity grid is not subject to the rules of the market. This should ensure that the focus of the electricity grids in Europe is not on profit, but on grid stability.
In the field of energy economics, creating a free market for energy is seen as a way to encourage competition and lower prices. The key places for this competition are the electricity exchanges, where electricity is traded. The European Energy Exchange (EEX) is the biggest of these exchanges in Europe. It serves as a main center for trading energy products and derivatives, connecting buyers and sellers, which helps in setting prices and making the European energy market more transparent.

There, the "Merit-Order System" is used to determine the price of electricity.

2.1.2 Merit-Order System

The merit order system is the system chosen for electricity pricing in the European Union.
In this context, the merit order represents the supply curve. The electricity price then results from the intersection of a generally price-independent demand and the merit order (Fraunhofer-Institut für Solare Energiesysteme ISE, 2024).
In perfect competition, all market participants are price takers. Under these conditions, it is rational for a supplier to offer its energy as soon as it is above the marginal costs, i.e. above the price that the *next* unit of energy costs. Then the power plant with the highest marginal costs, which is just needed to cover the demand, determines the electricity price. This power plant is also referred to as market clearing power plant.

The merit order system proceeds as follows:
Two curves are formed, namely a demand and a supply curve.
Supply curve: The demand curve is created by each power plant operator stating how many MW they plan to produce and the price for which they are willing to sell this energy. Typically, marginal costs are specified as the price, i.e. how much it costs to produce the *next* MWh. This information from all power plant operators is then summarised to form the demand curve. In (Figure 3) a simplified version of this can be seen. Naturally, renewable energies are on the left-hand side of this diagram, as the marginal costs for the production of renewable energy are de facto zero.

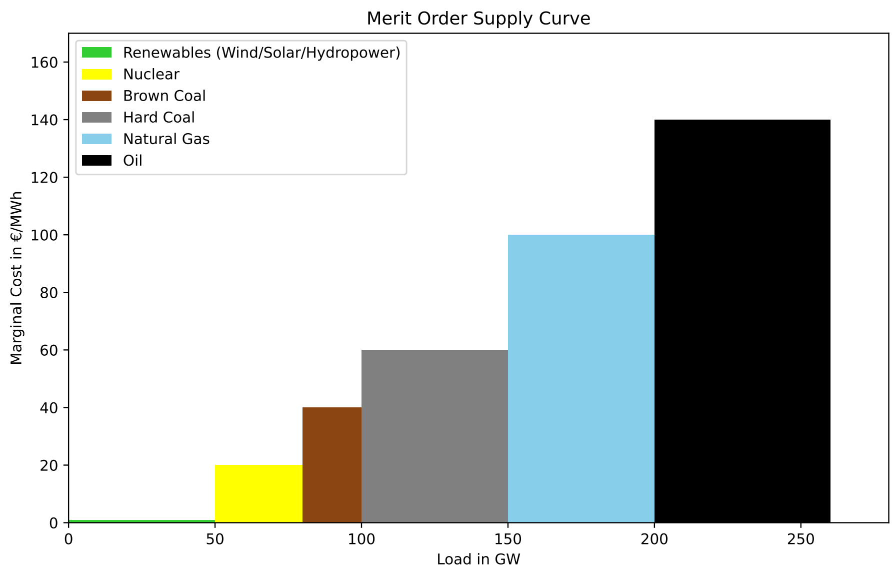


Figure 3: *Merit Order Supply Curve*

Demand curve:

Simultaneously, the demand curve is formed, this is also a staggered curve. Each electricity buyer specifies both the quantity, measured in megawatts (MW), they intend to purchase and the corresponding price at which they are willing to pay (Figure 4).

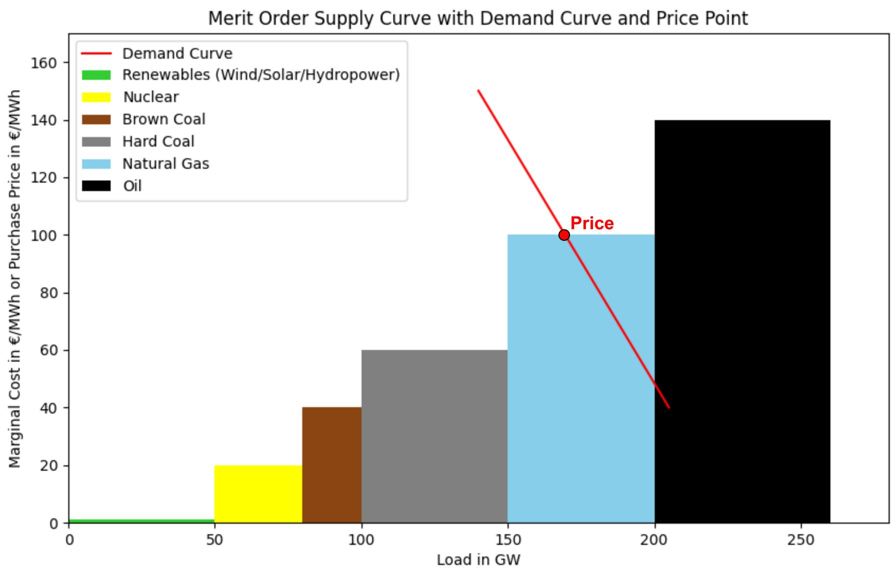


Figure 4: *Merit Order Supply Curve with Demand Curve and Price Point*

This is how the electricity price is calculated. *Each* participant pays the same price or gets it paid. So the operator of a solar park gets paid the same amount for one MWh as the operator of a nuclear power plant.

This system has advantages and disadvantages. One advantage is that renewable energies are indirectly promoted because they benefit from the higher marginal costs of other types of power plants. A disadvantage was observed during the Ukraine crisis (2022). Due to the sharp rise in gas prices and the resulting sharp rise in the cost of electricity from natural gas, electricity in Europe became significantly more expensive. Another effect of the merit order is that it is hoped that fossil fuels, which have higher marginal costs, will be pushed out of the market in the future.

2.2 Machine Learning for Regression Tasks

This chapter deals with the network architectures used in this thesis. For an introduction to this topic, the author recommends the book "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron (Géron, 2019. [1].
There are several options for predicting serial data, such as time series. In recent years, three architectures in particular have become established: Gated Recurrent Unit (GRU), Long-short Term Memory Networks (LSTM) and Transformers (Pra, 2023). LSTMs and transformers are explained in more detail here, as they were used in the work.

2.2.1 Word embedding

The examples given in the sections for the LSTMs (2.2.2) and transformers (2.2.3) come from Natural Language Processing (NLP), i.e. the processing/translation/etc. of human language. These examples are often very illustrative and should also be used here. To understand how an algorithm can process words, "word embedding" is explained first.

"Word embedding" means making words readable for an algorithm. To do this, you have to convert a word into numbers, or more precisely, a vector. With word embedding, you don't just take the letters and convert them into numbers, you also give the word context. For example, you could assign a k-dimensional vector to a word. Let's take the word "King" as an example. Each of these k numbers have a meaning. For example, the first number could mean: "Does this word have something to do with power". For "King" this would be positive (whatever positive means, it could mean that the number is getting bigger, for example). The second number would mean: "Does this word have anything to do with castles". Again, "King" would be positive. In reality, these vectors are not so easy to interpret. But what can be observed is that a functioning algorithm for "word embedding" assigns similar vectors to similar words, e.g. king and princess.

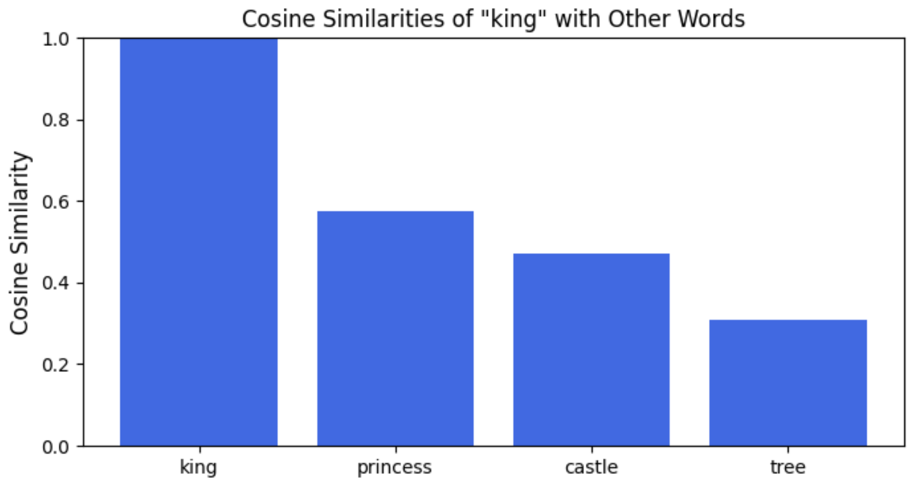


Figure 5: *Cosine Similarity of "King" with other words*

[1]The author also recommends the video series on neural networks from the Youtube channel 3Blue1Brown as a very basic introduction with good visualisations

In (Figure 5) it can be seen that "king" is most similar to "princess", second most similar to "castle" and most different to "tree". The cosine similarity was chosen as the measure for the similarities; this indicates how much the vectors point in one direction. The formula is given by:

$$\text{cosine similarity}(A, B) = \frac{A \cdot B}{\|A\| \|B\|}$$

Here, A and B are vectors. $A \cdot B$ denotes the dot product of the vectors, and $\|A\|$, $\|B\|$ are the length of the vectors.
The word embeddings were loaded from the python library "gensim" (Rehurek and Sojka, 2011) and they are 100-dimensional vectors.

2.2.2 Long-short Term Memory Networks

Long-short term memory networks (LSTMs) are a further development of recurrent neural networks (RNNs). LSTMs should solve the problem of the "vanishing gradient". "Vanishing gradient" means that the gradient, which is used to update the weights and biases, which allow the network to learn from data, is close to 0 and therefore "disappears".

Let's take a sentence "The bank of the river was very beautiful". This sentence must be used as a whole because if you only see the word "bank", you would think of a financial institution rather than a riverbed. This only becomes clear with the word "river". That's why it's important to give words context, so to put them in relation to other words.

The first development in this area were simple RNNs. RNNs not only take the input at time t, but also the output of $t-1$ (Figure 6). x_t denotes the input for the time step t. A denotes a neural network and h_t the output at time t. The loop should symbolize that the previous output is also fed into the network.

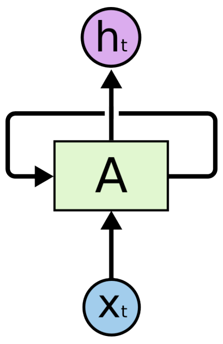


Figure 6: *RNN. Source: Olah, 2015*

To better understand this loop, this architecture can be unrolled (Figure 7).

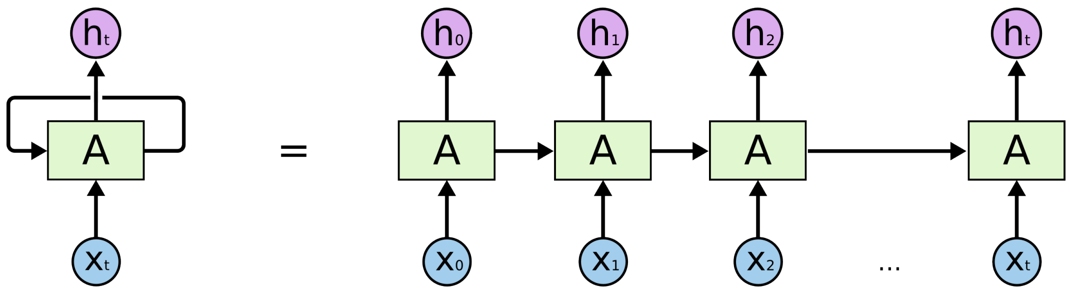


Figure 7: *RNN unrolled. Source: Olah, 2015*

A always represents the same neural network. The input is the concatenated output of the previous step h_{t-1} and the input x_t. Sometimes this simple structure is completely sufficient. For example, when a neural network is trained to predict the next word based on previous words. In the sentence fragment "The clouds are in the ..." it is pretty easy to say that the next word should be "sky". So the distance between the relevant words is comparatively small. A simple RNN would work well here (Figure 8) (Olah, 2015).

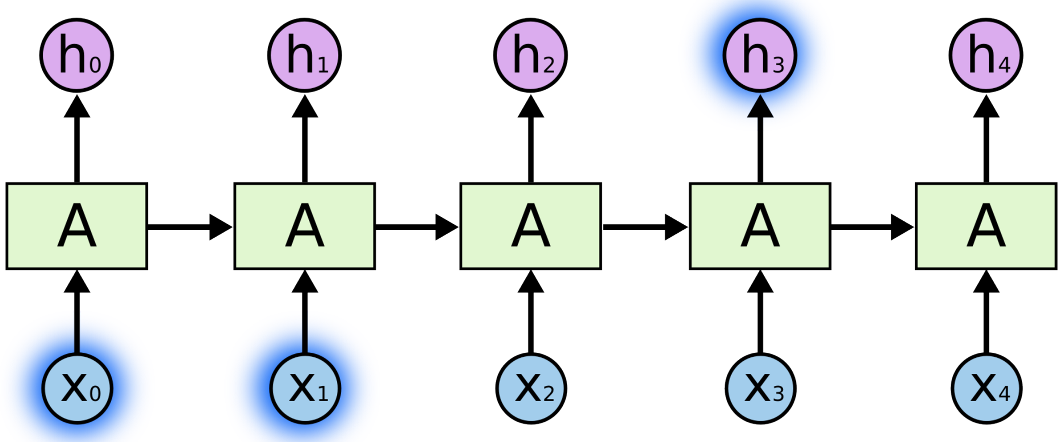


Figure 8: *RNN Short Term Dependency. Source: Olah, 2015*

Here the example would be that the input of x_0 and x_1 are important for the output h_3. In other words, a short term dependency.

But if we have longer sequences, for example "I grew up in France. I had a very nice childhood and this is also the reason why I fluently speak ...". Here, "french" would be a very likely answer. However, the words are further apart than in the previous example. This is called long term dependency (Figure 9). Simple RNNs are not good at modeling these long term dependencies. LSTMs are better at dealing with this problem because they are better at mapping long term dependencies by solving the vanishing gradient problem. (Olah, 2015).

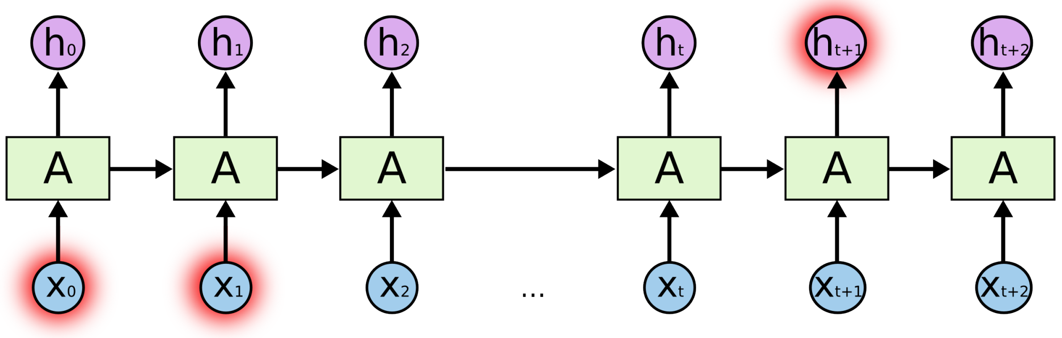


Figure 9: *Long Term Dependency. Source: Olah, 2015*

LSTMs were introduced by Hochreiter and Schmidhuber in 1997 (Hochreiter and Schmidhuber, 1997) and are a special form of RNNs. As shown, all RNNs form a chain of repeating modules. In standard RNNs, these modules can have a simple form, such as a single tanh layer (Figure 10) (Olah, 2015). A tanhh layer outputs a value between -1 and 1 (Figure 14)

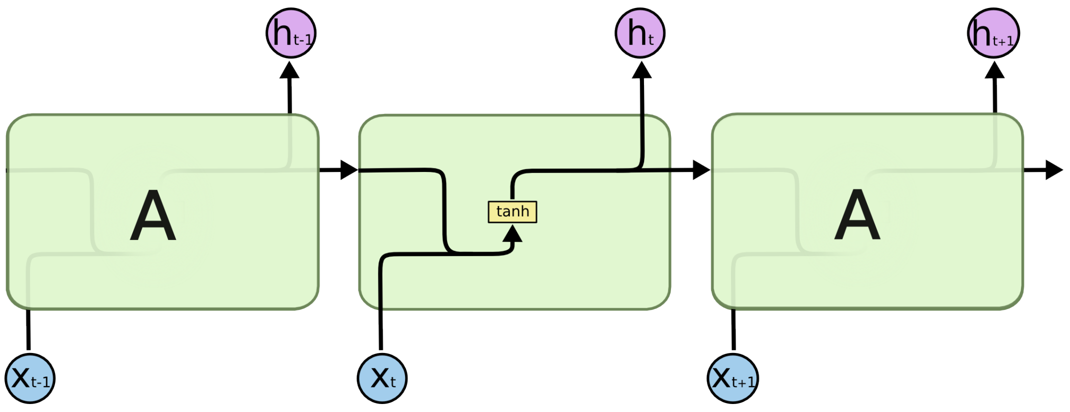


Figure 10: *Simple RNN. Source: Olah, 2015*

LSTMs also have a chain structure, but have a different internal structure (Figure 11) (Figure 12).

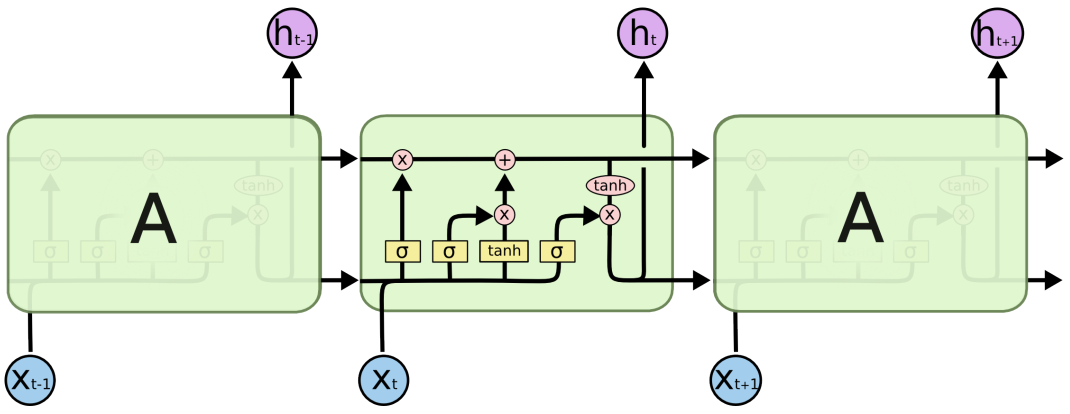


Figure 11: *Chain of LSTM. Source: Olah, 2015*

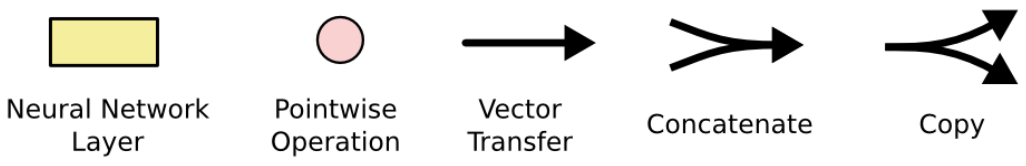


Figure 12: *Notation für LSTM structure. Source: Olah, 2015*

In the figure (Figure 11), each line carries a vector. The operations are shown in (Figure 12).

σ stands for a neural net layer with sigmoid activation function. This activation function returns a value between 0 and 1 as the function value (Figure 13).

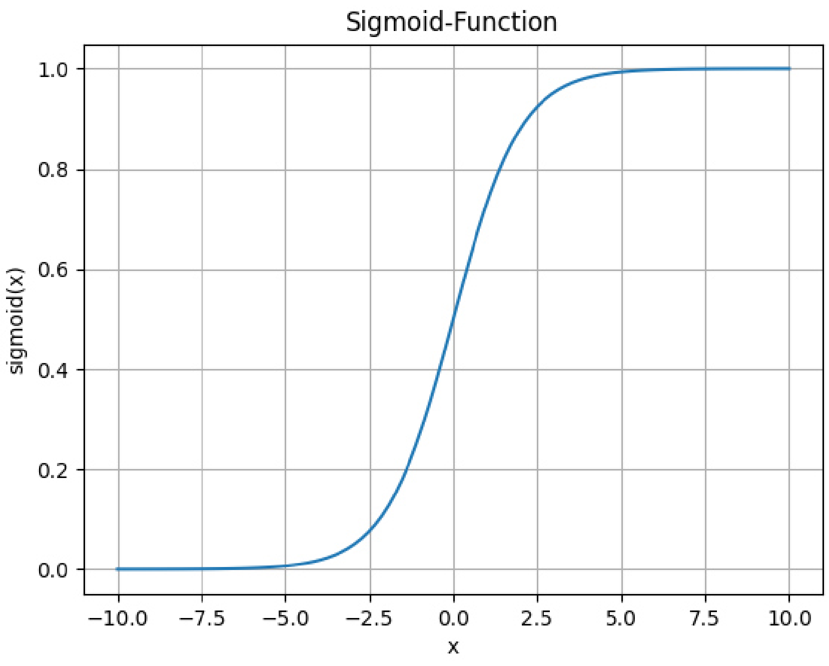


Figure 13: *Sigmoid Function.*

$tanh$ stands for a neural net layer with a tanh activation function. This activation function returns a value between -1 and 1 as the function value (Figure 14).

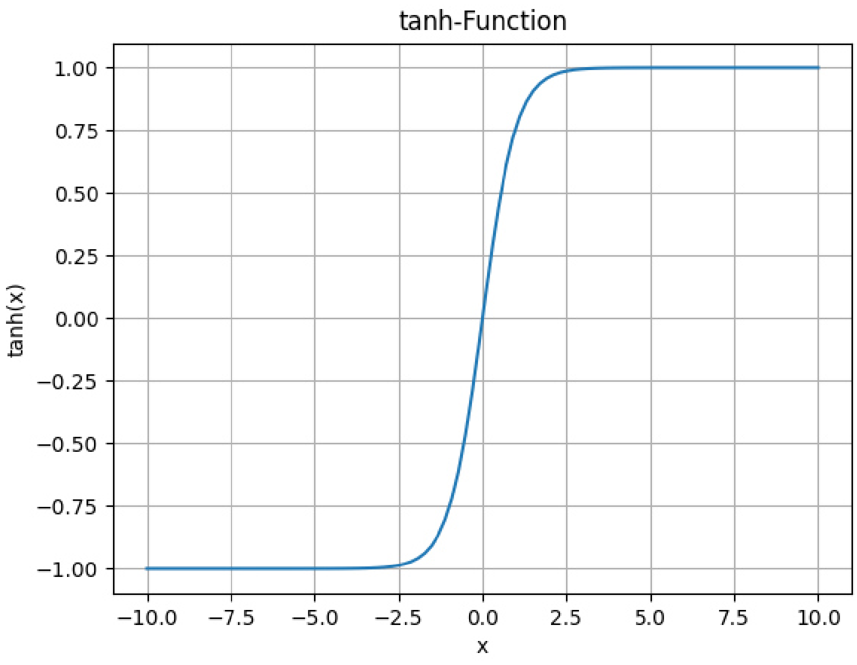


Figure 14: *tanh Function.*

The upper line represents the cell state. This is the heart of the LSTM (Figure 15).

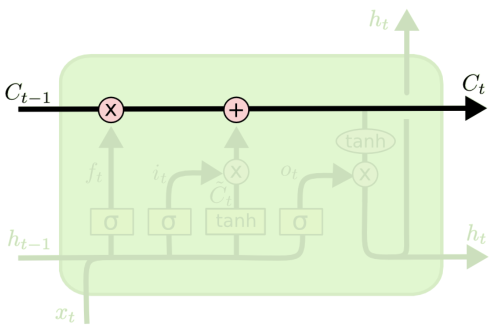


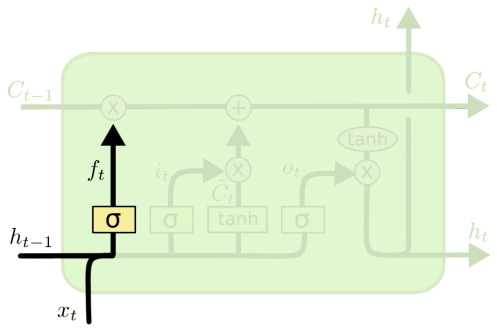
Figure 15: *Cell state of LSTM. Source: Olah, 2015*

The cell state can be changed via the so-called gates.

Forget Gate:

The forget gate has a sigmoid activation function (Figure 16). As shown in (Figure 13), the sigmoid function returns a function value between 0 and 1. As for all gates, the input is the concatenated vector of the hidden state of the previous time h_{t-1} and the input of the current time x_t. Thus, the dimension of the input of the forget gate layer (the σ in Figure 16) is equal to the concatenated vector of the hidden state h_{t-1} and the input x_t. The output vector f_t has a dimension that is equal to the dimension of the cell state C_t. Thus, a vector is output from the forget gate that contains a number between 0 and 1 for each number in the vector C. 0 means "forget this entry", 1 means "keep this entry completely". W_f denotes the weights and b_f the bias terms which are located in the forget layer and are adjusted during training. The formula shown in Figure 16 therefore has the following effect: The vectors h_{t-1} and x_t are concatenated. They are then multiplied by the weights W_f and the bias term

b_f is added. This creates a new vector from which the sigmoid activation function is then calculated and the result is the vector f_t.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \;+\; b_f\right)$$

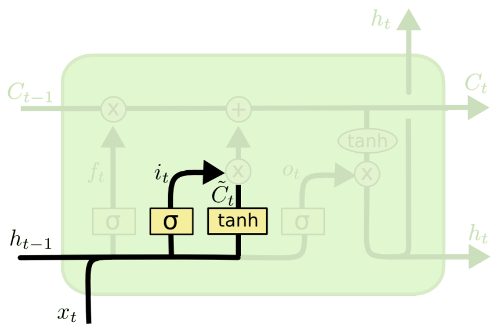
Figure 16: *Forget Gate. Source: Olah, 2015*

Input Gate and Candidate for Cell State Update:

In Figure 17 we see two operations. One is a sigmoid and the other is a tanh activated layer. This combination is used to update the cell state based on the new information. Two major reasons for this combination are:

- To control the flow of information: As in the other gates, the sigmoid function helps to control the flow of information because it can take values between 0 and 1
- Vanishing gradient problem: The tanh function has a stronger gradient. This solves the problem of the vanishing gradient for longer sequences. The derivative and thus the strength of the gradient is shown in figure (Figure 18).

$\tilde{C}_t$ denotes the vector that comes as output from the cell state update and i_t the output from the input gate. $W_{i,C}$ the weights and $b_{i,C}$ the bias term.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \;+\; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \;+\; b_C)$$

Figure 17: *Input Gate and Candidate for Cell State Update. Source: Olah, 2015*

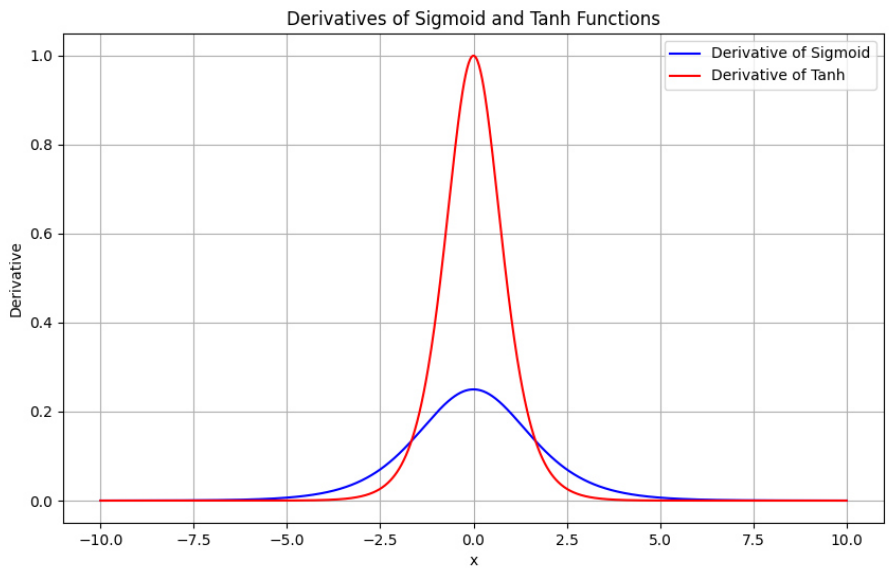


Figure 18: *Derivatives of Sigmoid and Tanh Functions.*

The two outputs i_t and $\tilde{C}_t$, are then both pointwise multiplied and then added to the cell state (Figure 19). The cell state is now updated.

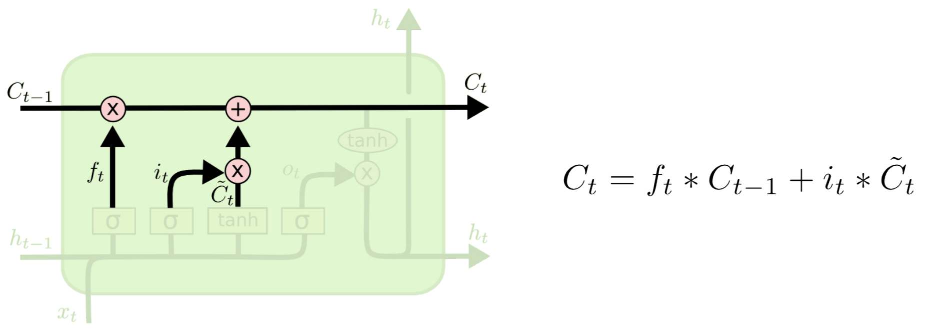
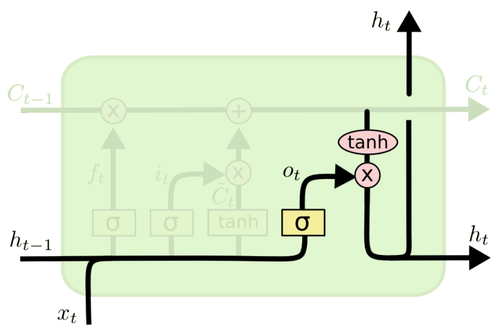


Figure 19: *New Cell State. Source: Olah, 2015*

Output gate:

The final output is a refined version of the cell state, subjected to a two-step filtering process (Figure 20). Initially, a sigmoid layer is applied which creates the vector o_t. This functions as a selector, determining the specific components of the cell state to be relayed forward. Subsequently, the cell state is processed through a $\tanh$ function, which normalizes its values to fall within the range of -1 and 1. This normalized state is then element-wise multiplied by the output of the sigmoid gate o_t, ensuring that only the selectively retained parts of the cell state are output.



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 20: *Output of LSTM. Source: Olah, 2015*

This new type of network architecture made it possible to solve many problems where long-term dependencies need to be mapped. These problems range from translation problems to time series data and much more. The main reason, as mentioned above, is the solution of the vanishing gradient problem (Kang, Zhang, and Liu, 2016).

2.2.3 Transformer

The basic idea for Transformers was presented by Vaswani et al. in the publication "Attention is all you need" (Vaswani et al., 2017). It shows that significant improvements in NLP applications can be achieved by implementing attention mechanisms. Attention mechanisms direct the "attention" of the network to certain parts of the data, so to speak. This means that the value of certain positions is considered more important for the prediction than others. This is by no means static, i.e. the network learns that, for example, the 3rd position is always the most important. Instead, the "attention" is directed to the right position depending on the input. This idea can be applied not only to language but also to any other form of serial data, including time series.

As already shown in the considerations on LSTMs, context plays a major role in NLP (2.2.2). "The bank of the river was very beautiful" was given as an example. In this example, the word "river" makes it clear that the bank is not a financial institution but a riverbank. So the attention of the word "bank" in this case should be drawn to "river", this is what context means.

The attention mechanism in transformers is called "Multi-Head Attention". A Multi-Head Attention is made up of several self-attention mechanisms.

Self-attention:

Take the shorter version of our example sentence, "bank of the river". First, the word embedding is carried out. So vectors are assigned to the individual words. Now context should be added to these vectors, i.e., the words should be linked to the other words. This is done using self-attention. The flow chart for this is as follows (Figure 21):

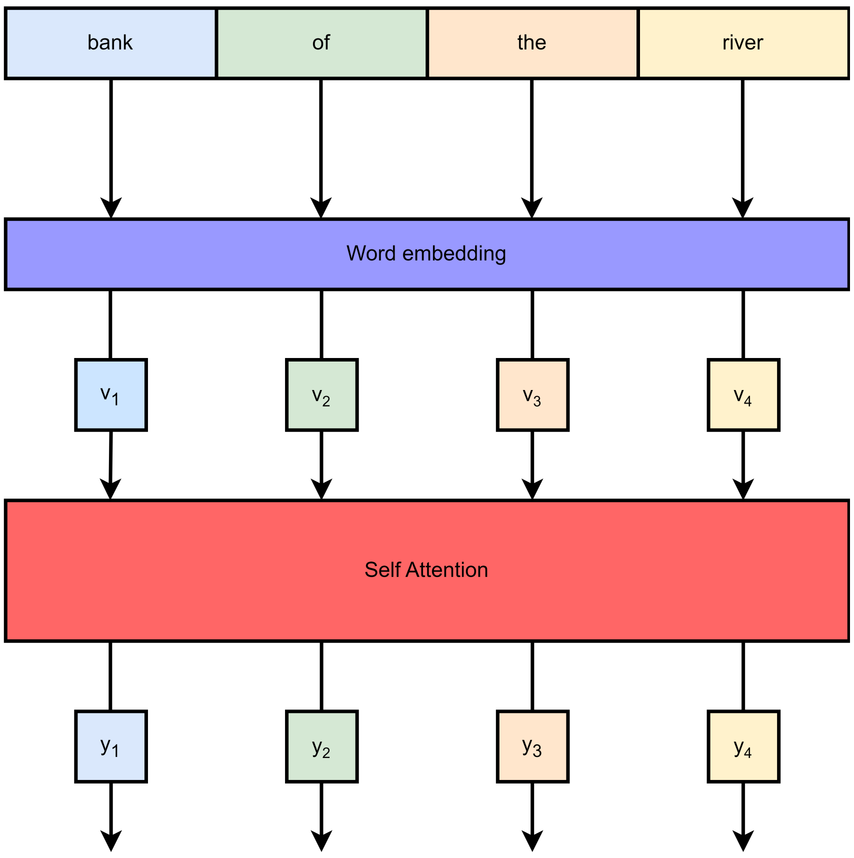


Figure 21: *Flow Chart for Self Attention*

In (Figure 21) you first the sentence "bank of the river" is given. Then the word embedding is performed and the words become the respective vectors $\vec{v_i}$. Then context is added to the vectors in the self-attention and they become the vectors $\vec{y_i}$.

The self-attention mechanism should now be explained in more detail (Figure 22):

It should apply:

$\vec{v_1} = "bank"; \vec{v_2} = "of"; \vec{v_3} = "the"; \vec{v_4} = "river";$

Actual word embeddings were also used here for visualization purposes. These were also imported from the python library gensim (Rehurek and Sojka, 2011). Each of these word embeddings is a 100-dimensional vector.

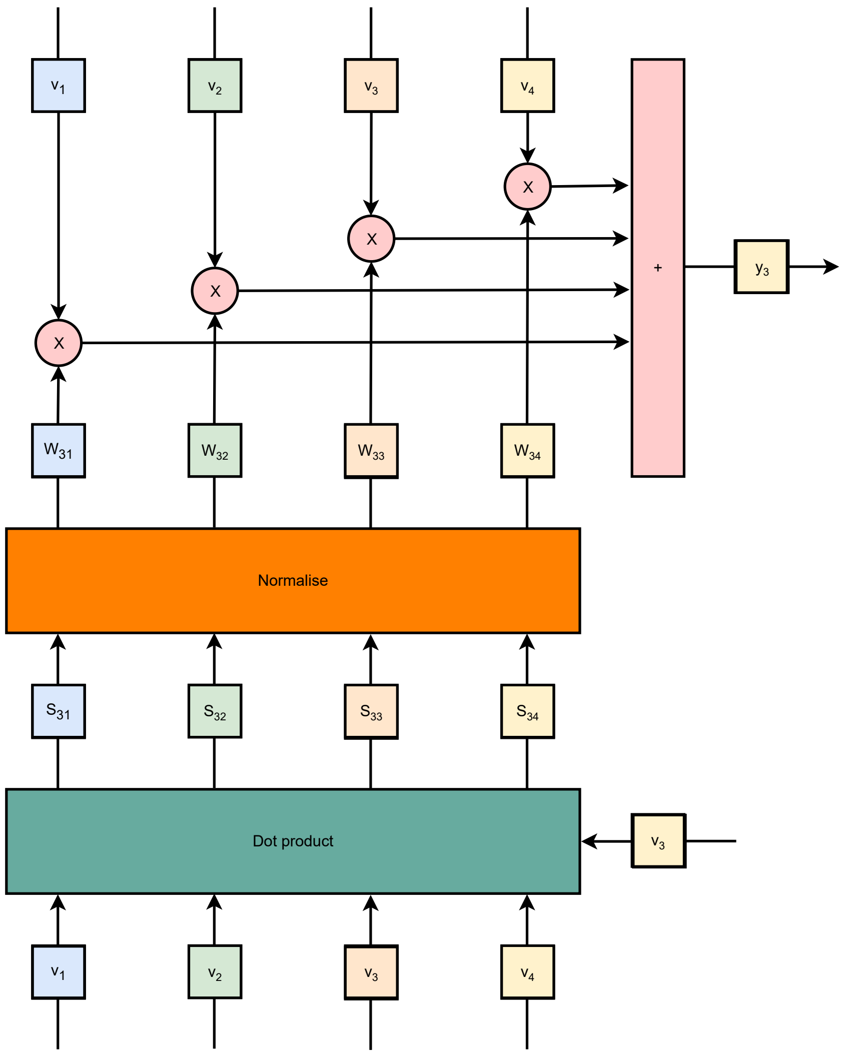


Figure 22: *Self Attention without weights*

In (Figure 22 shows the flow chart for the vector v_3. The same routine is performed for all vectors. First, the dot product is formed:

$$S_{ij} = \vec{v_i} \cdot \vec{v_j}, \quad i, j = 1, 2, 3, 4 \tag{1}$$

For our words, the dot product then looks like this(Figure 23:

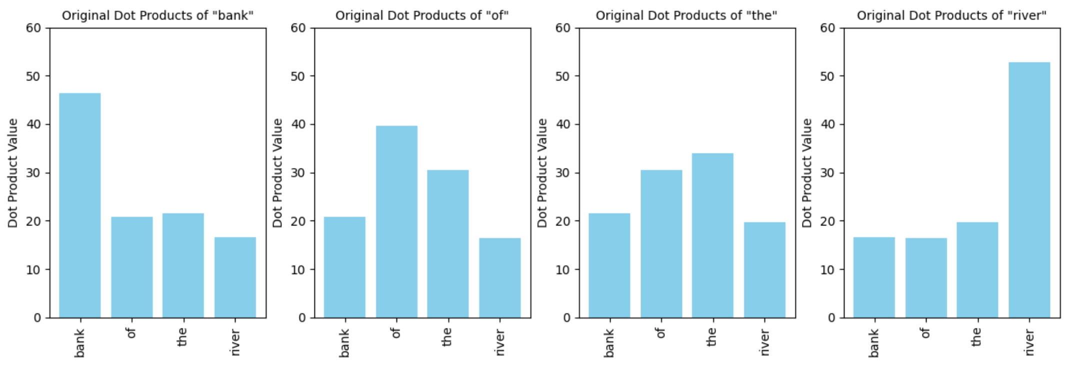


Figure 23: *Dot Product of Vectors*

This dot product must then be normalized. This is due to the fact, that neural networks better work with values that are in a certain range. To do this, first divide by the root of the dimension of the vector d_k (Equation 2). In our case, d_k is equal to 100.

$$S'_{ij} = \frac{S_{ij}}{\sqrt{d_k}} \tag{2}$$

Then the softmax function is applied.

$$\text{softmax}(W'_{ij}) = \frac{e^{S'_{ij}}}{\sum_j e^{S'_{ij}}} \tag{3}$$

Equation 3 ensures that the every input is between 0 and 1 and also normalizes them so that the sum is equal to 1.

With our word embeddings, this would then take this form (Figure 24):

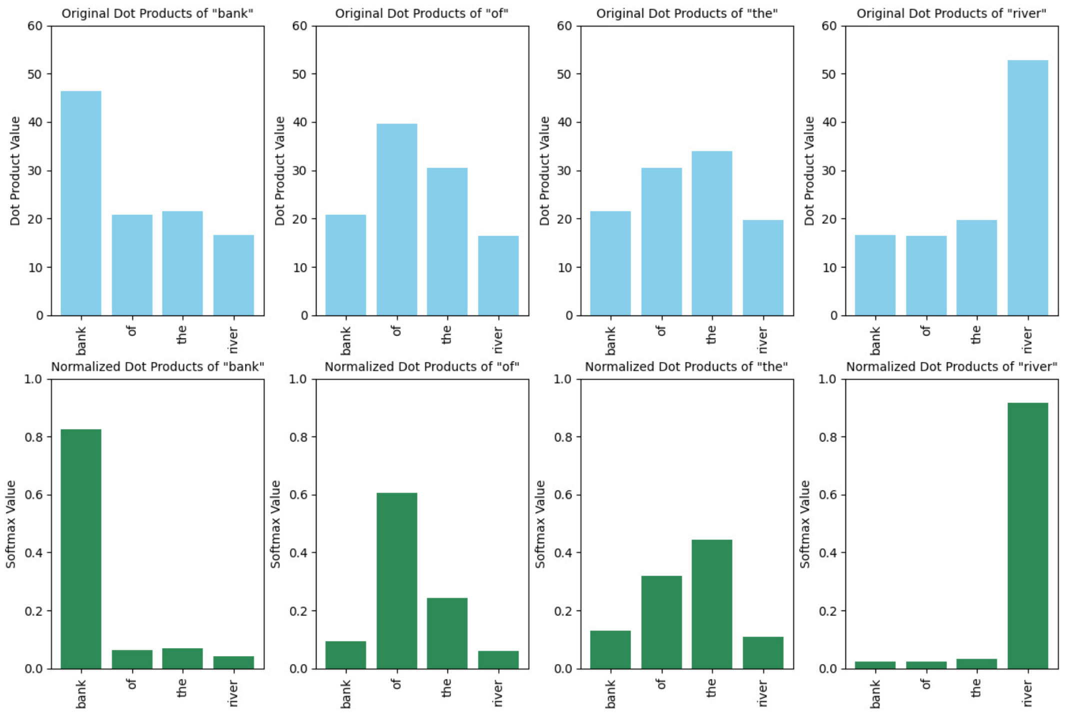


Figure 24: *Dot Product and Noramlized Dot Products of Vectors*

If we had not applied the Equation 2, the exponent for the self-correlation (e.g. "bank" with "bank") in Equation 3 would become so large that the normalized value would be equal to 1 and all others would be equal to 0. This is why this scaling is necessary. Because without it the vectors would in most cases just remain the same.

Now, as shown in Figure 22, we have to multiply these normalized dot products by the respective vectors. To illustrate the entire process, the entire operations for the $\vec{v_3}$ are shown again:

1st step, form dot products:

$$S_{3j} = \vec{v_3} \cdot \vec{v_j}, \quad i, j = 1, 2, 3, 4 \tag{4}$$

2nd step, normalize the dot products:

$$S'_{3j} = \frac{S_{3j}}{\sqrt{d_k}} \tag{5}$$

$$\text{softmax}(W_{3j}) = \frac{e^{S'_{3j}}}{\sum_j e^{S'_{3j}}} \tag{6}$$

3rd step, multiply the vectors with the normalized dot products and form the sum:

$$\vec{y_3} = W_{31}\vec{v_1} + W_{32}\vec{v_2} + W_{33}\vec{v_3} + W_{34}\vec{v_4} \tag{7}$$

This then leads to our new weighted vectors (Figure 25).

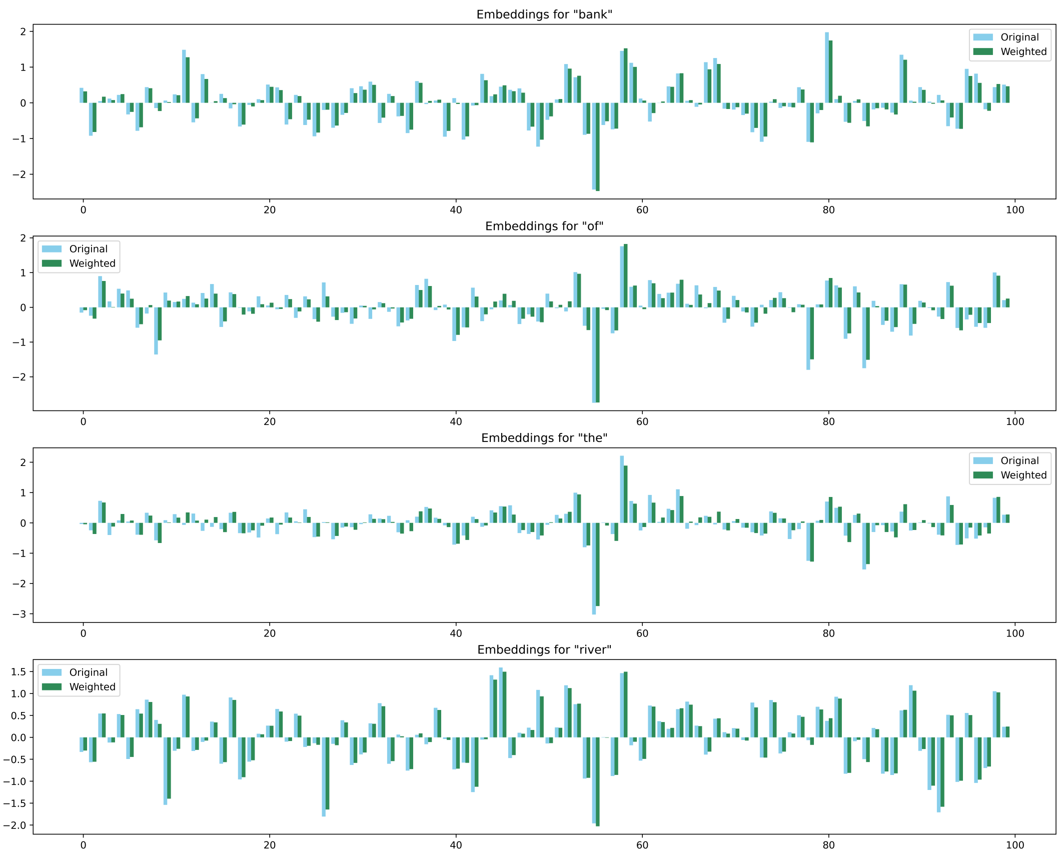


Figure 25: *Original vs. weighted word embeddings*

So far, the vectors have only been changed based on how similar the word embeddings are. However, in order to be able to give context to the different vectors , i.e. to learn the relationships in natural language, weights are introduced. These weights can then be changed during training according to the data. So the patterns within the data can be learned by adjusting this weights. This is a basic operation in nearly every neural network. For a good introduction to this topic, the author recommends the video series for neural networks of the YouTube channel 3Blue1Brown . There, basic neural networks are explained with very good visualisations.

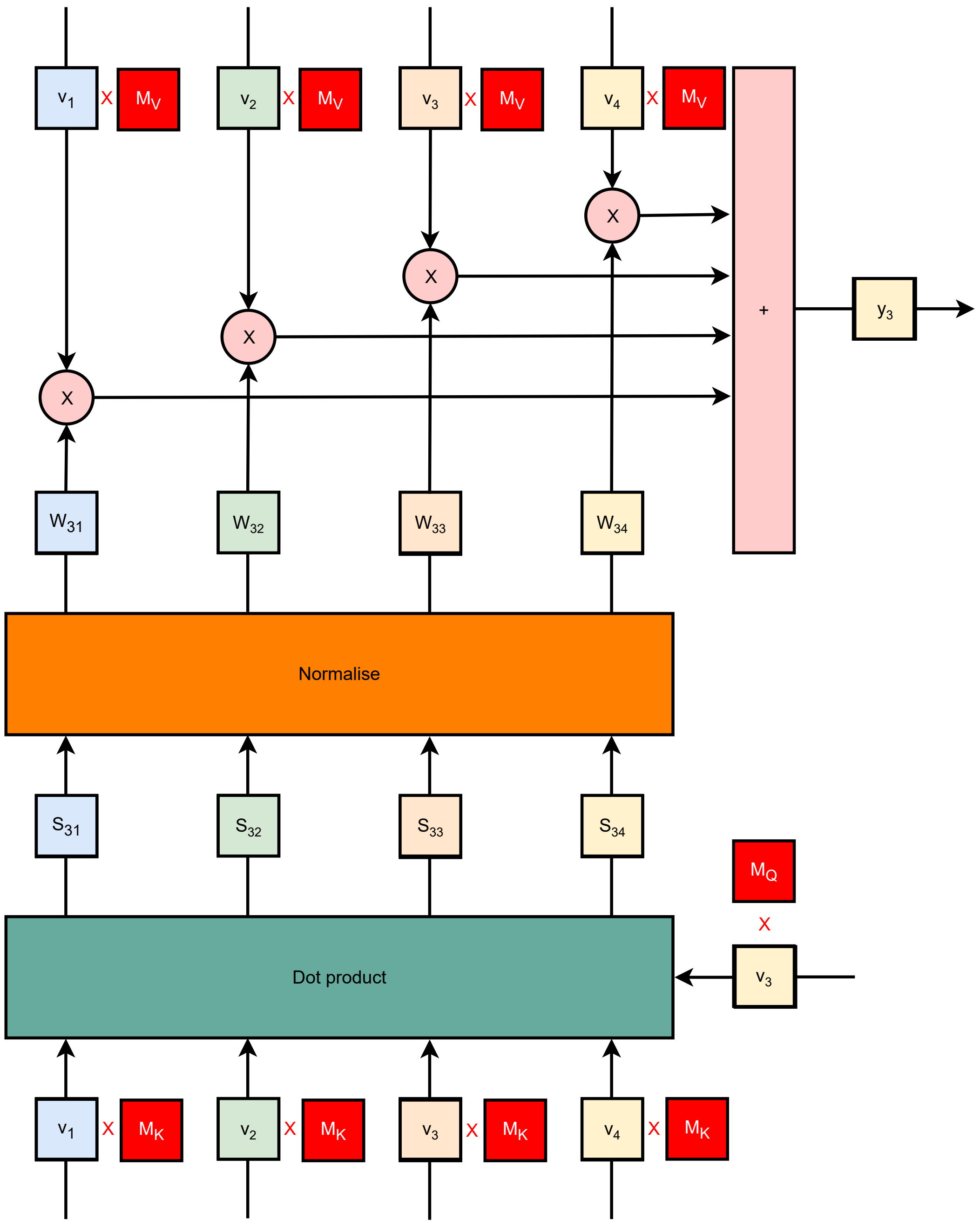


Figure 26: *Self Attention with weights*

The weights, M_K, M_Q and M_V, as seen in (Figure 26), are matrices. These matrices have the dimension $d_{\text{input}} \times d_{\text{head}}$. d_{input} stands for the dimension of the input vectors $\vec{v_i}$ and d_{head} is an adjustable hyperparameter. This makes it possible to change the respective vectors by multiplying them with these matrices. The values within the matrices are updated by a gradient signal. This is also a very common operation in neural networks.

The matrices that can be seen in (Figure 26) have the following names:

$$M_K\text{: Key Matrix}$$
$$M_Q\text{: Query Matrix}$$
$$M_V\text{: Value Matrix}$$

Name origin of query, key and value:
In the flow chart (Figure 26) you can see that a type of query is made by vector $\vec{v_3}$, similar to a database. This is where the term "query" comes from for the vector for which the self-attention is formed. The vectors with which the $\vec{v_3}$ forms the dot product can be seen as "keys" in a database. The vectors with which the normalized dot product is then multiplied can be seen as "values". This analogy does not quite hold up, but that is where the names for the matrices come from.

Now you have weights that you can train. This enables you to learn from data. To show how the vector changes, the following visualization was created.

We have two example sentences:

1. The bank of the river was beautiful.
2. The bank was far away from the river.

These were fed into an already trained model, the BERT (Bidirectional Encoder Representations from Transformers) model. The cosine similarity was then determined. This indicates how similar two vectors are.

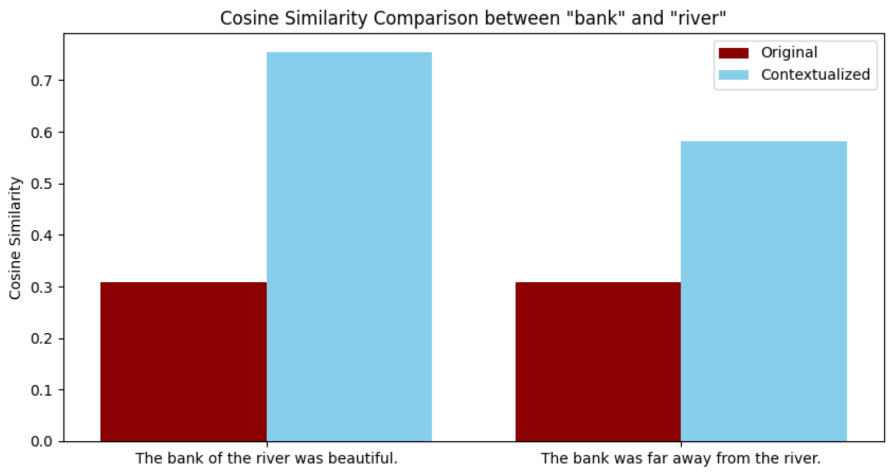


Figure 27: *Original vs. contextualized cosine similarity between "bank" and "river"*

In (Figure 27) you can see how the vector changes when context is added. In the first example sentence, "bank" and "river" influence each other directly because "river" changes the semantics of "bank". This leads to the vectors becoming more similar to each other. The tansformer recognizes, so to speak, that these two words belong together. In the second sentence, this influence is not nearly as strong as in the first, because the semantics of "bank" are not changed by "river". Nevertheless, the vector is changed in the transformer or more precise in the attention mechanisms of the transformer. However, it is difficult to interpret exactly why the vectors are changed by these values.

Since the multiplication with the respective key, query and value matrices has no bias terms, this multiplication can be regarded as a linear layer. This allows us to draw the Self Attention Block as a neural network:

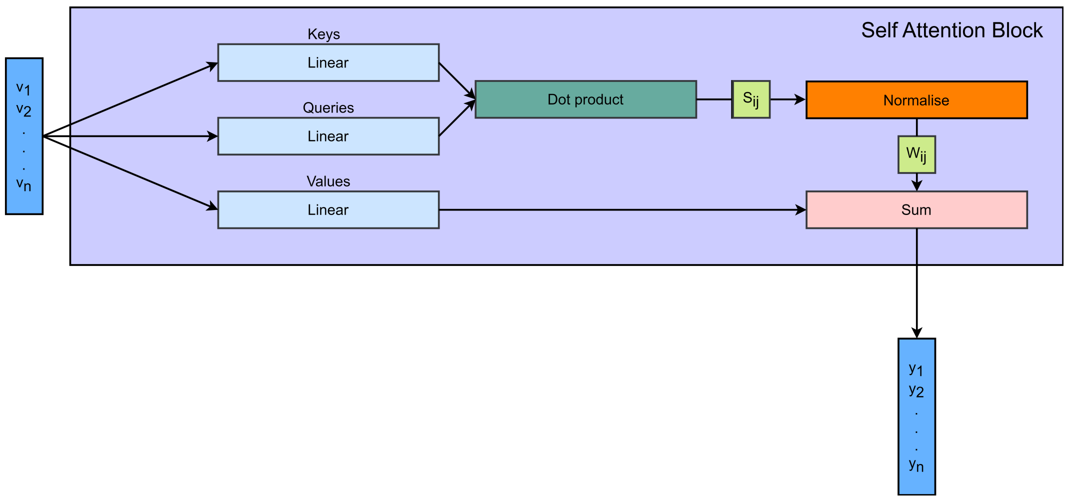


Figure 28: *Self Attention as Neural Network*

Now we have a so-called Self Attention Block (Figure 28). But if the sequences are longer than our simple example "bank of the river", it makes sense to use not just one of these attention mechanisms, but several. For example, the sentence "I gave my cat something to eat during the day." You could ask several questions, for example "Who gave the cat something to eat?", "When was the cat fed?", etc. In each of these cases, different words of the sentence become important.
This is why several of these Self Attention Blocks are used in parallel. This is the so-called "Multi-Head Attention" (Figure 29.

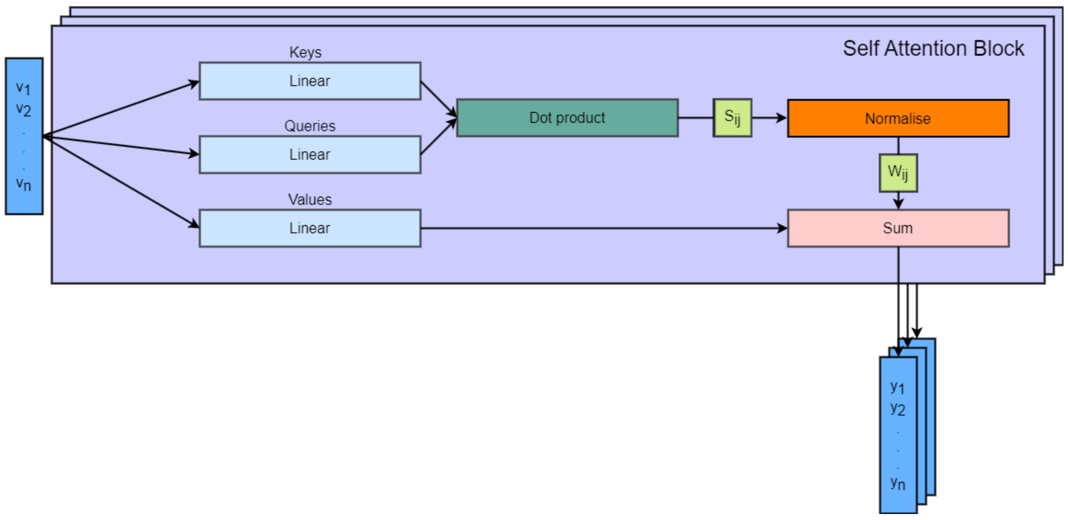


Figure 29: *Multi-Head Attention without Output Treatment*

Here, the same vectors (the word embeddings) are each placed in a Self Attention Block and the operations already shown are performed. However, each of these Self Attention Blocks has its own output. This must therefore still be handled. This is done by concatenating the vectors with each other and then passing them through a dense layer. This is a common method to preserve the information from all vectors as far as possible but to reduce the output to one vector (Figure 30).

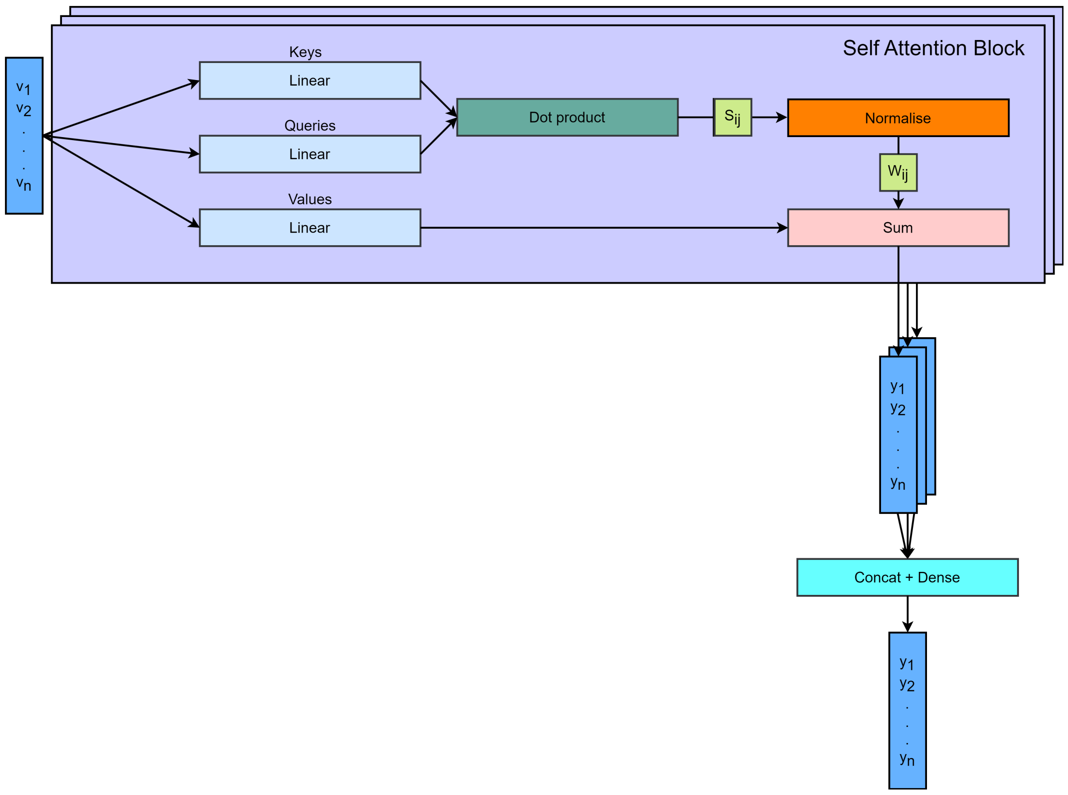


Figure 30: *Multi-Head Attention with Output Treatment*

This Multi-Head Attention is at the heart of the transformer proposed in "Attention is all you need" (Vaswani et al., 2017). Another advantage is not only the improved accuracy, but also the parallelization. This allows GPUs to be used for training instead of CPUs, which significantly reduces the training time and the prediction time of the already trained model.

This now leads us to the diagram as presented in the original paper (Vaswani et al., 2017(Figure 31).

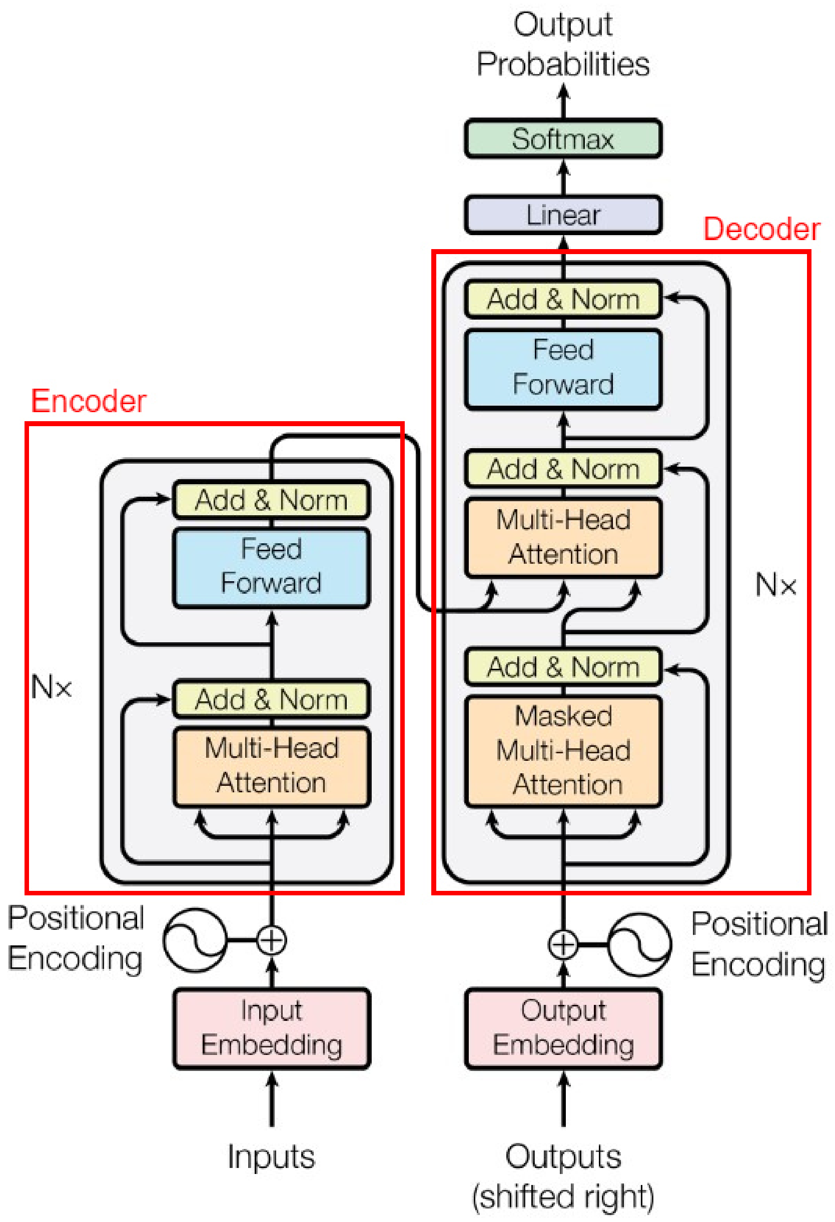


Figure 31: *Transformer from Attention is all you need. Source: Vaswani et al., 2017*

In this figure (Figure 31) you can see that there is an encoder and a decoder part. The reason for this is that this architecture was used for translations. This architecture is very common in translation models. The encoder encodes the information that is present in the text in the source language. This information is then decoded back into a text in the target language in the decoder. However, since we do not have a translation problem in this thesis, but the prediction of a time series, only the encoder part is used and hence only the encoder explained.

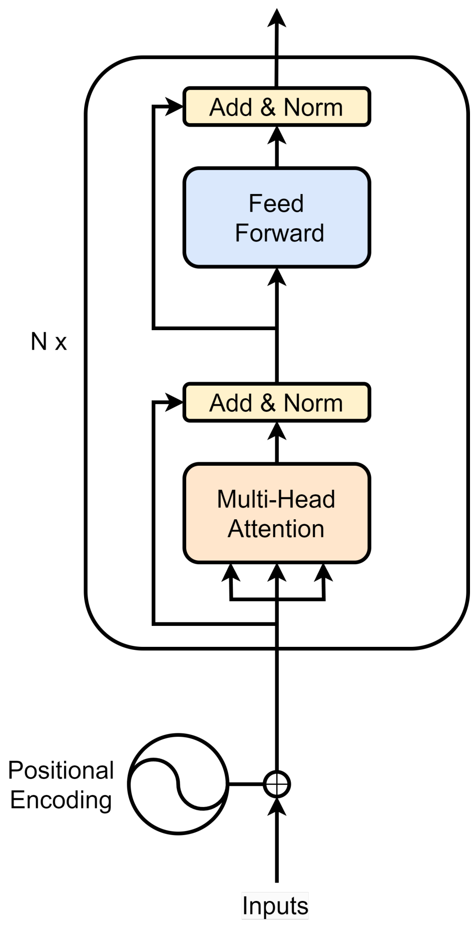


Figure 32: *Encoder only - Transformer*

In Figure 32, the first thing you see is the "Positional Encoding". This is used to communicate the position of the data within itself. For example of a word in a sentence. This is added to the input. Then comes the familiar Multi-Head Attention. Two operations can be seen in Figure 32 with "Add & Norm". These add the original vector to the processed vector once again and normalize it. The reason for this is that if many of these encoders can be connected in series (symbolized by "N x"), a vanishing gradient problem could occur. This operation prevents this. "Feed Forward" refers to a feed forward neural network. This was introduced because it adds a further non-linear transformation and thus increases the complexity of the patterns that can be learned (Geva et al., n.d.).

The Transformer architecture has significantly impacted AI, particularly in NLP. Originally designed for NLP tasks, Transformer models have since been adapted for diverse applications beyond language processing (Singh and Behera, 2022).

3 Methodology

Contents

In this section, we discuss the approach used for predicting the Day-Ahead Price. Initially, we detail the process of data generation and preparation. Subsequently, the selected Machine Learning algorithms employed in this prediction are presented.

3.1 Data

The data forms the foundation of the model. In this section, we elaborate the approach we followed to build upon this foundation.

3.1.1 Sources

The data used to train the model was obtained from two sources:

- European Network of Transmission System Operators for Electricity (ENTSO-E): This is the association of European energy grid operators. They provide data on the European electricity grid on their transparency website (`https://transparency.entsoe.eu`). The data originating from this website will from here on be referred to as "electricity market data".
- Austrian Gas Grid Management AG (AGGM):
  AGGM is responsible for the flow of natural gas in Austria. They publish data on the Austrian gas flow and also the European gas price on their transparency website (`https://platform.aggm.at`). The data originating from this website will from here on be referred to as "gas price data".

The data was generated from these websites using the respective API. These were then saved in .xlsx format for further processing.

3.1.2 Preparation

The price of the Central European Gas Hub (CEGH) should be used as the gas price. However, the data from the CEGH is only been available since 16.1.2021. Prior to that, the gas price of Austria for the balance group "East" could be accessed for delivery and purchase until 1.9.2022. As the price for delivery and purchase of the "East" balance group and the price from the CEGH overlap from 16.1.2021 to 1.9.2022, a correlation analysis could be carried out to see whether the price from the CEGH for the period before 16.1.2021 can be calculated from the prices from the Eastern balance group.

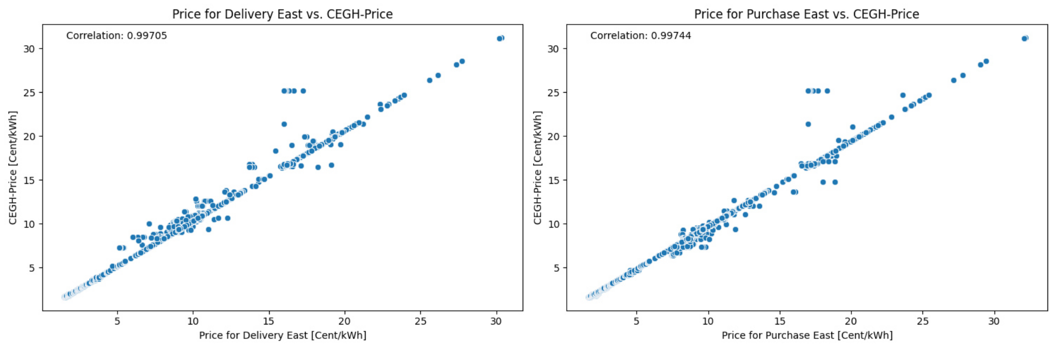


Figure 33: *Correlation between CEGH and the Prices for Delivery and Purchase "East"*

It can be seen in Figure 33 that there is a strong correlation between the prices. Therefore, a linear regression was performed to calculate the prices before 16.1.2022 of the CEGH from the prices of the balance group "East".

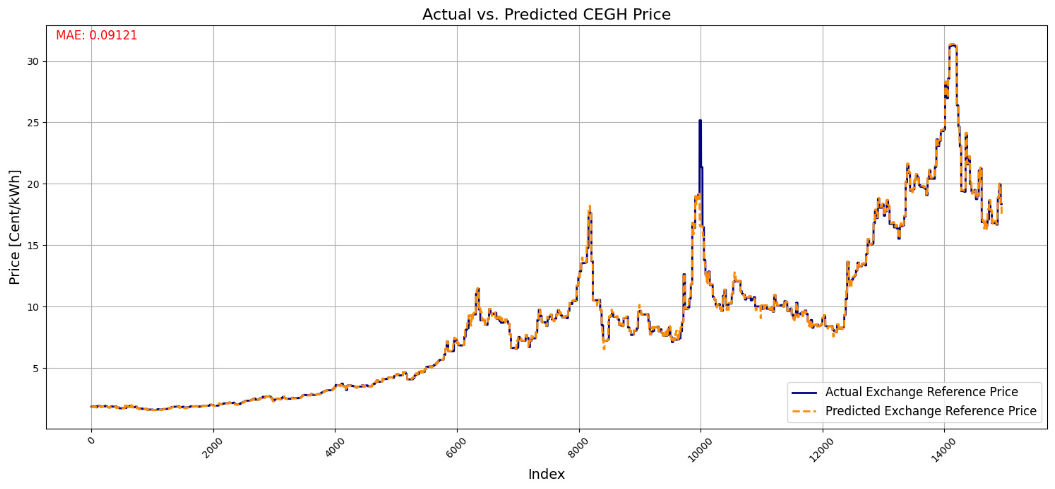


Figure 34: *Actual vs. Predicted CEGH Price*

Since the outlier which can be seen in Figure 34 affects less than 1% of the data, the regression can be considered acceptable and therefore the price from the CEGH before 16.1.2022 was calculated from the prices for the "East" balance group.

3.1.3 Description of the generated Data

The following data was generated by the ENTSO-E API for the period starting from 30.11.2018 00:00 until the 30.08.2023. The data is broken down hourly. The data was generated for the following European countries:

- Austria
- Belgium
- Germany-Luxembourg (Germany and Luxembourg form one bidding zone, this means they share the same electricity price and within this thesis can be thought of one country)
- Estonia
- Finland
- France

- Hungary
- Lithuania
- Latvia
- Netherlands
- Portugal
- Poland
- Romania
- Slovenia
- Greece

The reason why these countries were chosen is that these countries have the most accurately reported data. In other European countries there were problems with the reported data. For example, whole generation types were not reported or were filled with NaN-values. For example Albania only reports its data beginning from the 25.05.2022. The data that was generated were the following:

- **Actual Total Load:** The Actual Total Load is the current consumption in MW.
- **Actual Generation per Type:** The Actual Generation per Type denotes the current generation broken down by generation type (wind, solar, natural gas, etc.) in MW.
- **Day Ahead Prices:** The Day Ahead Price refers to the cost of purchasing one Megawatt-hour (MWh) of electricity today for consumption the following day, expressed in euros per Megawatt-hour (€/MWh). This price only refers to the cost of the energy and does not include the price for grid fees or taxes.

The generation types were the following:

- Biomass
- Fossil Brown coal/Lignite
- Fossil Coal-derived gas
- Fossil Gas
- Fossil Hard coal
- Fossil Oil
- Fossil Oil shale
- Fossil Peat
- Geothermal
- Hydro Pumped Storage
- Hydro Run-of-river and poundage
- Hydro Water Reservoir
- Nuclear
- Solar
- Wind Onshore
- Wind Offshore
- Waste
- Other renewable
- Other

Because the gas price comes from the CEGH, it can be considered global in the domain (Europe). In other words, the same gas price, that of the CEGH, was used for each country.

3.1.4 Data Processing

A cleanup had to be performed on the electricity market data. The procedure was as follows:

- Missing timestamps were filled with the mean value of the neighboring rows.
- Duplicate timestamps:
  If the values for each column are the same, one of the duplicate timestamps was removed.
  If the values were different, the mean value was calculated.
- NaN values ("Not a Number" values) were replaced by 0.

There were not many NaN values found. Most of the adjustments had to be made for duplicate timestamps. In total 1.62% of the data had to be adjusted. The average values for each day were also calculated. The data, which was then used to train the algorithms, was resolved daily. So for each day of the year there was one data point. One reason for this was that fewer predictions had to be made by the models, thus reducing the prediction time. In addition, the fluctuations in solar production, for example, led to additional noise in the predictions, which also reduced the accuracy of the models.

For better understanding some generation types were combined. "Fossil Brown coal/Lignite", "Fossil Hard coal" and "Fossil Peat" were combined to "Fossil Coal". "Fossil Gas" and "Fossil Coal-derived gas" were combined to "Fossil gas".

The generation types were converted into relative proportions. In other words, how large is the share of the respective generation type in the total generation (Equation 8):

$$\text{Relative Generation of Generation Type [.]} = \frac{\text{Generation per Type [MW]}}{\text{Total Generation [MW]}} \tag{8}$$

The own generation was calculated as follows (Equation 9):

$$\text{Own Generation [.]} = \frac{\text{Actual Load [MW]}}{\text{Total Generation [MW]}} \tag{9}$$

As machine learning algorithms are better able to deal with similar value ranges, min-max scaling has been applied to Actual Total Load, Total Generation, Day Ahead Price and Gas Price. This scales the values between 0 and 1 (Equation 10):

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{10}$$

In Equation 10, X_{min} denotes the minimum value in the data for the corresponding column and X_{max} the maximum value.

3.1.5 Description of Training Data

The finished training data consists of 17 features (Table 1). Features denote the variables that are used to predict the label. In our case the label is the Day Ahead Price:

Name	Description	Source
Gas Price	Gas Price Index from the CEGH	AGGM
Own Generation	Generated vs. Consumed Electricity	ENTSO-E
Biomass	Relative Biomass Production	ENTSO-E
Fossil Gas	Relative Fossil Gas Production	ENTSO-E
Fossil Oil	Relative Fossil Oil Production	ENTSO-E
Fossil Coal	Relative Fossil Coal Production	ENTSO-E
Nuclear	Relative Nuclear Production	ENTSO-E
Waste	Relative Waste Production	ENTSO-E
Geothermal	Relative Geothermal Production	ENTSO-E
Hydro Pumped Storage	Relative Hydro Pumped Storage Production	ENTSO-E
Hydro Run-of-river and poundage	Relative Hydro Run-of-river Production	ENTSO-E
Hydro Water Reservoir	Relative Hydro Water Reservoir Production	ENTSO-E
Wind Onshore	Relative Wind Onshore Production	ENTSO-E
Wind Offshore	Relative Wind Offshore Production	ENTSO-E
Solar	Relative Solar Production	ENTSO-E
Other renewable	Relative Other renewable Production	ENTSO-E
Other	Relative Other Production	ENTSO-E

Table 1: *Table of Features*

The label (the variable that will be predicted) is the Day Ahead Price. The data has been split, with the data for Austria reserved as the test set. From the training data, 10% is allocated for validation purposes.

3.1.6 Description of Test Data

The data for Austria (AT) was used as test data. These were not part of the training data. For the sensitivity analysis, the mean value of the various features of all countries used was calculated at each point in time. This will be called "All countries average" from now on. As an error metric the MAE was used instead of the MAPE because the MAE is not as sensitive to outliers as the MAPE. The calculation for the MAE is shown in 3.2.1.

3.1.7 Insights

Day Ahead Prices:

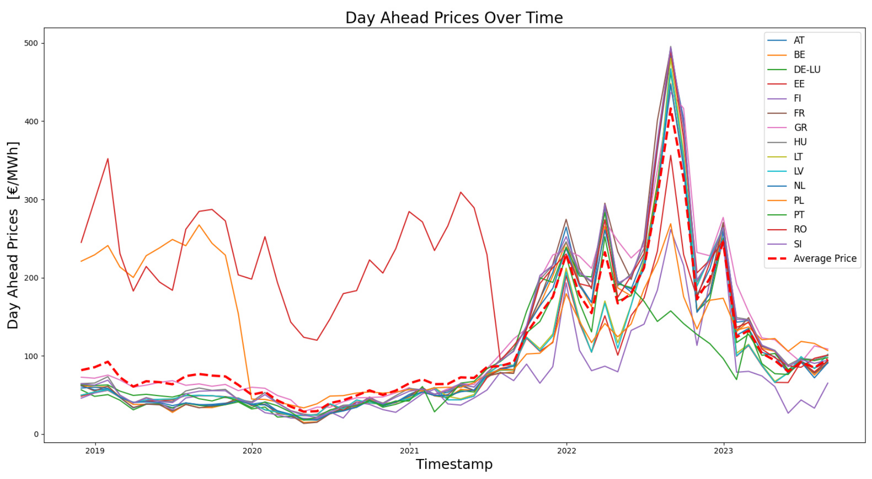


Figure 35: *Day Ahead Prices over Time for used Countries*

In Figure 35 can be seen that the Day Ahead Prices for RO (Romania) and BE (Belgium) behave very differently to the others. These countries were therefore removed from the training data as they led to high errors in the models. As shown later (3.3), a satisfactory accuracy could be achieved even without these countries.

Correlation: In Figure 36 a correlation of each feature with the Day Ahead Price was formed:

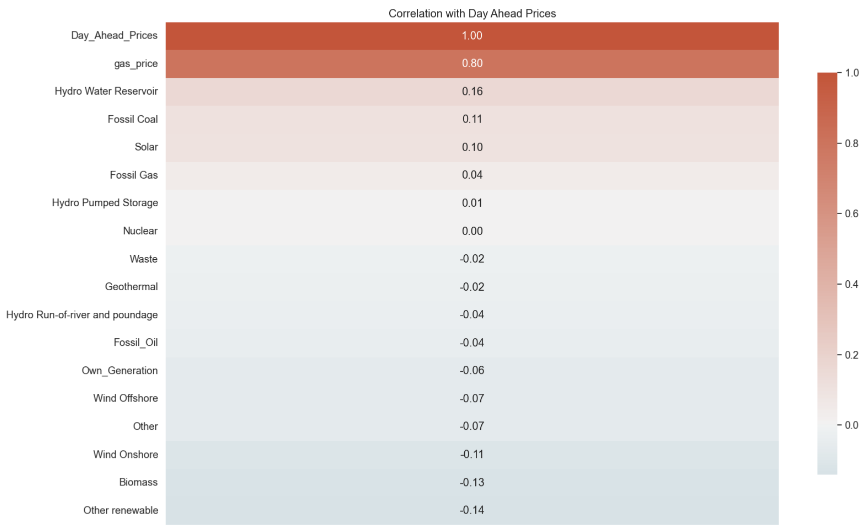


Figure 36: *Correlation with Day Ahead Prices*

As expected, the gas price has the strongest correlation. This is due to the merit order system (2.1.1). More insight on this will be provided in 4.2

It can also be seen that the Day Ahead Price and solar production show a positive correlation. This means that when solar production increases, the price of electricity also increases. More insight on this will be provided in 4.3.

3.2 Possible Approaches to predict Day Ahead Prices

In this section different approaches for predicting the Day Ahead Prices are shown. First, simple models are evaluated and compared to more complex ones.

3.2.1 Error metrics

As error metrics two different metrics are used in this thesis.
MAE: The Mean Absolute Error (MAE) is a measure of accuracy in forecasting models within statistics and machine learning. It calculates the average absolute difference between the actual and predicted values. In the MAE formula,

$$\text{MAE} = \frac{1}{T}\sum_{t=1}^{T}|y_t - \hat{y}_t| \tag{11}$$

T represents the total number of observations in the time series, y_t is the actual value at time t, and $\hat{y}_t$ is the predicted value at the same time. The MAE is particularly useful in time series analysis, providing a straightforward and clear assessment of forecasting accuracy.
MAPE: The Mean Absolute Percentage Error (MAPE) is a statistical measure used to assess the accuracy of a forecasting model. It is defined as the average of the absolute percentage differences between the predicted values and the actual values. This metric is particularly useful in scenarios where the comparison of the relative error between data points is more meaningful. The MAPE formula is expressed as:

$$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \times 100\% \tag{12}$$

where n is the number of observations, y_t is the actual value at time t, and $\hat{y}_t$ is the predicted value at time t.

3.2.2 Linear Regression

A linear regression is a statistical model for relating a dependent variable y to several independent variables x_n (Equation 13).

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n \tag{13}$$

- y: The dependent variable to be predicted.
- β_0: Intercept of the regression line with the y-axis, also known as the bias or intercept term.
- $\beta_1, \beta_2, \ldots, \beta_n$: The coefficients of the independent variables.
- $x_1, x_2, \ldots, x_n$: The independent variables.

The terms β_n can be adjusted by using data.
This linear regression was performed for the training data:

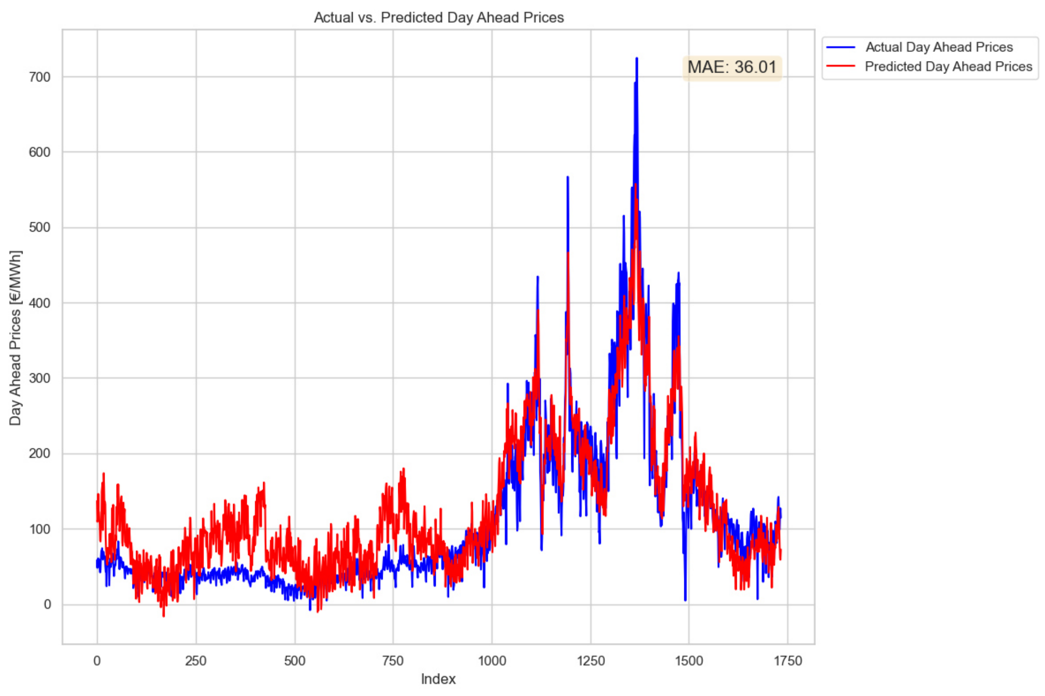


Figure 37: *Actual vs. Predicted Day Ahead Prices for Linear Regression*

As can be seen in Figure 37, linear regression cannot provide satisfactory results. Because of this the parameters have not been saved.

3.2.3 Polynomial Regression

A polynomial regression model is an extended linear regression model. It extends the linear regression by terms of power m (Equation 14).

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \ldots + \beta_m x_1^m + \beta_{m+1} x_2 + \beta_{m+2} x_2^2 + \ldots + \beta_{km} x_n^m \tag{14}$$

- y: The dependent variable to be predicted.
- β_0: Intercept of the regression line with the y-axis, also known as the bias or intercept term.
- $\beta_1, \beta_2, \ldots, \beta_{km}$: The coefficients of the independent variables and their polynomial terms.
- $x_1, x_1^2, \ldots, x_1^m$: The original independent variable x_1 and its polynomial terms up to degree m.
- m: The degree of the polynomial. The highest power of the independent variable.
- n: The number of independent variables.

In order to train a polynomial model, a linear model simply has to be extended by power terms during implementation. This can be realized e.g. with libraries like "sklearn"[2]. To check whether the code works correctly, first a polynomial model of first order was trained, which corresponds to a linear regression. The results were exactly the same as in Figure 37.

[2]https://scikit-learn.org/stable

When higher order polynomial models were trained, the results were equally unusable as the linear regression (Figure 38 Figure 39 Figure 40 Figure 41). Because of this, this parameters have also not been saved.

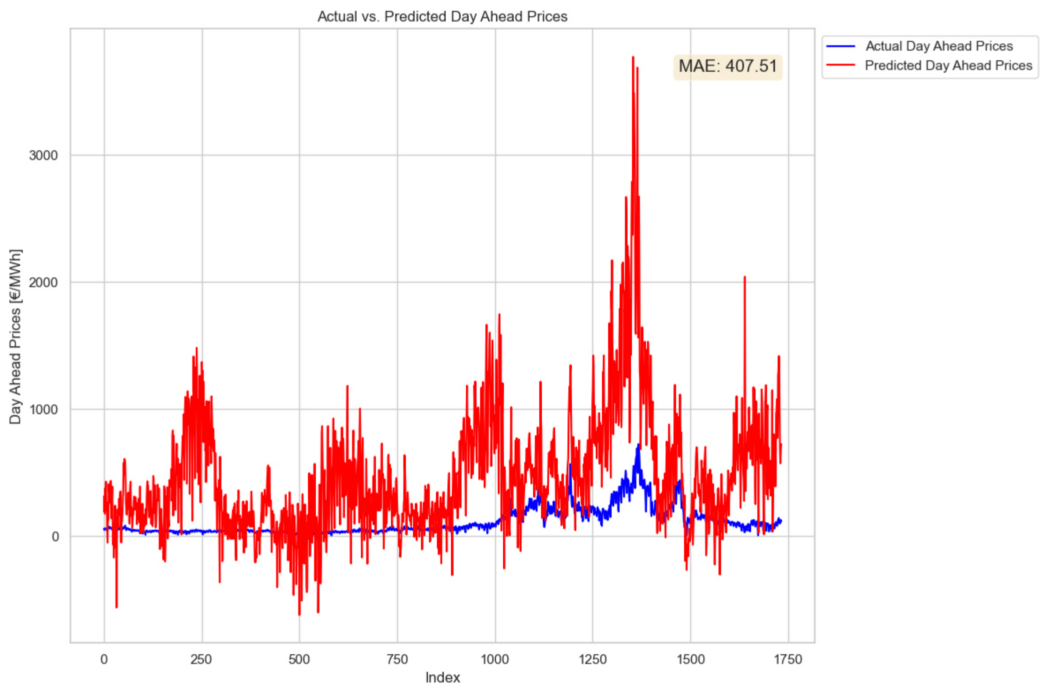


Figure 38: *Actual vs. Predicted Day Ahead Prices for 2nd order polynomial regression*

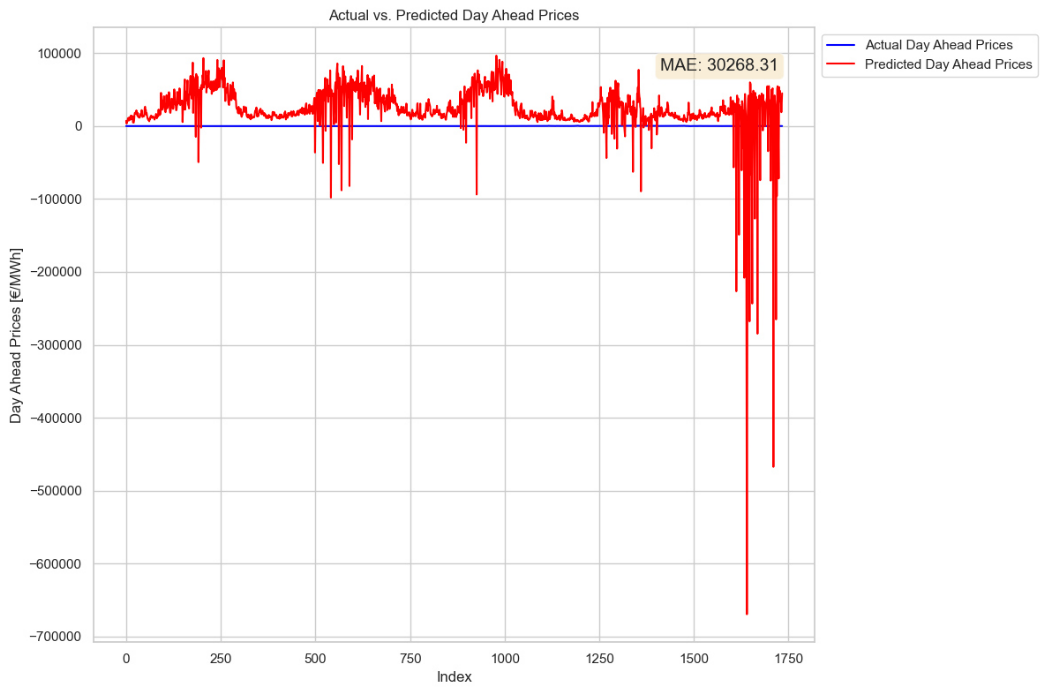


Figure 39: *Actual vs. Predicted Day Ahead Prices for 3rd order polynomial regression*

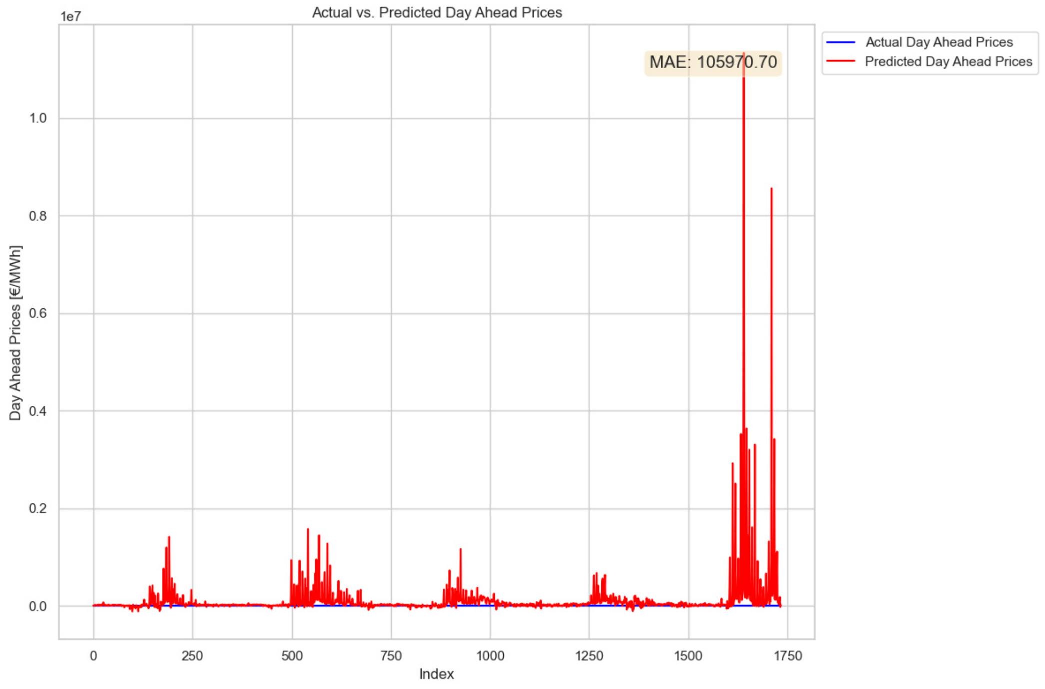


Figure 40: *Actual vs. Predicted Day Ahead Prices for 4th order polynomial regression*

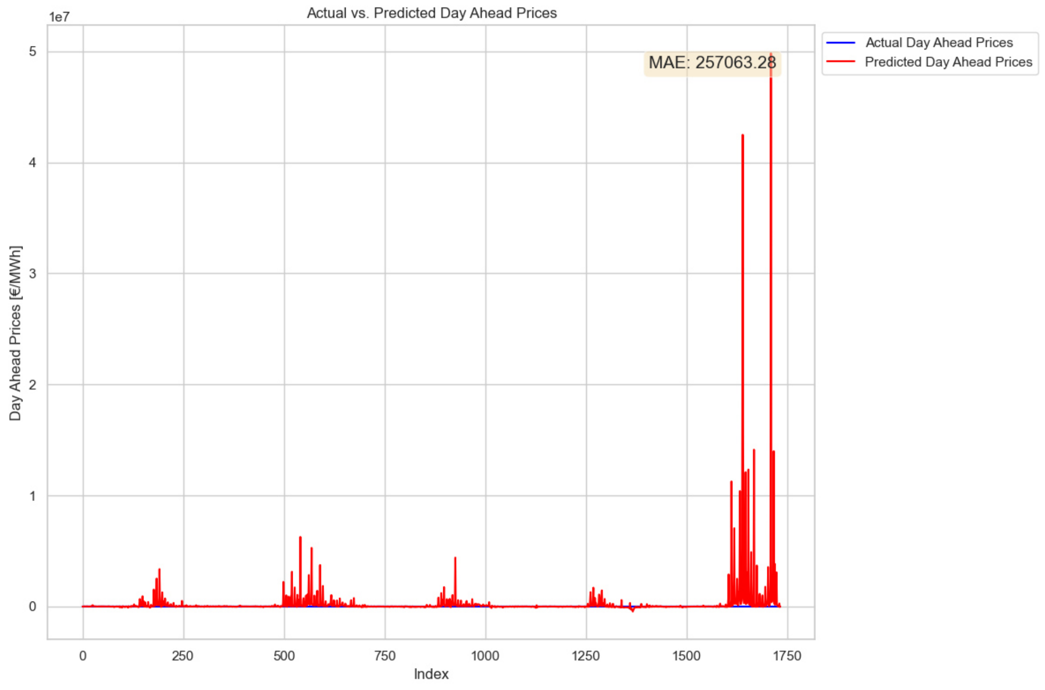


Figure 41: *Actual vs. Predicted Day Ahead Prices for 5th order polynomial regression*

3.2.4 Long-short Term Memory Networks

As shown in 3.2.2 and 3.2.3, more complex models must be used to achieve sufficient prediction accuracy. Initially, an LSTM model was employed (2.2.2), chosen for its ease of implementation compared to transformer models. LSTM models have demonstrated strong predictive accuracy (Figure 42) in comparison to the linear/polynomial Regression due to their ability to effectively capture and remember sequential patterns in data. Their gated architecture allows them to retain important information over longer sequences, making them well-suited for tasks involving time series or sequential data, such as predicting the Day Ahead Price.

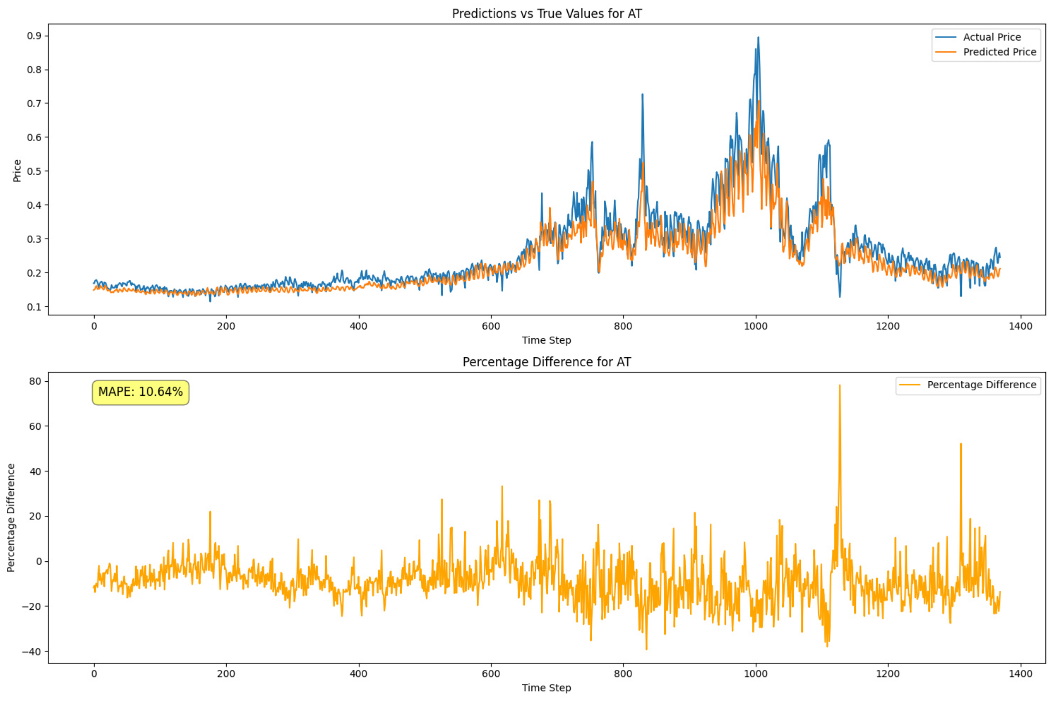


Figure 42: *LSTM: Predicted vs. True Values for AT*

Note: The Day Ahead prices only go from 0 to 1. This is due to the fact that no rescaling to the actual prices was carried out for test purposes.
Note: The calculation of the MAPE is shown in 3.2.1.

3.2.5 Transformer

Transformers have also been evaluated, as discussed in Section 2.2.3. They stand out for their capability to process data in parallel, unlike LSTMs which process data serially. This feature contributes to improved training and prediction times. Moreover, their attention mechanism allows them to capture long-term patterns more effectively. The enhanced prediction accuracy of transformers is demonstrated in Figure 43.

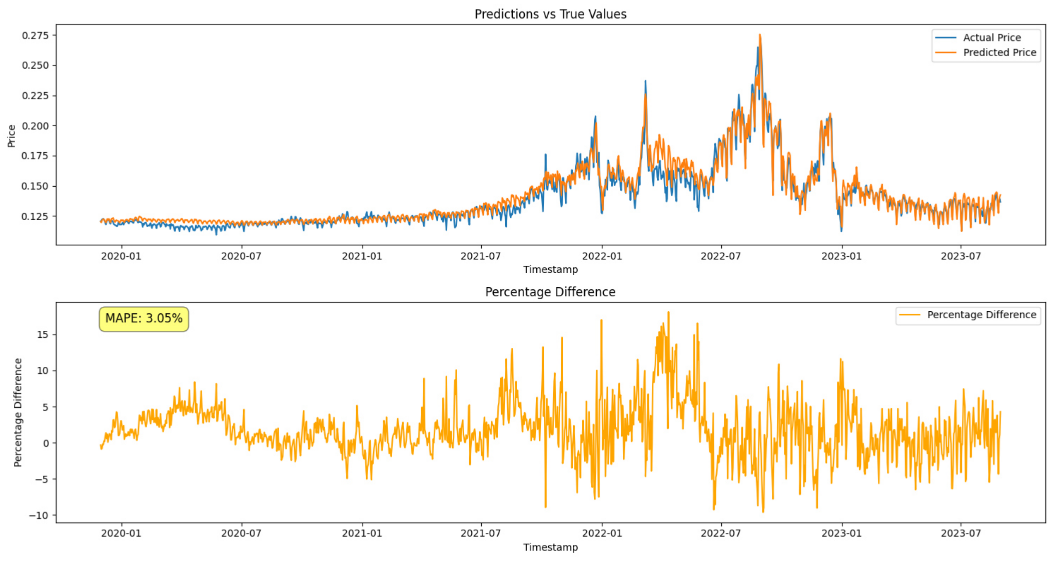


Figure 43: *Transformer: Predicted vs. True Values for AT*

Note: Hyperparameter tuning was carried out for the transformer model, but not for the LSTM model.
Note: The Day Ahead prices only go from 0 to 1. This is due to the fact that no rescaling to the actual prices was carried out for test purposes.
Note: The calculation of the MAPE is shown in 3.2.1.

3.2.6 Conclusion

Transformer models have shown significantly better performance on test data than LSTM models. However, it is important to emphasize that no careful hyperparameter tuning was performed for the LSTM, but the literature was searched for hyperparameters that were likely to show good performance. Thus, this should not be taken as an indication that transformers generally outperform LSTM models in time series prediction.

Nevertheless, transformers were chosen as the network architecture for the problem at hand. The main reason for this was the relatively long sequence length of 365. This sequence length was chosen so that a whole year is always mapped and the electricity price shows a strong annual seasonality (Ioannides et al., 2020). Although, as in 2.2.2 shown, LSTMs can handle the vanishing gradient problem better, problems with the gradient can still occur with long sequences (Zhou et al., 2020). This is why the Transformer architecture was chosen.

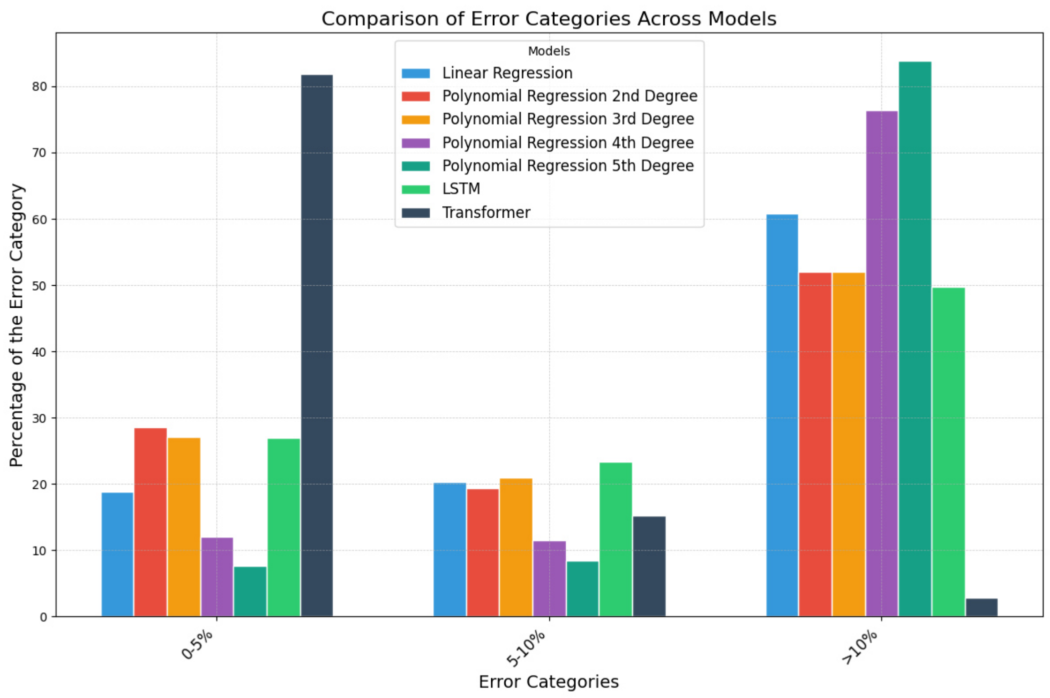


Figure 44: *Comparison of Error Categories Across Models*

A comparison of all models can be seen in Figure 44. It should be noted that although the LSTM seems to be underperforming in comparison to the polynomial models. The >10% errors are much closer to 10% than those of the polynomial models.

3.3 Chosen Approach

An ensemble of transformer models was chosen as the approach. This means that not just one model makes a prediction, but several. This allows a standard deviation and a mean value of the predictions to be formed.
The standard deviation of a sample is defined as:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

where:

s represents the standard deviation of the sample.
n is the number of observations in the sample.
x_i denotes the value of the i-th observation in the sample.
$\bar{x}$ is the mean (average) value of the observations in the sample.
The summation $\sum_{i=1}^{n}$ indicates that the following calculation (in this case, $(x_i - \bar{x})^2$) is performed for each i from 1 to n, and then summed up.

This formula calculates the dispersion of the values in the sample around their mean. This measure reflects the average distance of each observation in the sample from the mean. The role of the standard deviation as a measure of the distribution of data values is particularly meaningful in the context of a normal distribution. In such a distribution, approximately 68% of the values lie within one standard deviation of the mean, 95% within two standard deviations and 99.7% within three standard deviations. However, these specific percentages *only* apply to normally distributed data.
In the field of machine learning, the standard deviation is often used to assess the consistency and reliability of predictions of an ensemble of models. A low standard deviation in the predictions indicates that the models come to similar results, which can be interpreted as an indicator of a certain reliability of the prediction. However, it is important to emphasise that this does not necessarily guarantee the accuracy of the predictions. The possibility that all models in the ensemble are collectively wrong remains.
Ultimately, forecasts, such as the prediction of the Day Ahead Price, are always estimates with an inherent uncertainty. This uncertainty results from various factors, such as market volatility, unforeseen events and the inherent limitation of model accuracy. These aspects emphasise the importance of prudent handling of model forecasts and the consideration of possible uncertainties.

3.3.1 Hyperparameter Tuning

The hyperband tuner is integrated in the python library "Keras"[3]. It is used for tuning the hyperparameters. Hyperparameters would be, for example, how many "heads" are used in the multi-head attention or how many neurons are used in an LSTM layer. So these are the parameters of the model. There is no generally valid approach for tuning the hyperparameters. In essence, this is done by trial and error.
The Hyperband Tuner handles this as follows. It is based on the idea of the "multi-armed bandit". This idea is described by the following scenario: There are several one-armed bandits in a casino. Each of these slot machines has a different payout rate.

[3]https://keras.io/

How can you decide which machine you should put the most money (resources) into? This idea basically describes the two opposites of exploration and exploitation. In other words, how many resources should be used to explore new possibilities.
The Hyperband Tuner deals with this problem as follows: First, many possibilities are trained with little input. In our case, the input reflects the number of epochs used to train a set of hyperparameters. An epoch means that the model is trained once with the training data. In other words, how often does the model "see" the data. After the first round, there is "successive halfing", i.e., the most promising set of models moves on to the next round. The set of models does not necessarily have to be halved, this is a parameter of the tuner. More computing power is now applied to these models, this means these models are trained with more epochs. In our case about 3 times as much. This is also a parameter. This is repeated until the maximum number of epochs is reached. Hyperband also has so-called "brackets". In each bracket, the hyperparameters are set very differently at the beginning. Different initializations are advantageous when creating a model ensemble, as they lead to varied hyperparameters. This variety in hyperparameters contributes to distinct prediction strategies among the models, enhancing the overall robustness and accuracy of the ensemble's predictions.

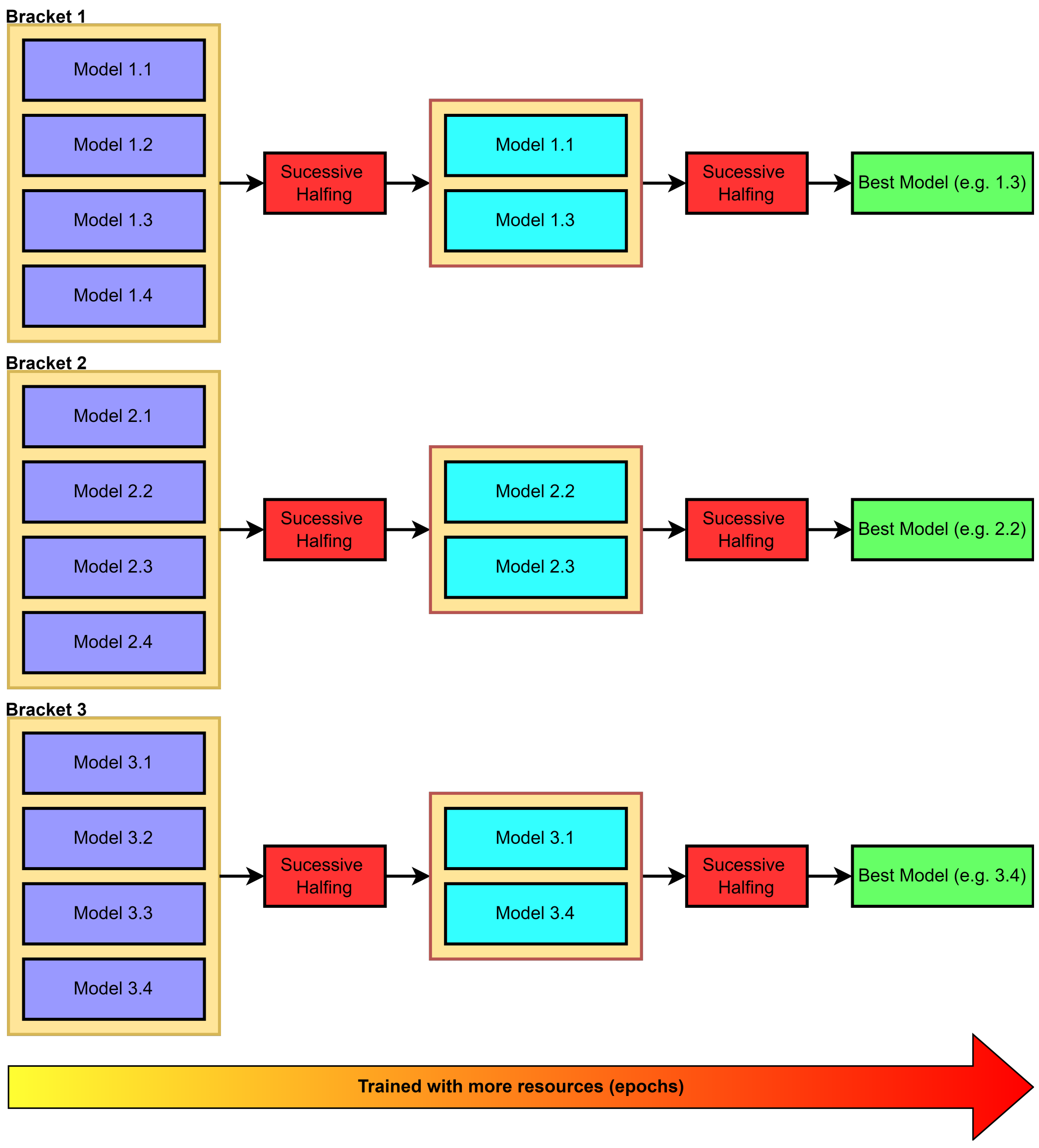


Figure 45: *Flowchart of Hyperband Tuner*

The methodology is further clarified in Figure 45. Here, each bracket starts with 4 models. As mentioned, each bracket is started with very different hyperparameters and within the bracket, each model has also different hyperparamters but in a smaller range than the brackets. Now these models are trained with few resources, in our case 16 epochs. This is followed by successive halving. In this case, the models are halved. So the best half goes to the next round (measured by the accuracy on the validation data). These are then trained with more epochs and so on. Until you end up with the best model of the bracket. These models could then be sorted by accuracy again.

In reality, there are significantly more models at the beginning of the brackets and more brackets are used. But the basic idea remains the same.

3.3.2 Chosen Models for Ensemble

The 8 best models were saved. For the exact implementation, please refer to A.1. The 8 best models were then tested on the test data. As a significant drop in performance was observed after the best 5 models, 5 models were selected to form the ensemble. To check whether the 5 best models on the test data also show the best performance as an ensemble, all combinations of the 8 models were tested and, as expected, the 5 models with the best individual performance on the test data also gave the best overall performance.

In Table 2 the hyperparameters of the best 5 models are shown.

Model	Model_0	Model_1	Model_2	Model_3	Model_4
head size	6	2	2	8	10
num heads	10	10	7	10	10
ff dim	128	128	32	64	64
num transformer blocks	2	1	2	5	5
mlp units	320	256	512	448	384
mlp dropout	0,4	0,1	0,3	0,3	0,1
dropout	0,3	0,2	0,1	0,2	0,3
learning rate	0,000151	0,000635	0,000614	0,000370	0,00062

Table 2: *Table of Hyperparameters of the Ensemble*

The sequence length of all models was 365.

Explanation of hyperparameters:

- "head size": This is the dimension of the vector d_{head}.
- "num heads": This is the number of Self Attention mechanisms in the Multi Head Attention.
- "ff dim": Refers to the dimensionality of the Feed Forward layer.
- "num transformer blocks": Specifies how many transformer blocks are connected in series.
- "mlp units": Specifies the neurons in the dense layer of the multi-head attention.
- "mlp dropout": A measure to prevent overfitting. This value specifies how many neurons are randomly deactivated in the dense layer.
- "dropout": A measure to prevent overfitting. Specifies how many neurons are randomly deactivated in the attention and feed forward networks.
- "learning rate": Specifies how strongly the model adjusts its weights during training.

4 Results

Contents

4.1 Performance

The ensemble made the following predictions for Austria (Figure 46):

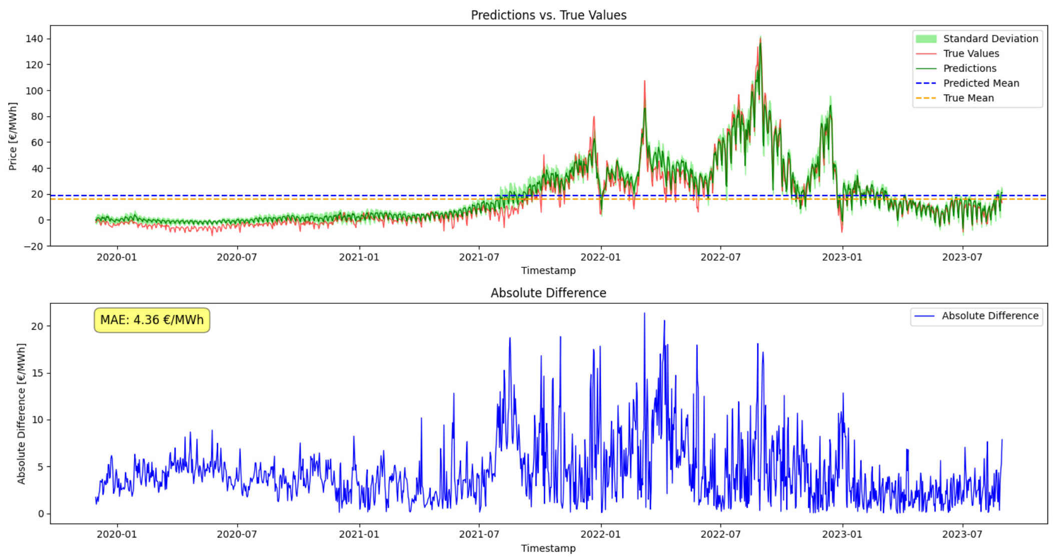


Figure 46: *Transformer Ensemble: Predicted vs. True Values for AT*

Although the accuracy of the ensemble may be slightly lower than that of a single model (3.2.5), its predictions are considered to be more reliable. This increased robustness comes from averaging the results of multiple models. In addition, calculating the standard deviation and mean of these predictions provides a quantitative measure of the ensemble's reliability, allowing a clearer assessment of prediction confidence.

4.2 Correlation with gas price

In order to investigate the high correlation with the gas price in more detail, an ensemble of transformer models was also trained, which were only provided with the gas price as a feature. A hyperparameter tuning was also carried out and the best 3 models were selected. The results are shown in Figure 47.

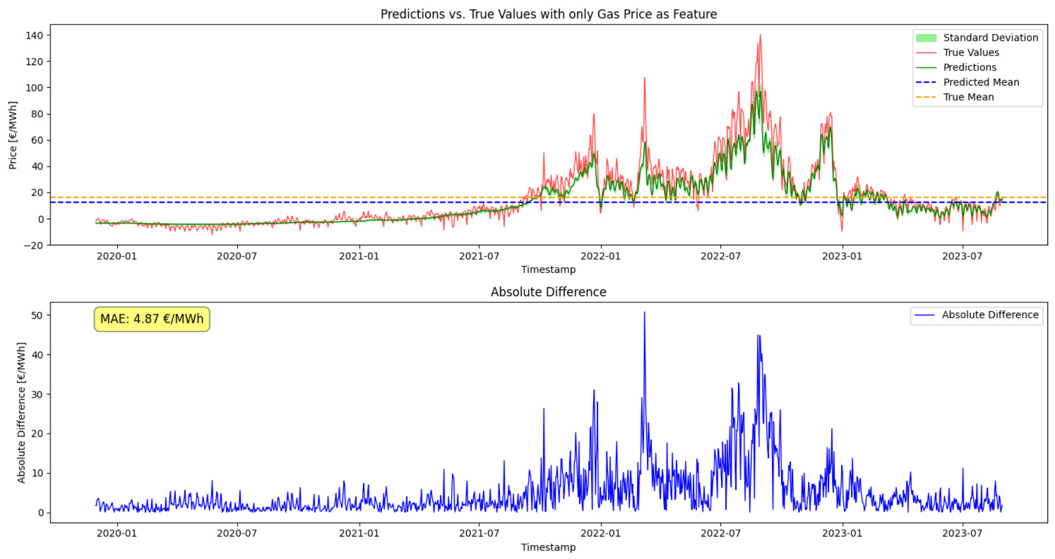


Figure 47: *Transformer Ensemble: Predicted vs. True Values for AT with only Gas Price as Feature*

It can be seen that the MAE increases from 4.36 €/MWh (Figure 46 to 4.87 €/MWh. In addition, the model adapts better when the gas price rises. The additional features therefore increase the accuracy of the predictions for test data.

4.3 Sensitivity Analysis

In the sensitivity analysis, one feature was changed at a time. As a MinMax scaling (where 0 is the smallest value in the data and 1 is the largest), all features from could be tested 0 to 100% . The reason for this was to find out how the individual features affect the Day Ahead Price. The names of the lines are the same in all figures:

- **Average Predicted Price**: Denotes the average price of the ensemble's predictions per timestamp.
- Standard Deviation: Describes the deviation of the ensemble's predictions.
- **Original Average Price**: Refers to the average day-ahead price of the underlying "All countries average" data.
- **Predicted Average Price**: Refers to the average price across all timestamps of all models.
- **Original Mean of "feature"**: Denotes the average value of the features in "All countries average".

Gas price:

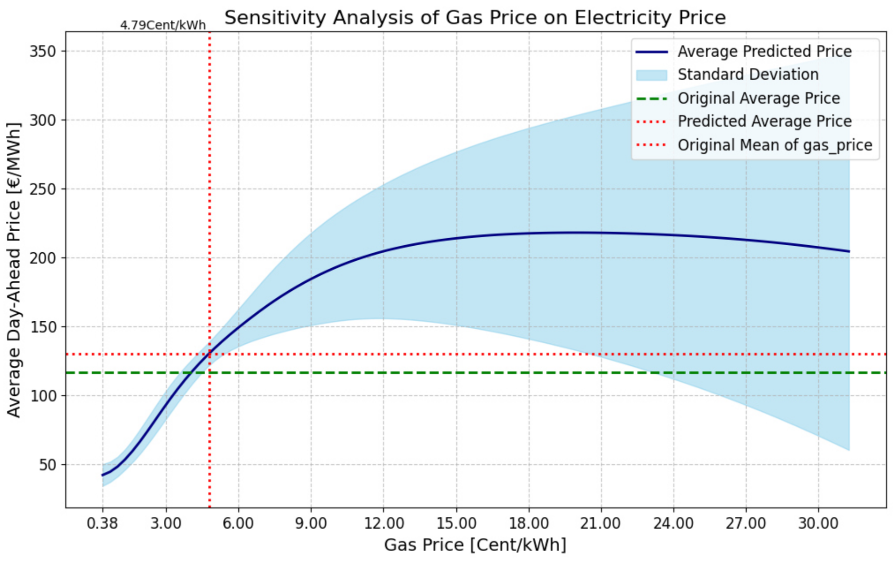


Figure 48: *Sensitivity analysis for Gas Price*

As in all analyses, it can be seen that the standard deviation increases sharply as the value moves away from the original mean. As expected, the Day Ahead Price rises when the gas price rises (Figure 48).

Own generation:

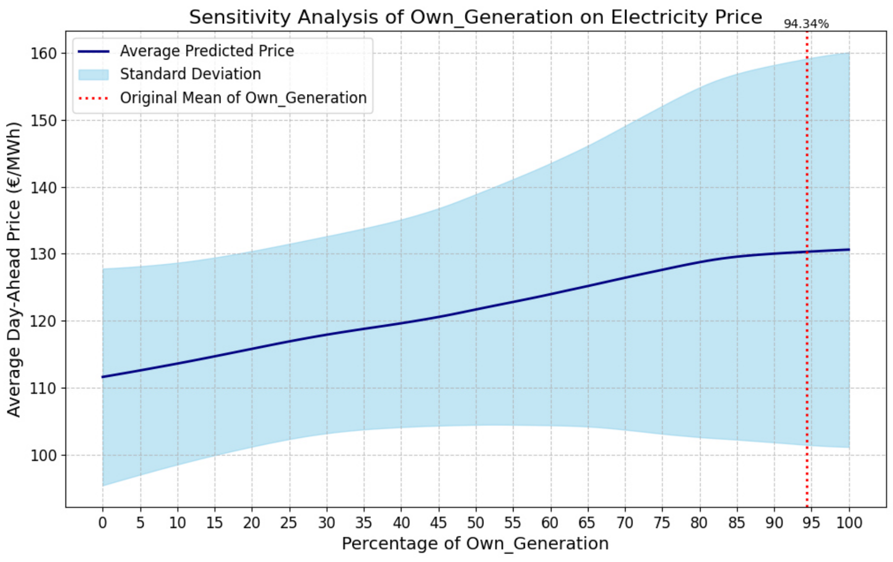


Figure 49: *Sensitivity analysis for Own Generation*

In (Figure 49) you can see that own generation does not have a strong influence on the Day Ahead Price also the standard deviation is high.

Generation Types:

Selected generation types are shown here, the others can be found in the appendix.

The procedure was the same for all generation types: The respective generation type was changed from 0 to 100%. The ratio of the other generation types remained the same.

Solar:

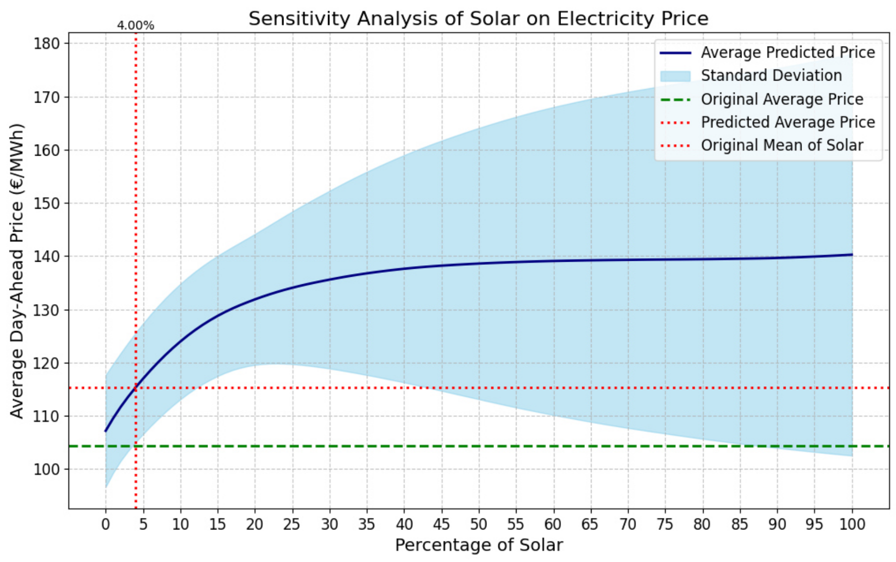


Figure 50: *Sensitivity analysis for Solar*

As already assumed in Figure 36 due to the positive correlation, solar production also shows the same behavior in the sensitivity analysis (Figure 50). This means that the Day Ahead price rises when solar production increases.

Wind Onshore:

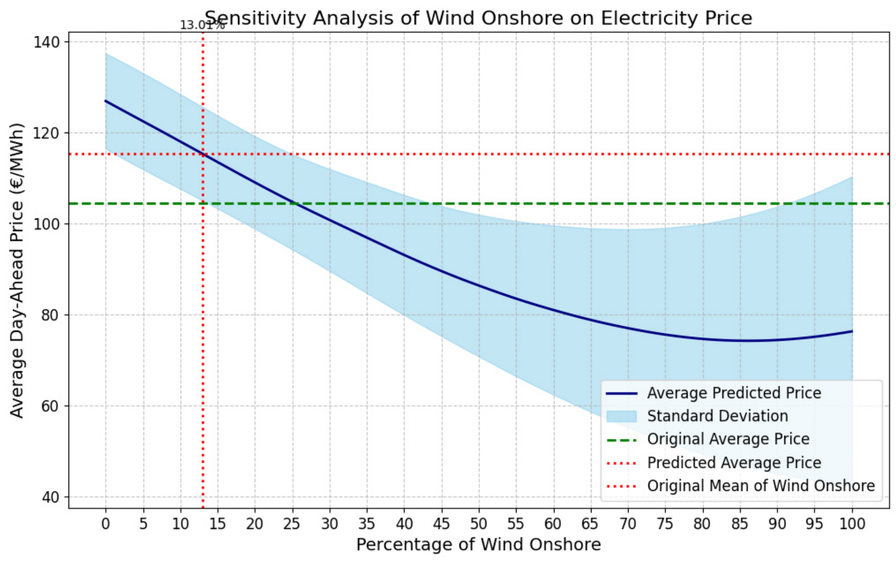


Figure 51: *Sensitivity analysis for Wind Onshore*

The production of wind onshore plants shows the behavior that the merit order system dictates and would also be expected from solar production, namely that the

Day Ahead Price decreases when production increases. Because the wind offshore production in the data was very small (0.88%), no well-founded statement can be made about it.

Biomass:

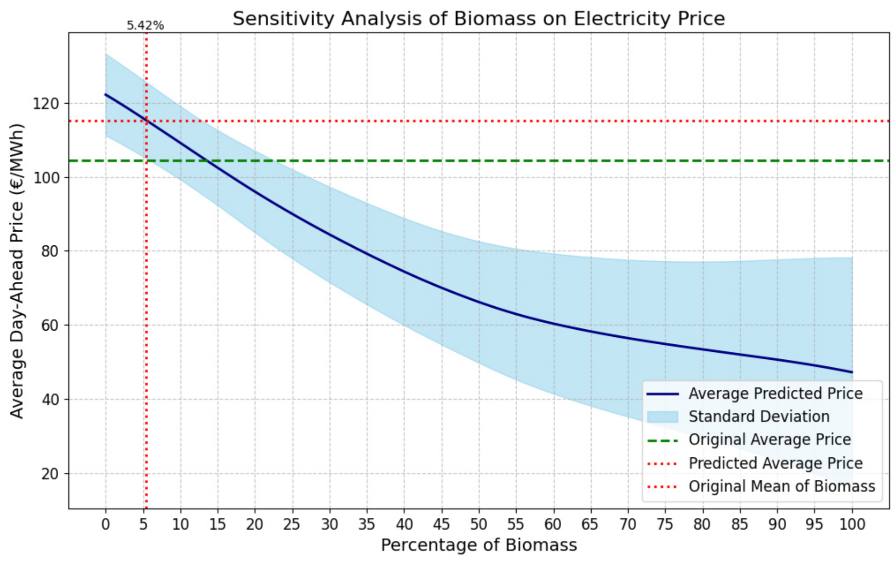


Figure 52: *Sensitivity analysis for Biomass*

Biomass also has a strong effect on the day-ahead price. Here, the price falls when production increases (Figure 52.

Fossil Gas:

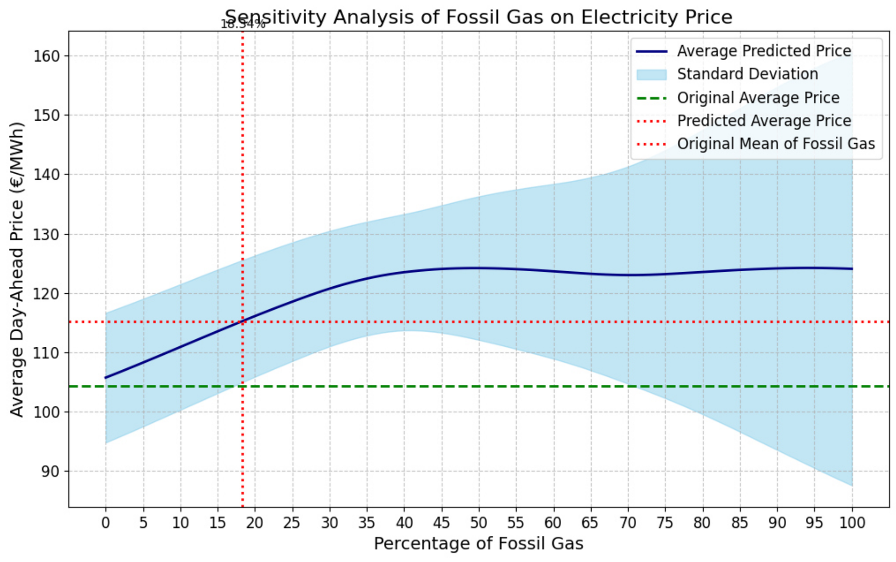


Figure 53: *Sensitivity analysis for Fossil Gas*

Fossil gas also shows the expected effect on Day Ahead Prices (Figure 53. Because the production of energy from gas is expensive, the price of electricity rises when production increases.

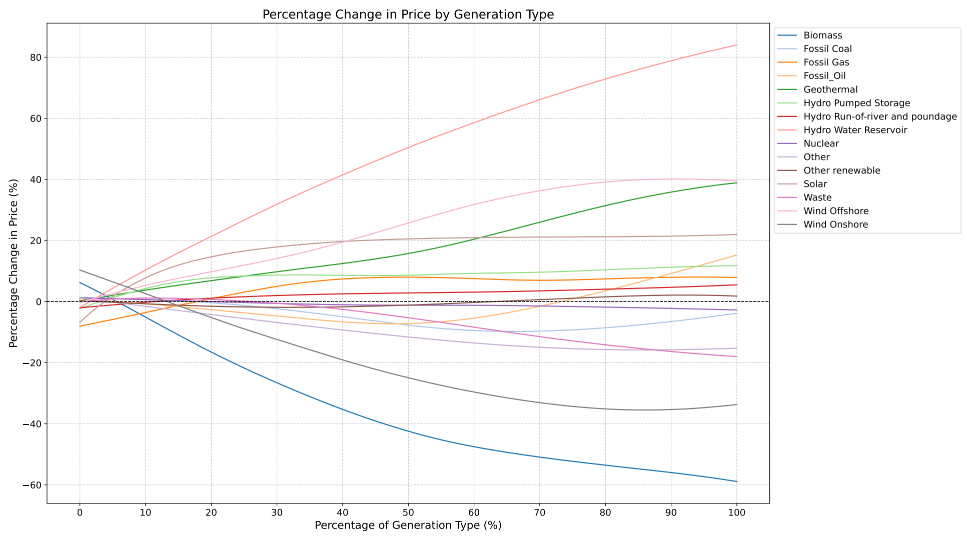
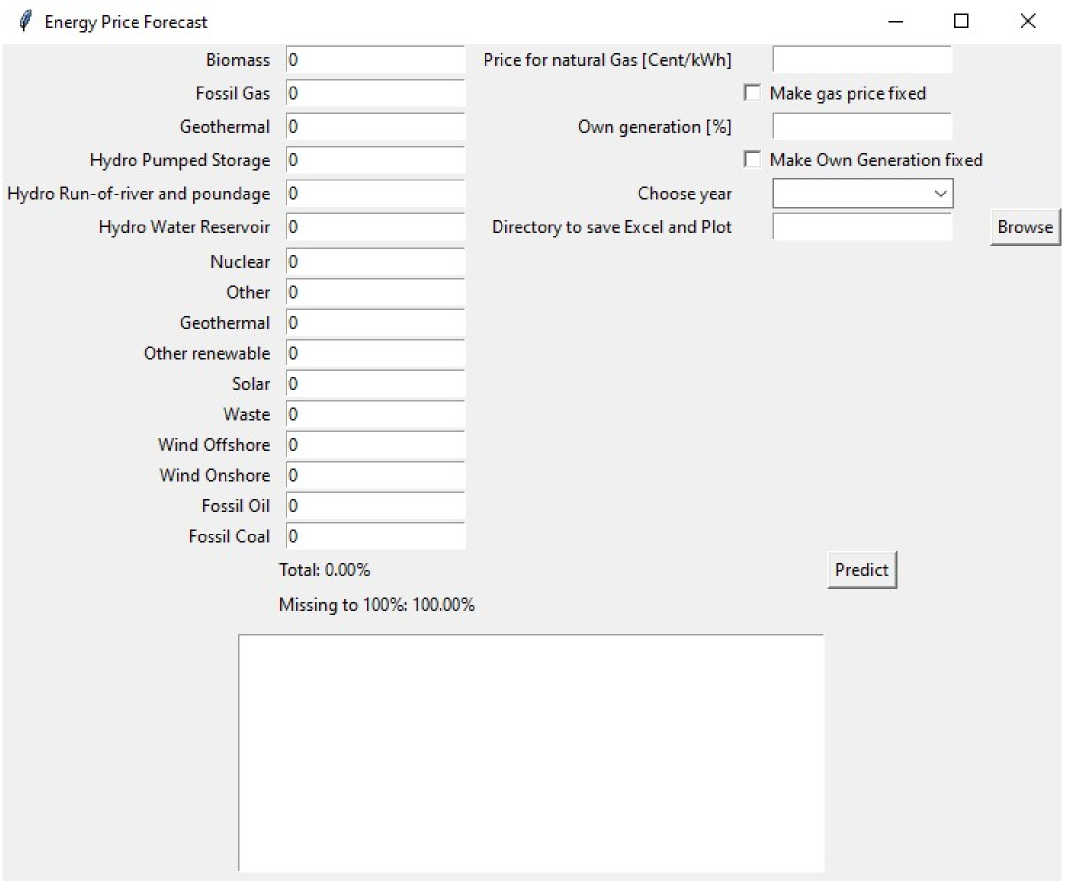


Figure 54: *Percentage Change in Price per Generation Type*

In Figure 54 you can see each production type and its mean value. This summarises all the diagrams of the sensitivity analysis. It is now clear that although the price rises with increasing solar production, even small amounts of solar production have an impact on the Day Ahead Price. The Day Ahead Price therefore already starts at a lower value. The increase that can then be observed could be explained by the increase in balancing energy, which is caused by the volatility of solar production.

4.4 Standalone Executable

To increase user-friendliness, a standalone executable (.exe) was created (Figure 55) to make predictions with the ensemble of transformer models. This is based on the Excel file "All countries average". Any electricity mix can be entered here, a self-generation (0-100%) and a gas price. The ensemble of 5 transformer models then creates a forecast. Since, as mentioned, the Excel file "All countries average" serves as the basis, the gas price, for example, is only scaled according to the user input. This results in the gas price having the same curve as in the data. You can also fix the gas price so that it always has the same value. However, this usually leads to a higher standard deviation. The value of the Own Generation can also be fixed. A year can also be selected, which then limits the data to the respective year in the data (2020-2023). If a directory to save to is entered, an Excel file containing the predictions and a plot of the predictions will be saved to that directory.


Energy Price Forecast

Biomass	0
Fossil Gas	0
Geothermal	0
Hydro Pumped Storage	0
Hydro Run-of-river and poundage	0
Hydro Water Reservoir	0
Nuclear	0
Other	0
Geothermal	0
Other renewable	0
Solar	0
Waste	0
Wind Offshore	0
Wind Onshore	0
Fossil Oil	0
Fossil Coal	0

Total: 0.00%

Missing to 100%: 100.00%

Price for natural Gas [Cent/kWh]

Make gas price fixed

Own generation [%]

Make Own Generation fixed

Choose year

Directory to save Excel and Plot

Figure 55: *Standalone Executable*

5 Conclusion, Discussion & Future Work

Contents

5.1 Conclusion

This work is a first step towards clarifying the complex relationships that form the electricity price. Transformer models demonstrate the ability to be able to map these relationships. It has been shown that renewable energies can not only be used to make the energy industry sustainable. They show the further advantage that they also reduce the price of electricity. The effect of all generation types can be seen in Figure 54.
As expected, the influence of the gas price on the Day Ahead Price was very high (Figure 48). The Own Generation does not have a strong effect on the Day Ahead Price as seen in Figure 49.
However, it must also be said that the standard deviation rises sharply for out-of-distribution data. This was also the reason why an ensemble of models was used. Because if only one model had been trained, it would not be possible to see how "certain" this model is. The ensemble of models can be viewed symbolically as a panel of experts. If all experts come to the same conclusion, then it is *more likely* that this prediction is also correct. However, there can be no 100% certainty. This is a problem that is inherent to many areas of machine learning.
The situation is similar when it comes to interpreting the results. The reasons given for the effect in solar production, for example, that the Day Ahead Price rises when solar production increases (Figure 50), *could* be explained by the increase in balancing energy, but it is not possible to verify this with the features used.
The transformer models are also unable to capture mutually exclusive options. For example, the Day Ahead Price in the models will still depend on the gas price even if only renewable energies are entered as input variables for the electricity mix. The reason for this is that this case is not covered in the training data.
As mentioned at the beginning, this work is only a first step towards understanding the complex processes that result in the electricity price and makes no claim to completeness.

5.2 Discussion

By reducing and simplifying the user inputs to the electricity mix, self-generation and the gas price, it was not possible to exploit the full potential of the transformer models. The ENTSO-E offers a lot more data that would be available for analysis. Examples include balancing energy or the flow of electricity between countries. However, this would have been very detrimental to the usability of the tool.
In addition, the analysis could be refined by adding further European countries. These were omitted because they did not report the data accurately. However, a precise analysis of which data could be trusted could improve the analysis. During the preparation of this master's thesis, it was observed that transformer models very probably have the potential to map the complex interrelationships of electricity pricing.

However, it was also observed that the standard deviation increases significantly as soon as the user inputs differ greatly from the values in the data. So you have to ask yourself how much you can extrapolate and how much you can trust these results.

5.3 Future work

As mentioned in section 5.2, the extensive ENTSO-E database should be used to further analyse the effects identified in this thesis. The analysis could be refined by applying different machine learning methods, using simpler, but possibly less precise models, which would have the advantage of better interpretability.
In addition, it would be useful to involve human experts to explain the observed effects. Given Europe's increasing focus on renewable energy, research could focus specifically on this area to investigate the challenges that a system based on renewable energy could pose.

Bibliography

Ahmed, Sabeen et al. (2023). “Transformers in Time-Series Analysis: A Tutorial”. In: *arXiv preprint arXiv:2205.01138*. DOI: https://doi.org/10.48550/arXiv.2205.01138.

Fraunhofer-Institut für Solare Energiesysteme ISE (2024). *Aktuelle Fakten zur Photovoltaik in Deutschland*. https://www.ise.fraunhofer.de/content/dam/ise/de/documents/publications/studies/aktuelle-fakten-zur-photovoltaik-in-deutschland.pdf. Zugriff am 19. Januar 2024.

Geissmann, T. and Adrian Obrist (2018). “Fundamental Price Drivers on Continental European Day-Ahead Power Markets”. In: *Energy eJournal*. DOI: 10.2139/ssrn.3211339.

Geva, Mor et al. (n.d.). “Transformer Feed-Forward Layers Are Key-Value Memories”. In: *arXiv* (). DOI: https://doi.org/10.48550/arXiv.2012.14913.

Godoy, Daniel Voigt (2022). *Deep Learning with PyTorch Step-by-Step*. ISBN: 979-8533935746.

Géron, Aurélien (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. ISBN: 9781492032649.

Harding, Michelle, Chiara Martini, and Alastair Thomas (2016). “Taxing Energy Use: Patterns and Incoherencies in Energy Taxation in Europe and the OECD”. In: pp. 233–264. DOI: 10.1007/978-3-319-21302-6_11.

Hirth, Lion, Jonathan Mühlenpfordt, and Marisa Bulkeley (2018). “The ENTSO-E Transparency Platform – A review of Europe’s most ambitious electricity data platform”. In: *Applied Energy* 225. DOI: https://doi.org/10.1016/j.apenergy.2018.04.048.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8. DOI: 10.1162/neco.1997.9.8.1735.

Ioannides, Filippos et al. (2020). “Electricity Pricing Using a Periodic GARCH-M Model with Conditional Skewness and Kurtosis Components”. In: *Econometric Modeling: Theoretical Issues in Microeconometrics eJournal*. DOI: 10.1016/J.ENECO.2021.105110.

Kang, Jian, Weiqiang Zhang, and Jia Liu (2016). “Gated recurrent units based hybrid acoustic models for robust speech recognition”. In: *2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pp. 1–5. DOI: 10.1109/ISCSLP.2016.7918456.

Kimdime (2006). *Map of European Transmission System Operators Organizations*. https://en.wikipedia.org/wiki/File:ElectricityUCTE.svg. [Online; accessed on Date of Access].

Kulkarni, N. and V. Bairagi (2018). “Classification Algorithms in Diagnosis of Alzheimer’s Disease”. In: pp. 61–71. DOI: 10.1016/B978-0-12-815392-5.00005-8.

Leuschner, Udo (2023). *The EU Directives for the Liberalization of the Internal Markets for Electricity and Gas*. [Online; accessed 19 December 2023]. URL: https://www.udo-leuschner.de/basiswissen/SB103-12.htm.

Li, Wei and D. Becker (2021). “Day-ahead electricity price prediction applying hybrid models of LSTM-based deep learning methods and feature selection algorithms under consideration of market coupling”. In: *Energy* 237, p. 121543. DOI: 10.1016/J.ENERGY.2021.121543.

Long, W. and W. Litzenberger (2012). “Fundamental concepts in High-Voltage Direct-Current power transmission”. In: *PES TD 2012*. DOI: 10.1109/

TDC . 2012 . 6281596. URL: https : / / consensus . app / papers / concepts - highvoltage - directcurrent - power - transmission - long / a15f97c0ccbc58559d2699b9f832c569/?utm_source=chatgpt.

Meeus, Leonardo (2020). *The Evolution of Electricity Markets in Europe*. Edward Elgar Publishing. ISBN: 9781789905465.

Muhammad, Tashreef et al. (2022). "Transformer-Based Deep Learning Model for Stock Price Prediction: A Case Study on Bangladesh Stock Market". In: *ar5iv*. URL: https://ar5iv.labs.arxiv.org/html/2208.08300.

Olah, Christopher (2015). *Understanding LSTMs*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Pra, Marco Del (2023). "Time Series Forecasting with Deep Learning and Attention Mechanism". In: *Towards Data Science*. URL: https://towardsdatascience.com/time-series-forecasting-with-deep-learning-and-attention-mechanism-2d001fc871fc.

Rehurek, Radim and Petr Sojka (2011). "Gensim–python framework for vector space modelling". In: *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3.2.

Singh, Paras Nath and Sagarika Behera (2022). "The Transformers' Ability to Implement for Solving Intricacies of Language Processing". In: *2022 2nd Asian Conference on Innovation in Technology (ASIANCON)*, pp. 1–7. DOI: 10.1109/ASIANCON55314.2022.9909423.

Tschora, Léonard et al. (2022). "Electricity Price Forecasting on the Day-Ahead Market Using Machine Learning". In: *Applied Energy*. DOI: https://doi.org/10.1016/j.apenergy.2022.118752.

Tunstall, Lewis, Leandro Werra, and Thomas Wolf (2021). *Natural Language Processing with Transformers*.

Vaswani, Ashish et al. (2017). "Attention is all you need". In: DOI: https://doi.org/10.48550/arXiv.1706.03762.

Yunsi, Ghita (2022). "Cryptocurrency Price Forecasting Using Transformer Model". In: *Capstone Project*.

Zhao, Huali, M. Crane, and Marija Bezbradica (2022). "Attention! Transformer with Sentiment on Cryptocurrencies Price Prediction". In: DOI: 10 . 5220 / 0011103400003197.

Zhou, Haoyi et al. (2020). "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting". In: *ArXiv* abs/2012.07436. DOI: 10.1609/aaai.v35i12.17325.

A APPENDIX ONE

Contents

A.1 Code

This code represents the data preparation for the Transformer model:

```python
import pandas as pd
import numpy as np
from sklearn.utils import shuffle
from collections import defaultdict
import glob
import os

# Define input and output directory paths
input_folder = "<input_data_folder_path>"
output_folder = "<output_data_folder_path>"

# Retrieve all Excel files from the input directory
files = glob.glob(os.path.join(input_folder, "*.xlsx"))

# Dictionary to store data frames, organized by country code
dfs_by_country = defaultdict(list)

# Read and organize files by country code
for file in files:
    country_code = os.path.basename(file).split("_")[0]
    df = pd.read_excel(file)
    df['Timestamp'] = pd.to_datetime(df['Timestamp']).dt.
        tz_localize(None)
    df.sort_values('Timestamp', inplace=True)
    dfs_by_country[country_code].append(df)

print('Data has been loaded')

# Data processing for each country
for country_code, df_list in dfs_by_country.items():
    for df in df_list:
        # Summarize and rename fossil oil-related columns
        columns_to_sum_oil = ["Fossil Oil", "Fossil Oil
            shale"]
        df['Fossil_Oil'] = df[columns_to_sum_oil].sum(axis
            =1)
        df.drop(columns_to_sum_oil, axis=1, inplace=True)

        # Summarize and rename fossil coal-related columns
        columns_to_sum_coal = [
            "Fossil Brown coal/Lignite", "Fossil Coal-derived gas",
            "Fossil Hard coal", "Fossil Peat"
        ]
        df['Fossil_Coal'] = df[columns_to_sum_coal].sum(axis=1)
        df.drop(columns_to_sum_coal, axis=1, inplace=True)

        # Calculate own generation ratio
        df['Own_Generation'] = df["Total_Generation"] / df["Actual_Total_Load"]
        df.drop(['Total_Generation', 'Actual_Total_Load'], axis=1, inplace=True)

        # Save the processed DataFrame to an Excel file
        output_file_path = os.path.join(output_folder, f"{country_code}_combined.xlsx")
        df.to_excel(output_file_path, index=False)

# Initialize dictionaries for features and labels
features = defaultdict(list)
labels = defaultdict(list)

# Drop 'Timestamp' column and split into features and labels
for country_code, df_list in dfs_by_country.items():
    for df in df_list:
        df.drop(['Timestamp'], axis=1, inplace=True)
        df_features = df.drop('Day_Ahead_Prices', axis=1)
        df_label = df['Day_Ahead_Prices']
        features[country_code].append(df_features)
        labels[country_code].append(df_label)

# Identify missing and infinite values in each DataFrame
for country_code, dfs in features.items():
    for df_index, df in enumerate(dfs):
        # Identify columns with NaN values and their count
        nan_counts = df.isnull().sum()
        nan_columns = nan_counts[nan_counts > 0].index.tolist()
        print(f"DataFrame {df_index} for Country Code {country_code} has NaN in columns: {nan_columns}")
        for column in nan_columns:
            print(f"Number of NaN values in column '{column}': {nan_counts[column]}")

        # Identify columns with infinite values and their count
        inf_counts = np.isinf(df).sum()
        inf_columns = inf_counts[inf_counts > 0].index.tolist()
        print(f"DataFrame {df_index} for Country Code {country_code} has inf values in columns: {inf_columns}")
        for column in inf_columns:
            print(f"Number of infinite values in column '{column}': {inf_counts[column]}")

# Create sequences for training
Xs, ys = [], []
sequence_length = 365
for country_code, dfs in features.items():
    for df_index, df in enumerate(dfs):
        for i in range(len(df) - sequence_length):
            Xs.append(df.iloc[i:(i + sequence_length)])
            if i % 1000 == 0:
                print(f'Sequence {i} for Country Code {country_code} created')

# Create label sequences
for country_code, dfs in labels.items():
    for df in dfs:
        for i in range(len(df) - sequence_length):
            ys.append(df.iloc[i + sequence_length])

# Shuffle and split the data
np.random.seed(42)
Xs, ys = shuffle(Xs, ys)
n = len(Xs)
train_size = int(n * 0.9)
val_size = n - train_size

X_train = Xs[:train_size]
y_train = ys[:train_size]
X_val = Xs[train_size:train_size + val_size]
y_val = ys[train_size:train_size + val_size]

print('Data split completed')

# Save training and validation data
np.savez('<path_to_save_training_data>/train_data.npz', sequences=X_train, labels=y_train)
print('Training data saved')
np.savez('<path_to_save_validation_data>/val_data.npz', sequences=X_val, labels=y_val)
print('Validation data saved')
```

This code represents the Transformer model with the included hyperparameter tuning:

```python
import numpy as np
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.callbacks import EarlyStopping,
    ModelCheckpoint
from tensorflow.keras.callbacks import TensorBoard
from keras_tuner import Hyperband

# Paths to training and validation data (anonymized for
    thesis)
train_data_path = "<path_to_train_data>"
val_data_path = "<path_to_val_data>"

# Load training and validation data
train_data = np.load(train_data_path)
val_data = np.load(val_data_path)

# Assign sequences and labels
x_train = train_data['sequences']
y_train = train_data['labels']
x_val = val_data['sequences']
y_val = val_data['labels']

print("Features shape:", x_train.shape)
print("Labels shape:", y_train.shape)

# Define the transformer encoder layer
def transformer_encoder(inputs, head_size, num_heads, ff_dim
    , dropout=0):
    # Conv1D layer to project inputs
    transformed_inputs = layers.Conv1D(filters=ff_dim,
        kernel_size=1)(inputs)
    x = layers.LayerNormalization(epsilon=1e-6)(
        transformed_inputs)
    x = layers.MultiHeadAttention(key_dim=head_size,
        num_heads=num_heads,
                                      dropout=dropout)(x, x)
    x = layers.Dropout(dropout)(x)
    res = x + transformed_inputs
    x = layers.LayerNormalization(epsilon=1e-6)(res)
    x = layers.Conv1D(filters=ff_dim, kernel_size=1,
        activation="relu")(x)
    x = layers.Dropout(dropout)(x)
    return x + res

# Function to build the transformer model
def build_model(input_shape, head_size, num_heads, ff_dim,
                num_transformer_blocks, mlp_units, dropout
                    =0, mlp_dropout=0):
    inputs = keras.Input(shape=input_shape)
    x = inputs
    for _ in range(num_transformer_blocks):
        x = transformer_encoder(x, head_size, num_heads,
            ff_dim, dropout)
    x = layers.GlobalAveragePooling1D(data_format='
        channels_first')(x)
    for dim in mlp_units:
        x = layers.Dense(dim, activation="relu")(x)
        x = layers.Dropout(mlp_dropout)(x)
    outputs = layers.Dense(1)(x)
    return keras.Model(inputs, outputs)

# Function to build hypermodel for tuning
def build_hypermodel(hp):
    head_size = hp.Int('head_size', min_value=2, max_value
        =10, step=1)
    num_heads = hp.Int('num_heads', min_value=2, max_value
        =10, step=1)
    ff_dim = hp.Int('ff_dim', min_value=32, max_value=128,
        step=32)
    num_transformer_blocks = hp.Int('num_transformer_blocks'
        ,
                                    min_value=1, max_value
                                        =5, step=1)
    mlp_units = hp.Int('mlp_units', min_value=64, max_value
        =512, step=64)
    mlp_dropout = hp.Float('mlp_dropout', min_value=0.1,
        max_value=0.5,
                          step=0.1)
    dropout = hp.Float('dropout', min_value=0.1, max_value
        =0.5, step=0.1)
    learning_rate = hp.Float('learning_rate', min_value=1e
        -4, max_value=1e-2,
                            sampling='log')

    model = build_model(x_train.shape[1:], head_size,
        num_heads, ff_dim,
                        num_transformer_blocks, [mlp_units],
                            dropout,
                        mlp_dropout)
    model.compile(optimizer=keras.optimizers.Adam(
        learning_rate=learning_rate),
                  loss='mean_squared_error')
    return model

# Initialize the Hyperband tuner
tuner = Hyperband(
    build_hypermodel,
    objective='val_loss',
    max_epochs=200,
```

```python
    directory='<hyperband_tuning_directory>',
    project_name='transformer_tuning_v2'
)

# Retrieve best hyperparameters
best_hps = tuner.get_best_hyperparameters(num_trials=8)

# Define callbacks for training
model_checkpoint = ModelCheckpoint('<model_checkpoint_path>',
                                   save_best_only=True)
early_stopping = EarlyStopping(monitor='val_loss', patience=8,
                               restore_best_weights=True)
tensorboard_callback = TensorBoard(log_dir='<tensorboard_log_dir>',
                                   histogram_freq=1)

# Train models with best hyperparameters
for i, hp in enumerate(best_hps):
    model = build_hypermodel(hp)
    model.fit(x_train, y_train, epochs=200, validation_data=(x_val, y_val),
              callbacks=[early_stopping, model_checkpoint, tensorboard_callback])

    # Save the trained model
    model.save(f'<models_directory>/Model_v3_{i}.h5')
```

A.2 Additional Figures

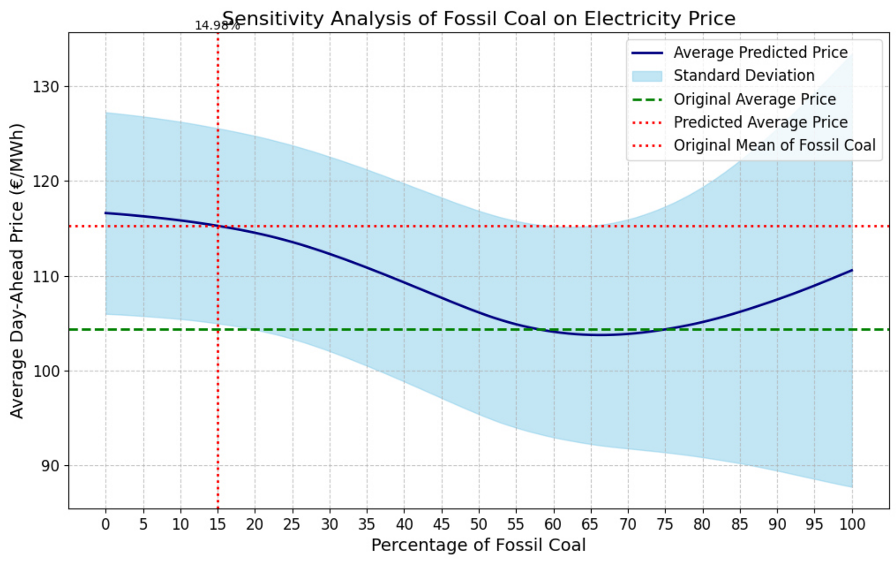


Figure 56: *Sensitivity analysis for the Fossil Coal on Electricity Price*

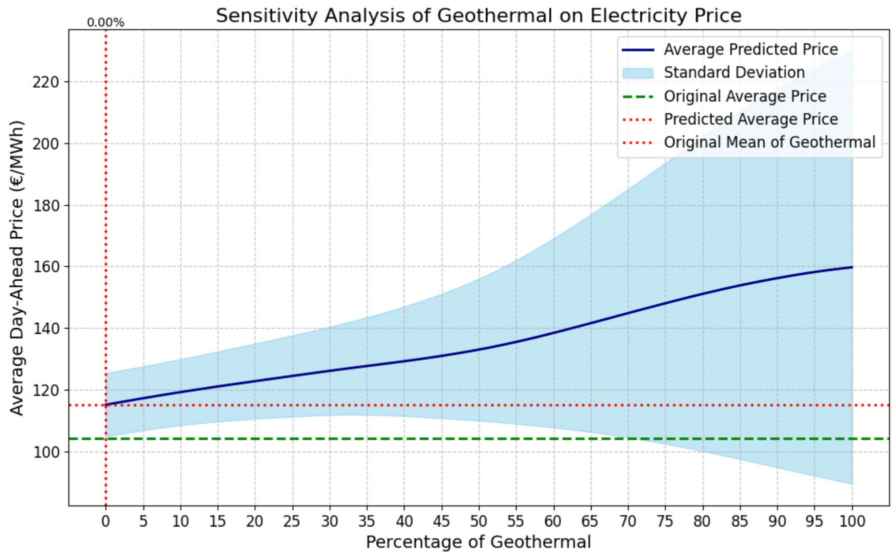


Figure 57: *Sensitivity analysis for Geothermal on Electricity Price*

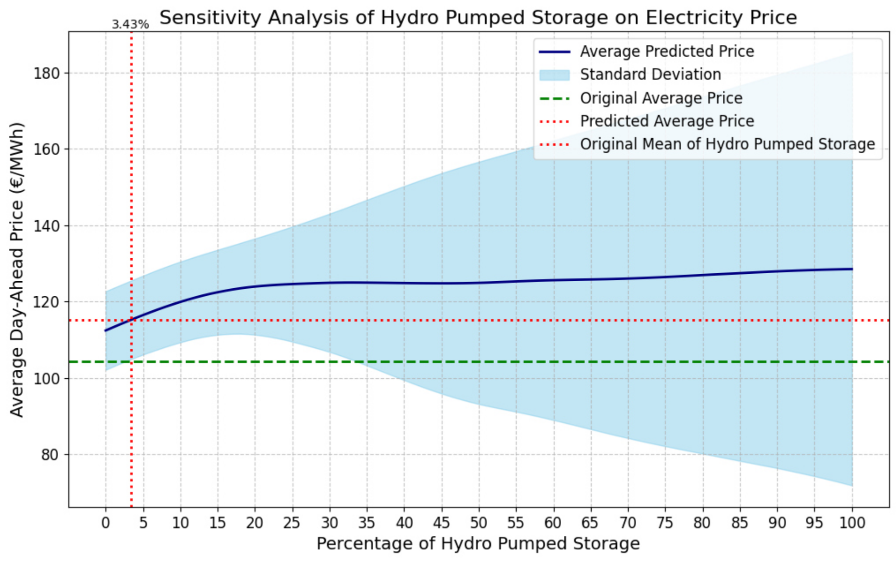


Figure 58: *Sensitivity analysis for Hydro Pumped Storage on Electricity Price*

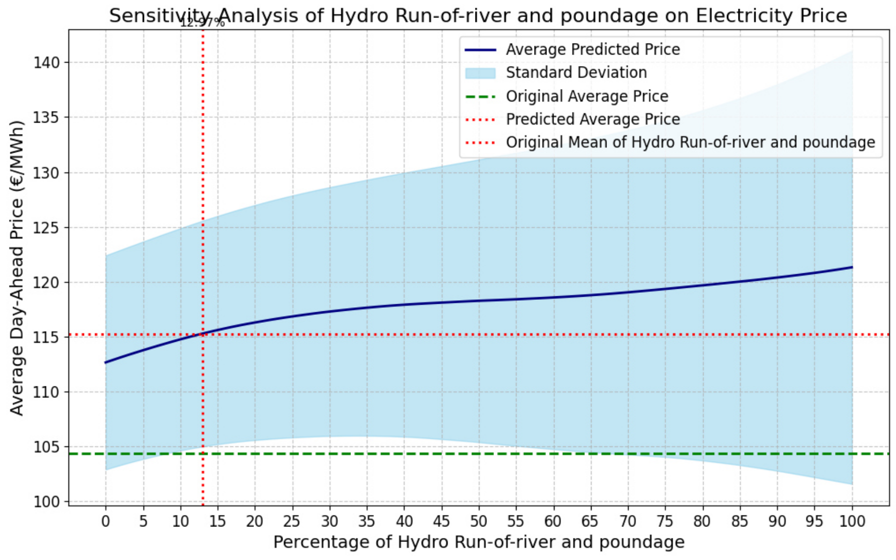


Figure 59: *Sensitivity analysis for Hydro Run-of-river and poundage on Electricity Price*

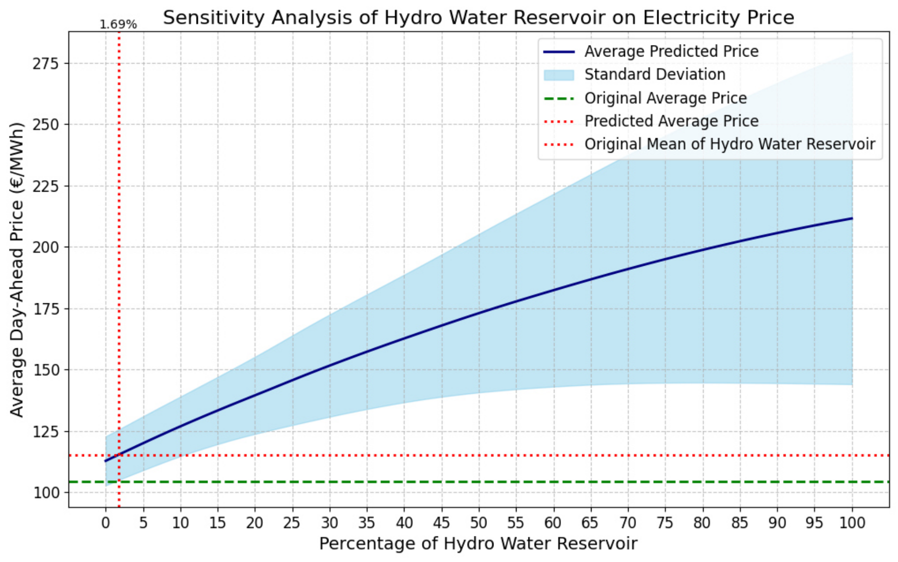


Figure 60: *Sensitivity analysis for Hydro Water Reservoir on Electricity Price*

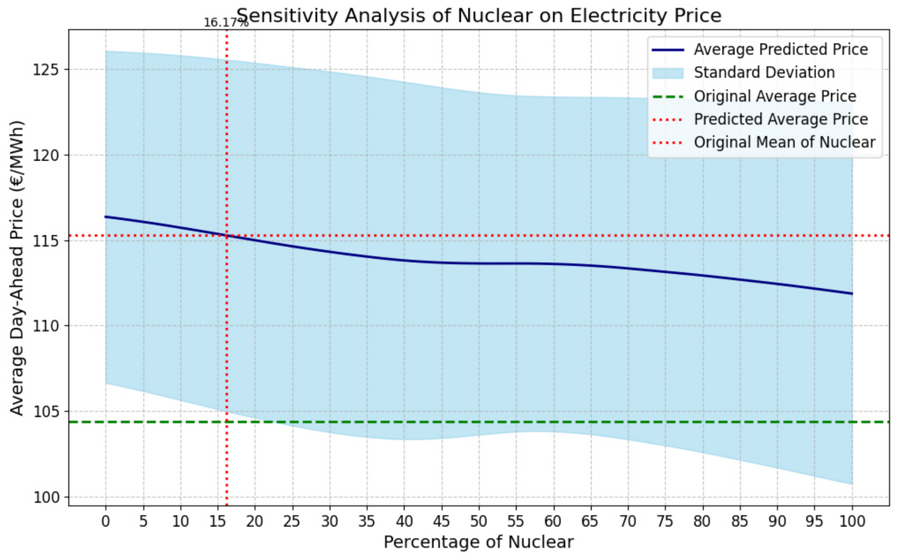


Figure 61: *Sensitivity analysis for Nuclear on Electricity Price*

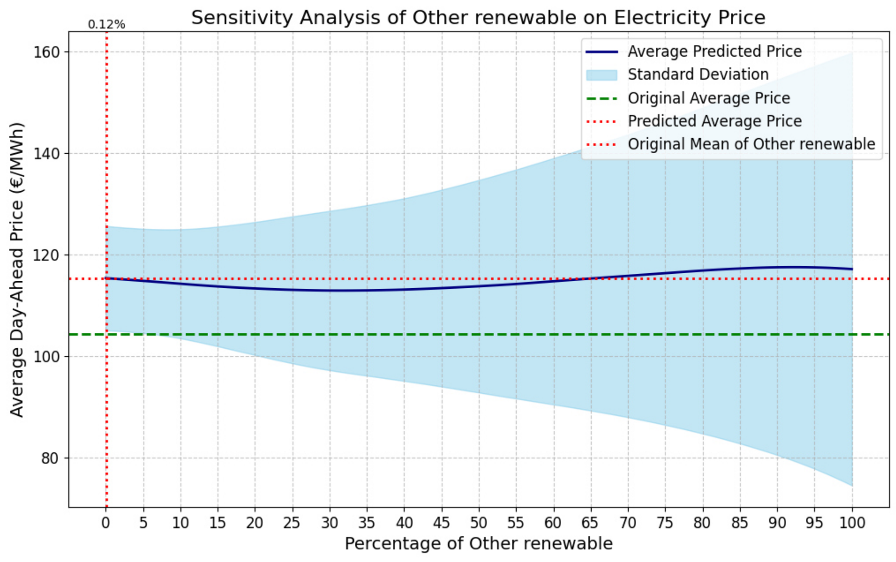


Figure 62: *Sensitivity analysis for Other renewable on Electricity Price*

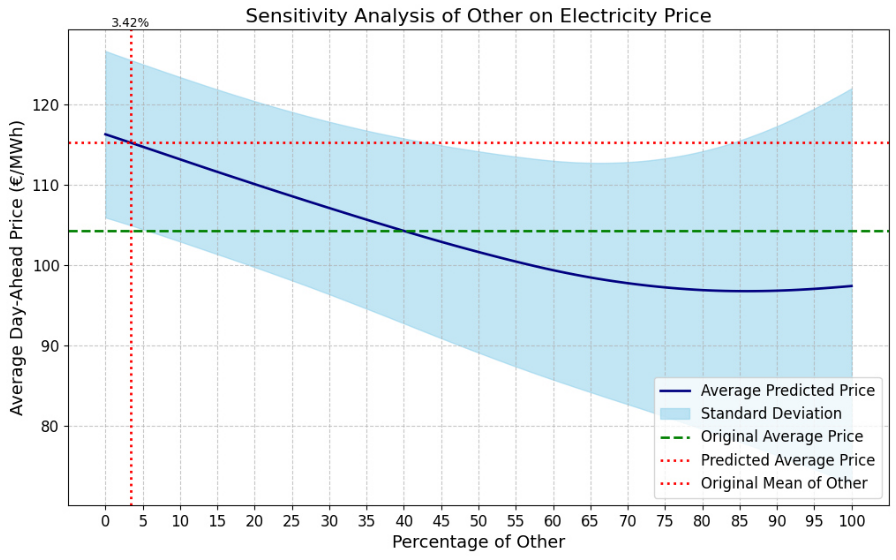


Figure 63: *Sensitivity analysis for Other on Electricity Price*

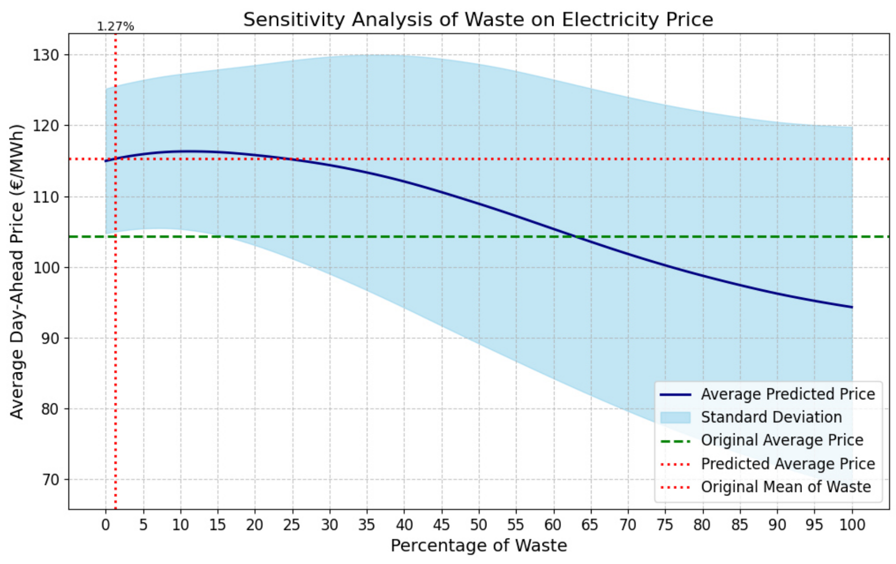


Figure 64: *Sensitivity analysis for Waste on Electricity Price*

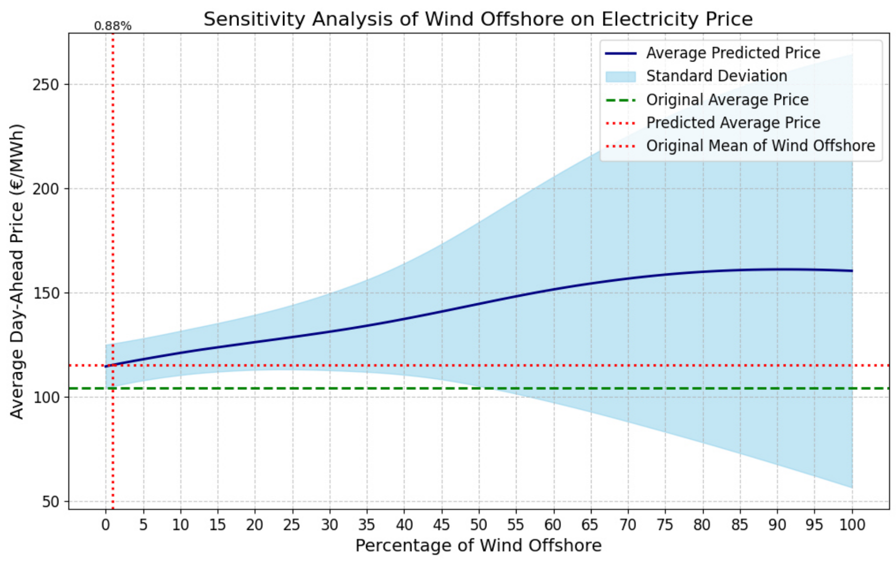


Figure 65: *Sensitivity analysis for Wind Offshore on Electricity Price*

B APPENDIX TWO

Contents

B.1 Nomiclature

Abbreviation	Complete Name
ENTSO-E	European Network of Transmission System Operators for Electricity
HVDC	High Voltage Direct Current
NLP	Natural Language Processing
LSTM	Long-short Term Memory Network
RNN	Recurrent Neural Network
CEGH	Central European Gas Hub

Nomiclature

B.2 Usage of Large Language Models

ChatGPT was used to optimise some of the code and also to rewrite some parts in writing.