Erasmus University Rotterdam

Erasmus School of Economics

Master Thesis Econometrics and Management Science: Business Analytics & Quantitative Marketing

Explaining Clickstreams by Layerwise Relevance Propagation in a 2D-Convolutional Neural Network

Mathijs Kroon (451067)

Supervisor:	dr. Kathrin Gruber
Second assessor:	dr. Paul Bouman
Date final version:	12th November 2023

Abstract

The online advertisement industry has grown significantly in the last couple of years. Predicting which specific advertisements are effective is becoming increasingly important and predicting if users click on specific advertisements has been thoroughly researched. Finding the reason why has been proven to be more difficult. The aim of this thesis is to uncover the black box associated with this problem. A way to uncover in image classification why certain classifications were made is by performing Layerwise Relevance Propagation (LRP) on a convolutional neural network (CNN). This technique has also shown promising results in structured data and will be adapted for the attribution problem mentioned above. Data from the online advertising company Criteo was used, which contains clickstreams of more than 6 million users. First, a 2D-CNN will be performed to predict which users click and hereafter LRP will be used to see which features were important for the user to click. The obtained model can effectively fit the existing data, however, it struggles to adapt to new data. With the fitted model, LRP is able to explain what drove a subsample of the users to click.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to dr. Kathrin Gruber. Without her invaluable support and guidance in every step, this thesis would not be lying in front of you. I have always greatly enjoyed the conversations with dr. Gruber. I also want to extend my sincere gratitude to my parents: Paul Kroon & Hester Kroon-Trouw, for their continuous support and patience during the pursuit of my academic career.

Contents

1 Introduction

The digital advertising market in the Netherlands grew with 13% in 2022, to more than 3.5 billion euros, according to a report by Deloitte (*Digital Advertising Spend 2022 the Netherlands*, 2023). In comparison, the total value of the offline advertising branch, such as magazines, radio, newspapers and TV was 1.6 billion euros. Six years earlier, in 2016, the combined sum was lower than the current online value. At that time, the total value of the advertising sector was 3.2 billion euros, with both online and offline accounting for 50% of the market. With an increasing digital advertising budget, companies will also want to invest in optimizing the allocation online.

The advertising budget can go to several digital channels, which include website advertising, boosting social media posts, email marketing and website search engines (Kurdi, 2022). HTTP cookies help websites keep track of your visits and activity, with third-party cookies also giving insights into past browser behaviour (Greenberg, 2003). These cookies, therefore, make it possible to observe which advertisements have been seen, before a customer decides to click on a specific advertisement. All of this information that is collected by websites is called the clickstream. From this behaviour, websites are able to better decide on whom to present which advertisement, such that it reaches optimal attention. Every advertisement seen by a user is also referred to as its journey and every separate advertisement is called a touchpoint. The process of finding the importance of certain advertisements, and their position for the customers to click or convert, is called the attribution problem (Shao & Li, 2011).

The attribution problem in the literature is historically based on rule-based heuristics, where those rules decide how much weight would be given to a specific touchpoint. Examples hereof include the First Touch Attribution and Last Touch Attribution, where either the first touchpoint or the final touchpoint receives all of the attribution. It can easily be understood that these methods are heavily biased and do not give any insights into a specific customer. Research shifted more towards algorithmic approaches, with the usage of the Shapley Value (Shapley, 1997). Another algorithmic approach included hidden Markov Models in the work of Anderl, Becker, Von Wangenheim and Schumann (2016), which started including differences in attribution values for different stages of the customer journey. With the ongoing inflow of data and more advanced methods which can make better use of the data, the application of these prior techniques has been restricted to specific factors. Therefore, the field of deep learning was introduced to uncover more complex patterns.

Modelling attribution with deep learning has been done with several approaches with specific goals in mind. Research by Arava, Dong, Yan, Pani et al. (2018) and Ren et al. (2018) has been focussed on predicting whether a series of touchpoints would lead to a click. The problem with

these kinds of methods is, that it remains difficult to pinpoint what eventually led to the click, whereas that is what is important for the allocation of digital advertising budgets. The methods are not regarded as "explainable", which is about opening the "black box", thus understanding what drove the model to make certain decisions (Rios, Gala, Mckeever et al., 2020). A model that is used in this field due to its ability to learn complex features and at the same time be able to explain these, is the Convolutional Neural Network. It has been used in a lot of image detection data (Montavon, Binder, Lapuschkin, Samek & Müller, 2019), and recent research by Rios et al. (2020) and Wang et al. (2017) has paved the way for structured data. The reason for choosing this model by the authors is that the technique of Layerwise Relevance Propagation (LRP) could be used to explain the findings of the model. The field of trying to explain the findings of a model is also referred to as Explainable Artificial Intelligence (E-AI). Layerwise Relevance Propagation is such an E-AI technique, where every decision of the model is evaluated by assigning relevance scores to the different inputs of the model. Via this way, heatmaps can be generated which can easily pinpoint for different users what was important for driving the decision.

The decision can still be click or conversion, the focus in this thesis will be on the click. The rationale behind this is that the focus of online advertising is not only to sell products, it could also be to generate attention for certain websites and services. Focussing on the click will keep an open scope and not limit the research only to the sales market. Solving the attribution problem for the clicks is already a difficult task which has been mainly researched by trying to get the right predictions for who clicked and who did not, not about explaining the click. Delving more into the field of (E-AI) could comprise the predictive accuracy of the model, this accuracy is however not the scope of this thesis. The scope of this thesis is more into explaining, for predictions that the model thinks for certain would have clicked, what drove that click. Using the E-AI techniques on structured data has already been done by Rios et al. (2020) for explaining customer churn, this did however not incorporate any form of sequential data. Research by Wu, Huang and Sutherland (2022) on predictive maintenance did include sequential data, the only downside of their model is that it uses the implicit assumptions that the further back in time, the smaller the attribution must be. Even though this assumption might hold for extremely large sequences, it disregards the significant amount of time an individual spends on the web in current times. The implication of that time is that it is not abnormal to assume that a user might see a lot of advertisements during an individual web visit, with all equal attribution values.

To contribute to the field of E-AI, this thesis attempts to answer the two following questions:

i) Can a 2D Convolutional Neural Network (2D-CNN) be used to accurately determine Click for online advertisements? and *ii) Can Layerwise Relevance Propagation be used to explain the findings from the 2D-CNN?*. The first question is there to ensure that the predictions that are going to be explained in the second question make sense. To answer these questions, an online advertising dataset is going to be needed. In the ideal scenario, one which is very rich in both users and categorical information. The one used in this thesis is one from Criteo, an international advertising company, first used by Diemert Eustache, Meynet Julien, Galland and Lefortier (2017). It contains 30 days of internet traffic for more than 6 million users. The categorical information is unfortunately anonymized, which is true for all free online datasets due to privacy reasons.

Empirical results demonstrate that the 2D-CNN is better at predicting clicks than simple rule-based and algorithmic approaches present in the literature. It scored better in terms of AUC, precision, accuracy and $F1$-measure compared to the LTA and Shapley Value approach. The model is unfortunately not able to fit very well, which should not directly disregard the findings of the model, as generalization is an issue in deep learning (Barbiero, Squillero & Tonda, 2020). The main goal of this thesis, again, is also not to get the highest accuracy or best-generalized model, but to explain the findings. Therefore, only the users for which the model with the highest prediction predicted that a click would occur, have been further analysed. The 100 best-predicted clicks have been backpropagated and relevance scores have been assigned to the features of the input journeys. From these combined scores, it is found which categories in what touchpoints were important for click.

The remainder of this thesis is structured as follows. Section 2 delves further into past research on the attribution problem, CNNs and E-AI. Hereafter, section 3 sheds more insights on the Criteo dataset and which data hereof is used. Section 4 explains how the different benchmark models, the 2D-CNN and LRP work and section 5 the results that follow from it. The final section 6 will summarize the results and discuss some suggestions for further research. The used code can be found in Appendix B.

2 Literature Review

Websites and tech companies have gathered enormous amounts of data and allowed researchers to investigate and test new research hypotheses. At the same time, increased computer power and the availability of cloud computing have allowed a significant amount of research to be done on these topics. This section dives into prior research and provides an overview of its current state. The literature review contains four sections. First, section 2.1 will provide information

about attribution modelling and the different methods used. Section 2.2 explains more about how different Neural Networks can be used to model attribution. Hereafter, section 2.3 will dive into different performance measures which can be used to assess the different models. The final section 2.4 will give the relevance of keeping the research explainable and how this has been done in the past.

2.1 Attribution Modelling

As mentioned in the introduction, the attribution problem is defined by finding the contribution every touchpoint had in the decision for a customer to click or not click. Several different solutions to the attribution problem have been proposed in the literature and these can be divided into three categories: *1)* Rule-Based Heuristics, *2)* Algorithmic approaches and *(3)* Deep Learning.

2.1.1 Rule-Based Heuristics

Rule-based approaches have been around for a long time and are more straightforward to understand. The first-touch attribution (FTA) and last-touch attribution (LTA) methods assign full credit to respectively the first or last touch point for the click (Ji, Wang & Zhang, 2016). The touchpoints which are encountered after or before that touchpoint are deemed unimportant. Research by Berman (2018) has shown that they are heavily biased. The time-decay model is another rule-based model, which gives more credit to touchpoints closer to the click. The last contribution will therefore receive maximum credit (Nisar & Yeung, 2018). Information regarding the categorical features of a touchpoint is however omitted and research by Lovett (2009) has shown it is more useful for short-lived deals or promotional offers. Since some researchers had a concern that the intrinsic value of each touchpoint cannot be credited easily, every impression is given equal weight in the Uniformly Distributed Attribution model (Nisar & Yeung, 2018).

All of these methods rely on simple, yet understandable, rules which can give a distorted view of reality and are not based on any actual features of the data.

2.1.2 Algorithmic Approaches

To overcome the issues from the previous section, research has shifted from rule-based heuristics to algorithmic approaches. The same research that shows the LTA is biased, comes with the Shapley Value as a well-working alternative, based on the game theory research by Shapley (1997). Each campaign is seen as a 'player' and conversion becomes the 'payoff', the Shapley Value is the average of all marginal contributions to all possible coalitions. Shapley Value Ana-

lysis then quantifies the outcome of the cooperative effect of the campaign and media platforms. With this method, Berman (2018) is able to increase advertising efficiency. The drawback of this method is that as the number of campaigns increases, the number of potential coalitions can grow exponentially. Research by (Zhao, Mahboobi & Bagheri, 2018) gave therefore two suggestions for improvement on this method. The first improvement is instead of making all possible coalitions, only create the ones that exist which include the specific channel. The second suggestion is an improvement of this method, which also included the ordering of the inputs.

Research by Anderl et al. (2016) takes a different turn, by looking at Markov Models. A dynamic Hidden Markov Model was designed, to further identify the importance of different campaigns. With this method, they were able to distinguish the importance of campaigns in different stages of the decision process. It also included other customer data for the touchpoints and a clear distinction between customer- and firm-initiated channels. A combination of the Markovian model and the Shapley Value is created in the work of Singal, Besbes, Desir, Goyal and Iyengar (2019). With their Counterfactual Adjusted Shapley Value metric for attribution, they were able to calculate attribution in a more robust way compared to previous research.

Hama, Mase and Owen (2022) researched the performance of the explaining power of the Shapley value in a deletion test, which is a test in which the relevance of a feature is defined by its importance in classifying an object. This would be tested by removing features and then reassessing the predictive accuracy of the input. They find that the Shapley value is not optimal for finding an optimal ordering here and therefore will not be helpful in explaining its finding. To counter this problem, they use Integrated Gradients and Local Interpretable Model-Agnostic Explanations to explain their classifications. Both are beyond the scope of this research, however, one of them which they mention and do not test is the Layerwise Relevance Propagation (LRP), which will be further explained in section 2.4. Another downside of the Shapley Value is that it only looks at campaigns, not at additional data related to the touchpoint and therefore an information loss occurs. Due to these limitations, another method will be explored in this research.

2.1.3 Deep Learning

To uncover relations between touchpoints and be able to incorporate categorical features, other frameworks have to be utilized. Deep learning models have been extensively used in contemporary research and are the solution to finding complex patterns. Attention mechanisms are used in these models to focus on relevant parts of the import data (Niu, Zhong & Yu, 2021). Arava et al. (2018) used attention mechanisms to create a Deep Neural Net with Attention for Multi-channel

Multi-touch attribution model, which predicted if a series of events could lead to a conversion. It also incorporated demographics, behaviour and control variables, to reduce the estimation bias. Around the same time, a Dual-attention Recurrent Neural Network was proposed by Ren et al. (2018), which is also able to look at sequential user patterns, impression-level and click-level user actions to derive conversion attribution. These types of neural networks are however not known for their interpretability. Research by Wu et al. (2022) on predictive maintenance and Arras, Montavon, Müller and Samek (2017) on sentiment analysis has shown promising results with this technique and LRP, which will be later discussed. The assumption is however made that the further back in time, the less relevance is given towards the data. The purpose of this research is to regard them as equal and to see what decisions the network makes. This is a more realistic assumption since the number of advertisements observed by a person in one web visit can be numerous and quickly after each other. If one would automatically assume that the more recent observations are more important, a bias can occur. The RNN will therefore be disregarded in this research.

There is however a type of Neural Network that has been mainly used for Explainable Artificial Intelligence (E-AI), which is a Convolutional Neural Network (CNN). It was first mainly used in explaining images (Montavon et al., 2019) and was later also used in explaining structured data (Rios et al., 2020) (Wang et al., 2017). The reason behind this was that the convolutional layers were better at feature extraction and able to do this with a reduced number of parameters, compared to a network with the same number of layers (Albawi, Mohammed & Al-Zawi, 2017). It works by not looking at single variables or features, but when its input goes to a node, it passes through a filter and can learn relations between inputs.

2.2 Convolutional Neural Network

The first usage of a convolutional neural network can be traced back to the work of Fukushima (1980), whose Neocogntiron is regarded as a predecessor of the networks used later. Due to limited computing power, the usage of the model was limited at that time. When this increased in the 1990s, the first notion of using convolution, was in the work of LeCun et al. (1989). A network by some of the same authors, the LeNet-5 (LeCun, Bottou, Bengio & Haffner, 1998), was an important breakthrough in popularizing CNNs and the reason they could hereafter be more widely used. This model was used successfully in recognizing hand-written digits. The results of Convolutional Neural Networks have been outstanding in the past 15 years, mainly in image processing and voice recognition. The usage of CNNs has also shifted towards structured data, examples include customer churn in the work of Rios et al. (2020) and heart disease

prediction by Singhal, Kumar and Passricha (2018). In the following section, the backgrounds of the different layers and set-ups will be given.

2.2.1 Layers and Set-ups

A Convolutional Neural Network is different from a standard Artificial Neural Network in the way that it contains two extra layers, a convolutional and a pooling layer. The first layer, respectively, is also the namesake of the model. This layer works as a filter, where it can filter the input matrix to obtain features or find important aspects of the data. When looking at the problem of facial recognition, a face objectively has several features, which would be eyes, a nose and a mouth. The nose can then again be divided into several optic features, such as two lines for a shaft and two circles for the holes. The goal of the filter is thus to extract the edges of the shaft and holes, which can later be combined into a nose. This filter is just a linear combination of the inputs and weights, however, non-linearity is needed, which ensures that the model can learn and recognize complex features (Sharma, Sharma & Athaiya, 2017). An activation function is added to these linear combinations to solve this. LeNet-5 discusses a sigmoid activation and uses a hyperbolic tangent, however, Krizhevsky, Sutskever and Hinton (2012) point out both have slow learning time and therefore introduced the usage of the Rectified Linear Units (ReLU) activation function for the CNN. The advantage of the ReLU is that it is not bounded for positive values and prevented the gradient from vanishing. The ReLU activation is hereafter in the literature chosen above the more classical hyperbolic tangent or sigmoid activation function (Ramachandran, Zoph & Le, 2017).

When these features are combined, the network can match these to a face. Combining these features happens in a separate layer between the convolutional layers, which is called the pooling layer. This layer can thus summarize the output. Fukushima (1980) called this process "subsampling" and LeCun et al. (1998) used in such way it would now be called "Average Pooling", which entails dividing the feature map into non-overlapping regions. The average of these regions is then calculated, which should reduce the dimensions and ensure that it is robust for shifts in the input. Krizhevsky et al. (2012) propose the "Max Pooling Layer", which does not take the average of a non-overlapping region, but the maximum. This is useful, as it could better focus on the most prominent features in that region.

There is no optimal setup of layers, size for the filters and the type of pooling that is used. Therefore, different architectures are presented in the literature for different applications. The earliest CNN was the LeNet-5 by LeCun et al. (1998), as discussed above. It is the simplest architecture, as it is simply stacked convolution and pooling layers. The model includes two

convolutional layers, two pooling layers and ends with two fully connected layers. Compared to more classical methods, like K-Nearest Neighbours and Support Vector Machines, LeNet-5 could give lower error rates with fewer parameters. The model used by Rios et al. (2020) can be compared to this architecture. To capture more complex relations, CNN's architectures have grown deeper. The level of depth is open for discussion and could be regarded as a danger for overfitting. Simonyan and Zisserman (2014) found out that pushing the depth to 16-19 layers, gave a significant improvement. The model they created to prove this, is the VGG-16 model, which helped them win the ImageNet Challenge 2014 (*ILSVRC2014 Results*, 2014) in the classification and localisation tracks.

2.3 Performance Measures

To evaluate the performance of the model, different performance measures are proposed throughout the literature. Most of the classical measures are based on the confusion matrix, which consists of 4 cells, namely true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). The most obvious measure would be accuracy, which is just the number of correctly estimated observations divided by the total number of estimates. However, for a binary model, an accuracy of 90% then suggests a correctly working model, however when 90% of the data is in one group this can easily be reached by always predicting that group. Other measures, such as precision and recall are therefore also common to use. Recall is the probability that the model will be able to find the positive values and precision that the positive values are indeed positive (Goutte & Gaussier, 2005). The F1-score is then a measure that combines both the information of precision and recall by taking their harmonic mean. It is given by:

$$F_1 = \frac{2TP}{2TP + FP + FN} \tag{1}$$

The accuracy, precision and F1-measure are used for determining the layers of the 1D-CNN by Rios et al. (2020). Since our data exhibits around the same number of positive and negative values as their Telecom Customer Churn Prediction Dataset, the same performance measures are going to be used in this research.

On top of these measures, three other measures will be used to determine the fit of the binary model, which are the area under the curve (AUC), the Brier score and the logarithmic score. The AUC is regarded in the literature as omnipresent to characterise the predictive power of the model (Lieli & Hsu, 2019). The curve refers to the Receiver Operating Characteristic curve, where the relationship between the true positive rate (TPR) and false positive rate (FPR) is given. When an increase in TPR does not increase the FPR by the same amount, the area

underneath the curve grows. The larger the area underneath the AUC, the better the fit of the model. The second one is the Brier score (Brier, 1950), which has become increasingly popular for comparing the accuracy of binary predictions in medical research (Rufibach, 2010). The score is the mean squared difference between predicted probabilities and the actual outcome, a Brier score should thus be close to 0 for better accuracy. The final measure is the logarithmic score, which measures the accuracy by taking the negative logarithm of the predicted probability assigned to the actual outcome (Gneiting & Raftery, 2007). Again, lower logarithmic scores imply a better model.

2.4 Explainable Artificial Intelligence

Keeping models explainable is important for several reasons. One of these is that this will ensure that the end user will be able to interpret the findings correctly and use it in the way it is intended. When this is being disregarded, models can produce unfair results and lead to discrimination (Lawless & Günlük, 2020). Another important reason is that E-AI helps to prevent 'Clever Hans' predictors, which might follow from biases in the training data (Lapuschkin et al., 2019). 'Clever Hans' follows the story of a German horse, which was believed to be highly intelligent. After further investigation, it was found that it only gave correct solutions, as it paid close attention to the reaction of the owner. The same goes for AI, where if we want to ensure that the models work correctly, one needs to ensure the model can generalise, instead of from what might be deduced from the creator.

E-AI has extensively been researched in the image recognition literature ((Zeiler, Krishnan, Taylor & Fergus, 2010) (Selvaraju et al., 2017) (Zhou, Khosla, Lapedriza, Oliva & Torralba, 2016)). A technique that has been used in the last years to explain which pixels in an image lead to classification is Layer Wise Relevance Propagation (LRP) (Bach et al., 2015). By propagating backwards to the network, weights are assigned to nodes and these can, in the end, be summed to give scores to different pixels. A heatmap can then be generated to visualize which pixels were important for classification. Predicting whether a brain MRI contains a tumour and in which part it is located is done by Ahmed, Asif, Saleem, Mushtaq and Imran (2023) and has great potential in the medical world. This technique can also be used in non-image contexts, as noted in the section 2.2.

3 Data

The data used in this thesis is obtained from Criteo, which is an international advertising company specialising in display advertisements. The data was used in a paper written for

Criteo by Diemert Eustache, Meynet Julien et al. (2017). Their research is also focused on the bidding aspect of advertisements, which is beyond the scope of this research. Criteo can be seen as the broker that provides websites with advertisements and advertisers a place to showcase their advertisements. The dataset includes over 16 million data points collected from 30 days of internet traffic. Every datapoint is one touchpoint from a user, which is represented by a *userID*. This unique user ID allows us to create user paths for different users and map them through time. Every touchpoint also includes a timestamp, a campaign, which is categorical, 9 other different categorical variables, whether or not a click was present and several other information on cost and conversion, which will be disregarded for this research. Information on the different categorical variables is captured in table 1.

Category	Campaign	1	2	3	4	5	6	7	8	9
Size	675	9	70	1.829	21	51	30	57.196	11	30

Table 1: Table with sizes of categorical variables

The discrepancy between the sizes of different categories can immediately be observed. The enormous size of Category 7 and to a lesser extent of Category 3, could potentially cause the model to overfit. The reason behind this is that there is only a small chunk of customers with a specific category in the train set and they all clicked, the model will probably expect the same category in the test set to always click, which could also be a potentially unimportant category. For this reason, categories 3 and 7 will be omitted from the research.

The size of the paths of different users also greatly differs. There are in total 6.142.256 unique users who collectively share 16.468.027 touchpoints. The shortest path is of length 1 and the longest path is of length 880, which are respectively present in more than half of the paths and only once. The different path lengths present are shown in figure 1. A big fraction of the data, more than 88%, has 5 or less touchpoints. Users with a small amount of touchpoints, do not reveal much of their preferences and might as well be accidental clicks. On the other hand, users with an enormous amount of touchpoints in the 30 days, are also not of much use. The reason behind this is that they are already extremely busy on the web and their behaviour cannot be compared to the general public. Therefore, paths which have a length of more than three clicks per day, 90 in total, are also omitted. This will leave us with 243.986 users. All of these users will thus have a path with a length between 10 and 90. Of this group, 38.25% clicked on the advertisement and 61.75% did not click on the advertisement.

The next step will be generating the 'images' which are to be used for the convolutional neural network. It is called images since the input will be shaped in such a way, that it can be compared to the input images used in the models discussed in section 2.2.1. For the CNN to

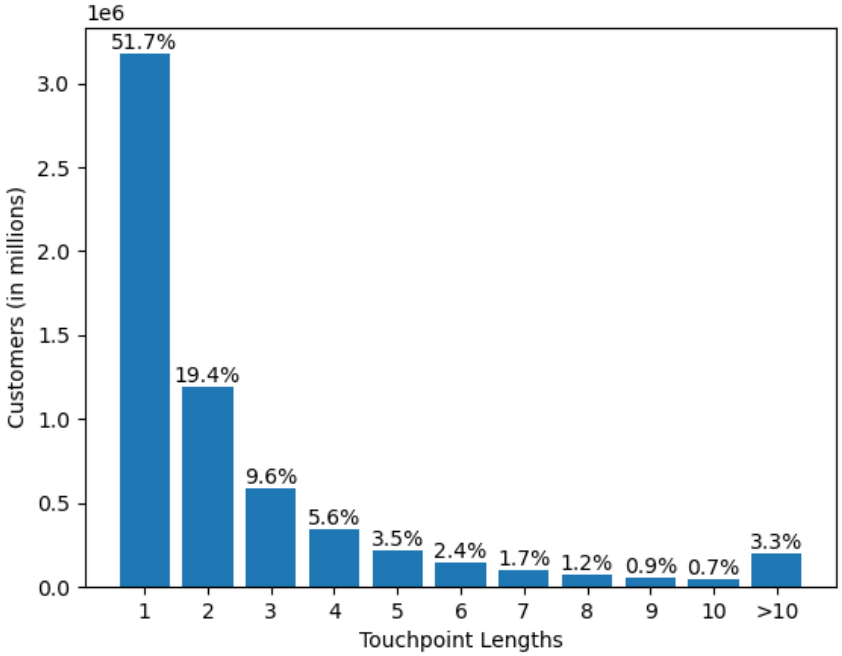


Figure 1: Touchpoint Lengths

be able to train in an efficient and manageable way, a fixed path length of 10 is chosen. This implies for paths which have more than 10 touchpoints, the final 10 touchpoints will be chosen. The logical explanation behind this would be that after having seen 10 advertisements, how conscious will you be of the 10 advertisements which preceded them. Due to computational issues, it will not be possible to create heatmaps for all of the 243.986 users. An image will thus consist of 10 rows for the different touchpoints. Every row then consists of 9 variables, which are: Campaign, Category 1, 2, 4, 5, 6, 8, 9 and the click. All of the different clicks are incorporated in the row, which allows CNN to also learn which rows click and which do not. Only for the final touchpoint, the click will not be given, since this would lead to endogeneity errors. The final click is therefore manually set to 1, which is chosen as it better fits the model than 0.

Generating heatmaps for every user will require a lot of computational power and time, which is not manageable and is beyond the scope of this thesis. A subsample with a size of 100 users will therefore be selected to use for the final Layerwise Relevance Propagation. It will be more insightful to take this sample of users predicted with high certainty that he/she is going to click, as this is in line with the goal of this thesis in explaining what drove a user to click. Therefore, the 100-user sample will be taken of the 10.000 users with the highest predicted probability to click. The smallest probability is then 85%. The subsample needs to inherit the same traits as the original data, therefore stratified subsampling will be used. A dependent variable will be

chosen and the subsample will inherit the same distribution for the dependent variable as the larger sample. The dependent variable will be the occurrences of touchpoint lengths, to ensure that all of the different users will be incorporated.

4 Methodology

In this section, the different methods will be discussed that form an approach to explain the clicks in a 2D-Convolutional Neural Network. First, bench-mark methods will be evaluated in section 4.1 to set a standard for the results. In the following section 4.2, first the basics of a convolutional neural network will be explained and hereafter a description of the different architectures. The final section 4.3, describes the Layerwise Relevance Propagation process.

4.1 Benchmark Models

The two benchmark models which will be used will be the Last Touch Attribution (LTA) and the Shapley Value. The LTA will be based on the classic approach of assigning all of the credit to the final touchpoint, which will be discussed in section 4.1.1. In the next section 4.1.2, the Shapley Value approach will be based on the Simplified Shapley Value method by Zhao et al. (2018).

4.1.1 Last Touch Attribution

The LTA, as also described by Ji et al. (2016), gives full credit to the final campaign of the touchpoint. The attribution for a campaign is then calculated by the following sum:

$$attribution_i = \frac{total\ number\ of\ clicks_i}{total\ number\ of\ touchpoints}, \tag{2}$$

where i indicates a certain campaign. This would imply that if two-thirds of the total number of customers had been presented campaign X as their final campaign, campaign X received an attribution of 66%. With attribution, different performance metrics can be calculated. The attribution can then be seen as the probability a customer is likely to click, whereas, with the example from above, people with campaign X would be predicted to click.

Since the number of campaigns is huge and an information loss occurs since other information is discarded, a second calculation for the LTA is done, where categorical information is included for calculating the attribution. The same formula as formula 2 is used, where i is one of the unique values from the nine different categories. To calculate a score for a customer, the different attributions for the campaign and different categories are added up and normalized between 0

and 1. Customers who are above a threshold, which is chosen based on results with the highest performance measures, are then expected to click and those below do not click. These results can then be used to calculate the needed measures.

4.1.2 Shapley Value

The Shapley Value, explained by Zhao et al. (2018) in the context of advertising attribution, is the weighted average of its marginal contribution over all possible coalitions for each channel. The marginal contribution is identified as:

$$M(j, S) = v(S \cup \{x_j\}) - v(S), \tag{3}$$

where $v(S)$ is the utility function of S, which represents a campaign. The utility function of $v(S)$ will be altered to the click function $c(S)$, which is just 1 if the collection of campaigns led to a click and zero otherwise.

As mentioned in section 2.1, this method has to take all of the coalitions of the campaigns present. This will not be feasible, since there are over 600 different campaigns. Therefore, the specific method is going to be based on the Simplified Shapley Value Method as presented in (Zhao et al., 2018). The formula for calculating the Shapley value for a specific campaign is:

$$\phi_i = \sum_{S \subseteq Pn\{x_i\}} \frac{1}{|S| + 1} c(S \cup \{X_i\}), i = 1, ..., p \tag{4}$$

In this formula, S is every coalition of a certain campaign i in combination with a fixed amount of p - 1, which are other campaigns that co-occur in the path of a customer with i. The size of p will be deduced from the data. The Shapley Value of a campaign is then the weighted average of all of the individual contributions of all coalitions, where the contribution is measured by a click C. The coalitions excluding the certain campaign will therefore not be calculated. This difference can be observed when comparing the two marginal contributions, where it can be observed that in equation 4, the $c(S)$ is not deducted, contrary to equation 3.

When all of the attributions are calculated for different campaigns in the journey, the total value of attributions can be calculated for a new customer. The sum of all of the attributions can then be converted to a probability of how likely this customer will be to click. After again looking at the different performance measures, a threshold will be determined and customers with a sum of attributions above this threshold are expected to click.

4.2 2D-Convolutional Neural Network

The method proposed in this thesis will be based on the idea of the 1D-CNN for structured data by Rios et al. (2020), however, with a 2D architecture, based on the architectures from LeCun et al. (1998) and Simonyan and Zisserman (2014). In the first four sections 4.2.1, 4.2.2, 4.2.3, 4.2.4, the different layers which are present and their parameters will be discussed. In the final section 4.2.5, the training techniques are going to be elaborated.

4.2.1 Input Layer

The starting point for our inputs is the image which is constructed in the data pre-processing step. The difference with the image from e.g. the LeNet-5 network, is that had a 28×28 input and the depth was 1, the model proposed in this thesis is 10×9 with also a depth of 1.

Due to anonymization, the categorical data was given a number, but the size of this number does not carry any value. The layer is therefore inputted through an embedding layer, similar to the work of Wang et al. (2017). The idea of embeddings is to transform the input objects, which could be words, sentences or numbers into a numerical representation (Chen, Perozzi, Al-Rfou & Skiena, 2018). In our case, these will be the different category values and campaigns. It can be viewed as a form of one-hot encoding, in which a value is represented by a vector with a 1 for the unique value and 0 otherwise (Pang, Lee & Vaithyanathan, 2002). For embeddings, however, a value between -1 and 1 is usually given and in contrast to one-hot encoding, one embedding is not the length of all of the unique values. It does exist out of multiple numbers, but this size is pre-determined per value. Since a specific value for a category in the first touchpoint, should not be given the same weight as one in a later stage, every value in a touchpoint gets an embedding, E_i. There is no optimal setting for the maximum embedding size. However (Lakshmanan, Robinson & Munn, 2013), suggest the following formula as a rule of thumb:

$$E_i = max(1.6 * \sqrt{unique\_values_i}, 600) \tag{5}$$

The 1.6 and 600 are included to ensure that even when the amount of unique values is lower in a text, enough embeddings are going to be created. Note, that with this formula the embeddings are mainly used in contexts with objects larger than one number and might carry a contextual meaning. A larger embedding size is more suitable for such a setting since this could capture complex relations and an almost infinite vocabulary can be inputted, however, this is not the case for the data used in this thesis. Equation 5 will therefore not be used, but an alteration is made, where a minimum is put on the number of embeddings:

$$E_i = min(unique\_values_i, max\_embed\_size) \tag{6}$$

The idea is based on the fixed length of 50 used in the thesis of Tang et al. (2014). The *max_embed_size* will be chosen through optimization and will be significantly lower since again we do not have an infinite amount of words to choose from, but a fixed set of options. The different weights that are put on the different embeddings, will be randomly chosen at the start.

Every campaign or category in a touchpoint will then be given an embedding $E_{c,t} = e_{1,c,t}, ..., e_{s,c,t}$ for s in *max_embed_size*. When all of these embeddings are then concatenated again to create the starting picture, a matrix will be constructed with height $H = 10$, following from the number of touchpoints chosen, and width $W = \sum_i E_i$. The input will then have size $10 \times \sum_i E_i$ and is pictured in figure 2.

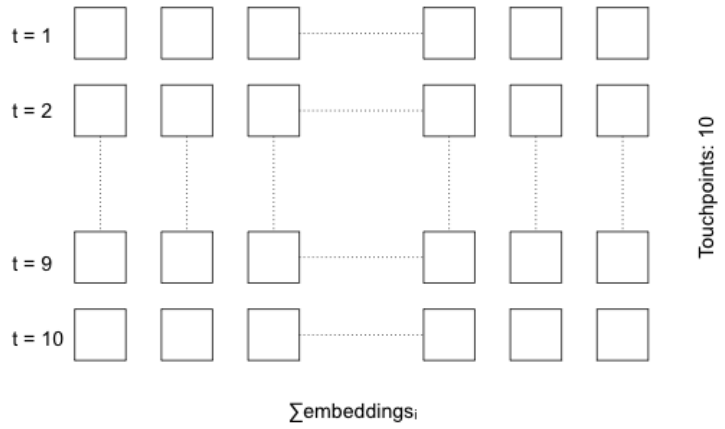


Figure 2: Input Structure

4.2.2 Convolutional Layer

To give an explanation of the usage of the convolutional layer, it is best to first discuss the reasons why this layer was added in the first place. In fully connected architectures, the way in which the input is presented to the network is ignored (LeCun et al., 1998). However, in image recognition, pixels that were next nearby could be correlated and combined to become local features. These local features would then become a set of categories, which were in the case of LeNet-5 corners and edges. As discussed in section 2.1.2, this technique is not only used in images and Rios et al. (2020) used this feature to learn important features of structured data. This also makes sense for the data used in this thesis, as information in the same touchpoint, e.g. the campaign and a certain category grouped together, could be an important reason for clicking, which otherwise might be overlooked. Another reason mentioned by LeCun et al. (1998), is that the convolution ensures the network can learn spatially invariant features. Spatially invariant implies that it

does not matter where the object is located in the picture, it will be able to recognize this. How this will work, will be further explained when the idea of the kernel is made clear. The reason, however, why this is important for the data in this thesis, is that a combination of a previous campaign with a click might be important, regardless of its placement in time.

With the importance of convolution in mind, the way it works can now be explained. Every convolutional layer exists of k filters, called the kernels. The job of such a kernel is to detect a feature and every kernel will therefore slide over the entire input. Sliding implies that the dot product is taken from the kernel and the input data. For simplicity, an easy example will be given of an image of 6×6 with a kernel size of 2×2. This kernel will carry four weights, which will be optimized in the training steps. The kernel will go over the entire image, starting at the top left and take the dot product with the kernel, which results in one value that will be filled in the top left feature map. Repeat this process until every 2×2 area is covered in the input image. The process with three examples for illustration is shown in figure 3. This 2×2 area is meant to capture the spatial invariant features, discussed before.

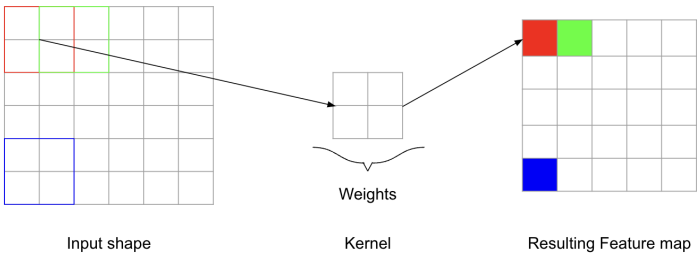


Figure 3: Convolution Process

The formula for the convolution step is given by (Yu, Wang, Chen & Wei, 2014):

$$y_k = f(w_k \cdot x) \tag{7}$$

where y_k is the k-th feature map. This feature map is then thus the result of the activation function $f(\cdot)$ for the dot product of the weights w_k of the k-th features map and inputs x. The ReLU will be used as an activation function, as discussed in section 2.2.1, and is given by the following equation:

$$f(x) = max(0, x) \tag{8}$$

After the first convolution step, an issue arises, which is that edges will be less frequently used than non-edge inputs, which could lead to a potential bias. To counter this issue, padding will be used, which is just adding zero rows around. The zeros rows are the padding and Simonyan

and Zisserman (2014) concluded that this substantially increased the receptive field. Figure 4 shows how the padding ensures that the shape of the input remains equal due to padding. This process is called zero-padding or same-padding. When this is not used, it is called valid-padding.

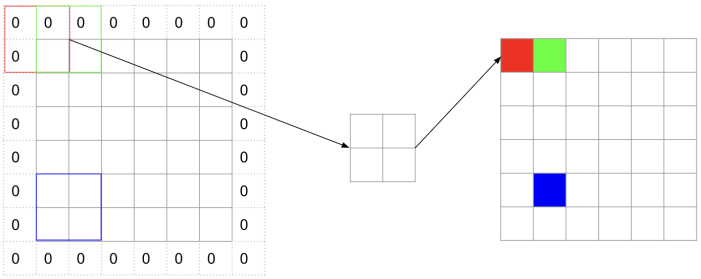


Figure 4: Padding

After having sketched the case for the 6×6 example, the attention can be shifted towards the case of this thesis. The input data which resulted from the embedding layer were 10 stacked touchpoints of size $1 \times \sum_i E_i$. Multiple settings for the kernel will be tried to find features. What could be especially interesting, is to set the kernel's height to one and the width to the same width as the embeddings. This could help find features that are distinct for different touchpoints. The limitation of this would be that this results in 10 dot products of a row with the weights, which would only be 10 values and might be a too significant information loss. This is illustrated in figure 5. Smaller kernels, which convolve within the touchpoints, will therefore also be tried. When the kernel width is the same size as the embeddings, no padding is needed, however, when the kernel is smaller, padding can be needed.

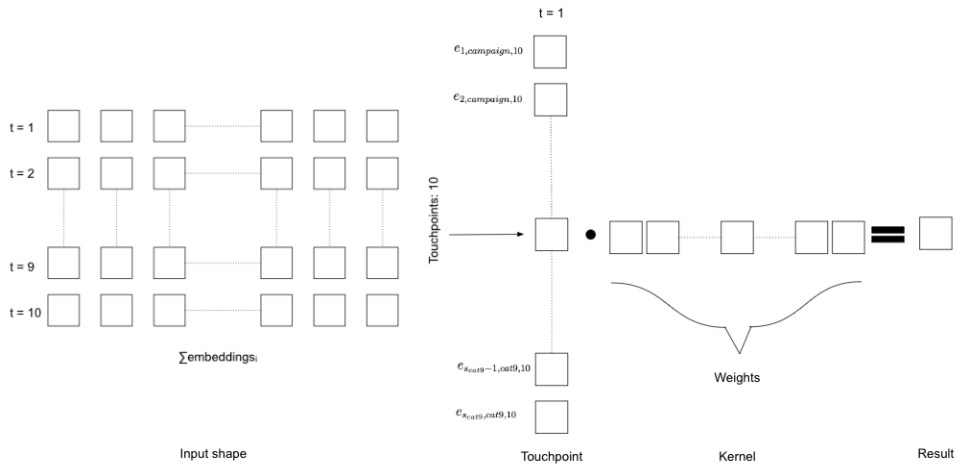


Figure 5: Convolution Result for One Touchpoint

The height of the convolution can however also be of a different size than 1, to include multiple touchpoints in one convolution. This could help find features inherent to multiple touchpoints and will be tested for several different combinations. The first combination will include only

the previous touchpoint and therefore a padded row of zeros needs to be implemented at $t = 0$ and at $t = 11$. Hereafter, other combinations with touchpoints around will be tested. For every touchpoint that would be added, one extra zero row needs to be padded. Another interesting combination would be to test whether features might exist in combination when one touchpoint is skipped, e.g. when the filter looks at touchpoint 1 and touchpoint 3, then to 2 and 4. This is actually just changing the step size and this stepsize is in terms of a CNN also called the stride. Skipping one touchpoint implies changing the stride from 1 to 2.

4.2.3 Pooling Layer

After having found the different features which might be present in the data in the convolutional layer, it will be important to differentiate between the ones that are relevant for predicting clicks and the ones that are not important. Therefore, a pooling layer can be added which can make this decision. Another reason it is frequently used is to reduce dimensions and the location of features becomes less important (LeCun et al., 1998).

As mentioned in section 2.2.1, there are two types of pooling layers: Average Pooling and Max Pooling. Pooling works in the same way as the convolutional layer with the kernels, the output is different however. Yu et al. (2014) propose the following definition for average pooling:

$$y_{k,i,j} = \frac{1}{|R_{i,j}|} \sum_{(p,q) \in R_{i,j}} x_{k,p,q}, \tag{9}$$

in which $y_{k,i,j}$ is the result for the k-th feature map in the area around (i, j), which is $R_{i,j}$ and of which its size depends on the used kernel. The sum is then taken of the different elements in that region, which are $x_{k,p,q}$ and divided by the total number of elements.

To focus better on the most prominent features, max pooling (Yu et al., 2014) will be used in this thesis, of which its formula is given by:

$$y_{k,i,j} = \max_{(p,q) \in R_{i,j}} x_{k,p,q}, \tag{10}$$

where the maximum is just taken in the part of the feature map $R_{i,j}$. By taking the maximum, the model can separate main and side issues.

4.2.4 Dense and Output Layer

To go from the different convolutional and pooling layers to a binary output, different types of layers have to be added. The first is a dense layer, also known as a fully connected layer. The dense layer takes all of the neurons in a previous layer and connects them to the neurons in

the next layer, with the goal of matching general patterns (Gu et al., 2018). There is no fixed rule on how many neurons this layer should have and the number of dense layers that should be present. Famous architectures, such as LeNet-5 and VGG-16, usually have 1 to 3 layers.

The final layer is the output layer. In this layer, the classification is made as to whether a touchpoint journey will result in a click or not. This is a binary problem and the Sigmoid activation function will therefore be used.

4.2.5 Training the Network

After creating a set-up and establishing all of the weights per layer, the network can be trained and the weights will be updated as long as the network can keep learning. The starting point of all of the weights will be randomized by Keras, which is the software package that is going to be used. In batches of a pre-defined size, the input images will be put through the network, also known as forward propagation, and the output will be compared to the real output. The loss function that will be used to evaluate this comparison will be the standard binary cross-entropy loss since we are dealing with binary classification, which is given by (Ho & Wookey, 2019):

$$J_{bce} = -\frac{1}{M}\sum_{m=1}^{M}[y_m * log(h_\theta(x_m)) + (1 - y_m) * log(1 - h_\theta(x_m))], \tag{11}$$

where y_m, x_m and h_θ are respectively the label, its input and the network. The first part ensures false positives are incorporated and the second part the false negatives. Based on this output, backpropagation is executed and the weights are updated accordingly. Optimizing these weights iteratively is however a lengthy process and there is the risk of getting stuck in a local minimum. For this reason, different optimizing algorithms were invented. VGG-16 and the Imagenet both use the Stochastic Gradient Descent (SGD), however, an improvement on this was made by Kingma and Ba (2017), with the Adaptive Moment Estimation optimizer. The rule for optimizing the weights is:

$$w_t = w_{t-1} - \eta\frac{\hat{m_t}}{\sqrt{\hat{v_t}} + \epsilon}, \tag{12}$$

where w_t are the weights, η is the pre-defined learning rate and $\hat{m_t}$ and $\hat{v_t}$ are moving average of respectively the gradient and squared gradient. Tuning the learning rate with information on past gradients improves convergence. Also, by adaptively changing the gradients compared to globally in SGD, the memory requirements are also significantly lower. For these reasons, the Adam optimizer is chosen. The number of times the entire sample is propagated through the network is called the number of epochs. The performance measures are evaluated at every

epoch and when it stops improving, it implies the network is overfitting.

Overfitting the network implies that the network is only suited for the given data and will not be working for predicting the click for new data. Several precautions will be taken to be better prepared for this issue. One is to randomly drop nodes and the arc which are connected during training. The idea is that this thinned network will not create nodes for every exception and be better at generalization (Srivastava, Hinton, Krizhevsky, Sutskever & Salakhutdinov, 2014). Dropout can be initialized after each layer with a percentage of how many nodes should be dropped. Another method is to add regularizing terms to the layer weights, which can be done in two ways. The first is the L1 regularization, also known as Lasso, which adds a penalty term times the weights to the loss function and forces it into feature selection (Vidaurre, Bielza & Larranaga, 2013). L2 regularization is the ridge regularizer, which adds a penalty term times the squared weights of the loss function in order to prevent extremely large weights (Hastie, 2020).

4.3 Layerwise Relevance Propagation

The reason the 2D-CNN setup is chosen is that the Layerwise Relevance Propagation technique can be used. The hope here is to be able to show for the entire dataset as well as for individual samples what was important for conversion. It works by propagating backwards in the network and giving individual *Relevance* scores to every node. The score is calculated with the following formula:

$$R_i^{(l)} = \sum_j \frac{x_i^{(l)} w_{ij}^{(l,l+1)}}{\sum_i x_i^{(l)} w_{ij}^{(l,l+1)}} R_j^{(l+1)} \tag{13}$$

The $x's$ in the equation are the inputs given, the $w's$ are the weights that are determined by the model and the $l's$ is the layer in which a node is residing. A relevance score for node i in layer l is thus calculated by adding up all of the different next weights times the current input and dividing this by the combined total of the other weights and inputs. So one node will be the sum of all of the relevance scores of the nodes following. It will make sense that the relevance score for more used nodes, and in the end more used features, will be higher as the input of these nodes will be higher as they are more often activated.

There are two alterations to the LRP equation, where the first is the Epsilon Rule (LRP-ϵ) (Montavon et al., 2019). The new formula is given in this case by:

$$R_i^{(l)} = \sum_j \frac{x_i^{(l)} w_{ij}^{(l,l+1)}}{\epsilon + \sum_i x_i^{(l)} w_{ij}^{(l,l+1)}} R_j^{(l+1)} \tag{14}$$

A predefined epsilon is added to the denominator and reduces the relevance when the effects on neuron j are weak or contradictory. It is therefore more used in the middle layers to filter out spurious relations found by the kernels.

The other alteration is the Gamma Rule (LRP-γ). In the Gamma rule, positive contributions are added to the outcome with a factor γ, which creates the following equation:

$$R_i^{(l)} = \sum_j \frac{x_i^{(l)} \rho(w_{ij}^{(l,l+1)})}{\sum_i x_i^{(l)} \rho(w_{ij}^{(l,l+1)})} R_j^{(l+1)} \tag{15}$$

where, $\rho(\theta) = \theta + \gamma\theta^+$, and $\theta^+ = max(0, \theta)$. Favouring the positive relevances can ensure that the heatmaps will be easier to understand since the positive relevances will be higher. The LRP-γ will, therefore, be used in the lower layer.

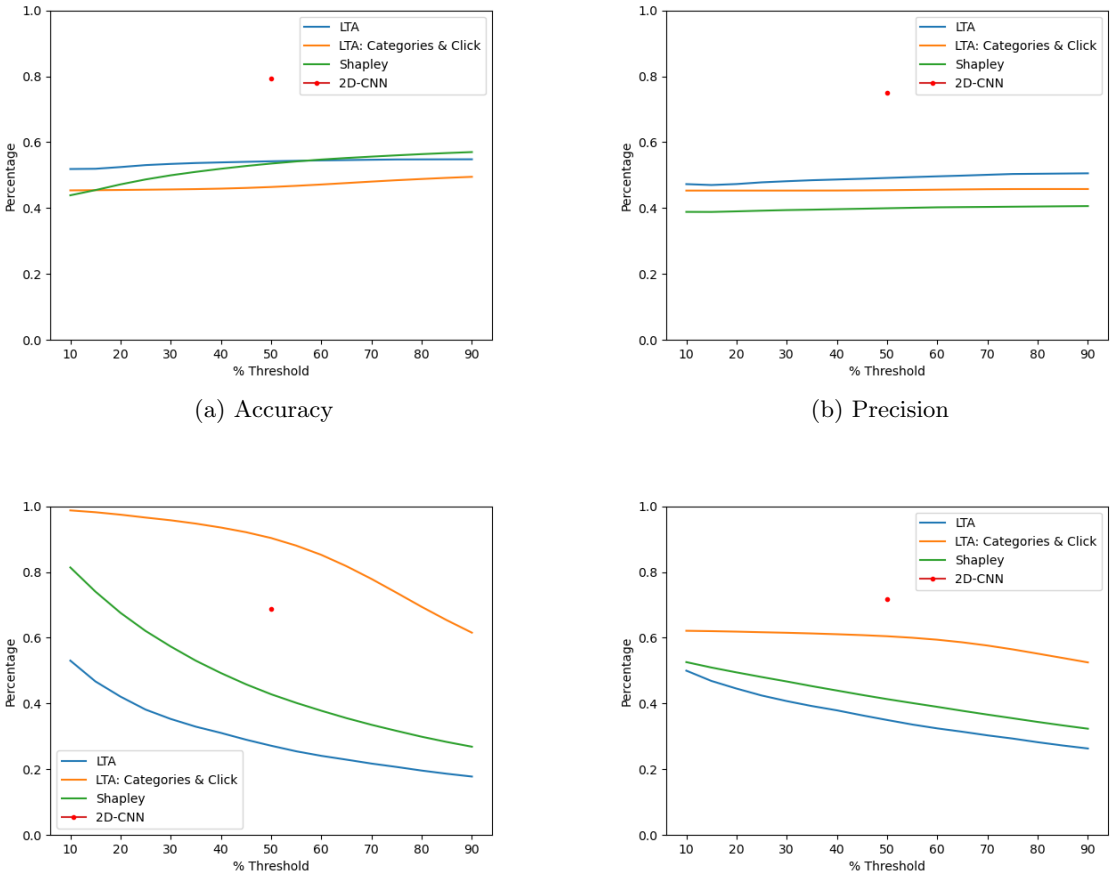
The chosen values for ϵ and γ will both be 0.25, which is in accordance with Montavon et al. (2019). The middle layer for the Epsilon rule will be the convolutional layer since this is where the spurious relations might occur. The lower layer for the Gamma rule will be the input layer, as this is the final step before the heatmap is finished.

5 Results

The following section presents the empirical results of the proposed methodology. This section will be split into three parts, first, in subsection 5.1, the results of the different benchmark models will be given. Hereafter, in subsection 5.2, the results of the proposed 2D-CNN will be discussed. The comparison of these models will be made in Section 5.3. This thesis aimed to increase the explainability of neural networks, and these results are shared in the final subsection 5.4. The results were created on a MacBook Pro with M1 and the packages can be found in Appendix C.

5.1 Benchmark Models

Two different benchmark models have been tested, the LTA and the Simplified Shapley Value. The results of both methods are in respective sections 5.1.1 and 5.1.2. As stated in section 4.1.1, different thresholds for a probability to click are tested to find the optimal threshold. The performance measures for these different thresholds for all of the different models are plotted in Figure 6. The graphs show for the different performance measures which model performs best.



(a) Accuracy

(b) Precision

(c) Recall

(d) F-score

Figure 6: Accuracy, Precision, Recall and F_1-Score measures for four different models across varying benchmarks

5.1.1 LTA

The LTA is tested for two different types of data, first for only the campaign, and later also based on the different categories. The blue lines in Figure 6 are for the LTA based on the campaign only. The accuracy and precision curves are nearly flat for the different percentages, which indicates that the decisions are being made at random. The F1 score and Recall are higher for the percentage between 10% and 20%, yet quickly drop hereafter. This can be explained by the fact that with a higher threshold, more observations are predicted to be negative, therefore increasing false negatives which is in the denominator of both. In Figure 6 the orange line is the situation when categories are added to the decision-making. It can be observed that the accuracy and precision exhibit similar behaviour compared to the situation before adding the categories and are expected to be close to random. The fact that the recall is close to one is due to the fact that the model will in this case predict all cases to be positive.

The performance time of both models is within seconds. Due to the bad performance of the

measures for the different percentages, both LTA models will not be regarded as informative.

5.1.2 Simplied Shapley Value

The Simplified Shapley Value will be evaluated in the same way as the previous benchmark model, in Figure 6 it is the green line. The Simplified Shapley Value only takes a few seconds to run, which can be attributed to the implementation of the simplification. It can be observed that there is a more clear point where an optimal value would be chosen in the middle, at around 55%. Even though this is an optimal point, it is still not a point that proves this is an optimal model as the measures are still close to being random.

5.2 2D-Convolutional Neural Network

In the following Table 3, the results are presented for the different setups of the model. The model is trained with an 80/20 training and test split, however, it is unsuccessful in fitting the model to the test set. The measures to prevent overfitting from section 4.2.5, dropout and kernel regularizers, have been initialized, however, did only prevent the model from fitting. When overfitting measures prevent the model from fitting, underfitting could also be present, which occurs when the model is too simple to fit the data. Different set-ups are presented in this section, to try and find the optimum between under- and overfitting, however with the current model none were successful in fitting.

In the following Table 2 some of the tested hyperparameters are given in bold the ones which were best for the starting set. These were hereafter used for the choice of different layers. The number of epochs is fixed since the model did not show any significant change hereafter.

Parameter	Values
Epochs	10
Embedding size	3, 5, **10**
Batch size	**32**, 64, 128

Table 2: Hyperparameters Settings for the 2D-CNN

Different model architectures were tested, where the first main choice is in the type and number of layers, kernel sizes and number of filters. To begin with the type of layers. The choice is between adding pooling layers after the convolutional layer or skipping these. For the same architectures as in Table 3, the model performed worse when a pooling layer was introduced. For this reason, the pooling layer is skipped in the results. Apart from the ones in Table 3, other kernel sizes have also been tested, which include looking at 2 touchpoints instead of 1 for the full-row kernel and also looking at more than 2 observations in the past. All

Layers	Padding	Kernel	Filters	Dense	AUC	Accuracy	Precision	Recall	F1
1	Valid	(2,1)	50	10	0.836	0.766	0.718	0.641	0.678
	Valid	(2,1)	150	50	0.832	0.762	0.714	0.635	0.672
	na	(1,69)	50	10	0.819	0.754	0.704	0.614	0.656
	na	(1,69)	150	10	0.838	0.769	0.725	0.640	0.680
2	na	(1,69)	50						
	Valid	(2,1)	50	50	0.840	0.771	0.728	0.641	0.682
	na	(1,69)	250						
	Valid	(2,1)	100	50	0.849	0.779	0.739	0.652	0.693
	na	(1,69)	300						
	Valid	(2,1)	200	100	0.856	0.784	0.784	0.658	0.699
	na	(1,69)	50						
	Same	(2,1)	50	50	0.832	0.764	0.720	0.627	0.671
	na	(1,69)	250						
	Same	(2,1)	100	50	0.852	0.781	0.744	0.652	0.696
	na	(1,69)	300						
	Same	(2,1)	200	100	0.857	0.784	0.746	0.660	0.701
3	na	(1,69)	50						
	Same	(2,1)	10						
	Same	(2,1)	2	2	0.849	0.778	0.732	0.663	0.696
	na	(1,69)	150						
	Same	(2,1)	50						
	Same	(2,1)	25	10	0.851	0.780	0.734	0.666	0.698
	na	(1,69)	200						
	Same	(2,1)	100						
	Same	(2,1)	50	20	0.841	0.771	0.724	0.649	0.685
	na	(1,69)	300						
	Same	(2,1)	150						
	Same	(2,1)	50	20	0.867	0.793	0.751	0.688	0.718

Table 3: Results of the different setups for the CNN. Different number of layers, filters, dense layer nodes and kernel sizes, with either *Valid* and *Same* padding, from section 4.2.2, have been tried.

of these did not yield any extra performance gains and therefore the choice was made for the models which are now in Table 3. It can be observed that adding complexity to the model has resulted in improved performance measures. All of the best measures are for the setup with the most layers and filters. An important note with these findings is that the differences are very small and it can be argued whether the increase in time and complexity is worth the 0.2%. The last option of the double-layer architecture took 38 seconds per epoch, whereas it took over 9 minutes per epoch for the triple-layer architecture. In Section 2.2.1, 16-19 layers were suggested, however, this was the case for larger and more complex image data and would not work in this context, since the data is smaller and simpler. The training time would then also experience an exponential increase.

5.3 Comparison Benchmark model and 2D-CNN

In the next Table 4 the AUC, Brier- and logarithmic score of the different benchmarks and the 2D-CNN are given. It can be observed for all of the benchmark models that the AUCs are close to 0.5 which indicates that the choices from the model are not significantly different from a random choice. The AUC for the 2D-CNN of 86.7% indicates that the model is well in distinguishing between the two classes. Also in terms of the Brier and logarithmic score, it outperforms the other three models.

Model	AUC	Brier	Logarithmic
LTA: Campaign	0.531	0.355	1.697
LTA : Campaign & Category	0.519	0.295	0.807
Simplified Shapley Value	0.522	0.287	0.968
2D-CNN	0.867	0.136	0.423

Table 4: Different AUC values for the four different models: LTA: Campaign, LTA: Campaign & Category, Simplified Shapley Value & 2D-CNN

The optimal values for the 2D-CNN have also been added to the plots of the benchmark models in Figure 6 to further show the difference in performance between the benchmarks and the 2D-CNN. The different values for the different thresholds have not been included in these graphs for the 2D-CNN. The reason behind this is that since the model uses a sigmoid activation to create the predictions, the predictions are automatically scaled to the 50% threshold.

The 2D-CNN outperforms all three models on almost all of the measures, whereas only the LTA with categories included outperforms on recall. Noted with this measure should be that for only predicting positive values, it will always be high and the combination of this with the other measures shows it should not be trusted solely. From these measures combined, it can be concluded that the 2D-CNN is a better model compared to the benchmarks for the specific dataset. Unfortunately, it is not well in generalizing and further research is needed if one wants to achieve that goal.

5.4 Layerwise Relevance Propagation

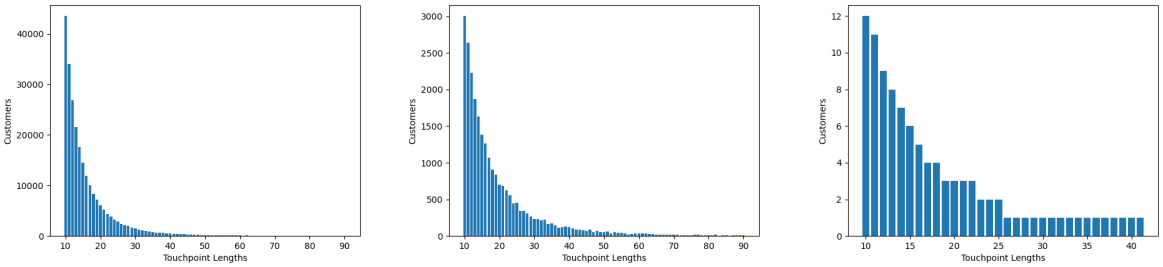
Since the results from the 2D-CNN showed that it failed to generalize well, the simplest architecture is chosen for the LRP. In combination with the computational burden it proved to be to back-propagate every entry through the network, a one-layer 2D-CNN is chosen as the model. A choice is made for five filters and two nodes in the dense layer. The results of this network are in Table 5. This model took 3 minutes to run 10 epochs. In Appendix A the ineffectiveness to fit a validation is shown for 100 epochs, which took 46 minutes. Here it can also be observed that the model succeeds in fitting the train set and therefore will only be used in the following

results.

AUC	Accuracy	Precision	Recall	F1	Brier	Logarithmic
0.800	0.739	0.684	0.588	0.632	0.177	0.528

Table 5: Performance Measures 1 Layer 2D-CNN

As specified in section 3, a stratified subsample of 100 observations will be taken of all of the observations above 0.85%. In Figure 7a the distribution of the different lengths present in the data for customers between 10 and 90 touchpoints is presented, in Figure 7b for the 10.000 highest predictions to have clicked and in Figure 7c of the stratified subsample of 100. Running this sample took 107 minutes, which comes down to about a minute per user. It can be observed that the distribution for the different subsamples behaves in the same way, only in Figure 7c less smooth, which is expected as it is heavily downsampled. A note with this downsampling is that this is not based on any of the other features of the data. The other features which are present or dominant might differ, however since we are also focussing on a specific subsample of the data, those with a high prediction of clicking, this is expected.

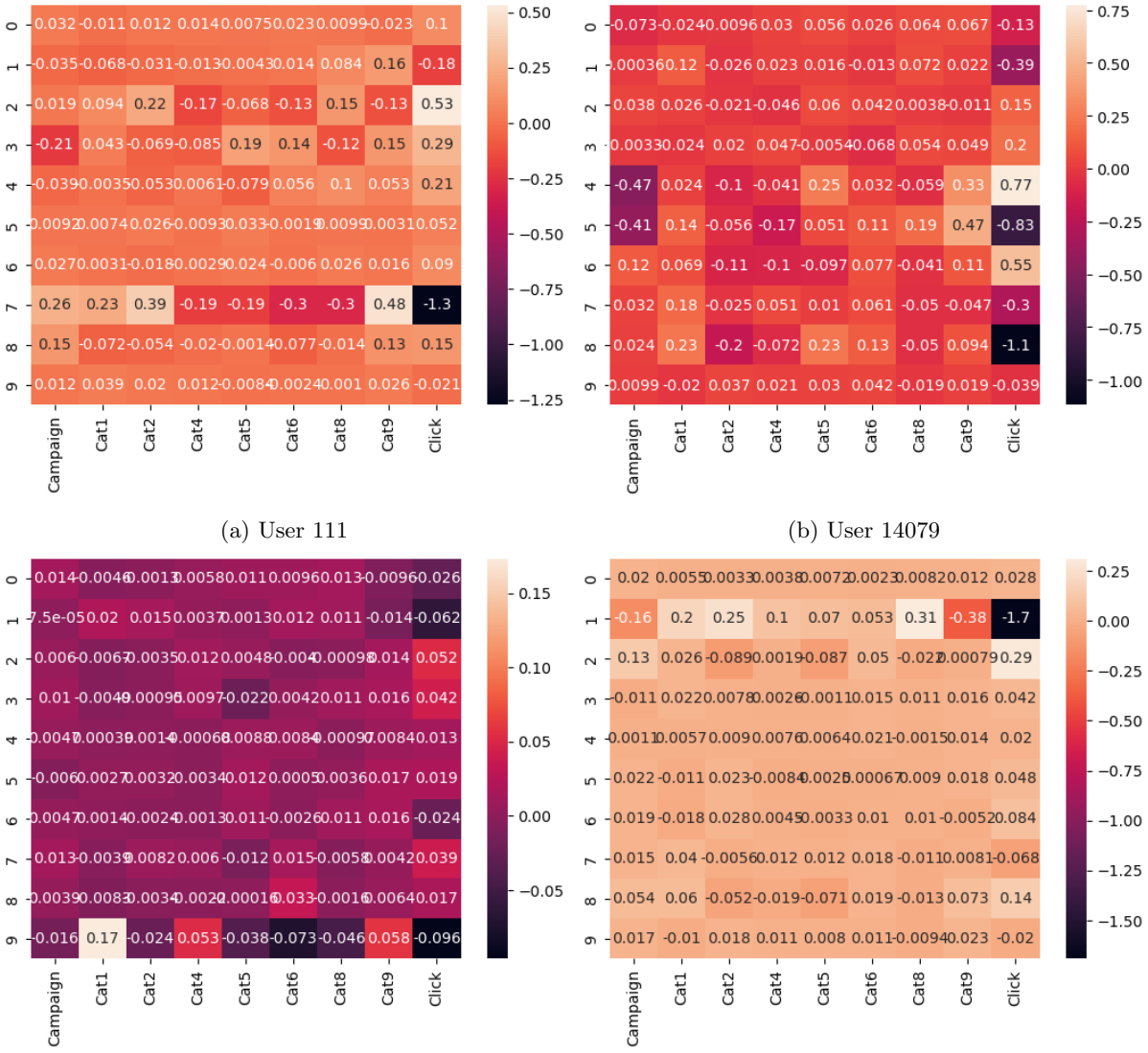


(a) Full Data (n = 243.986) (b) First subsample (n = 10.000) (c) Final subsample (n = 100)

Figure 7: Histograms of Distribution of Touchpoint Lengths for (a) The Full Data, (b) The First Subsample and (c) The Final Subsample

After propagating backwards towards the network and finding the relevance scores, heatmaps are generated based on the scores. In Figure 8 four sample heatmaps are shown. The starting point for the heatmap is the image constructed of the variables and paths of a user, as discussed in Section 3. The different touchpoints are represented on the y-axis and the features on the x-axis. Every cell in the heatmap represents a feature from a specific touchpoint. The brighter a cell is, the more relevant the variable is for clicking and vice versa for dark cells. It can be observed that the relevance scores show different behaviour for different users, which implies that for different users, different variables were important for conversion.

It can be observed that the heatmaps for the different sample customers are quite scattered around the journeys and it is therefore interesting to see at which coordinates the highest relevance scores for the different customers are found. From this finding, it can be deducted



(a) User 111

(b) User 14079

(c) User 17115

(d) User 24736

Figure 8: Four illustrative heatmaps from users with touchpoints on y-axis and features on x-axis. The brighter a certain cell is, the more relevant the specific feature is.

which variables or at which path, the customer is most likely to click. These findings are presented in Table 6 and 7.

Touchpoint	1	2	3	4	5	6	7	8	9	10
Frequency	28%	10%	9%	7%	9%	9%	7%	6%	13%	2%

Table 6: Relative Frequences Most Relevant Touchpoints

Category	Campaign	1	2	4	5	6	8	9	Click
Frequency	27%	1%	2%	0%	2%	6%	2%	2%	58%

Table 7: Relative Frequences Most Relevant Categories

What immediately can be observed is that for 58% of the users, the *click* of a previous

touchpoint gave the highest relevance score. Whether or not someone clicked before is thus a good indication for clicking in the future. On its own, this will not be seen as an interesting finding, however, in combination with the findings of Table 6 more insights can be drawn. Also, the type of campaign had for 27% of the users the highest relevance score. The highest relevance scores are found in the 10^{th} and 9^{nd} to last touchpoint. Putting more emphasis on these touchpoints for this specific group of customers can thus be beneficial to advertisers. The specific values that ranked highest among the different categories have also been evaluated, however, due to the limited number of observations, 100, no obvious value stood out.

When all of the heatmaps for the different customers are added in a single picture, an overview will be given of how the different cells behave in contrast to each other. This overview is given in Figure 9.

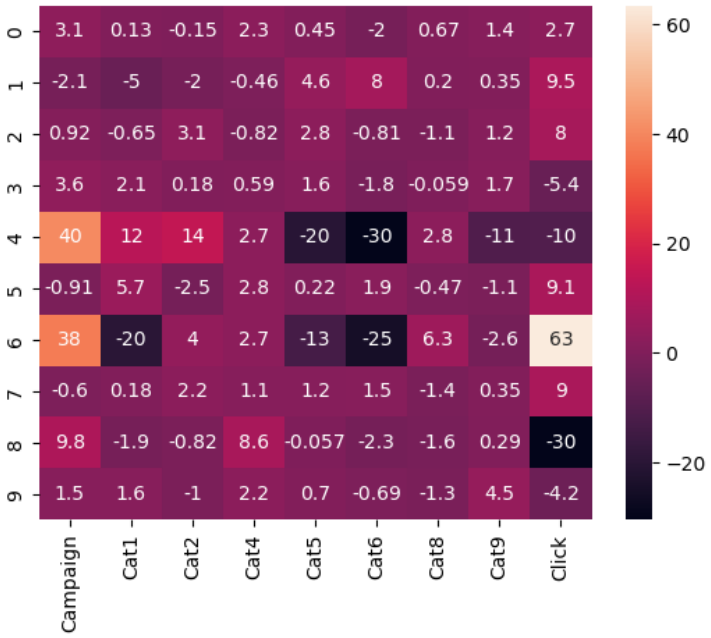


Figure 9: Sum of all heatmaps from the subsample of 100 users

The values with the highest and lowest relevance are interesting for an advertiser to focus on. The overview can, however, give a distorted view since it could be that some variables are extremely positive for one and negative for the other, which would cancel each other out and give relevances close to zero. A min-max normalization is therefore applied to the heatmaps and the result can be found in Figure 10.

As the scores are normalized over the columns, the relation within a touchpoint is a bit distorted. The normalization is more interesting to compare the relevance scores over the touchpoints. The aggregated heatmap sheds insight into which features to focus on at specific points in a clickstream. It can be observed that some have more importance in earlier touchpoints,

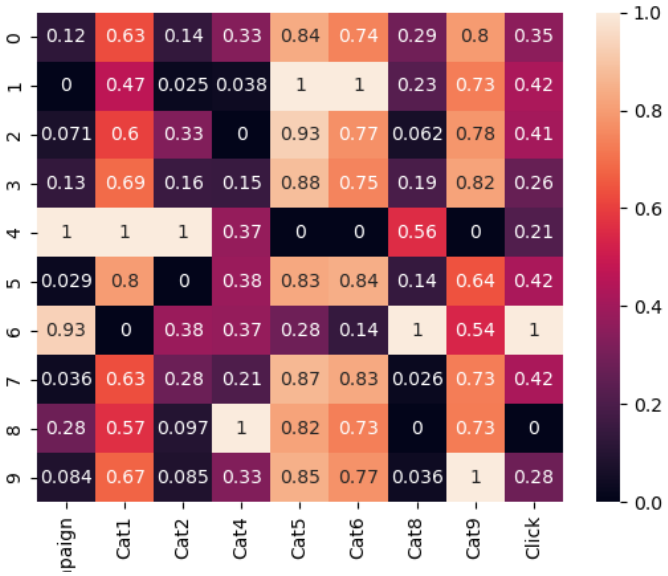


Figure 10: Sum of all heatmaps from the subsample of 100 users, normalized

whereas others are more important in the later touchpoints.

6 Conclusion

In this thesis, a novel method is proposed to open the "black box" associated with deep learning and the attribution problem. The idea is, without making assumptions for the last ten touchpoints, to find what drove a customer to click. The method that is going to be used is a Convolutional Neural Network after which Layerwise Relevance Propagation is used to explain the findings of the network. In previous literature, the CNN has been tested for structured data, only research on sequential data is missing. This research aims to fill that gap.

To evaluate the results, the Criteo dataset is benchmarked against two more industry standard approaches, the LTA and the Simplified Shapley Value approach. Results show both benchmarks fail to provide accurate predictions and that the CNN is able to fit on the given dataset. It only fails to generalize thereafter and overfits. More complexity to the model implied better performance measures, with as a downside an increasing computing time. The characteristics of the data are different from the characteristics of the image data previously used with a CNN, therefore rapidly expanding the number of layers would not be the best idea. Therefore, the choice was made to stop after 3 layers, since the improvements found at this level were not significant enough.

Since the goal of this thesis is to explain the findings and not find the best-generalised model, a simple model is hereafter used for the LRP. This simple model only had one convolutional

layer which contained five filters. The reason behind this is that the LRP with these settings already took about 1 minute per user. The findings of the LRP show that for the most sure findings of the network, it is able to generate insights on a customer level and also on a group level. These insights can hereafter be used by advertisers to better tailor their advertisements for specific users.

Referring back to the original questions asked in Section 1, *(i)* the 2D-CNN can accurately determine which users click on an online advertisement. A note with this finding is that the model is not able to generalize yet, with the current set-up it will only fit the data. Then for *ii*, LRP can be used to explain the findings from the 2D-CNN. Due to computational issues, a simpler model was used to prove this for a subset. With more computing power, LRP can be performed on more complex models and for larger datasets. For the sake of this thesis, it was only relevant to prove it worked.

The biggest limitation follows hereof, is the anonymized nature of the dataset. An interesting field of further research would therefore be to conduct this research with non-anonymized data. When the relations between categories are better known, with clear distinctions between those that are objectively closer and farther apart, the network its ability to fit new data can be enhanced. This can be done by placing categories that are more or related, close to each other, such that filters can be made more effectively. Value could also be added by exploring other factors for a more effective CNN, potentially by integrating elements inspired by the Dual-attention Recurrent Neural Network (Ren et al., 2018). This network assumes that older touchpoints are less relevant and that might need to be adjusted, following the result of the aggregated heatmap.

References

Ahmed, F., Asif, M., Saleem, M., Mushtaq, U. F. & Imran, M. (2023). Identification and prediction of brain tumor using vgg-16 empowered with explainable artificial intelligence. *International Journal of Computational and Innovative Sciences*, *2*(2), 24–33.

Albawi, S., Mohammed, T. A. & Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (icet)* (pp. 1–6).

Anderl, E., Becker, I., Von Wangenheim, F. & Schumann, J. H. (2016). Mapping the customer journey: Lessons learned from graph-based online attribution modeling. *International Journal of Research in Marketing*, *33*(3), 457–474.

Arava, S. K., Dong, C., Yan, Z., Pani, A. et al. (2018). Deep neural net with attention for multi-channel multi-touch attribution. *arXiv preprint arXiv:1809.02230*.

Arras, L., Montavon, G., Müller, K.-R. & Samek, W. (2017). Explaining recurrent neural network predictions in sentiment analysis. *arXiv preprint arXiv:1706.07206*.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R. & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, *10*(7), e0130140.

Barbiero, P., Squillero, G. & Tonda, A. (2020). Modeling generalization in machine learning: A methodological and computational study. *arXiv preprint arXiv:2006.15680*.

Berman, R. (2018). Beyond the last touch: Attribution in online advertising. *Marketing Science*, *37*(5), 771–792.

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly weather review*, *78*(1), 1–3.

Chen, H., Perozzi, B., Al-Rfou, R. & Skiena, S. (2018). A tutorial on network embeddings. *arXiv preprint arXiv:1808.02590*.

Diemert Eustache, Meynet Julien, Galland, P. & Lefortier, D. (2017). Attribution modeling increases efficiency of bidding in display advertising. In *Proceedings of the adkdd and targetad workshop, kdd, halifax, ns, canada, august, 14, 2017* (p. To appear). ACM.

Digital Advertising Spend 2022 the Netherlands. (2023, 5). Retrieved from `https://view.deloitte.nl/TMT-AdSpendStudy.html`

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, *36*(4), 193–202.

Gneiting, T. & Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, *102*(477), 359–378.

Goutte, C. & Gaussier, E. (2005). A probabilistic interpretation of precision, recall and f-score,

with implication for evaluation. In *European conference on information retrieval* (pp. 345–359).

Greenberg, E. (2003). What are cookies. , *33*, 76. doi: 10.1097/00152193-200306000-00059

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, *77*, 354-377. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0031320317304120` doi: https://doi.org/10.1016/j.patcog.2017.10.013

Hama, N., Mase, M. & Owen, A. B. (2022). Deletion and insertion tests in regression models. *arXiv preprint arXiv:2205.12423*.

Hastie, T. (2020). Ridge regularization: An essential concept in data science. *Technometrics*, *62*(4), 426–433.

Ho, Y. & Wookey, S. (2019). The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE access*, *8*, 4806–4813.

ILSVRC2014 Results. (2014). Retrieved from `https://image-net.org/challenges/LSVRC/2014/results`

Ji, W., Wang, X. & Zhang, D. (2016). A probabilistic multi-touch attribution model for online advertising. In *Proceedings of the 25th acm international on conference on information and knowledge management* (pp. 1373–1382).

Kingma, D. P. & Ba, J. (2017). *Adam: A method for stochastic optimization.*

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

Kurdi, B. (2022). The role of digital marketing channels on consumer buying decisions through ewom in the jordanian markets. *International Journal of Data and Network Science*. doi: 10.5267/j.ijdns.2022.7.002

Lakshmanan, V., Robinson, S. & Munn, M. (2013). Machine learning design patterns?: solutions to common challenges in data preparation, model building, and mlops. 390.

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W. & Müller, K.-R. (2019). Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, *10*(1), 1096.

Lawless, C. & Günlük, O. (2020). Fair and interpretable decision rules for binary classification. In *Neurips workshop.*

LeCun, Y., Boser, B., Denker, J., Henderson, D., Hubbard, W. & Jackel, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, *2*.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lieli, R. P. & Hsu, Y.-C. (2019). Using the area under an estimated roc curve to test the adequacy of binary predictors. *Journal of Nonparametric Statistics*, *31*(1), 100–130.

Lovett, J. (2009). A framework for multicampaign attribution measurement. *Forrester Research. February*, *19*.

Montavon, G., Binder, A., Lapuschkin, S., Samek, W. & Müller, K.-R. (2019). Layer-wise relevance propagation: an overview. *Explainable AI: interpreting, explaining and visualizing deep learning*, 193–209.

Nisar, T. M. & Yeung, M. (2018). Attribution modeling in digital advertising: An empirical investigation of the impact of digital sales channels. *Journal of Advertising Research*, *58*(4), 399–413.

Niu, Z., Zhong, G. & Yu, H. (2021). A review on the attention mechanism of deep learning. *Neurocomputing*, *452*, 48–62.

Pang, B., Lee, L. & Vaithyanathan, S. (2002, July). Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 conference on empirical methods in natural language processing (EMNLP 2002)* (pp. 79–86). Association for Computational Linguistics. Retrieved from `https://aclanthology.org/W02-1011` doi: 10.3115/1118693.1118704

Ramachandran, P., Zoph, B. & Le, Q. V. (2017). Searching for activation functions. *CoRR*, *abs/1710.05941*. Retrieved from `http://arxiv.org/abs/1710.05941`

Ren, K., Fang, Y., Zhang, W., Liu, S., Li, J., Zhang, Y., ... Wang, J. (2018). Learning multi-touch conversion attribution with dual-attention mechanisms for online advertising. In *Proceedings of the 27th acm international conference on information and knowledge management* (pp. 1433–1442).

Rios, A., Gala, V., Mckeever, S. et al. (2020). Explaining deep learning models for structured data using layer-wise relevance propagation. *arXiv preprint arXiv:2011.13429*.

Rufibach, K. (2010). Use of brier score to assess binary predictions. *Journal of clinical epidemiology*, *63*(8), 938–939.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the ieee international conference on computer vision* (pp. 618–626).

Shao, X. & Li, L. (2011). Data-driven multi-touch attribution models. In *Proceedings of the 17th acm sigkdd international conference on knowledge discovery and data mining* (pp.

258–264).

Shapley, L. S. (1997). A value for n-person games. *Classics in game theory*, *69*.

Sharma, S., Sharma, S. & Athaiya, A. (2017). Activation functions in neural networks. *Towards Data Sci*, *6*(12), 310–316.

Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Singal, R., Besbes, O., Desir, A., Goyal, V. & Iyengar, G. (2019). Shapley meets uniform: An axiomatic framework for attribution in online advertising. In *The world wide web conference* (pp. 1713–1723).

Singhal, S., Kumar, H. & Passricha, V. (2018). Prediction of heart disease using cnn. *Am Int J Res Sci Technol Eng Math*, *23*(1), 257–261.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T. & Qin, B. (2014). Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 1555–1565).

Vidaurre, D., Bielza, C. & Larranaga, P. (2013). A survey of l1 regression. *International Statistical Review*, *81*(3), 361–387.

Wang, X., Yu, L., Ren, K., Tao, G., Zhang, W., Yu, Y. & Wang, J. (2017). Dynamic attention deep model for article recommendation by learning human editors' demonstration. In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining* (pp. 2051–2059).

Wu, H., Huang, A. & Sutherland, J. W. (2022). Layer-wise relevance propagation for interpreting lstm-rnn decisions in predictive maintenance. *The International Journal of Advanced Manufacturing Technology*, 1–16.

Yu, D., Wang, H., Chen, P. & Wei, Z. (2014). Mixed pooling for convolutional neural networks. In *Rough sets and knowledge technology: 9th international conference, rskt 2014, shanghai, china, october 24-26, 2014, proceedings 9* (pp. 364–375).

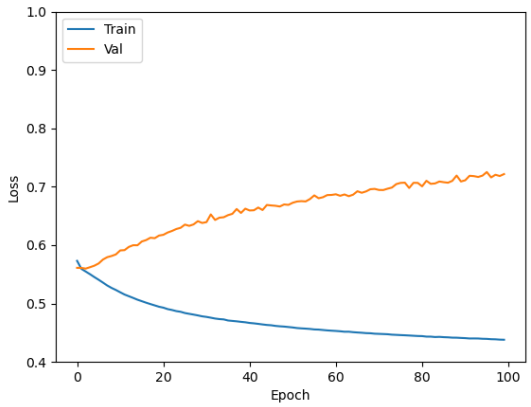
Zeiler, M. D., Krishnan, D., Taylor, G. W. & Fergus, R. (2010). Deconvolutional networks. In *2010 ieee computer society conference on computer vision and pattern recognition* (pp. 2528–2535).

Zhao, K., Mahboobi, S. H. & Bagheri, S. R. (2018). Shapley value methods for attribution

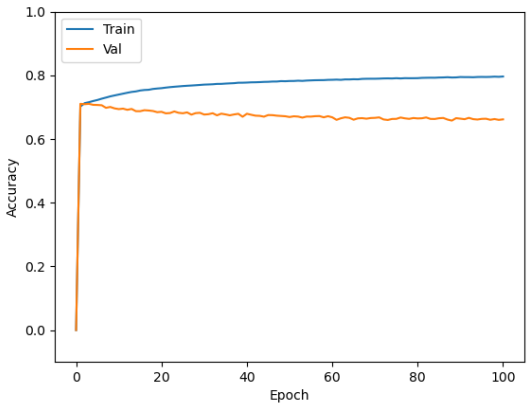
modeling in online advertising. *arXiv preprint arXiv:1804.05327*.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A. & Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2921–2929).

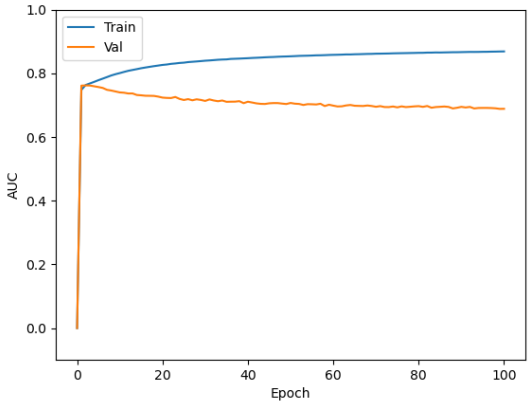
A Performance Measures Simple One-Layer 2D-CNN Per Epoch



(a) Loss



(b) Accuracy



(c) AUC

Figure 11: Loss, Accuracy and AUC for both Train and Validation set.

B Programming Code

A brief description of the code used in this thesis can be found below. All of the scripts are in the Jupyter Notebook format and written to be used independently. The versions of the packages that were used are in Appendix C.

LTA.ipynb, contains the code for both the LTA: Campaigns model and the LTA: Campaigns and Categories model. Also contains the code to create the graphs, were the results from Shapley.ipynb and CNN.ipynb are used.

Shapley.ipynb, contains the code for the Simplified Shapley Value. The code uses the SimplifiedShapleyAttributionModel.py which was obtained from *https://github.com/ianchute/shapley-attribution-model-zhao-naive*.

CNN.ipynb, contains the different CNN models which were tested. Different techniques

tried can be found in the comments. Creating one hyperparameter optimization was unfortunately not feasible due to computational issues.

LRP.ipynb, contains the process to perform the LRP on a created 2D-CNN. The backpropagation in this process is specifically made for the chosen one-layer 2D-CNN.

C Packages Used

The versions of the following packages are used throughout this thesis. Note that Keras is an interface for the TensorFlow library.

Package	Version
keras	2.13.1
tensorflow	2.13.0rc1
polars	0.18.9
pandas	2.0.2
numpy	1.24.3
scikit-learn	1.2.2

Table 8: Overview of packages