Unveiling the Power of Self-Attention for Shipping Cost Prediction: The Rate Card Transformer

P Aditya Sreekar SREEKARP@AMAZON.COM*
Amazon

Sahil Verma VRSAHIL@AMAZON.COM*
Amazon

Varun Madhavan VARUNMADHAVAN@IITKGP.AC.IN†
Indian Institute of Technology, Kharagpur

Abhishek Persad PERSADAP@AMAZON.COM
Amazon

Abstract

Amazon ships billions of packages to its customers annually within the United States. Shipping cost of these packages are used on the day of shipping (day 0) to estimate profitability of sales. Downstream systems utilize these days 0 profitability estimates to make financial decisions, such as pricing strategies and delisting loss-making products. However, obtaining accurate shipping cost estimates on day 0 is complex for reasons like delay in carrier invoicing or fixed cost components getting recorded at monthly cadence. Inaccurate shipping cost estimates can lead to bad decision, such as pricing items too low or high, or promoting the wrong product to the customers. Current solutions for estimating shipping costs on day 0 rely on tree-based models that require extensive manual engineering efforts. In this study, we propose a novel architecture called the Rate Card Transformer (RCT) that uses self-attention to encode all package shipping information such as package attributes, carrier information and route plan. Unlike other transformer-based tabular models, RCT has the ability to encode a variable list of one-to-many relations of a shipment, allowing it to capture more information about a shipment. For example, RCT can encode properties of all products in a package. Our results demonstrate that cost predictions made by the RCT have 28.82% less error compared to tree-based GBDT model. Moreover, the RCT outperforms the state-of-the-art transformer-based tabular model, FTTransformer, by 6.08%. We also illustrate that the RCT learns a generalized manifold of the rate card that can improve the performance of tree-based models.

1. Introduction

Amazon ships packages in the order of billions annually to its customers in the United States alone. The route planning for these packages is done on the day of shipping, *day 0*. As part of this plan, the shipping cost for each package is estimated by breaking down the package journey into smaller legs, and calculating the cost of each leg using a rate card. Day 0 cost estimates are used to compute initial profitability estimates for accounting purposes,

* These authors contributed equally to this work.
† Work done during internship at Amazon

e.g.the estimate of profit/loss for each item as a result of a specific sale to a customer. These profitability estimates are used by several downstream services for decision making and planning.

However, the day 0 estimates may differ from the actual cost due to factors like improper rate-card configuration, incorrect package dimensions, wrong delivery address, etc. Inaccurate cost estimates cause skewed profitability estimates, which in turn leads to suboptimal financial decisions by downstream systems. For example, if the shipping cost of an item is consistently overestimated, then the item could be removed from the catalog. On the other hand, underestimated cost can lead pricing systems to lower the price of the item, leading to losses. Further, inaccurate estimation also leads us to promote wrong products to the customer, causing bad customer experience. To improve these shipping cost estimates, we propose a Transformer based deep learning model that accurately predicts the shipping cost at day 0.

In the context of shipping, a package is characterized by its physical dimensions, weight, and contents. It also include details about the carrier responsible for transporting it and the intended route. Additionally, a package is associated with a variable number of attributes that describe the item(s) inside and the various charges related to its shipment. Collectively, we refer to these attributes as the *rate card* associated with the package. For tabular datasets like package rate cards, tree based models like Gradient Boosted Decision Trees (GBDT), XGBoost (Chen and Guestrin, 2016), etc., are considered as state-of-the-art (SOTA) models. However, their effectiveness heavily relies on high-quality input features (Arik et al., 2019) which can require extensive feature engineering. For our use case, this problem is further accentuated by the fact that the target concept depends on high order combinatorial interactions between rate card attributes. For example, if the rate card is improperly configured for large containers with flammable substances shipped from Washington DC to New York by ABC carrier, then the model has to learn to associate property combination $< size = large,\ item = flammable,\ source = Washington,\ destination = New\ York,\ carrier = ABC >$ with high deviation between estimated and actual costs. When dealing with feature combinations, considering all possible higher-order interactions between package properties may be impractical due to the exponential increase in the number of interactions with each increase in order, leading to the curse of dimensionality (Bishop, 2006). Another shortcoming of tree based models is their inability to handle a variable length list of features. A package may contain multiple items, and its ship cost can be broken down into multiple charges types. Previous experiments demonstrated that adding features engineered from multiple items and charges improved GBDT's performance. However, due to inability of tree based models to handle variable list of features, complete information from them could not be learned.

In this paper, inspired by the recent success of transformers in tabular domain (Huang et al., 2020; Somepalli et al., 2021; Gorishniy et al., 2021), we propose a novel architecture called the Rate Card Transformer (RCT) to predict ship cost on day 0. The proposed model is specifically designed to learn an embedding of rate card associated with a package. The RCT leverages self-attention mechanisms to effectively capture the interdependencies between various components in a rate card by learning interactions between input features. Specifically, our contributions in this work include:

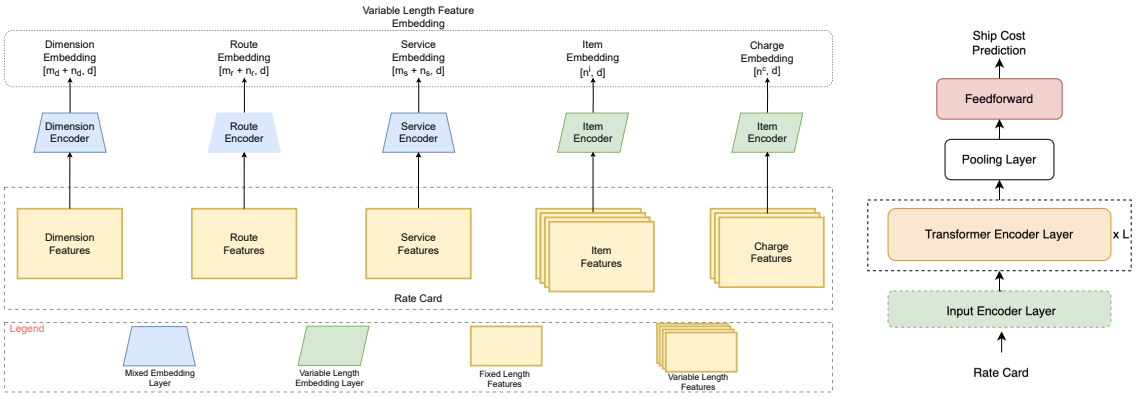
- Propose a novel architecture, *Rate Card Transformer* (RCT), which leverages transformer architecture to learn a manifold of the rate card, to predict shipping cost on Day 0. Further, it is demonstrated that RCT outperforms both GBDTs and the state-of-the-art tabular transformer, FT-Transformer, (Gorishniy et al., 2021) in shipping cost prediction.

- Extensive experiments are performed to show that the learned embeddings are a sufficient representation of the rate card manifold, and self-attention layers are effective feature interaction learners. Ablation studies are performed to analyze the impact of number of transformer layers and attention heads on model performance.

2. Related Works

Tree-based algorithms are widely used in machine learning for tabular data. Decision trees recursively split the data into multiple parts based on axis-aligned hyper-planes (Hastie et al., 2009). Random Forests (RF) (Breiman, 2001) and Gradient Boosted Decision Trees (GBDT) (Friedman, 2001) are the most commonly used tree based ensembles. RF fits multiple decision trees on random subsets of the data and averages/polls the predictions to alleviate the overfitting characteristic of decision trees. GBDT, XGBoost (Chen and Guestrin, 2016), and CatBoost (Prokhorenkova et al., 2018) are boosted ensemble models that sequentially build decision trees to correct errors made by previous trees, leading to improved performance on complex datasets with non-linear relations.

Recently, there has been a lot of interest in deep learning models for tabular data. Some methods introduce differentiable approximations of decision functions used in decision trees to make them differentiable (Hazimeh et al., 2020; Popov et al., 2019). These methods outperform pure tree based problem for some problem statements, however, they are not consistently better (Gorishniy et al., 2021). Other methods have used attention mechanisms to adapt DL methods to tabular data (Arik et al., 2019; Huang et al., 2020; Gorishniy et al., 2021; Somepalli et al., 2021; Chen et al., 2022). TabNet (Arik et al., 2019) proposes a sparse attention mechanism that is stacked in multiple layers to mimic the recursive splitting of decision trees. Inspired from the success of self-attention transformers (Vaswani et al., 2017) in many domains (Devlin et al., 2019; Dosovitskiy et al., 2021; Gong et al., 2021) methods like TabTransformer (Huang et al., 2020), FT-Transformer (Gorishniy et al., 2021) and SAINT (Somepalli et al., 2021) were proposed. TabTransformer embeds all categorical variables into a unified embedding space, and a sentence of categorical embeddings is passed through self-attention transformer layers. FT-Transformer further extends this by attending to numerical features as well, by using continuous embedding. SAINT builds on FT-Transformer by proposing a new kind of attention which captures interactions between samples of a batch. However, SAINT does not provide any advantage over FT-Transformer for our problem statement, because intersample attention is only effective when the number of dimensions is higher in comparision to the number of samples, thus we do not compare RCT against SAINT (Somepalli et al., 2021).



(a) (b)

Figure 1: (a) Input encoder layer of Rate Card Transformer. (b) RCT Architecture

3. Methodology

3.1. Problem Statement

Rate card information of a shipment is a collection of feature types - dimension, route, service, item, and charge, as illustrated in Fig. 1a. Each type contains m numerical and n categorical features, represented as $\mathbf{x} \in \mathcal{S}[m, n]$. Formally, the rate card representation of a package is $\mathbf{P} = \{\mathbf{d}, \mathbf{r}, \mathbf{s}, \{\mathbf{i}_k\}_{k=1}^{n^i}, \{\mathbf{c}_k\}_{k=1}^{n^c}\}$ where $\mathbf{d} \in \mathcal{S}[m_d, n_d]$ are dimensional features, $\mathbf{r} \in \mathcal{S}[m_r, n_r]$ are route features, $\mathbf{s} \in \mathcal{S}[m_s, n_s]$ are service features, $\mathbf{i}_k \in \mathcal{S}[m_i, n_i]$ are features of k^{th} item, $\mathbf{c}_k \in \mathcal{S}[m_c, n_c]$ are features of k^{th} charge, and n^i, n^c are the number of items and charges in the package.

The objective is to make an estimate $\hat{C}$ of the unknown actual shipping cost C, given the *day 0* heuristic estimate C^A and the rate card $\mathbf{P}$. A functional mapping $f(\mathbf{P}, C^A; \theta) \approx \hat{C}$ with parameters θ is learned from a dataset $\mathcal{D} = \{\mathbf{P}_j, C_j, C_j^A\}_{j=1}^{N}$, of N packages shipped in the past.

3.2. Background

The Transformer architecture (Vaswani et al., 2017) is constructed by stacking multiple encoder blocks, where each block takes a sequence of embeddings as input and outputs a sequence of context aware embeddings. The encoder block consists of a multi-head self-attention (MHSA) layer followed by a position-wise feed-forward layer, with residual connections and layer norm before each layer. The MHSA layer comprises multiple self-attention units called heads, which learn interactions between input embeddings.

The encoder layers are powerful feature aggregators when the input sequence consists of features. The encoder leverages MHSA layers to produce an interaction-aware representation of the input features. This is accomplished through the use of self-attention heads, which computes a weighted sum of the input feature embeddings. To determine the summation weights, called attention scores, the attention mechanism projects the input features using learned matrices into three subspaces, query q_i, key k_i and value v_i. The attention score, between two features are computed using Eq.1.

The attentions score $a_{i,j}$ quantifies the interaction between features i and j. The raw attention scores are softmax normalized to ensure that they sum up to 1. The interaction-aware representation of feature i, o_i, which considers interaction between all features, is computed using the equation Eq. 2.

$$a_{i,j}(q_i, k_j) = softmax(\frac{q_i \cdot k_j}{\sqrt{d}}) \tag{1}$$

$$o_i = \sum_j a_{i,j} v_j \tag{2}$$

The output sequence is then recursively passed through subsequent encoder layers, allowing each successive layer to learn higher order feature interactions. The transformer's depth controls the complexity of the learned representation, as deeper layers capture more complex interactions between features. Further, multiple self-attention heads are used in MHSA, enabling each head to attend to different feature sub-spaces and learning interactions between them, cumulatively learning multiple independent sets of feature interactions.

3.3. Rate Card Transformer

The rate card of a package consists of multiple features types, namely dimensional, route, service, item, and charge (Fig. 1a), where each feature type comprises multiple numerical and categorical features. The dimensional, route and service features are referred to as *fixed length feature* types, because each of them have a fixed number of features. Fixed length feature types are embedded to a sequence of tokens using a *mixed embedding layer* (MEL). For example, dimensional features $\mathbf{d} \in \mathcal{S}[m_d, n_d]$ are embedded to a d-dimensional token sequence of length $m_d + n_d$. The MEL contains multiple embedding blocks, one for each feature in the feature type being embedded. Embedding lookup tables are used for embedding categorical features, while numerical features are embedded using continuous embedding blocks, as introduced in (Gorishniy et al., 2021).

Unlike fixed length features, a package is associated with variable number of items and charges, thus these are referred to as variable length features. Each item is first embedded through a mixed embedding layer, creating a sequence of tokens. The sequence is reduced to a single token, creating a single embedding for each item. These set of layers is collectively referred to as *variable length embedding layer*. Charges are also embedded similarly. After all features have been embedded, a token sequence of length $[(m_d + n_d) + (m_r + n_r) + (m_i + n_i) + n^i + n^c]$ is constructed. This is called the *rate card embedding*.

The sequence of feature tokens is passed as input to a stack of L Transformer encoder layers that are able to learn complex, higher order interactions between the features. Finally, the pooled Transformer output is fed to a feedforward layer to predict the shipping cost $\hat{C}$ as shown in Fig. 1b.

We call the complete architecture the *Rate Card Transformer* (RCT). Trained to minimize the L1 loss between the predicted and actual shipping cost (Equation 3), RCT learns an effective representation of the dynamic rate card that allows it to accurately predict the shipping cost.

$$\underset{\theta}{\operatorname{argmin}} \underset{\mathbf{P},C,C^A \sim \mathcal{D}}{\mathbb{E}} \left[\left| C - f\left(\mathbf{P}, C^A\right) \right| \right] \tag{3}$$

4. Experiments

In this section, the performance of the RCT is demonstrated on a dataset of packages shipped in 2022. The mean absolute error (MAE) between the predicted and actual shipping cost is selected as the performance metric, as it is representative of the absolute error in monetary terms. In this paper, the MAE values are normalized by the MAE of day 0 heuristic estimate, which is expressed as MAE percentage ($MAE_{\%}$). This metric emphasizes the improvement achieved against the heuristic baseline.

$$MAE_{\%} = \frac{\sum_i^N |C_i - \hat{C}_i|}{\sum_i^N |C_i - C_i^A|} \times 100 \tag{4}$$

4.1. Experimental Setup

4.1.1. Architecture and Hyperameters

The embedding dimension was set to 128, and 6 transformer encoder layers were used, each with 16 self-attention heads. Adam optimizer (Kingma and Ba, 2014) with a starting learning rate of 0.0001 and a batch size of 2048 was used. To improve convergence, the learning rate was reduced by a factor of 0.7 every time the validation metric plateaued. The model code was implemented using the PyTorch (Prokhorenkova et al., 2018) and PyTorch Lightning (Falcon and The PyTorch Lightning team, 2019) frameworks.

4.1.2. Data Preparation

A training dataset of 10M packages was sampled from packages shipped during a 45-day period in 2022. The data was preprocessed by label encoding categorical features and standardizing numerical features. The test dataset contains all packages (without sampling) that were shipped during a separate, non-overlapping week from 2022.

4.1.3. Benchmark Methods

We compare the performance of RCT against various models with increasing level of complexity: GBDT, AWS AutoGluon (Erickson et al., 2020), Feedforward neural network, TabTransformer and FT-Transformer. For GBDT model, numerical features were not standardized, and target encoding (Micci-Barreca, 2001) was used to encode categorical features instead of label encoding. AWS AutoGluon was configured to learn an ensemble of LightGBM (Ke et al., 2017) models. A feedforward neural network containing 5 layers was used, the input to which was generated by embedding and concatenating dimension, route and service features. Publicly available implementations [1] of TabTransformer and FT-Transformer were used, and all hyperparameters were made consistent with RCT. Since the baselines do not handle collections of items and charges, we only used dimension, route and service features.

1. https://github.com/lucidrains/tab-transformer-pytorch

Table 1: (a) compares the performance of the RCT against various benchmarks, (b) compares the performance of GBDT baseline with GBDT trained with RCT embeddings. $MAE_{\%}$ is calculated as shown in Equation 4.

(a)

Model	$MAE_{\%}$
GBDT baseline	78.29
AutoGluon	77.59
Feed Forward	67.13
TabTransformer	69.58
FT-Transformer	59.33
RCT	**55.72**

(b)

Model	$MAE_{\%}$
GBDT + manual features	78.29
GBDT + manual features + RCT embedding	68.50
GBDT + RCT embedding	69.21

Table 2: $MAE_{\%}$ ***comparison between RCT and FT-Transformer (SOTA for self-attention models)***

Layers	**nheads**	**d_model**	**RCT** $MAE_{\%}$	**FT-Transformer** $MAE_{\%}$
1	4	32	73.82%	76.31%
3	8	64	61.95%	63.11%
6	16	128	55.73%	59.34%

4.2. Baseline Comparisons

Table 1a compares RCT against the baseline models discussed in section 4.1.3. The models in the table are organized in increasing order of model complexity. Both tree based models, GBDT and AutoGluon, are performing at a similar level. Deep learning models consistently outperform tree based models, indicating that the proposed architecture is efficient for shipping cost prediction. Transformer based models have lower $MAE_{\%}$ scores than feedforward neural network, showing that transformers learn effective interaction. The RCT model outperforms both transformer models - TabTransformer and FT-Transformer (SOTA), suggesting that a custom architecture which encodes the latent structure of the rate card is contributing to the improved performance. Table 2 compares the performance of FT-Transformer and RCT models at different model sizes. The results show that RCT outperforms FT-Transformer across all tested model sizes, showing indicates that encoding rate card structure provides performance benefits across varying model capacities.

4.3. Does RCT learn effective representation of rate cards?

Transformers have been shown to have strong representation learning capabilities in a variety of tasks. In this experiment, we investigate the effectiveness of rate card representation learned by RCT. To evaluate this, we compare the performance of our GBT model with and without the learned rate card representation as an input feature.

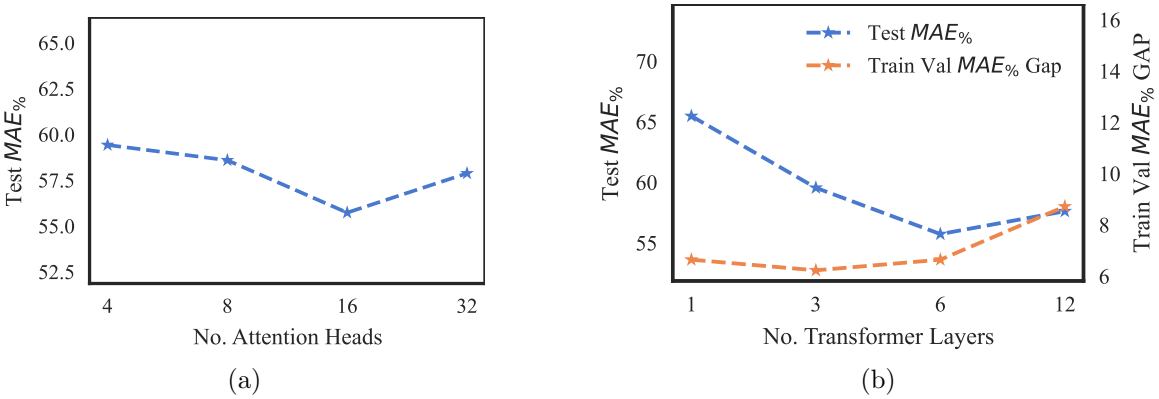


Figure 2: Figure a plots test $MAE_{\%}$ vs number of attention heads. Figure b plots test $MAE_{\%}$ and train-val $MAE_{\%}$ gap vs number of transformer layers. $MAE_{\%}$ is calculated as shown in Equation 4.

The pooled output of the final Transformer layer is treated as the learned representation of the rate card. Adding this feature improved the performance of the GBDT by 9.79% (refer Table 1b). Further, it was observed that even when all manually engineered features are dropped, the GBDT still performs comparably, with an MAE percentage of 69.21%. This indicates that the learned representations of rate cards are not only effective at capturing better feature information, but are also sufficient representation of the package rate card. However, even with this feature, the GBDT has a 13.5% higher $MAE_{\%}$ than the RCT. This is likely because the RCT is trained end-to-end, while the GBDT uses features learned as part of a separate model.

4.4. Does self-attention learn better interactions than feed forward neural networks?

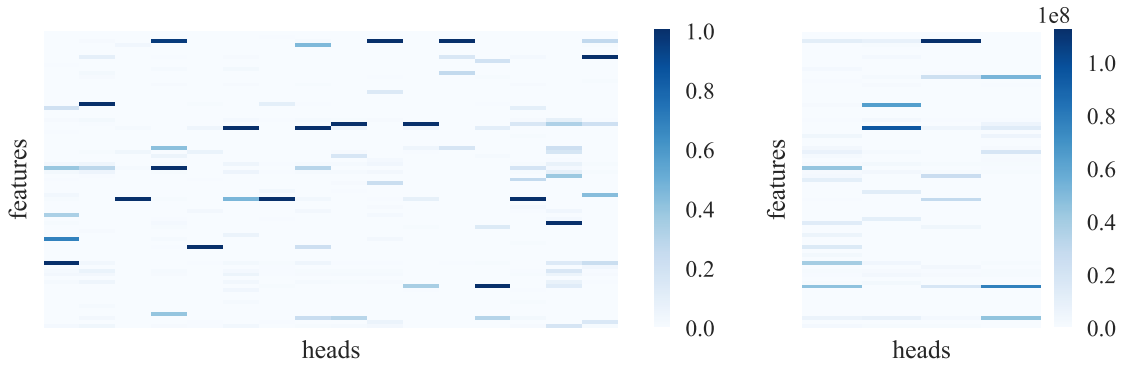
In section 4.2, it was observed that feed forward (FF) neural networks were outperformed by transformers, leading to the hypothesis that self-attention is a superior interaction learner. This section aims to explore this hypothesis further by utilizing FF instead of self-attention to encode dimension, route and service features while limiting the width of self-attention to only the item and charge features. The output encodings of both FF and self-attention are concatenated and fed into an FF layer to predict shipping cost. As the self-attention width is decreased, it fails to capture the interactions between all rate card features. The resulting model exhibits a higher $MAE_{\%}$ of 64.73% in comparison to the RCT's 55.72%. These results suggest that FF models are inferior interaction learners in comparison to transformers.

4.5. Analysis of Self-Attention

In section 3.2, we discussed the proficiency of transformers in feature aggregation, owing to self-attention. In this section, ablation experiments are conducted to analyze the effect of attention depth and attention head count. Increasing the number of attention heads allows the model to learn more independent feature interactions. For this experiment, the

Algorithm 1: Extraction of most attended feature per head

Data: Package Dataset $\mathcal{D}$, self-attention heads $\mathcal{H} = \{\mathrm{h}_i\}$ and N features
Result: Heat map of most attended features
$heat\_map \leftarrow []$;
for $\mathrm{h} \in \mathcal{H}$ **do**
  $attention\_importance \leftarrow [0]_{i=0}^{N}$;
  for $\mathbf{P} \in \mathcal{D}$ **do**
    Compute attention map $A = \{a_{i,j}\}$ of h for $\mathbf{P}$;
    Increment $attention\_importance[j]$ by 1 for top five interaction $\{i, j\}$ from A;
  end
  $attention\_importance \leftarrow \frac{attention\_importance - min(attention\_importance)}{max(attention\_importance) - min(attention\_importance)}$ Append $attention\_importance$ to $heat\_map$
end



(a) 16 Attention Heads

(b) 4 Attention Heads

Figure 3: Heatmaps generated from 1. Each column shows the relative importance of each feature in a head, and each column corresponds to a different head.

model capacity is fixed at 128 dimensions, so an increase in the number of heads also reduces the complexity of interactions learned per head. Thus, choosing optimal head count is a trade-off between learning independent interactions and the complexity of each learned interaction. The trade-off can be observed in Fig. 2a, where the performance improves from 4 heads to 16 heads because the attention learned by each head is complex enough. However, the performance degrades when attention heads are increased from 16 to 32 because the complexity of heads has reduced substantially, negating the benefit of learning more independent interactions.

Next, we illustrate the effect of increasing the attention depth by adding transformer encoder layers. Deeper transformer networks learn more complex higher-order interactions, thereby enhancing the model's performance, as observed in Fig. 2b. However, increasing the number of layers from 6 to 12 reduces the model's performance due to overfitting, caused by the rise in learnable parameter count. The evidence for overfitting can be found in Fig.

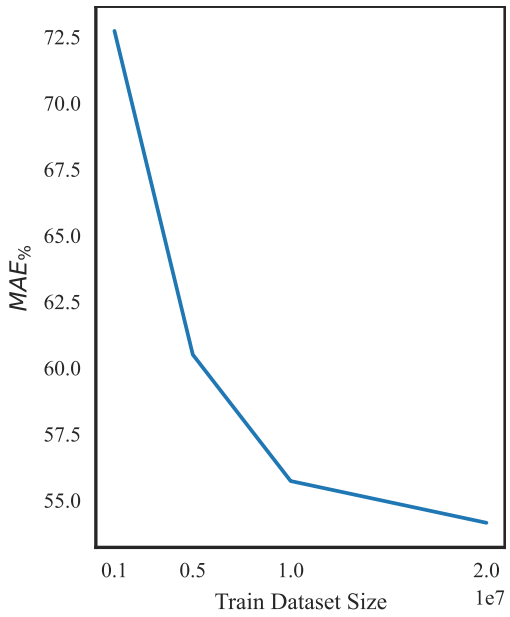


Figure 4: Scaling of RCT with data

2b, where the gap between train and val MAE increases by 30% when moving from 6 to 12 layers.

Finally, in Fig. 3, we display the heat maps generated using Algorithm 1. These heat maps illustrate the number of times each feature was attended to as part of the top five most attended features. Each column corresponds to a head, and each row corresponds to a feature. The heat map on the left was generated using RCT with $nheads = 16$, and the one on the right was generated with $nheads = 4$. Comparing both the heat maps, it can be seen that Fig. 3a has less number of active feature interactions per column, confirming our hypothesis that a larger number of attention heads leads to each head learning independent interactions between features.

4.6. How does the Transformer scale with more data?

To minimize the experimentation costs, all experiments in this paper were conducted using a training dataset of size 10 million. However, it is important to use the best performing model, the training dataset size can be increased to achieve optimal performance.

To verify the scalability of RCT with data, we trained the model on different training dataset sizes and plotted the results in Fig. 4. The results demonstrate that RCT's performance continues to improve with larger datasets. Therefore, we can confidently expect that models trained on larger datasets will outperform the model explored in this paper.

5. Conclusion and Future Work

In this paper, we presented a novel framework based on the Transformer architecture for predicting shipping costs on day 0. Our proposed framework encodes shipping attributes of a package, i.e., the package rate card, into a uniform embedding space. These embeddings

are then fed through a Transformer layer, which models complex higher-order interactions and learns an effective representation of the package rate card for predicting shipping costs. Our experimental results demonstrate that the proposed model, called RCT, outperforms GBDT model by 28.8%. Furthermore, demonstrate the RCT performs better than SOTA model FT-Transformer for our problem statement. We also show that when rate card representation learned by RCT is added to GBDT model, its performance is improved by 12.51%. This underscores the fact that RCT is able to learn sufficient representation representations of rate card information.

In this work, the route information used was limited to the start and end nodes alone. Future work could explore the use of Graph Neural Networks to encode information about the complete route. Further, the performance of the RCT might be improved by exploring ways to include the item ID as a feature, such as the use of item embeddings which are available internally.

Also, while the RCT was trained to predict only the ship cost, it can be modified to predict all the attributes of the invoice by adding a Transformer decoder layer. This would enable other applications like invoice anomaly detection. Additionally, future research could investigate whether the package representations learnt by the RCT can be used to improve the performance of other related tasks or to quantify the model uncertainty in each prediction via approaches like the one proposed in Amini et al. (2019).

References

Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. *CoRR*, abs/1910.02600, 2019. URL http://arxiv.org/abs/1910.02600.

Sercan O Arik, Engin Gedik, Kenan Guney, and Umut Atilla. Tabnet: Attentive interpretable tabular learning. In *Advances in Neural Information Processing Systems*, pages 10951–10961, 2019.

Christopher M Bishop. Pattern recognition and machine learning. In *Springer*, chapter 2, pages 36–43. 2006.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Jintai Chen, Kuanlun Liao, Yao Wan, Danny Z Chen, and Jian Wu. Danets: Deep abstract networks for tabular data classification and regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3930–3938, 2022.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1*

(Long and Short Papers), pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data, 2020.

William Falcon and The PyTorch Lightning team. PyTorch Lightning, 3 2019. URL https://github.com/Lightning-AI/lightning.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Yuan Gong, Yu-An Chung, and James Glass. AST: Audio Spectrogram Transformer. In *Proc. Interspeech 2021*, pages 571–575, 2021. doi: 10.21437/Interspeech.2021-698.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943, 2021.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

Hussein Hazimeh, Natalia Ponomareva, Petros Mol, Zhenyu Tan, and Rahul Mazumder. The tree ensemble layer: Differentiability meets conditional computation. In *International Conference on Machine Learning*, pages 4138–4148. PMLR, 2020.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Daniele Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *SIGKDD Explor. Newsl.*, 3(1):27–32, jul 2001. ISSN 1931-0145. doi: 10.1145/507533.507538. URL https://doi.org/10.1145/507533.507538.

Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*, 2019.

Liudmila Prokhorenkova, Gleb Gusev, Alexey Vorobev, Anna Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 31:6638–6648, 2018.

Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.