Tartu University

Faculty of Science and Technology

Institute of Technology

Nathan Mison

Using Machine Learning to Measure Digital Audiences

Master's thesis (30 EAP)
Robotics and Computer Engineering

Supervisor:

Msc Josep Badia
Phd Kairit Sirts

Tartu 2023

Resümee/Abstract

Masinõppe kasutamine digitaalse vaatajaskonna mõõtmiseks

Viimastel aastatel on masinõppe valdkond olnud tunnistajaks enneolematule huvi ja rakenduste kasvule erinevates valdkondades. Vaatajaskonna mõõtmise liider Kantar Media eesmärk on rakendada masinõpet publiku mõõtmise tehnikate täiustamiseks ja revolutsiooniliseks muutmiseks. See lõputöö demonstreerib korduvate närvivõrkude ja tähelepanumehhanismide rakendamist. Vaatamata olemasolevatele vaatajaskonna mõõtmise lahendustele on neil piiranguid kõrvalekallete andmete igakülgsel käsitlemisel. See lähenemisviis näitab, et sügava õppimise võimendamine annab märkimisväärse testi täpsuse 97 protsenti. Siiski märgitakse ka, et teatud kõrvalekalded kujutavad endast püsivaid ennustamisprobleeme.

CERCS: P176 Tehisintellekt, P175 Informaatika, süsteemiteooria [4]

Märksõnad: digitaalsed vaatajaskonnad, masinõpe, süvaõpe, korduvad närvivõrgud, tähelepanu

Using Machine Learning to Measure Digital Audiences

In recent years, the field of machine learning has witnessed an unprecedented surge in interest and application across diverse domains. Kantar Media, a leader in audience measurement, aims to apply machine learning to refine and revolutionize audience measurement techniques. This thesis demonstrates the application of recurrent neural networks and attention mechanisms. Despite existing solutions in audience measurement, they exhibit limitations in comprehensively addressing outlier data. The approach demonstrates that leveraging deep learning yields a remarkable test accuracy of 97 percent. However, it is also noted that certain outliers present persistent predictive challenges.

CERCS: P176 Artificial intelligence, P175 Informatics, systems theory [4]

Keywords: digital audiences, machine learning, deep learning, recurrent neural networks, attention.

Contents

List of Figures

1 Introduction

1.1 Aims of the Project

Kantar Media is a multinational company performing a multitude of services worldwide, of which one task is to measure online audiences for video traffic [1] [3]. A primary component of this measurement process involves the deployment of a focal meter, installed on WIFI routers. The focal meter is situated in residences volunteered by individuals presumed to be representative of the broader population of a given country. This approach is designed to afford a nuanced understanding of the audience composition within a designated geographic context. The acquired data, serves as a the basis for determining advertising pricing strategies [2], exemplifying the utility of this method in contributing to the commercial landscape.

In addressing this challenge, the optimal resolution involves the focal meter intercepting all

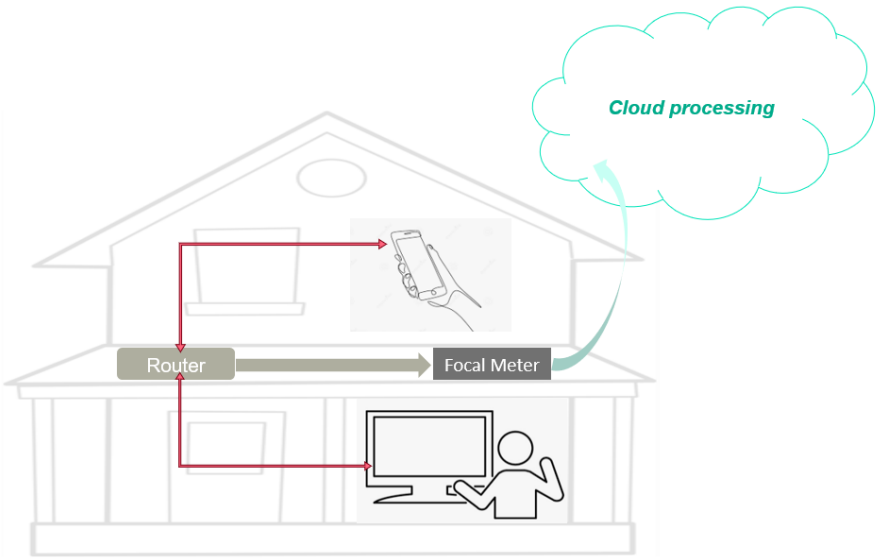

Figure 1.1: Illustration of the task facing Kantar [24]

data packets transmitted through the WIFI network. However, the focal meter's capability is constrained to observing only the outbound traffic from the router, and does not have access to the contents of packets transmitted with the https protocol. As a consequence, only limited information, such as the domain name and quantity of certain packets, is discernible. The primary objective is to find the destination of the address based on the available domain name data. Although Kantar currently employs a method to identify the end servers, its accuracy is compromised and susceptible to obsolescence due to various factors. Consequently, Kantar is

actively exploring alternative approaches, with machine learning emerging as a prominent candidate for replacing the existing methodology.

My task is to continue the work on this project previously handled by Josep Badia, my supervisor at Kantar,

1.2 Challenges to Overcome

Notably, not all visited addresses are pertinent for Kantar, as the company is interested in measuring audiences for a specific set of streaming services, including but not limited to Netflix, Amazon, and YouTube, among others, totaling 25 websites of interest. Currently, Kantar employs a solution comprising a series of conditional statements, utilizing if else loops. Each domain name of interest is initially associated with a distinct tag, manually linked to the server corresponding to each country in which Kantar operates. Subsequently, when a tag is identified, the focal meter quantifies the volume of data packets traversing through it. If the packet count surpasses a predefined threshold indicative of substantial video traffic, it is assumed that video content is likely being accessed on the identified domain.

Kantar faces several challenges in its pursuit of an effective solution for online audience measurement [6] [5]. Firstly, the variation in buffering behaviors across devices introduces complexity [17] [18], as certain applications may exhibit different domain names during similar time intervals, particularly notable in the case of iOS devices, which pose an important challenge. Furthermore, the intricacies arising from shared servers, as exemplified by Amazon's ownership of both Twitch and Prime Video, necessitate a nuanced approach to accurately distinguish between the two entities.

Another main consideration is the exclusion of background applications from the analysis. The goal is to identify the content actively viewed rather than to enumerate all open applications. Ensuring uniform accuracy across all classes of interest is most important, as discrepancies in classification accuracy among different websites could undermine client satisfaction. The challenge is to achieve consistent and high accuracy levels for each website classification. Additionally, the dynamic nature of web hosting infrastructures, with multiple servers distributed globally and potential variations based on geographic zones, makes developing a solution that exhibits robustness across diverse global locations an important task.

While Kantar has made significant improvements in addressing several challenges, a notable obstacle remains in the discrepancy of protocols, particularly with iOS devices, leading to suboptimal outcomes in certain instances. The challenge lies in achieving consistent accuracy across different devices, with iOS devices posing specific difficulties. Another concern is the assessment of Kantar's true accuracy, which is primarily conducted in controlled lab conditions. The standard testing protocol involves individuals documenting their viewing activities on a device, and then evaluating whether Kantar's solution accurately discerns the content being accessed. However, the controlled nature of lab settings may not entirely replicate real life conditions. It may introduce potential inaccuracies related to factors such as background applications, rapid app navigation, and improper device shutdown. Kantar's self estimated working accuracy ranges between 80 to 90 percent for most classes across diverse devices. However,

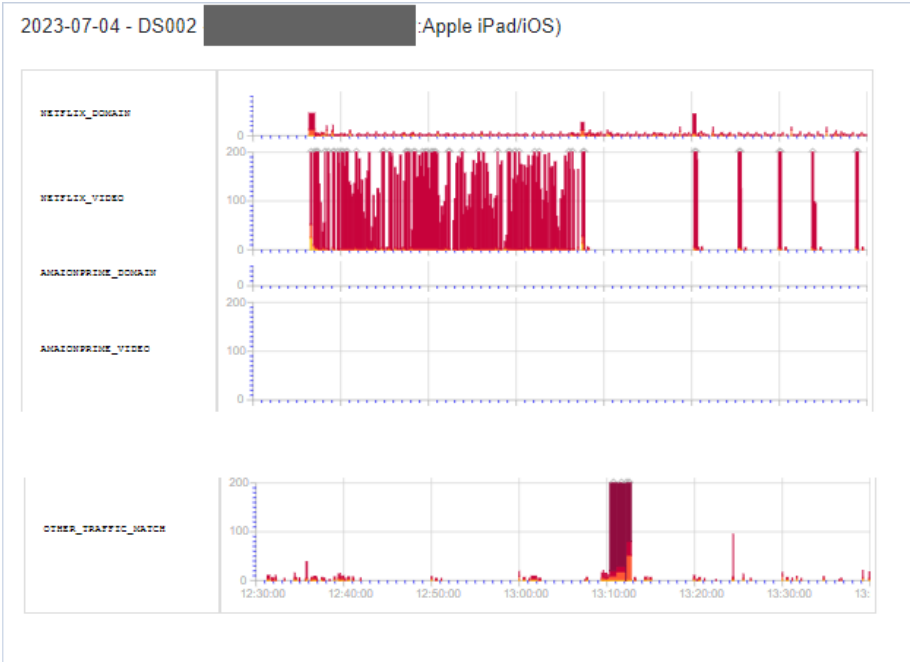

Figure 1.2: Example of a Netflix higly aggressive bufferization on iOS. The first content is streamed "normally", while the second is bufferized aggressively

challenges persist, as accuracy can plummet to as low as 30 percent for certain classes, such as Netflix on iOS devices.

The primary objective of this project is the development of a machine learning model capable of associating a specific class with each timestamp throughout the day, thereby discerning the content transmitted through the router at any given time. The ultimate goal is to achieve accurate predictions for multiple classes consistently across a diverse range of devices and countries. It is imperative that the model demonstrates parity in accuracy across all classes, ensuring uniform performance irrespective of the device or geographic location. This means not only accurate identification of classes but also the establishment of a robust and adaptable framework capable of handling the unique challenges posed by various devices and countries. The successful realization of this aim holds the potential to significantly enhance Kantar's online audience measurement capabilities, and leading to a more comprehensive and accurate understanding of user behavior across the digital landscape.

1.3 Related Works

Measuring online audiences presents an intricate challenge, with various methodologies currently in use, including those employed by Kantar. Surprisingly, the application of machine learning to this domain remains largely unexplored, with limited or no published literature on the subject. To address this gap, a perspective is proposed that considers the data acquired through the focal meter and its associated labels as constituting a simple language.

Furthermore, it is crucial to acknowledge the temporal constraints inherent in the data, where

Figure 1.3: Example of a background app(Netflix) still generating data on the network while a parallel one is playing (Twitch). Also note the iOS Netflix bufferization pattern, different from the first example.

preceding iterations may exert influence on subsequent results, presenting a temporal dependency akin to a time series. Consequently, this investigation aligns with scientific literature encompassing natural language processing, NLP and time series analysis. By using techniques from these domains, we aim to redefine the problem, viewing online audience measurement as a language understanding task influenced by temporal dynamics.

In NLP literature, several influential methodologies have risen to solve language related tasks. Transformer architectures, leveraging attention mechanisms, have gained widespread adoption [9] [15]. Recurrent Neural Networks (RNNs) have also shown to be a prominent choice [10], [12], [14], and hybrid approaches, combining both attention and RNNs, have been explored [13]. In the context of time series analysis, RNNs are used [7], often in conjunction with attention mechanisms [16] [26]. The consensus from these works suggests that the combination of RNNs and attention mechanisms leads to superior performance, with each component complementing the strengths of the other. Moreover, employing sequence to sequence tasks has demonstrated significant efficacy [10], [21].

Given this landscape, the primary objective of this project is to start with a sequential RNN framework and subsequently enhance its performance by adding an attention mechanism. By using insights from both NLP and time series literature, this approach seeks to use the strengths of sequential modeling and attention mechanisms to achieve an effective solution for online audience measurement.

2 Data

2.1 Collecting the Data

The dataset used by Kantar and in this project comprises two distinct components: data obtained from the focal meter and corresponding ground truth data, both collected independently. This data acquisition process took place within a controlled laboratory environment, where an individual systematically used with various streaming services on multiple devices throughout the day. The focal meter, linked to the WIFI router, recorded all outgoing data packets emerging from these interactions.

The ground truth data is derived from meticulous documentation by the individual, who recorded the precise start and finish times of app usages. These diaries are the most reliable approximation of ground truth, providing unequivocal insights into the activities occurring on a given device at specific points in time. With it, the raw focal meter data contains the entirety of outgoing internet packages, capturing the digital footprint of the user's interactions. This dual source dataset, combines observed focal meter data with meticulously recorded ground truth diaries, forms the basis for the development and evaluation of the machine learning models aimed at predicting online user behaviors.

2.2 Data from the Focal Meter

The primary component of the dataset comes from the raw data captured by the focal meter. Mostly presented in the form of domain names to the server, this dataset includes additional contextual information. The key attributes are: DateAud: The date on which the data was collected. TsLocal: The timestamp in local time corresponding to when the data was recorded. TsUtc: The timestamp in Coordinated Universal Time (UTC). URL: The domain name under consideration, the prediction target. Mc Address: The address of the device generating the request. Meter ID: The identifier associated with the focal meter. Whitelist Tag: A tag generated from the URL, constituting the primary method utilized by Kantar to find the corresponding website. Duration: The duration of the request in seconds. Tcp Data, Tcp Sig, Tcp Ssl, and Udp: Counts of the number of packages received in TCP and UDP protocols [19] [20].

During testing conditions, data collection contains every output from the WIFI network. In these cases, the majority of the data is expected to be categorized as "other" as most of the time devices are either inactive or doing an activity not on the list. Also, when recording a date, data spanning from 2:30 am on that day to 2:30 am on the following day is considered as part of that particular day.

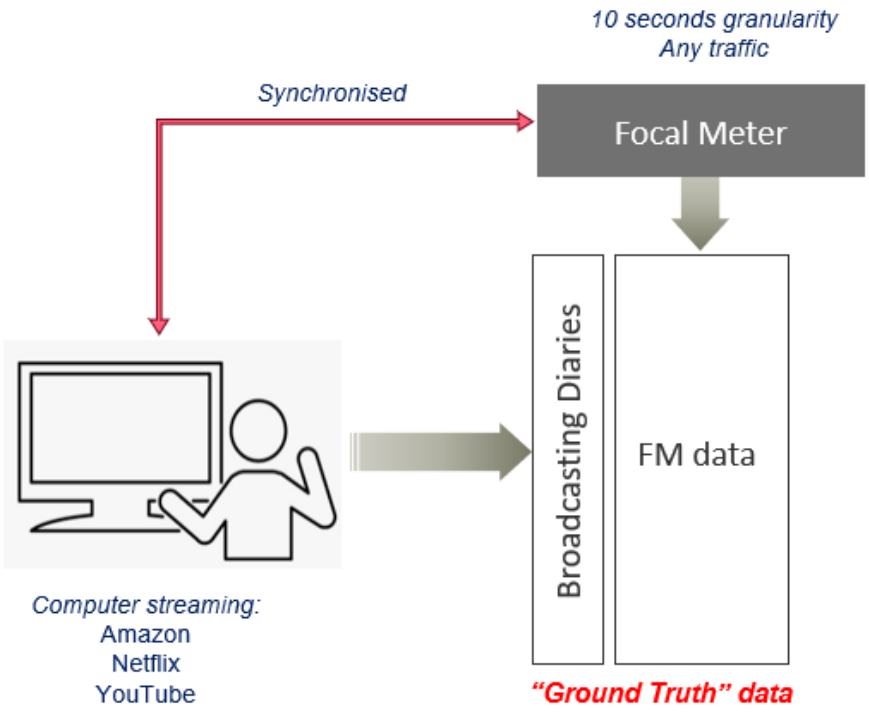

Figure 2.1: Explanation on how the data is collected [24]

2.3 Ground Truth Data

Obtaining accurate ground truth data poses a significant challenge due to the difficulty of replicating real life conditions. It is the complement to the focal meter data.

Similar to the focal meter dataset, the ground truth data includes the following attributes: DateAud: The date on which the data was recorded. McAddress: The MAC address of the device involved in the activity. StartTime and EndTime: The local times at which the activity commenced and concluded. Activity: The specific activity undertaken by the device during the recorded period.

The range of activities includes those falling within the desired classes as well as a general "other" category, encompassing any activities not explicitly desired including the absence of activity. This inclusive categorization ensures comprehensive coverage, even when no discernible activity occurs. Same as for the focal meter data, the labeling spans the entire day, from 2:30 am to 2:30 am the following da. All activities are set into the designated classes or as "other".

2.4 The Transforms

The original code [24] implemented these transforms using SQL databricks, but I have since redone all of them using pandas.

Preparing the data for effective utilization by a neural network requires a series of transformations. The initial step involves separating the data based on MAC address, as each device's data is independent. This separation is therefore essential, and enables the option for individualized training on each device in the next stages.

Determining the desired resolution of the end result is an important consideration. While the ground truth data is labeled with up to one second precision, classifying every single second may not be necessary or optimal. therefore, an arbitrary choice is made to divide each day into intervals of ten seconds for every MAC address. This results in 8,640 intervals per day. Each interval is then labeled based on the most present activity observed during that period. Consequently, every timestamp within these ten second intervals is assigned a class corresponding to its label in the ground truth data.computational efficiency.

Regarding the recorded data, the comprehensive URL is streamlined to focus only on the constituent words within it. For instance, the URL "https://edge.microsoft.com/" is disassembled into two words: "edge" and "Microsoft." Simultaneously, other parameters such as TCP data, duration, and the like are transformed into words; for instance, a TCP data value of 1 is represented as "tcpdata01." Integrating all these words produces a representation containing all information related to a package received by the focal meter, resulting in a phrase.

To organize and synchronize the data, the timestamp in Coordinated Universal Time is discarded, aligning with the local time annotations used in the diaries. The next step involves a merging process in which both the ground truth data and the raw data from the focal meter are merged. The raw data is organized into ten second intervals, with all words from the requests within each interval merged into a single phrase. Subsequently, the ground truth and raw data, both now organized into ten second intervals, can be merged based on their temporal alignment. Each data point collected from the focal meter is prefixed with a corresponding label.

Considering the buffering phenomenon associated with video playback, where a single request may signify that the video was being watched for an extended duration, it becomes necessary to incorporate temporal context for effective model training. To address this, for each timeband, the phrases from the ten preceding timebands and the one immediately following are concatenated. Consequently, each timeband contains its own phrase along with information from the twelve surrounding timebands, providing a total temporal span of two minutes and twenty seconds of focal meter data within a single timeband. Therefore enabling to capture the necessary context for predicting labels accurately.

In the final preparation step, a dictionary is constructed. Since RNNs operate with numerical inputs, all unique words present in the concatenated phrases are added to the dictionary and indexed. This process facilitates the conversion of words to numerical indices and vice versa. The resulting dataset, is now ready for training machine learning models to predict user activities based on focal meter data.

3 Model Architecture

For the model the first approach is too use an RNN for sequence to sequence tasks. As mentioned before this method is able to perform the task required [10], [12], [14] with a reasonable amount of computation power. In this projects case, calling the model sequence to vector would be more accurate as the classification is an output. To achieve this the Pytorch deep learning framework is used. Originally created by Josep at Kantar, I fine-tuned the model and meticulously reviewed all the choices to ensure they aligned with scientific literature.

3.1 The Model

The Embedding Layer is the first layer in the network. It transforms discrete, categorical data into continuous vectors. The layer takes in token indices, and maps each token to a dense vector of a specified size. This mapping enables the model to interpret the tokens in a way that captures semantic similarities and relationships. For example, words with similar meanings tend to have closer embeddings in this high dimensional space. An embedding dimension of 128 is chosen, enabling to capture enough information while also keeping the computational load in check.

Following the embedding layer, the RNN layers are the core of the model. An RNN is particu-

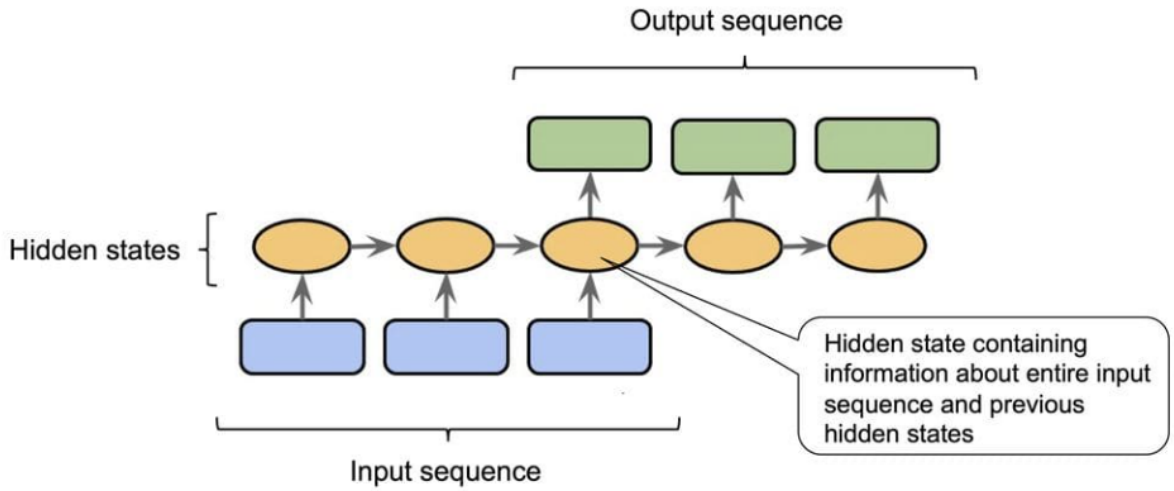

Figure 3.1: Architecture of an RNN encoder decoder architecture for sequence to sequence task, in the projects case due to the classification task, the output is not a sequence [28]

larly suited for processing sequences, as it maintains an internal state that captures information about the parts of the sequence it has seen so far. GRU cells are chosen as the perform as well as LSTM cells while needing less computation [8]. The model has 8 layers of GRU cells, enhancing its ability to capture more complex patterns in the data at a cost of a potentially exploding gradient. The hidden size of 128 is chosen, as it is enough to capture the dependencies in the data. The RNN is bidirectional, processing the sequence in both forward and backward directions. This bidirectionality allows the model to have context from both past and future data points, providing a more comprehensive understanding of the sequence.

After going through the RNN layers, the output needs to be transformed into a more usable form for classification end goal. This is where the linear layers come in. These layers are fully connected neural network layers that can transform the RNN output into a vector that corresponds to the desired output format, in this case, the number of classes in the classification task. The final output layer then maps this transformed output to the final output space. This would actually make the model more of a sequence to sequence to sequence task

Finally, to improve the model's generalization capability and reduce overfitting, dropout layers are used [25] [27]. These layers randomly zero out a fraction of the output from a layer. This random omission of units during training forces the network to not rely on any single input and hence to learn more robust features that are useful in general.

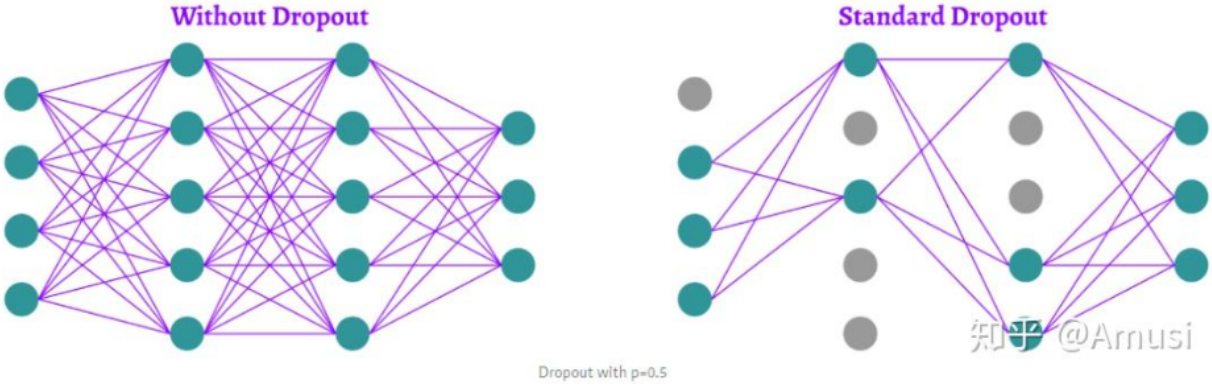

Figure 3.2: Example of the activation of a simple neural network with and without dropout [29]

3.2 Training

For training a negative log likelihood loss function is used as it performs well on classification tasks [22], as it outputs log probabilities across a predefined set of classes. The function offers flexibility for adjusting the learning rate, batch size.

The training function always prepares the dataset for training by aligning its size with the specified batch size. During each training epoch, the model is trained on batches of data, and the hidden state is managed to avoid backpropagation through the entire dataset. The Adam optimizer is used as, it is a popular choice in deep learning [23] [25], as it facilitates efficient gradient based parameter updates.

The training function further incorporates dropout regularization to reduce overfitting during

training [25]. The function keeps track of training and validation metrics, including loss and accuracy, across epochs. The inclusion of validation allows for the assessment of the model's performance on an independent dataset.

4 Experimental Setup

Once the machine learning model is developed, the next phase involves training the model on the prepared dataset. During this training process, the primary objectives are to determine the optimal training strategies that yield the best results and, if necessary, to fine tune or modify the model for improved performance.

4.1 Data Splits

For this project, data was collected from four distinct devices with the objective of developing a model capable of predicting the labels for all devices based only on the training from a single device. The dataset contains varying durations of recorded data across these devices: 57 days from a smartphone, 37 days from a PC, 23 days from a smart TV, and 10 days from an iPad. This disparity in data availability presents a challenge, mainly for the smart TV and iPad, as not every class label is represented daily. Consequently, establishing an ideal train test validation split for these devices is problematic. it could potentially result in certain class labels being absent from one or more of the split datasets.

Prior to data splitting, several important considerations must be addressed. Mainly, in a production environment, the model is trained on historical data, therefore necessitating that the test set comprises data from later dates. This approach ensures that the model's predictions are based on preceding training dates. Additionally, the main objective of the project is to facilitate cross device prediction, where a model trained on data from one device is employed to predict outcomes on multiple other devices. Therefore, the true test set actually consists of the data from devices distinct from the one used for training. Consequently permanently an important testing set for every device is permanently available, however since iPad is the most problematic device it is still difficult to assess models trained on its data.

For the PC and smartphone datasets, the high amount of data allows a more generous allocation to the validation and testing sets. Specifically, a 70/15/15 split is implemented for the PC data, assigning 25 dates to the training set, and 6 dates each to the validation and testing sets. This distribution is carefully calculated to ensure diversity in the representation of all class labels across each dataset. In the case of the smartphone data, a comparable distribution strategy is adopted, 25 dates are allocated to the training set, 6 to the validation set, and the remainder to the testing set. It is important to note that expanding the training set to too large dates may pose some problems. Firstly, it significantly increases the training duration. Secondly, a larger training set may inadvertently lead to overfitting, wherein the model not only learns the underlying patterns but also the noise and outliers in the data, which can adversely affect its generalization capability.

For the iPad dataset, a constrained data splitting strategy is adopted due to limited availability of the data. Consequently, 8 dates are designated for the training set, while only a single date is allocated to each of the validation and testing sets. This configuration, while not ideal for ensuring the representation of all class labels in the validation and testing sets, is deemed sufficient. The primary validation of the model's performance is predicated on its ability to predict across different devices. Therefore, having only one date in the testing set can be considered adequate, especially if this date encompasses the 'Netflix' class, which has been identified as particularly challenging. The inclusion of this class in the testing set is crucial to ascertain the model's capability to predict on the iPad data. Regarding the smart TV dataset, a less conservative approach is taken. 17 days are assigned to the training set, and the remaining 6 dates are evenly distributed between the validation and testing sets.

4.2 Hyperparameters

The next phase in improving the model's accuracy involves the optimization of hyperparameters. Given the constraints of time, it is assumed that variances in model performance across different devices due to device specific hyperparameter tuning is negligible. Consequently, all hyperparameter optimization experiments are conducted exclusively on the PC dataset. The optimized hyperparameters identified through this process are then applied to the models for other devices.

The optimal hyperparameters found for the model are: a training duration of 100 epochs, a learning rate set at 1e-6, and a dropout probability of 0.2. Utilizing these parameters, the model, when trained on the PC dataset, achieves an accuracy of 97 percent on its testing set. A notable achievement of this configuration is the absence of overfitting, as evidenced by consistent performance across both validation and testing sets. However, the extended training time of around 12 hours required for 100 epochs poses a practical challenge. To address this, a modified training regimen is proposed, reducing the number of epochs to 20 while adjusting the learning rate to 5e-6. This adjustment typically results in only marginally reduced performance of less then one percent. But it offers a more time efficient approach while still maintaining a high level of accuracy. This compromise between training duration and model performance is required for effective model development as it allows the training of more experiments.

4.3 Further Tests

When evaluating the baseline model, which was trained on PC data, it performed exceptionally well on the PC testing set. However, its effectiveness significantly diminished when applied to predict iPad data. This discrepancy shows the need for additional experimentation to enhance the model's accuracy across all devices. The first step in this process involves assessing the relevance of TCP and UDP packets as features. Given the possibility that these packets might not contribute meaningfully to the model's performance, and could potentially be acting as dead weights. Training the model without these features is possible. This will help determine their actual impact on the model's predictions. Another area of investigation is the temporal context

used in the model. Currently, the model utilizes data from 11 previous iterations and one subsequent iteration for each timestamp. The efficiency of this approach can be re evaluated to see whether simplifying the input to just the data of the current iteration can be enough, therefore reducing the model's complexity without compromising much its performance. To further enhance accuracy, particularly for a RNN architecture, the integration of an attention mechanism can be helpful as mentionned in the Related Works chapter. Attention mechanisms have been proven improve the performance of RNNs by enabling them to focus on the most relevant parts of the input sequence [9] [15] [13], therefore improving their ability to capture long range dependencies and nuances in the data, exactly as in the Netflix outlier case.

5 Results

The current model shows great performances when tested on the same device as its training data, achieving a 0.97 accuracy on the PC test set. This high level of accuracy is also observed when the model is applied to data from the smart TV and the smartphone, with 0.94 accuracy on the TV and 0.96 on the smartphone. However, as anticipated, the model's accuracy declines when predicting data from the iPad, with accuracy falling to 0.86. This decline is further amplified in the case of an outlier date as illustrated in figure 1.2, where accuracy drops sharply to 0.68. This significant decrease highlights the model's struggle to accurately predict certain iPad data, particularly those with outlier characteristics.

5.1 Additional Tests

The initial strategy to improve prediction accuracy on iPad data involves training the model only with data from the iPad. In this setup, the training set comprises of only 8 dates, with one date allocated to the validation and testing sets each. Making sure to include a problematic date in the testing set is important to evaluate the model's robustness in predicting complex scenarios. Employing this approach, the model achieves a 0.90 accuracy on the problematic test date. To better apprehend of the model's performance when trained on iPad data, a series of identical models are trained, with the only variation being the shuffling of dates in the dataset. The aggregated outcomes from these models show an average accuracy of 0.93 on iPad data, which is a notable improvement. However, the accuracy drops to 0.83 for both the smart TV and the smartphone, and further declines to 0.72 for the PC data. These results suggest that training the model only on iPad data, with its short dataset of 8 days, does not provide sufficient learning for the model to fully capture the outlying cases of iPad buffering patterns.

The next solution is therefore to train with both PC and iPad data, with 8 iPad dates in the training set alongside 8 PC dates. When doing this method, the test accuracy on PC is of 84 percent and 86 percent on iPad. However the 86 percent is only on the problematic date so the true overall accuracy is probably slightly higher. It can be noticed that it cannot properly predict PC data, therefore 4 more PC dates are added to the training. This raises the accuracy on PC data to 95 percent and to 85 percent on iPad. The next approach to improve model performance involves training with a merged dataset from both PC and iPad. In this method, the training set contains an equal distribution of 8 dates each from PC and iPad data. This strategy seeks to balance the learning process across both device types. The results from this combined training show a test accuracy of 0.84 on PC data and 0.86 on iPad data. The 0.86 accuracy on the iPad is achieved on the same problematic date, meaning that the true overall accuracy for iPad data might be slightly highers.

The initial model shows limitations in accurately predicting iPad data. To address this, the training set is augmented with 4 dates from the PC data, therefore increasing the total PC data in the training set to 12 dates. This adjustment results in a good improvement in model performance, increasing the accuracy on PC data to 0.95, having an accuracy of 0.85 on iPad data.

In more efforts to improve model performance, two methods were tested. Firstly, using only the current timestamp's Phrase. This approach, focused only on the phrase of the current timestamp and excluded previous and subsequent iterations, gave a 0.96 accuracy on PC but only 0.76 on iPad. Secondly, excluding TCP and UDP packet counts. Eliminating the numerical data of TCP and UDP packets from the model led to a 0.97 accuracy on PC and 0.82 on iPad. While more efficient in training, those methods showed lower accuracy and were not pursued further.

5.2 Attention

An attention mechanism in a model allows the network to focus on different parts of the input sequence for each step of the output sequence [27]. It's particularly useful in tasks where the relevance of input elements can vary across different parts of the output [15]. For example, in machine translation, certain words in the input sequence are more relevant when generating specific words in the output sequence, in this case some input phrases may be more relevant for certain classes or device.

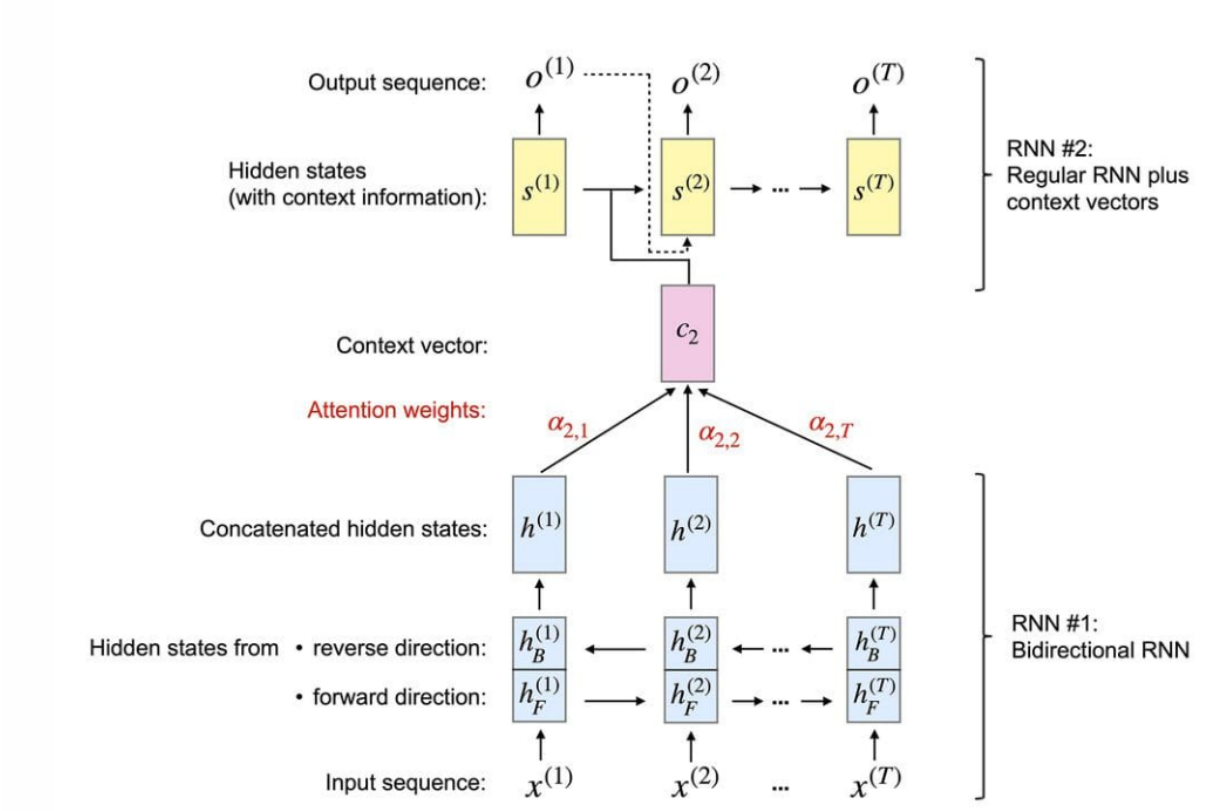

Figure 5.1: Architecture of an RNN with attention, note that in this projects case the output is a vector as the task is classification [28]

The model is the same as before, but an attention mechanism is added. It starts with an embedding. This layer is designed to map the decoder inputs, into a high dimensional space. It serves as a first preparation step, transforming the decoder input into a form suitable for attention computation. Then comes a linear layer that computes the attention weights. This is achieved by taking the concatenated vector of the embedded attention input and the hidden state of the RNN. The linear layer processes this concatenated vector, and the resulting output is passed through a softmax function. The softmax output represents the attention weights, indicating the importance or relevance of each part of the input sequence for the current output prediction.

After applying the attention weights to the RNN outputs to get a weighted sum, a linear layer reshapes the attention applied output. It is an important step for aligning the dimensions of the attention output with the next layers. Finally an additional RNN layer to process the output. Post attention processing involves passing the reshaped, attention enhanced output through another RNN layer. This layer refines the output by taking in account the attention focused features, hopefully capturing more complex dependencies and contextual information that the standard RNN layer might miss.

Implementing the attention mechanism in the model results in better outcomes. When the model is trained exclusively on PC data, it achieves a 0.97 accuracy on the PC test set. The model's performance on iPad data also shows improvement with an overall accuracy of 0.91, but it struggles on the problematic iPad date, where the accuracy drops to 0.75. On the other hand, when the model is trained with a mix of both PC and iPad data, the accuracy on the PC test set remains high at 0.97. More notably, the accuracy on the problematic iPad date sees a significant improvement, reaching 0.83. While the attention mechanism does enhance the model's performance in cross-device predictions, its impact is more substantial when the model is trained solely on PC data compared to a combined training approach with both PC and iPad data.

Trained on	**iPad Test Accuracy**	**PC Test Accuracy**	**Extra Information**
PC	0.82249	0.97063	no tcp and udp packages
PC	0.76977	0.95635	only phrase of timestamp
PC and iPad	0.83747	0.85106	8 dates PC and iPad
iPad	0.93277	0.72648	none
PC and iPad	0.85196	0.94820	none
PC	0.85985	0.97622	none
PC	0.91854	0.96618	attention
PC and iPad	0.82768	0.96704	attention, 12 dates PC and iPad

Table 5.1: Model Test Accuracies

6 Conclusion and Future Works

In summary, the feasibility of employing machine learning, particularly RNNs, for surveying digital audiences is entirely doable. These RNNs, whether augmented with attention mechanisms or not, are largely effective in this purpose. However, it is important to note that certain outlying cases can present challenges to the model's accuracy and reliability.

To improve the performance and overcome these challenges, additional methods can be employed. Firstly, integrating a more complex attention mechanism, such as the multi head attention found in transformer architectures. This form of attention allows the model to handle multiple aspects of the data simultaneously, offering a more nuanced understanding of complex patterns. Secondly, increasing the volume of training data from the iPad could be beneficial. More data would provide the model with a wider array of usage scenarios, enabling it to better learn and predict iPad-specific user behaviors, particularly in those outlier cases. As an alternative, performing some data augmentation would enable the model to train on additional data without more being collected. Finally, using ensemble learning. This would involve using a combination of models, each leveraging their strengths of the specific device or network they were trained on.

Acknowledgements

Dyma for introducing me to machine learning

Naveed offering to co supervise the thesis despite not being a machine learning expert.

Bibliography

[1] Kantar.com, audience measurement https://www.kantar.com/expertise/audience-measurement

[2] Hanne Teigum, Kantar.com, Unlocking the relationship between the viewer and the screen
https://www.kantar.com/inspiration/advertising-media/unlocking-the-relationship-between-the-viewer-and-the-screen 25.04.2023.

[3] Kantar, 100 years and counting: how centennial media brands stay relevant, Kantar.com https://www.kantar.com/inspiration/100-years-and-counting—how-centennial-media-brands-stay-relevant 05.10.2023

[4] Common European Research Classification Scheme (CERCS) https://www.etis.ee/Portal/Classifiers/Index/26?

[5] M Gómez Aguilar, FJ Paniagua Rojano, P Farias Batlle, "The behaviour of the television audience on social networks. An approach to its profile and the most talked about programmes", *Revista Latina de Comunicación Social*, **70**, 2015, pp. 539 to 551, DOI:http://dx.doi.org/10.4185/RLCS-2015-1058en.

[6] Hill Shawndra, "TV Audience Measurement with Big Data", *Big Data*, **2**, 2014, pp. 76 to 86, DOI:10.1089/big.2014.0012.

[7] Yamak, Peter T. and Yujian, Li and Gadosey, Pius K., "A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting", *Association for Computing Machinery*, **19**, 2020, pp. 49 to 55, DOI:10.1145/3377713.3377722.

[8] Nosouhian, S., Nosouhian, F., Kazemi Khoshouei, A., "A Review of Recurrent Neural Network Architecture for Sequence Learning: Comparison between LSTM and GRU", *Preprints*, DOI:https://doi.org/10.20944/preprints202107.0252.v1.

[9] A. Vaswani et al, "Attention is All you Need" in *Advances in Neural Information Processing Systems*, 2017.**30**, pp. 6000–6010,

[10] Kyunghyun Cho and Bart van Merrienboer and Çaglar Gülçehre and Fethi Bougares and Holger Schwenk and Yoshua Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation", *CoRR*, **abs/1406.1078**, DOI:https://doi.org/10.48550/arXiv.1406.1078.

[11] Philip B. Weerakody, Kok Wai Wong, Guanjin Wang, Wendell Ela, "A review of irregular time series data handling with gated recurrent neural networks", *Neurocomputing*, **44**, 2021, pp. 161 to 178, DOI:https://doi.org/10.1016/j.neucom.2021.02.046.

[12] M.Tarwani, Kanchan and Edem, Swathi, “Survey on Recurrent Neural Network in Natural Language Processing”, *International Journal of Engineering Trends and Technology*, **48**, 2017, pp. 301 to 304, DOI:10.14445/22315381/IJETT-V48P253.

[13] J. Xiao and Z. Zhou, “Research Progress of RNN Language Model”, *International Conference on Artificial Intelligence and Computer Applications*, **ICAICA**, 2020, pp. 1285 to 1288, DOI:10.1109/ICAICA50127.2020.9182390.

[14] Wenpeng Yin and Katharina Kann and Mo Yu and Hinrich Schütze, ”Comparative Study of CNN and RNN for Natural Language Processing”, *CoRR*, **abs/1702.01923**, 2020, DOI:https://doi.org/10.48550/arXiv.1702.01923.

[15] A. Galassi, M. Lippi and P. Torroni, ”Attention in Natural Language Processing”, *IEEE Transactions on Neural Networks and Learning Systems*, **32**, 2021, DOI:10.1109/TNNLS.2020.3019893.

[16] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, Garrison W. Cottrell, ”A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction”, *CoRR*, **abs/1704.02971**, 2017, DOI:https://doi.org/10.48550/arXiv.1704.02971.

[17] A. Dan, D. M. Dias, R. Mukherjee, D. Sitaram and R. Tewari, “Buffering and caching in large-scale video servers”, *Digest of Papers. COMPCON’95*, **Technologies for the Information Superhighway**, 1995, pp. 217-224, DOI:10.1109/CMPCON.1995.512389.

[18] T. Hoßfeld, C. Moldovan and C. Schwartz, “To each according to his needs: Dimensioning video buffer for specific user profiles and behavior”, *2015 IFIP/IEEE*, **International Symposium on Integrated Network Management**, 2015, pp. 1249-1254, DOI:10.1109/INM.2015.7140476.

[19] Gary C. Kessler, “An Overview of TCP/IP Protocols and the Internet”, *InterNIC*, , 2010, URL:http://www.sci.brooklyn.cuny.edu/ parsons/courses/3120-fall-2012/notes/tcp-ip-notes.pdf.

[20] H. Zheng and J. Boyce, “An improved UDP protocol for video transmission over Internet-to-wireless networks”, *IEEE Transactions on Multimedia*, **3**, 2001, pp. 356-365, DOI:10.1109/6046.944478.

[21] Gheith Abandah; Abdel-Karim, Asma., “ACCURATE AND FAST RECURRENT NEURAL NETWORK SOLUTION FOR THE AUTOMATIC DIACRITIZATION OF ARABIC TEXT”, *Jordanian Journal of Computers and Information Technology*, **6**, 2020, DOI:10.1109/6046.944478.

[22] Donglai Zhu, Hengshuai Yao, Bei Jiang, Peng Yu, “Negative Log Likelihood Ratio Loss for Deep Neural Network Classification”, *CoRR*, 2018, **abs/1804.10690**, DOI:https://doi.org/10.48550/arXiv.1804.10690.

[23] Sebastian Bock, Josef Goppold, Martin Georg Weiß, “An improvement of the convergence proof of the ADAM-Optimizer”, *CoRR*, 2018, **abs/1804.10587**, DOI:https://doi.org/10.48550/arXiv.1804.10587.

[24] Josep Badia, personal communication, October, 2023

[25] Dmytro Fishman, Pavel Chizov, Introduction to Machine Learning course, Autumn 2022

[26] Joonas Ariiva, et al, Seminar in computer vision, Spring 2023

[27] Raul Vincente Zafra, Neural Networks course, Spring 2023

[28] KDnuggets on March 10, 2022 in Partners, Kantar.com,Adding an attention mechanism to RNNs https://www.kdnuggets.com/2022/03/packt-adding-attention-mechanism-rnns.html.

[29] Amusi, https://zhuanlan.zhihu.com/p/146876678.

Appendices

Figure 6.1: Examples of background activity that are not created by an active utilisation of the device

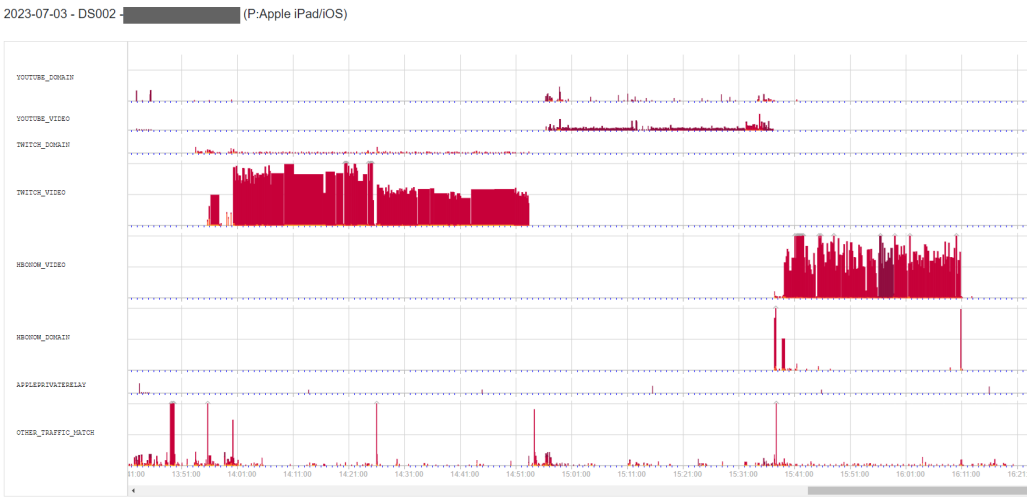

Figure 6.2: Example of a very different volume of data exchanged between two providers, one using TCP and the other UDP

Non-exclusive licence to reproduce thesis and make thesis public

I, Nathan Mison

1. grant the University of Tartu a free permit (non-exclusive licence) to: reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, my thesis

   "Using Machine Learning to Measure Digital Audiences"

   supervised by Josep Badia, Kairit Sirts

2. I grant the University of Tartu the permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work from 23/12/2025 until the expiry of the term of copyright,

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Nathan Mison
23.12.2023