Multi-touch Attribution for Complex B2B Customer Journeys using Temporal Convolutional Networks

Aniket Agrawal
anikagra@adobe.com
Adobe Systems
India

Nikhil Sheoran
sheoran2@illinois.edu
University of Illinois at Urbana-Champaign
United States

Sourav Suman
sosuman@adobe.com
Adobe Systems
India

Gaurav Sinha
sinhagaur88@gmail.com
Adobe Research
India

ABSTRACT

Customer journeys in Business-to-Business (B2B) transactions contain long and complex sequences of interactions between different stakeholders from the buyer and seller companies. On the seller side, there is significant interest in the multi-touch attribution (MTA) problem, which aims to identify the most influential stage transitions (in the B2B customer funnel), channels, and touchpoints. We design a novel deep learning-based framework, which solves these attribution problems by modeling the conversion of journeys as functions of stage transitions that occur in them. Each stage transition is modeled as a Temporal Convolutional Network (TCN) on the touchpoints that precede it. Further, a global conversion model Stage-TCN is built by combining these individual stage transition models in a non-linear fashion. We apply Layer-wise Relevance Propagation (LRP) based techniques to compute the relevance of all nodes and inputs in our network and use these to compute the required attribution scores. We run extensive experiments on two real-world B2B datasets and demonstrate superior accuracy of the conversion model compared to prior works. We validate the attribution scores using perturbation-based techniques that measure the change in model output when parts of the input having high attribution scores are deleted.

CCS CONCEPTS

• **Information systems** → **Online advertising**; **Web mining**; • **Applied computing** → **Electronic commerce**.

KEYWORDS

multi-touch attribution, temporal convolutional networks, layer-wise relevance propagation, b2b customer journeys

WWW '22 Companion, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9130-6/22/04…$15.00
https://doi.org/10.1145/3487553.3524670

ACM Reference Format:
Aniket Agrawal, Nikhil Sheoran, Sourav Suman, and Gaurav Sinha. 2022. Multi-touch Attribution for Complex B2B Customer Journeys using Temporal Convolutional Networks. In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3487553.3524670

1 INTRODUCTION

In today's era of big data, businesses are tracking and storing huge amounts of customer data coming from interactions via different channels such as *paid-search, organic-search, blog-posts, form-submission, chat-bots,* etc. This data can then be stitched at a customer level to build the entire customer journey as a sequence of interactions (called touchpoints[1]). A natural and important question is to identify the touchpoints and channels which contribute significantly towards customer conversion. This problem, referred to as "multi-touch attribution" (MTA) [13], is a key technical challenge for marketing and sales teams, and if done correctly, can provide considerable benefits to businesses [1]. In this paper, we are interested in the Business-to-Business (B2B) setting, where products are sold by one company to another company, e.g. enterprise software, cloud technology, etc. A customer in this setting goes through several interactions in a non-linear fashion [3], and transitions between multiple stages before an eventual purchase decision happens. A natural extension to the above MTA question is to attribute purchase credit to these stage transitions as well. Further, another interesting question is to understand the contributions of touchpoints and channels that led to these transitions. We address these questions in more detail, but we first give an overview of stages and stage transitions in B2B customer journeys and the customer funnel, which encapsulates all of these.

B2B Funnel and Stage Transitions: Marketing automation softwares (MAS) enable businesses to track touchpoints that have happened during a customer's journey in real-time. In these journeys, multiple people from the buying company interact with the seller via multiple online as well as offline channels. Depending on the kind of interactions, an MAS divides the journey into stages which can be viewed as components of the B2B customer funnel. Transitions between these stages are indicative of the progress made towards the purchase transactions, and attributing appropriate credit to them is of very high value. Moreover, in the B2B setting,

[1] we will use terms interaction and touchpoint interchangeably

the seller might also be interested in identifying the most influential touchpoints for each stage transition. Below we give examples of some popular stage transitions that an MAS tracks.

(1) Customer-Known: In the beginning, unknown users from the buyer company interact with the seller via marketing channels. Some of them provide identifying information such as name/email and get converted to prospects. This transition is called *Customer-Known*.
(2) Lead Creation: Some of the prospects show concrete interest by filling out forms and providing contact information. Based on the engagement level, they get converted into leads. This transition is called *Lead-Creation*.
(3) Opportunity Creation: Leads interact with sales representatives and, based on their interest in purchasing, they get converted to contacts with whom the sales teams engage further. This transition is called *Opportunity-Creation*.
(4) Journey Closed: After multiple interactions between contacts and the sales team, a decision on purchase takes place. This transition is called *Journey Closed*.

A customer journey is called "completed" if they go through some or all of the stage transitions and their purchase decision is known. We are now ready to describe the main problem statement tackled in this work.

The B2B-MTA Problem: Given access to a collection of completed customer journeys (as a sequence of touchpoints), their eventual conversion (purchase) label, and timestamps of all stage transitions, we aim to solve the following MTA problems for all converted journeys, i.e., journeys where a purchase happened.

- **Problem 1:** Attribute credit of eventual conversion to the different stage transitions.
- **Problem 2:** Attribute credit of each stage transition to touchpoints and channels that led to it.
- **Problem 3:** Attribute credit of eventual conversion to individual touchpoints and channels.

Challenges: As highlighted in [7], B2B journeys span over long time periods, sometimes spanning years depending on the size of the seller business and industry it operates in. This will lead to several touchpoints with irregular time gaps between them. Further, B2B journeys are highly non-linear [3] due to the presence of multiple stakeholders in the buying company, each trying to execute their own role in the buying process, leading to long-term sequential dependencies between the touchpoints. Popular data modeling techniques often struggle to capture such complexities, and therefore building a data-driven technique is very challenging.

Contributions: We attempt to solve the B2B-MTA problem in this paper and make the following contributions:

(1) We build a deep learning-based model called Stage-TCN to model conversion of B2B customer journeys as a non-linear function of different stage transitions in the journey.
(2) We adapt the Layer-wise Relevance Propagation (LRP) technique to our models for attributing conversion credit to stage transitions, touchpoints, and channels. While doing so, we also attribute credit of stage transitions to touchpoints that precede them.
(3) We conduct experiments on two proprietary real-world B2B datasets to demonstrate that accuracy metrics of Stage-TCN are much better compared to baselines [8, 11, 13, 17, 18].
(4) Using the region perturbation technique from [2, 12], we demonstrate that our attribution scores are much more reliable than the ones from the baselines. While doing so, we also provide deep insights into the contributions of different touchpoints/channels/stage-transitions in our datasets.

2 RELATED PRIOR WORK

The simplest set of attribution techniques are rule-based. Most common among these are first-touch/last-touch, where all credit is given to a single touchpoint [4]. An extension called linear attribution gives equal credit to all touchpoints. The Time-decay model, on top of linear attribution, re-weights scores in proportion to their recency with respect to conversion. U-shaped, W-shaped, and full-path models consider rules to give higher credit to certain touchpoints and distribute the leftover among the remaining. These models are simple to understand and implement, but they lack the sophistication needed to tackle the B2B specific challenges mentioned above (since they ignore the data distribution). Many recent works [8, 11, 13, 17] on the MTA problem have explored data-driven techniques. These model the underlying distribution of touchpoints and purchase labels and derive attribution scores using the model. For example, [13] proposed a bagged logistic regression model for conversion prediction and computed attributions based on their model weights. A survival analysis-based technique was designed in [17], which was improved in [8] by taking the intrinsic conversion rate of users into account. Another popular way of modeling attributions in online advertising is via Shapley values, and their modifications from cooperative game theory [5, 14]. A drawback of all these methods is that they fail to capture sequential dependencies in the journey, which is a major challenge in the B2B setting, as mentioned earlier. To tackle this, an attention-based Recurrent Neural Network (ARNN) model was designed in [11] where attention weights were used to compute touchpoint attributions. Another RNN-based method with attributions computed using Shapley values was designed in [6]. While these methods perform very well on digital advertising datasets, they have comparatively lower performance on B2B journeys. The RNNs used in these capture sequential dependencies but only have limited access to past touchpoints which is limiting in the B2B setting due to the presence of longer sequential dependencies. It is also well known that RNNs face the problem of vanishing/exploding gradients on long journeys such as those in the B2B world, which could greatly hamper performance.

3 PROPOSED METHODOLOGY

In this section, we provide complete details of our methodology. The high level strategy of our method is to model conversion of completed[2] B2B journeys. We first supply all details of our conversion model. Following this, we perform multi-touch attribution using a model explanation technique which scores stage transitions, touchpoints and channels. Even though in Section 1 we described four specific stage transitions, here we consider the general case

[2] i.e. purchase decision has been taken

and assume at-most r such possible transitions exist for each journey. We assume the following input format for each journey in our dataset.

(1) A sequence of $r+1$ numbers $n_1, \ldots, n_r$ and y, where n_i $i \in \{1, \ldots, r\}$ correspond to the number of touchpoints that precede the r stage transitions respectively, and y is the purchase label. Note that the last stage transition is always the closure of journey (with or without purchase) and thus n_r denotes length of the journey as well.
(2) A sequence $(x_1, \ldots, x_n)$ $(n = n_r)$ of touchpoints where x_1 occurs before x_2 and so on. Each x_i, $i \in \{1, \ldots, n\}$ is a vector in $\mathbb{R}^p \times \mathbb{N}^q$, i.e. the first p co-ordinates are numeric and the last q co-ordinates are categorical.

Next, we describe salient features of our Stage-TCN model.

Stage-TCN Model: At a very broad level, our model contains r different components (sub-networks) $S_1, \ldots, S_r$, each representing one stage transition. The network S_i, takes the sub-sequence $(x_1, \ldots, x_{n_i})$ as input and outputs a vector $Y_i \in \mathbb{R}^2$. At the output node, $Y_1, \ldots, Y_r$ are concatenated and a dense layer is applied giving

$$Y_{out} = (Y_{out}[0], Y_{out}[1]) = Dense(Y_1, \ldots, Y_r) \in \mathbb{R}^2.$$

This is followed by a softmax σ to get Y_{pred}, the probability of conversion. The loss function used for training is sparse softmax cross entropy with logits over Y_{out}.

$$\begin{aligned} Y_{pred} &= \sigma(Y_{out}) = \frac{e^{Y_{out}[1]}}{e^{Y_{out}[0]} + e^{Y_{out}[1]}} \\ Loss &= -\sum_{i=0}^{1} Y_{True}[i] \log(\sigma(Y_{out}[i])) \end{aligned} \tag{1}$$

where $Y_{True}[i]$ is the label corresponding to i^{th} class. Figure 1 provides an illustration of the above. Next, we explain the individual networks $S_1, \ldots, S_r$.

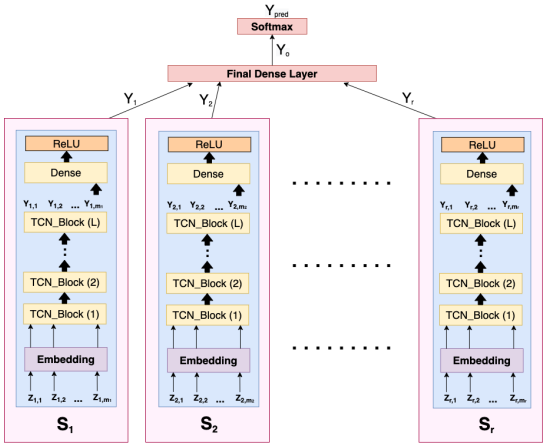


Figure 1: Stage-TCN model

Architecture of component S_i: Each network S_i is modeled as a TCN (Appendix A) that takes the sequence $(x_1, \ldots, x_{n_i})$ as input and outputs Y_i. It's architecture is exactly like the layered TCN model described in Appendix A. Assuming that the TCN takes m_i inputs[3] (ordered in time), we need to modify the input when m_i, n_i are not equal. If $n_i > m_i$, then the input sequence is truncated from the beginning and $(x_{n_i - m_i + 1}, \ldots, x_{n_i})$ is used as the input. If $n_i < m_i$, then the sequence is padded with zeroes in the beginning to make it of length m_i. For simplicity, let's assume $(z_{i,1}, \ldots, z_{i,m_i})$ is the final input sequence for the TCN. Recall that each input touchpoint is assumed to be in $\mathbb{R}^p \times \mathbb{N}^q$ i.e. first p features are numeric and the last q are categorical. The discrete features are passed through an embedding layer, mapping them into numeric vectors of length e (called embedding dimension). Therefore, our input to the TCN block is a m_i length sequence of vectors in $\mathbb{R}^{p+eq}$. We assume the TCN has L hidden layers (where L is tuned as a hyperparameter). In order to capture long term memory, we use dilated convolutions described in Equation 2 (Appendix A) with dilation factor 2^l for layer l along with a large filter size k. Assume that the output at the L^{th} (final) layer of the TCN is $y_{i,1}, \ldots, y_{i,m_i}$, i.e.,

$$y_{i,1}, \ldots, y_{i,m_i} = f_{TCN}(z_{i,1}, \ldots, z_{1,m_i})$$

where f_{TCN} denotes the TCN model. We slice the output and extract $y_{i,m_i} = (y^1, \ldots, y^d) \in \mathbb{R}^d$. Here d denotes the number of filters used at the L^{th} layer which we also tune as a hyperparameter. Finally, we apply a linear network with weights $(v_0, \ldots, v_d)$ and $(w_0, \ldots, w_d)$ (to be learned) on this to get

$$Y_i = (v_0 + \sum_{i=1}^{d} v_i y^i, w_0 + \sum_{i=1}^{d} w_i y^i)$$

which is fed into Equation 1. Figure 5 in Appendix A shows a detailed illustration of TCN network architecture. This completes our description of the Stage-TCN model.

3.1 Attribution Model

Our attribution methodology applies a Layer-wise Relevance Propagation (LRP) framework (Appendix B) on trained Stage-TCN models. The main task in LRP is to set relevance back-propagation rules from the output all the way down to the inputs. For all journeys that convert, we compute relevance of all the nodes (using specific propagation rules) with respect to output $Y_{out}(1)$ since it essentially captures the positive class. Since we are only interested in assigning positive attributions in our B2B journeys, we perform a mild post-processing to remove the negative attributions and normalize across nodes appropriately. These processed scores are then interpreted as the desired attribution scores. For attributing credit of stage transitions to touchpoints, we use relevance of touchpoints relative to the stage-transition relevance. This leads to a complete resolution of our problem statement in Section 1. We explain each step in more detail below.

Attribute conversion to stage transitions: We compute the attribution of i^{th} stage transition using relevance R_i of the output node of S_i with respect to $Y_{out}(1)$. Since $Y_{out}(1)$ is computed via a dense layer on outputs $Y_1, \ldots, Y_r$, we use the dense layer propagation

[3] m_i will be tuned as a hyperparameter

rule from Definition B.1 in Appendix B. Note that since $Y_i \in \mathbb{R}^2$, relevance $R_i = (R_{i,1}, R_{i,2}) \in \mathbb{R}^2$ as well[4]. We normalize the positive relevances in set $\{R_{i,j} : i \in \{1, \ldots, r\}, j \in \{1, 2\}\}$ to sum up to $Y_{out}(1)$ and clip the negative relevances to 0. Finally, to get attribution of stage transition in component S_i, we simply add these modified $R_{i,1}$ and $R_{i,2}$ together to get the attribution A_i.

Attribute stage transitions to touchpoints: This is done by back-propagating relevance R_i that was computed above, all the way down to the touchpoints of S_i i.e. $(z_{i,1}, \ldots, z_{i,m_i})$[5]. We use the dense layer propagation rule from Definition B.1 followed by the propagation rule for dilated convolutional layers and skip connections outlined in Definitions B.2 and B.3 (Appendix B) respectively. Propagating relevance of just one of the S_i (i.e. R_i) will only attribute credit of the i^{th} stage transition to the embeddings $\in \mathbb{R}^{p+eq}$ of input touchpoints in S_i. Similar to the stage transition attribution, we clip all the negative co-ordinates of these relevance vectors to 0 and normalize the positive ones to sum up to A_i. Finally, we add these modified co-ordinates for each touchpoint $z_{i,j}$ in S_i and get it's attribution score $A_{i,j}$ relative to the i^{th} stage transition.

Attribute conversion to touchpoints: To calculate this, at each touchpoint we accumulate the relative attributions that were obtained for the different stage transitions as described above. The accumulated scores thus obtained are interpreted as attribution of conversion to all the touchpoints.

Attribute conversion to channels: Once we have the touchpoint level attribution scores, for each journey, we accumulate the attributions of all touchpoints that occur via the channel and obtain channel level attribution scores (for each journey). We also aggregate this over all the journeys to obtain aggregate channel attributions.

4 OUR DATASETS

In Section 5, we conduct experiments on two real world B2B customer journey datasets. The first one (called Dataset 1) is from a Fortune 500 software company that sells marketing automation software to its customers. The second dataset (Dataset 2) is from a leading global technology company that provides enterprise marketers a platform to enhance their visual brand content at scale. Dataset 1 contains about ~55K customer journeys that occurred between December 2010 and August 2019. We use ~44K of these for training our models, ~5.5K for validation/model-selection and the remaining (~5.5K) as test data on which we report our metrics. Dataset 2 contains about ~16K customer journeys that occurred between January 2000 and August 2020. We use ~10K of these for training our models, ~3K for validation/model-selection and the remaining (~3K) as test data. In both our datasets, the number of touchpoints per journey (called journey-length) varies from as small as 5 to values in 1000s. Each touchpoint contains information about the channel through which it occurred along with the timestamp at which it occurred. Total number of channels in Dataset 1 is 53 and that in Dataset 2 is 38. Apart from the channels and timestamps, we also have access to some other supplementary features which help us identify interactions where the stage transitions happened. The current dataset has 4 such stage transitions; (a) Customer-Known, (b) Lead-Created, (c) Opportunity-Created, and (d) Journey-Closed. These are explained in detail in Section 1. We create features for each interaction containing information about its channel, time taken from previous interaction and whether stage transition happened during this interaction (also which one). This information can be used by our Stage-TCN model to separately represent each stage transition with a TCN. For the baseline models (which does not create such separate representations), this information is provided at each interaction directly to make sure that all models receive the same information and our model does not have any advantage in terms of data access.

5 EXPERIMENTS AND RESULTS

In this section, we conduct two kinds of experiments. The first compares accuracy metrics of our conversion model, Stage-TCN with many of the recent baselines mentioned in Section 2 demonstrating that our conversion model is the most accurate with respect to the metrics used. In our second experiment, we compute attribution scores as described in Section 3 and use perturbation based techniques (Appendix C) to provide quantitative insights into why our touchpoint, channel and stage transition attribution scores are very reliable and better than the baselines. We also supplement our results with various insights about attributions for different stage transitions and channels relative to these transitions, providing qualitative justification as to why a separate representation of each stage transition is very useful. Here is a list of baseline models we compare Stage-TCN with:

- **LR** is a Logistic Regression model proposed in [13]. We use channel frequencies and number of interactions preceding each stage transition as features. The attribution scores are obtained at channel-level using model coefficients.
- **AMTA** is the Additional Multi-Touch Attribution [8] model that uses Survival Analysis to model the effect of channel interactions (originally ad exposures) on conversion.
- **ARNN** is a sequence-based model [11] that models conversion using an RNN with an attention layer. The attention weights are used as the attribution score for each touchpoint.
- **LRATT** is a sequence-based model [18] that learns Logistic Regression based global attention weights that are used as attention over a Bi-RNN sequence model.

We create features for each touchpoint as described in Section 4. We build upon the code released by [11][6] and [18][7] to evaluate these baselines. All the models were trained on Tesla V100-SXM2 GPUs (16GB /32GB memory) with Intel Xeon CPU E5-2686.

5.1 Conversion Model Accuracy

Let $X_1, \ldots, X_n$ denote the sequence of touchpoints and Y denotes the eventual label, we model the probability $P(Y = 1|X_1, \ldots, X_n)$ as a function of $X_1, \ldots, X_n$ using the baseline models and our technique described in Section 3. We tune the hyperparameters of our models by searching over a random grid of parameters containing 100 settings each and training the model for each of these settings on the training data. For each of the models the best hyperparameters are selected using the AUC-ROC (Area under ROC

[4] it contains relevance of both the co-ordinates of Y_i
[5] actually we only propagate till output of the embedding layer
[6] https://github.com/rk2900/deep-conv-attr
[7] https://github.com/joey1993/Global-Attention-RNN

	Dataset 1		Dataset 2	
Models	AUC-ROC	Accuracy	AUC-ROC	Accuracy
LR	0.68	0.67	0.64	0.77
AMTA	0.67	0.63	0.68	0.77
ARNN	0.74	0.67	0.77	0.77
LRATT	0.66	0.60	0.78	0.59
Stage-TCN (Ours)	0.79	0.72	0.89	0.85

Table 1: Accuracy results on Dataset 1 and Dataset 2

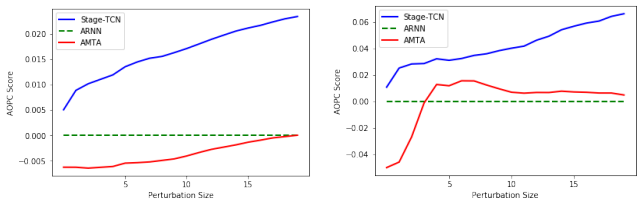
curve) score on the validation data. We use the following metrics for comparison; (a) AUC-ROC score on the test data, and (b) Accuracy score i.e. number of correctly predicted labels in the test data. To compute accuracy (on validation data) we need to threshold the conversion probabilities. We maximize accuracy for thresholds in set $\{0.01, 0.02, \ldots, 0.99\}$. Tables 1 and 1 show the performance of all the models for Datasets 1 and 2 respectively, computed on the test data. As can be seen from the tables, Stage-TCN clearly outperforms all the baseline models on both the datasets in terms of AUC-ROC and Accuracy.

5.2 Validation of Attribution Scores

Touchpoint Attributions: For each converted journey, our method outputs attribution scores of each touchpoint depicting their relative[8] contribution to conversion. We validate these systematically by computing AOPC scores described in Appendix C. The main idea is to successively delete touchpoints in decreasing order of their attribution scores and accumulate the change in conversion probabilities (using an accurate conversion model). If attribution scores are correct representations of a touchpoint's influence on conversion, then deleting highly influential touchpoints should create a significant drop in conversion probability. Since Stage-TCN provides the best accuracy results, we use it as the conversion model while calculating AOPC scores for baselines as well. Figures 2a and 2b depict the AOPC scores vs. number of touchpoints deleted (called perturbation size) of our attribution method compared with the attributions provided by the baselines[9] **ARNN** and **AMTA**. As mentioned earlier, **LR** and **LRATT** do not provide attribution scores for touchpoints and therefore we do not compare them here. Since we wish to plot AOPC for reasonably large perturbation size (~ 20), we restrict our datasets to converted journeys with lengths greater than 50. All scores were computed on the respective test datasets. We would like to note that while test data from Dataset 1 had sufficiently large number of such[10] journeys (~700), the one from Dataset 2 had only 35 such journeys. As can be seen from Figures 2a and 2b, on both datasets, the AOPC scores of the ordering obtained from our attribution method is significantly higher compared to that obtained from **ARNN** and **AMTA** indicating that we do a much better job of attributing conversion credit to touchpoints.

Channel Attributions: In this analysis, for each model we first compute the aggregate attribution scores of different channels towards conversion by traversing each journey and accumulating attributions of touchpoints that occur via any given channel. Then,

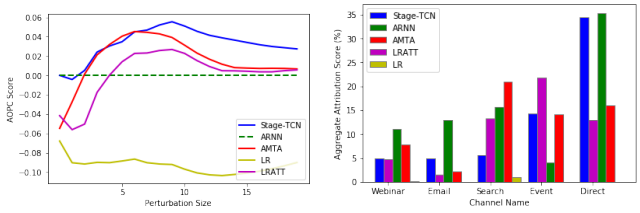
[8] relative to other touchpoints in the journey
[9] we plot the AOPC scores relative to AOPC of **ARNN**
[10] converted and with length greater than 50



(a) Dataset 1 **(b) Dataset 2**

Figure 2: AOPC vs. Perturbation for touchpoint attributions

for each channel, we aggregate across all the journeys[11] and denote them as attributions of the channels towards conversion. We compute AOPC curves to validate these attributions by creating the counterfactual situation where we delete all touchpoints that occur from the top channels (according to the above aggregate attributions) via and accumulate the change in conversion probabilities. Similar to the touchpoint attribution analysis, to compute these AOPC scores we again use Stage-TCN as the conversion model due to it's high accuracy. Figure 3a depicts the AOPC scores vs. number of channels deleted (called perturbation size) of our method compared with the attributions provided by baselines[12] **ARNN**, **AMTA**, **LRATT** and **LR** on Dataset 1. As can be seen from Figure 3a, the AOPC scores of the ordering obtained from our attribution method is strictly better compared to the orderings obtained from the baselines. For smaller values of perturbation, AOPC scores of the **AMTA** model comes close to ours, while for larger perturbations, we are clearly better. In Dataset 2, unfortunately, most journeys have very few channels and therefore larger perturbations lead to deletion of quite a lot of touchpoints leading to highly unreliable AOPC scores. Therefore, we only show the above plot for Dataset 1.



(a) AOPC vs. perturbation size **(b) Top 5 channels**

Figure 3: Plot on left shows AOPC score vs. perturbation size for channel attributions and the bar chart on right illustrates aggregate channel attributions for top channels

Since the AOPC scores only care about the ranking of channels (decreasing order of attribution scores), our comparisons above do not capture the actual values of these scores. In order to compare the actual scores, we pick 5 channels that were among the top ones according to our Stage-TCN based attribution method and compare their aggregate attribution scores across different models. Results are shown in Figure 3b.

[11] and normalize with respect to the other channels
[12] we plot the AOPC scores relative to AOPC of **ARNN**

Stage Transition	Attribution
Customer Known	2.7%
Lead Created	1.4%
Opportunity Created	84.3%
Journey Closed	11.6%

Stage Transition	Email	Direct
Customer Known	3%	81%
Lead Created	79%	1%
Opportunity Created	3%	6%
Journey Closed	0%	75%

Figure 4: Table on the left contains aggregate stage transition attribution scores and the one on the right shows relative attribution of popular channels

Stage Transition Attributions: Unfortunately, none of the baseline models provides attribution scores for stage transitions and therefore for this analysis we only highlight insights about stage transition attribution scores obtained from our Stage-TCN model. Recall from Section 3 that for each converted journey, our attribution method outputs attribution scores of each stage transitions depicting their relative[13] contribution to conversion. We first aggregate these over all the converted journeys (in test data) to get aggregate attribution scores for each stage transition. In the left table of Figure 4, we show these aggregate attribution scores for Dataset 1. It clearly shows that at the aggregate level, the "Opportunity Created" stage transition contributes most to conversion (according to our technique), followed by "Journey Closed". The first two stage transitions i.e. "Customer Known" and "Lead Created" have much lower contributions which is understandable since they are further from conversion. However, we would like to note that, when we dig deeper into the attribution scores (at journey level) for these stage transitions, we observe that in more than 14% (9% respectively) of converted journeys "Customer Known" ("Lead Created" respectively) gets more than 10% attribution. Alternatively, in more than 24% (73% respectively) of converted journeys "Opportunity Created" ("Journey Closed" respectively) gets less than 1% attribution. This implies that while at the aggregate level, the last two stage transitions dominate significantly, for many converted journeys they do not contribute much as all. Similarly, even though the first two stage transitions have very low attributions at the aggregate level they have significant attributions for a large chunk of converted journeys. This illustrates that our attribution mechanism is accounting for contributions of all stage transitions as we would desire in the B2B MTA application.

Attributions Relative to Stage Transitions: Using our method described in Section 3, we also compute attribution scores of touchpoints relative to stage transitions. By accumulating relative attributions of touchpoints from the same channel (within a journey) we also obtain attribution scores for channels relative to stage transitions. By aggregating over all converted journeys, we also get the aggregate attribution scores of channels relative to stage transitions. In the right table of Figure 4, we present these aggregate relative attribution scores (towards the different stage transitions) for 2 very influential channels in Dataset 1 and show that different channels can be important for different transitions. As can be seen, channel "Email" has very high attribution score relative to the "Lead Created" stage transition and no contribution to "Journey Closed". Similarly the channel "Direct" (direct contact) contributes significantly both to "Customer Known" and "Journey Closed" stage transitions.

[13] relative to other stage transitions

REFERENCES

[1] Blake Brysha Amy Abascal. 2015. *The Definitive Guide to Multi-touch Revenue Attribution.* Brightfunnel. http://www.brightfunnel.com/resources/ebooks/BrightFunnel_Definitive_Guide_To_MultiTouch_Attribution.pdf

[2] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE* 10, 7 (07 2015), 1–46. https://doi.org/10.1371/journal.pone.0130140

[3] Nick Toman Brent Adamson. 2020. *5 Ways the Future of B2B Buying Will Rewrite the Rules of Effective Selling.* Gartner, Inc. https://emtemp.gcom.cloud/ngw/globalassets/en/sales-service/documents/trends/5-ways-the-future-of-b2b-buying.pdf

[4] Jordan Con. 2016. *B2B Marketing Attribution 101.* Bizible, Inc. https://engage.marketo.com/rs/460-TDH-945/images/BZ-B2B-Marketing-Attribution-101-ebook.pdf

[5] Brian Dalessandro, Claudia Perlich, Ori Stitelman, and Foster Provost. 2012. Causally Motivated Attribution for Online Advertising. In *Proceedings of the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy* (Beijing, China) *(ADKDD '12)*. Association for Computing Machinery, New York, NY, USA, Article 7, 9 pages. https://doi.org/10.1145/2351356.2351363

[6] Ruihuan Du, Yu Zhong, Harikesh S. Nair, Bo Cui, and Ruyang Shou. 2019. *Causally Driven Incremental Multi Touch Attribution Using a Recurrent Neural Network.* Research Papers 3761. Stanford University, Graduate School of Business. https://ideas.repec.org/p/ecl/stabus/3761.html

[7] Matt Halchak. 2017. *2017 B2B Buyer's Survey Report, Demand Gen Reports.* DemandBase. https://www.demandgenreport.com/resources/research/2017-b2b-buyers-survey-report/

[8] Wendi Ji and Xiaoling Wang. 2017. Additional Multi-Touch Attribution for Online Advertising. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (San Francisco, California, USA) *(AAAI'17)*. AAAI Press, 1360–366.

[9] Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. 2017. Temporal Convolutional Networks for Action Segmentation and Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[10] Colin Lea, René Vidal, Austin Reiter, and Gregory D. Hager. 2016. Temporal Convolutional Networks: A Unified Approach to Action Segmentation. In *Computer Vision – ECCV 2016 Workshops*, Gang Hua and Hervé Jégou (Eds.). Springer International Publishing, Cham, 47–54.

[11] Kan Ren, Yuchen Fang, Weinan Zhang, Shuhao Liu, Jiajun Li, Ya Zhang, Yong Yu, and Jun Wang. 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* (Torino, Italy) *(CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 1433–1442. https://doi.org/10.1145/3269206.3271677

[12] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. 2017. Evaluating the Visualization of What a Deep Neural Network Has Learned. *IEEE Transactions on Neural Networks and Learning Systems* 28, 11 (2017), 2660–2673. https://doi.org/10.1109/TNNLS.2016.2599820

[13] Xuhui Shao and Lexin Li. 2011. Data-Driven Multi-Touch Attribution Models. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Diego, California, USA) *(KDD '11)*. Association for Computing Machinery, New York, NY, USA, 258–264. https://doi.org/10.1145/2020408.2020453

[14] Raghav Singal, Omar Besbes, Antoine Desir, Vineet Goyal, and Garud Iyengar. 2019. Shapley Meets Uniform: An Axiomatic Framework for Attribution in Online Advertising. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 1713–1723. https://doi.org/10.1145/3308558.3313731

[15] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. In *Arxiv*. https://arxiv.org/abs/1609.03499

[16] Fisher Yu and Vladlen Koltun. 2016. Multi-Scale Context Aggregation by Dilated Convolutions. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1511.07122

[17] Ya Zhang, Yi Wei, and Jianbiao Ren. 2014. Multi-touch Attribution in Online Advertising with Survival Theory. In *2014 IEEE International Conference on Data Mining*. 687–696. https://doi.org/10.1109/ICDM.2014.130

[18] Yichao Zhou, Shaunak Mishra, Jelena Gligorijevic, Tarun Bhatia, and Narayan Bhamidipati. 2019. Understanding Consumer Journey Using Attention Based Recurrent Neural Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 3102–3111. https://doi.org/10.1145/3292500.3330753

A TEMPORAL CONVOLUTIONAL NETWORKS (TCN)

TCNs were recently introduced as an architecture for convolutional sequential prediction [9, 10]. They have been successfully applied in applications such as action segmentation [9] and have also shown strong performance on tasks such as music and language modeling by outperforming recurrent architectures such as LSTMs and GRUs. The goal in a sequence modeling task is to build a neural network that takes a temporal sequence $(\bar{x}_1, \ldots, \bar{x}_n)$[14] as input along with corresponding outputs $y_1, \ldots, y_n$, and outputs a sequence $\hat{y}_1, \ldots, \hat{y}_n$ as prediction for the desired outputs while satisfying the causal constraint that is for all $i \in \{1, \ldots, n\}$, y_i only depends on the "past" i.e. on inputs $\bar{x}_1, \ldots \bar{x}_i$ and not on any future inputs $\bar{x}_{i+1}, \ldots, \bar{x}_m$. To achieve this, TCNs use 1D fully convolutional networks with "causal convolutions", which convolve output at time i only with elements at time $< i$, in the previous layer. In general, these causal convolutions will not be able to look back very long in the past and therefore dilated convolutions [15] are used, which can have exponentially large receptive fields [16]. The dilated convolutional operation on a 1D sequence $\boldsymbol{x} \in \mathbb{R}^n$ using filter $f : \{0, \ldots, k-1\}$ at element s can be defined as

$$(\boldsymbol{x} * f)(s) = \sum_{i=0}^{k-1} f(i)\boldsymbol{x}_{s-di} \tag{2}$$

where d is the dilation factor, filter size is k and $s - di$ depicts past direction. By increasing the dilation factor d (common practice is to increase it exponentially with depth i.e. $d = O(2^i)$ at layer i) and choosing a large filter size k, we can increase the receptive field of the TCN and therefore allow for an extremely large effective history. A typical TCN will have many (say L) layers of such dilated convolutions. Inputs and outputs to each layer have the same length by using the same padding. Figure 5 contains an illustration of TCN with two hidden layers of dilated convolutions and a TCN block with a residual connection. Some advantages of TCNs are - parallelization while training and evaluation, flexibility in receptive field size to have better control on model's memory, more stable gradients compared to RNNs since back-propagation is not in the temporal direction, low memory requirement, etc.

B LAYER-WISE RELEVANCE PROPAGATION

Layer-wise relevance propagation (LRP) [2] is a very popular technique of explaining neural network predictions and has been shown to be quantitatively and qualitatively better at explaining predictions of deep neural networks as compared to other methods [12]. For each input's prediction, the predicted quantity gets decomposed into "relevance scores" of hidden nodes at each layer (including the input layer). Then, for each layer l, these scores are interpreted as contribution of nodes at layer l towards prediction. Following the notation in [2], we denote the relevance score of node d at layer l as $R_d^{(l)}$. The first condition that $R_d^{(l)}$s are expected to satisfy is that at each layer l, they sum up to the prediction output, i.e.

$$f(\bar{x}) = \sum_{d \in l} R_d^{(l)}$$

[14] by temporal we mean ordered in time i.e. $\bar{x}_1$ happens before $\bar{x}_2$ and so on

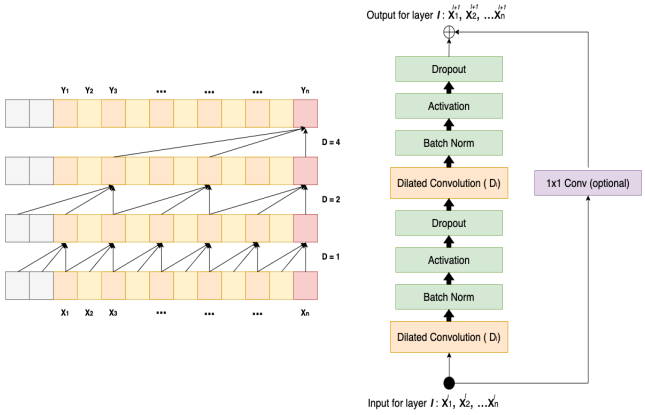


Figure 5: Temporal Convolutional Network

These scores can be further broken into messages $R_{i \leftarrow j}^{(l,l+1)}$ sent from node j at layer $l+1$ to node i at layer l. For node i on layer l, sum of all messages coming from nodes having i as an input, equals it's relevance. Similarly for node j on layer $l+1$, it's relevance can be written as sum of all the messages going from j to it's input nodes.

$$R_i^{(l)} = \sum_{j: i \text{ is input for } j} R_{i \leftarrow j}^{(l,l+1)} \tag{3}$$

$$R_j^{(l+1)} = \sum_{i: i \text{ is input for } j} R_{i \leftarrow j}^{(l,l+1)} \tag{4}$$

To calculate relevance of each node in the network, we need to set a rule for relevance propagation. This is done in a backward fashion and amounts to deciding a rule to break down relevance of a node R_j^{l+1} into the messages $R_{i \leftarrow j}^{(l,l+1)}$. Once this is done for all nodes at a given layer $l+1$, the messages are appropriately summed (Equation 3) at nodes in layer l and their relevances are computed. At the output node relevance is assumed to be equal to the prediction $f(x)$ of the network. Using the propagation rules, this is back-propagated to the lower layers all the way to the inputs. Note that for any input node i and a hidden node j in the network, such that i, j are connected by a path, we can think of relevance of i towards output of j as accumulation of messages on paths from i to j. When j is the output node, we get relevance of inputs on the output prediction. For different kind of nodes, one can choose different propagation rules. As an example, for node j at layer $l+1$ computing a linear function $z_j = \sum_i z_{ij} + b_j$[15] of it's inputs z_{ij}s, a possible decomposition is

$$R_{i \leftarrow j}^{(l,l+1)} = \frac{z_{ij}}{z_j} R_j^{(l+1)}$$

Next, we outline a few propagation rules used in Section 3.

Definition B.1 (LRP for dense layer). *Consider a dense layer calculating $Y = W\bar{a} + b$, where $\bar{a}$ is the vector of activations from previous layer and W is the weight matrix of the dense layer. Given relevance vector $\bar{R}$ comprising of relevances of this layer's output, we*

[15] b_j is the bias term

propagate it to inputs from the previous layer as follows.

$$z = W\bar{a} + b + \epsilon, \quad s = \frac{\bar{R}}{z}, \quad c = W^T s$$

and relevance to be propagated is $R = a * c$. Note that in $\bar{R}/z$ we perform component-wise division and $*$ denotes component-wise multiplication.

Definition B.2 (LRP for dilated convolutional layer). *Using the same notation as above, LRP of dilated convolutional layer is defined as*

$$z = Conv(\bar{a}, kernel = W) + b + \epsilon, \quad s = \frac{\bar{R}}{z},$$
$$c = Conv\_bp(filter = w, grads = s)$$

and relevance to be propagated is $R = a * c$. Note that Conv is the convolution operator and Conv_bp is the backward propagation of convolution.

Definition B.3 (LRP for skip connection). *Output of a skip connection is $Y = U + H(U)$ where H is a deep network and U is an input. Writing $H(U)$ as V and interpreting element-wise addition as a special case of Dense layer and applying LRP for dense layer (with $\epsilon = 0$) defined above, relevances of U and V are given as*

$$R(U) = \frac{U * R(Y_{output})}{U + V}, \quad R(V) = \frac{V * R(Y_{output})}{U + V}$$

where $*$ is the element-wise product of vectors.

C REGION PERTURBATION

In this section we describe the region perturbation technique developed in [12] which extends a similar technique from [2]. We adapt the description in [12] to our situation i.e. for evaluation of attribution mechanisms for B2B customer journeys. Let a journey be given as a time ordered sequence of touchpoints $\boldsymbol{x} = (\bar{x}_1, \ldots, \bar{x}_n)$ and $O = \{\bar{r}_1, \ldots, \bar{r}_L\}$ be a set of subsets of the set $\{\bar{x}_1, \ldots, \bar{x}_n\}$. Further, for every $\bar{r}_i \in O$, let $h_i = \mathcal{H}(\boldsymbol{x}, f, \bar{r}_i)$ be a heatmapping function (computed using the attribution methodology) derived from a class discriminant f used for classification. This heat mapping function indicates how important the subset $\bar{r}_p$ of touchpoints is for representing the eventual predicted label of the journey. Without loss of generality, we assume that the heatmap values are in decreasing order i.e. $h_1 > \ldots > h_L$. Analogous to [12], we define a perturbation process known as the most relevant first (MoRF) process by the following recursive formula

$$\begin{aligned} &\boldsymbol{x}^{(0)}_{MoRF} = \boldsymbol{x} \\ &\forall 1 \leq k \leq L : \boldsymbol{x}^{(k)}_{MoRF} = g(\boldsymbol{x}^{(k-1)}_{MoRF}, \bar{r}_k) \end{aligned} \tag{5}$$

where an ordered sequence of touchpoints $\boldsymbol{x}^{(k)}_{MoRF}$ is created by using a function g that removes the touchpoints in set $\bar{r}_k$ from the sequence $\boldsymbol{x}^{(k-1)}_{MoRF}$. Note that relevance (heatmap values) of the $\bar{r}_i$ are assumed to be in decreasing order and therefore most relevant subset is removed first, then the second most relevant is removed and so on. In order to compare different attribution methodologies (and the heat maps they generate), similar to [12], we define the metric called area over the MoRF perturbation curve (AOPC)

$$AOPC = \frac{1}{L+1} \langle \sum_{k=0}^{L} f(\boldsymbol{x}^{(0)}_{MoRF}) - f(\boldsymbol{x}^{(k)}_{MoRF}) \rangle_{p(\boldsymbol{x})}$$

where $\langle . \rangle_{p(\boldsymbol{x})}$ denotes average over all the journeys in the dataset. When O is ordered in decreasing order of relevance (i.e. the attribution based heatmap values h_p), we would expect AOPC to be large. In the experiments in [12], LRP seemed to have significantly higher AOPC compared to baselines such as random ordering of O, deconvolution and sensitivity, and therefore is argued to be a much better explanation mechanism. More details can be found in [12].