Quantifying Explanations of Neural Networks in E-Commerce Based on LRP*

Anna Nguyen✉[0000−0001−9004−2092], Franz Krause[0000−0003−2869−6815], Daniel Hagenmayer, and Michael Färber[0000−0001−5458−8645]

Karlsruhe Institute of Technology, Karlsruhe, Germany
anna.nguyen@kit.edu
franz.krause@student.kit.edu
daniel.hagenmayer@student.kit.edu
michael.faerber@kit.edu

Abstract. Neural networks are a popular tool in e-commerce, in particular for product recommendations. To build reliable recommender systems, it is crucial to understand how exactly recommendations come about. Unfortunately, neural networks work as black boxes that do not provide explanations of how the recommendations are made.
In this paper, we present *TransPer*, an explanation framework for neural networks. It uses novel, explanation measures based on *Layer-Wise Relevance Propagation* and can handle heterogeneous data and complex neural network architectures, such as combinations of multiple neural networks into one larger architecture. We apply and evaluate our framework on two real-world online shops. We show that the explanations provided by *TransPer* help (i) understand prediction quality, (ii) find new ideas on how to improve the neural network, (iii) help the online shops understand their customers, and (iv) meet legal requirements such as the ones mandated by GDPR.

1 Introduction

The breakthrough with neural networks as a pattern recognition technique has lead its way into many industry sectors. Especially in e-commerce, it can be used as recommender system for advanced searches [12], personalization of shopping experiences and direct marketing [20], or advanced sales forcasting and predictions [14]. Improving the predictions and the usefulness of those recommenders can increase sales and customer satisfaction. Additionally, there is increasing legal pressure in favor of privacy and data protection. For example, the General Data Protection Regulation [10] (GDPR) states that data subjects should be enabled to check the collection, processing, or use of their data. Thus, businesses may be legally required to make their recommender systems transparent.

Multilayer Perceptrons (MLP) have been applied in recommender systems learning feature representations as an extension to collaborative filtering [11].

* Supported by the German Research Ministry (BMBF), the Smart Data Innovation Lab (01IS19030A), and the company econda GmbH.

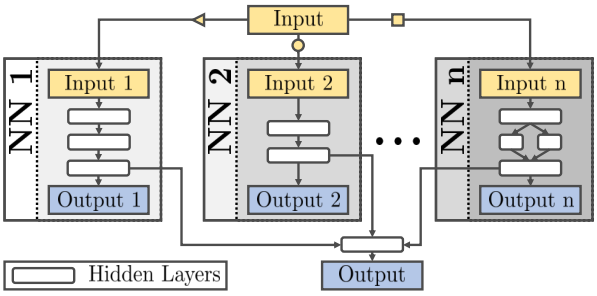

Fig. 1: Model of a neural network with different input data types

In combination with convolutional layers, they are applied to generate fashion outfits for e-commerce or to personalize outfit recommendations based on learned embeddings in Convolutional Neural Networks (CNN) [3, 7]. Recurrent Neural Networks (RNN) have shown success in modelling sequential data and have been used for personalized product recommendations based on the purchase patterns of customers [17], learning embeddings of fashion items [13] and modelling user behaviour to predict clicks [5].

However, neural networks are black box models, i.e., the predictions can not be explained. In order to tackle this, it is beneficial to make them more transparent and therefore, more human-understandable. Typically, the Gradient-based Sensitivity Analysis [21] is used to explain the predictions of neural networks. By optimizing the gradient ascent in the input space, it is possible to determine which inputs lead to an increase or decrease of the prediction score when changed [23, 25]. Although applications based on this method enable a statement regarding positive or negative influence of an input on a prediction, they do not reveal a quantitative decision-relevant input score such as Guided Backpropagation [24], DeconvNet [19], or DeepLIFT [22]. These algorithms use the trained weights and activations within the forward pass to propagate the output back to the input. This way, it is possible to determine which features in an input vector contribute to the classification and to what extent. Exploiting this, ObAlEx [18] is an explanation quality metric which measures to what extent the classified object is aligned to the mentioned explanations. Nonetheless, all these methods are solely applied to CNNs with image data where single pixels are then highlighted. Another back-propagating algorithm is the Layer-Wise Relevance Propagation (LRP) that has already been successfully used in interaction with MLPs and CNNs [1, 2, 15]. LRP computes the relevance of each input neuron to an output by performing a value-preserving backpropagation of the output. Furthermore, this method is even applicable on RNNs with sequential data [4, 16] which often occurs in processing customer profiles in e-commerce.

Contribution. Our contribution is threefold. First, we provide an explanation framework called TransPer[1] for e-commerce businesses in online shopping

[1] We provide the source code online at https://github.com/Krusinaldo9/TransPer.

(e.g., for product recommendation) to provide transparency to the neural networks used. Based on a custom implementation of *Layer-Wise Relevance Propagation*, our approach can not only handle individual neural networks types, but also more complex architectures that contain multiple neural subnetworks, such as shown in Fig. 1. This is required in the presence of highly heterogeneous input data (e.g., product images, chronological shopping interactions, personal information) where different neural network types are necessary (e.g., CNN, RNN, MLP). We not only take into account the relevance of the activations of the neurons, but also the bias. This has not been considered in depth in the literature. Second, we define quantity measures to evaluate the helpfulness of these explanations. The individuality measure can be used to determine those parts of the input that are particularly relevant for the decision. The certainty measure quantifies how certain the system is about its prediction. The diversity measure states whether there are clear top predictions. Third, we evaluate our approach on real-world scenarios. To this end, we used data from two real-world online shops provided by our partner *econda*, an e-commerce solution provider. We show that TransPer helps in (i) understanding the prediction quality, (ii) finding ideas to improve the neural network, and (iii) understanding the customer base. Thus, TransPer brings *trans*parency to *per*sonally individualised automated neural networks and provides new knowledge about customer behaviour. We believe that this helps to fulfill GDPR requirements.

The remainder of this paper is structured as follows. After introducing preliminary definitions and concepts in Section 2, we go on to describe the problem setting and formally define an online shop in Section 3, to introduce our quantity measures in Section 4. We evaluate our approach on the basis of a real-world scenario in Section 5 before ending with some concluding remarks.

2 Preliminaries

In this section, we present the fundamentals for the application of our approach. To begin with, we consider a trained neural network with $K \in \mathbb{N}$ layers as shown on the left-hand side of Fig. 2. We refer to Π_k as the set of all neurons in the k-th layer, σ as a nonlinear monotonously increasing activation function, z_i^k as the activation of the i-th neuron in the k-th layer, $w_{ij}^{k,k+1}$ as the weight between the neurons z_i^k and z_j^{k+1}, and b_j^k as the bias term w.r.t. z_j^{k+1}. Assuming that we know the activations in Π_k, the activations in Π_{k+1} can be determined via forward pass as follows:

$$z_j^{k+1} = \sigma\left(\left(\sum_{i \in \Pi_k} z_i^k w_{ij}^{k,k+1}\right) + b_j^k\right) \tag{1}$$

For non-connected neurons z_i^k and z_j^{k+1} we assume $w_{ij}^{k,k+1} = 0$. If a network has no bias, then $b_j^k = 0$.

Layer-Wise Relevance Propagation is a method that represents a backward analysis method [2]. Knowing the activations z_j^{k+1} in layer $k+1$, we can determine

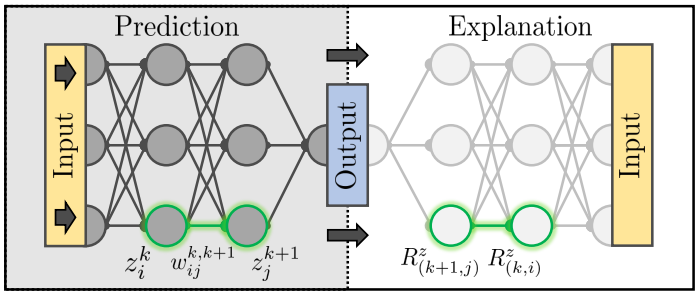

Fig. 2: Exemplary run of LRP. The left-hand side shows the calculation of neuron activations in the forward pass. These activations are then part of the calculation of its relevances in the backward analysis depicted on the right-hand side.

to what extent the neurons in Π_k and the biases b_j^k have contributed, or how relevant they were. The idea behind the standard implementation of the LRP algorithm can be found on the right-hand side of Fig. 2 and is defined as

$$R^z_{(k,i)} = \sum_{j \in \Pi_{k+1}} \frac{z_i^k w_{ij}^{k,k+1}}{\left(\sum_{i \in \Pi_k} z_i^k w_{ij}^{k,k+1} \right) + b_j^k} R^z_{(k+1,j)}, \tag{2}$$

$$R^b_{(k,j)} = \frac{b_j^k}{\left(\sum_{i \in \Pi_k} z_i^k w_{ij}^{k,k+1} \right) + b_j^k} R^z_{(k+1,j)}. \tag{3}$$

For a layer $k+1$, we assume for each neuron j that a relevance can be assigned in the form of a real-valued number $R^z_{(k+1,j)}$. Using Equation 2, we obtain the relevance, i.e., quantitative contribution, of the i-th neuron in the k-th layer to the overall relevance of layer $k+1$. Furthermore, Equation 3 provides the relevance of the bias b_j^k of the j-th neuron in layer $k+1$.

In certain applications, customized variations of the standard LRP algorithm presented above can be considered to increase the performance. In particular, with respect to the explainability of CNNs, it has been found that adapted LRP methods lead to better results than the standard LRP method [1, 2, 15]. These are characterized, e.g., by the use of tuning parameters or penalty terms for negative neuron activations. Regarding RNNs, however, hardly any results exist concerning the use of such variations. Therefore, in relation with the use cases in Section 5, we provide results of a test study comparing well-known customizations with the standard method.

3 Formal Model of an Online Shop

In this section, we define an online shop with regard to a suitable neural network which can handle specific characteristics. Especially, we include heterogeneous

input data such as interest in products or interactions with products which additionally can have different input lengths. In order to generalize our definition, we consider a neural network consisting of several neural subnetworks to cover different cases as can be seen in Fig. 1. Considering all this, we define our online shop as follows.

Definition 1 (Online Shop Model). *We define an online shop T as a tuple*

$$T = (C, P, (P^*, \Phi), \Lambda, \Lambda^*, S, (\Omega_c)_{c\in C}, (\omega_c)_{c\in C}, (f_c)_{c\in C})$$

with the following entries:

a) We denote C as the finite set of all customers of the shop.
b) Let P be the finite set of all products that the shop offers.
c) Then, let P^* be a subset of P or P itself, i.e., $P^* \subseteq P$, and Φ denotes the real-valued output space $[0,1]^{|P^*|}$.
d) We denote Λ as the set of information types that the shop T can have about one of its customers $c \in C$ and assume that this amount is finite.
e) We define Λ^* as a finite set of disjoint subsets $\Lambda_1, ..., \Lambda_n$ of Λ which corresponds to neural networks $S = \{s_1, \cdots, s_n\}$.
f) For a customer $c \in C$ we define an associated real-valued input space

$$\Omega_c = \mathbb{R}^{m_1(c)} \times ... \times \mathbb{R}^{m_n(c)}$$

with the mappings $m_i : C \to \mathbb{N}$ for $i \in \{1, .., n\}$ with respect to s_i.
g) Considering a particular customer $c \in C$, we define his input as $\omega_c \in \Omega_c$.
h) For a customer $c \in C$, we also define the mapping $f_c : \Omega_c \to \Phi$ where $f_c(x)$ is the recommender's output vector for an input $x \in \Omega_c$.

Assume we have an online shop T with customers C. The online shop has a catalogue of offered products P. Though, not all products are predicted for example only seasonally available ones or most purchased ones in the last week denoted by P^*. These are used as output space Φ in the neural network, i.e., if $\Phi(p) > \Phi(p')$ then product p is recommended. Now, consider the types of information Λ the online shop can have about their customers such as already purchased products, interactions, or ratings. As mentioned in Section 1, certain network types are more suitable for specific data types. Therefore, this information is then classified into disjoint information types, such as sequential data Λ_1, graphical data Λ_2, etc., and summarized in Λ^*. So, if an online shop T has heterogeneous user data, Fig. 1 would consist of neural subnetworks $s_1, \cdots, s_n$. With homogeneous data, we would have a special case of the previous one. Hence, we have:

1. $\Lambda = \Lambda'$ and $\Lambda^* = \{\Lambda'_1, ..., \Lambda'_n\}$. (heterogeneous data) (4)
2. $\Lambda = \Lambda'_i$ for some i and $\Lambda^* = \{\Lambda'_i\}$. (homogeneous data) (5)

The different Λ'_i can have different input lengths depending on the sequence length of the interactions or the size of the images. So, we use the mappings m_i

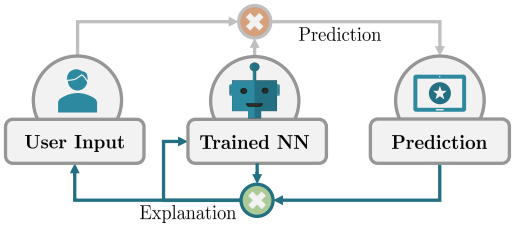

Fig. 3: TransPer Overview.

to deal with it and summarize them in Ω. Thus, for a customer $c \in C$, we obtain the neural network's output vector $y = f_c(\omega_c)$.

Considering the different data types, the online shop has three possibilities to define a suitable neural network: (i) The online shop uses n different data types, i.e., heterogeneous data, and needs n different neural subnetworks. An overall decision is obtained by concatenating the hidden layers at a suitable positions, see Fig. 1. (ii) Second, the online shop decides to just use one data class, i.e., homogeneous data, and therefore has just one neural subnetwork in Fig. 1. However, important information can be lost from the other data classes. (iii) Third, it is possible to define suitable neural subnetworks for $n > 1$ data classes, train them separately and then save their weights. These n trained neural subnetworks can be concatenated and trained again with the entire data, using the already trained weights and biases as initial values. This approach is therefore a combination of the two mentioned possibilities above. Thus, $n+1$ neural subnetworks are obtained in total, with one resulting from the concatenation of the n individual neural subnetworks. The output vector then depends on whether one uses the concatenated network s_{n+1} or one of the neural subnetworks $s_1, ..., s_n$. This third possibility will be relevant for our use case.

4 Explanation Approach

The goal of our approach is to evaluate the explanation of product recommendations of a shop-adapted neural network in order to better understand the decision. Given an input from a user of an online shop and a trained neural network as recommender, TransPer performs a backward analysis based on an individual prediction. In this way, it can be explained to what extent components of the trained network or certain inputs were relevant. This process can be seen in Fig. 3. In the following, we will (i) describe how these explanations can be gained with LRP, (ii) specify how to analyze the input with Leave-One-Out method and (iii) define quantity measures to evaluate the explanations.

4.1 Explanation via Layer-Wise Relevance Propagation

Following the notation of Section 2 and Definition 1, we assume that $K \in \mathbb{N}$ denotes the number of layers in the neural network, i.e., the first layer is the in-

put layer and the K-th layer is the output layer. Furthermore, for $k \in \{1, ..., K\}$ let $|\Pi_k| = I_k \in \mathbb{N}$ be the number of neurons in the k-th layer, i.e., I_1 describes the number of input neurons and I_K the number of output neurons. Indeed, in the context of classifiers, each neuron of the output layer represents one element of the target set. For example, for an input x, the neuron (K, i^*) with the highest prediction score $f(x)_{i^*}$ as output is the actual recommendation. In this context, it is then of interest to find out to what extent the neurons of the lower layers contributed to the decision $f(x)_{i^*}$. For our approach, we define the initial relevance vector $R^z_{(K,\cdot)} := (R^z_{(K,i)})_{i \in \{1,...,I_K\}}$ with

$$R^z_{(K,i)} = \begin{cases} f(x)_{i^*} & \text{if } i = i^* \\ 0 & \text{otherwise} \end{cases}$$

which can be used to iteratively compute the relevance for layers $K-1, ..., 1$ using Equation 2 and Equation 3. Finally, we obtain $R^z_{(1,\cdot)}$ as the input layer's relevance vector and can thus determine to what extent an input neuron is decision-relevant (see Fig. 2). Note that a negative relevance in an input neuron diminishes the prediction i^* whereas a positive relevance underpins it. In contrast to most LRP approaches, we also consider the relevance of the bias $R^b_{(k,j)}$ of the j-th neuron of the $(k+1)$-th layer. Our LRP method is characterized as follows:

$$\begin{aligned}
\sum_{i \in \Pi_k} R^z_{(k,i)} + \sum_{j \in \Pi_{k+1}} R^b_{(k,j)} &= \sum_{i \in \Pi_k} \sum_{j \in \Pi_{k+1}} \frac{z_i^k w_{ij}^{k,k+1}}{\left(\sum_{i \in \Pi_k} z_i^k w_{ij}^{k,k+1}\right) + b_j^k} R^z_{(k+1,j)} \\
&\quad + \sum_{j \in \Pi_{k+1}} \frac{b_j^k}{\left(\sum_{i \in \Pi_k} z_i^k w_{ij}^{k,k+1}\right) + b_j^k} R^z_{(k+1,j)} \\
&= \sum_{j \in \Pi_{k+1}} \frac{\sum_{i \in \Pi_k} z_i^k w_{ij}^{k,k+1}}{\left(\sum_{i \in \Pi_k} z_i^k w_{ij}^{k,k+1}\right) + b_j^k} R^z_{(k+1,j)} \\
&\quad + \sum_{j \in \Pi_{k+1}} \frac{b_j^k}{\left(\sum_{i \in \Pi_k} z_i^k w_{ij}^{k,k+1}\right) + b_j^k} R^z_{(k+1,j)} \\
&= \sum_{j \in \Pi_{k+1}} \frac{\left(\sum_{i \in \Pi_k} z_i^k w_{ij}^{k,k+1}\right) + b_j^k}{\left(\sum_{i \in \Pi_k} z_i^k w_{ij}^{k,k+1}\right) + b_j^k} R^z_{(k+1,j)} \\
&= \sum_{j \in \Pi_{k+1}} R^z_{(k+1,j)}.
\end{aligned}$$

As $f(x)_{i^*} = \sum_{j \in \Pi_K} R^z_{(K,j)}$ is satisfied by assumption, we obtain

$$\begin{aligned} f(x)_{i^*} &= \sum_{i \in \Pi_{K-1}} R^z_{(K-1,i)} + \sum_{j \in \Pi_K} R^b_{(K-1,j)} \\ &= \sum_{i \in \Pi_{K-2}} R^z_{(K-2,i)} + \sum_{j \in \Pi_{K-1}} R^b_{(K-2,j)} + \sum_{j \in \Pi_K} R^b_{(K-1,j)} \\ &= \cdots \\ &= \underbrace{\sum_{i \in \Pi_1} R^z_{(1,i)}}_{=:R^z} + \underbrace{\sum_{k=1}^{K-1} \sum_{j \in \Pi_{k+1}} R^b_{(k,j)}}_{=:R^b}, \end{aligned} \tag{6}$$

i.e., the sum of the final relevancies R^z and R^b equals the original output score. By comparing the two summands in Equation 6, the LRP algorithm also provides a method to find out how much relevance R^z, R^b can be assigned to the input neurons and the trained bias, respectively.

4.2 Input Analysis with Leave-One-Out Method

In this section, we want to find out why well-functioning recommenders actually work and provide new insights into the customers' shopping behavior. Additionally, we want to know why an insufficiently functioning recommender delivers meaningless predictions. Therefore, we need to further analyze the explanations gained from LRP regarding their helpfulness, i.e., the impact of an input on the prediction. Using the Leave-One-Out method [26], we evaluate the input relating to the explanations. By consistently leaving one product out by setting its input value to zero, we can observe its effect on the predictions and explanations, see Fig. 4. Assuming a trained neural network, we perform the following steps:

(i) We start with a particular customer and the associated input x which is mapped to an output vector y via the trained network.
(ii) According to Equation 6, for a given output neuron y_{i^*} with $i^* \in \{1, .., I_K\}$ (e.g., the one with the highest prediction score), we compute the associated input relevancies $(R^z_{(1,j)})_{j \in \{1,..,I_1\}}$ and the overall relevance of the bias R^b. Thus, we consider the set of relevancies $R := \{R^b\} \cup \{R^z_{(1,j)} : 1 \leq j \leq I_1\}$.
(iii) For a salient subset of the relevancies $R^* \subset R$ (e.g., the inputs with the highest/lowest relevancies), we set the associated input neurons (marked red in Fig. 4) in x to 0 and obtain the adapted input vector x^*.
(iv) As in Step (i), we map the input x^* to the corresponding output y^* via the same trained network and obtain the test output y^*.

Thus, with steps (i)-(iv), we obtain the input vectors x and x^*, the output vectors y and y^*, and the set of relevancies R. They are used in Section 4.3 to enable the explainability of neural network predictions according to the online shop in Definition 1.

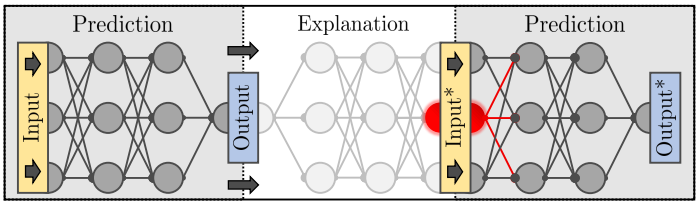

Fig. 4: Selection and analysis of the most relevant inputs via LRP

4.3 Explanation Quantity Measures

Methods such as A/B testing exist to test the performance of a recommender system [9, 6]. They aim at evaluating the predictions trained on a fixed group of customers with new test customers. Ideally, positive feedback on the training process is obtained. However, the results can be unsatisfactory as well. In both cases, it is of interest to know how the predictions come about and how certain inputs influence them specifically. Using Equation 6 and the definitions

$$R^z_+ := \sum_{i \in \Pi_1} \max\{0, R^z_{(1,i)}\}, \quad R^z_- := \sum_{i \in \Pi_1} \min\{0, R^z_{(1,i)}\},$$

we obtain the network's top prediction within the setting of Definition 1

$$y_{i^*} := f_c(\pi_c)_{i^*} = R^z + R^b = R^z_+ + R^z_- + R^b. \tag{7}$$

In the following we consider two disjoint subsets $C_1, C_2 \subset C$. C_1 represents a set of customers where the inconsistencies to be analysed occur. In contrast, this is not the case for customers from C_2. With Equation 7, it is then possible to define measures that can be used to analyse such irregularities in specific test cases. W.l.o.g we always assume for the output value $y_{i^*} > 0$. Based on these considerations, we define three measures to quantify the relevance of the input.

(i) **Definition 2 (Individuality Measure).** $\sigma_T : C \to \mathbb{R}$ *with*

$$\sigma_T(c) := \frac{R^z}{R^z + R^b} = \frac{R^z}{y_{i^*}}.$$

The individuality measure can be used to determine to what extent the input was relevant for the decision. Via Equation 7, we obtain $1 = R^z/y_{i^*} + R^b/y_{i^*}$ and define that a prediction y_{i^*} is *maximally individual*, if $\sigma_T(c) = 1$ holds. In contrast, y_{i^*} is considered to be *minimally individual*, if $\sigma_T(c) = 0$ holds. In this case only the bias was relevant. For $\sigma_T(c) \in (0, 1)$ we generally have $R^z, R^b > 0$, so both of these components contribute positively to y_{i^*}. If R^z or R_b are negative, this component argues against prediction y_{i^*} and we either have $\sigma_T(c) \in (-\infty, 0)$ or $\sigma_T(c) \in (1, \infty)$. Note that due to $y_{i^*} > 0$ it can not occur that R^z and R_b are negative.
With σ_T it is for example possible to attribute inconsistencies to overly homogeneous training data. Consider a shop offering men's and women's

products. Let men be C_1 and women be C_2. If the training data is largely assigned to men, women could often get men's products suggested because the recommender's bias was trained on men. Then, for $c_1 \in C_1$ and $c_2 \in C_2$, the following would apply: $|1 - \sigma_T(c_1)| < |1 - \sigma_T(c_2)|$.

(ii) **Definition 3 (Certainty Measure).** $\nu_T : C \to (0, 1]$ *with*

$$\nu_T(c) := \begin{cases} R^z/R^z_+, & \text{if } R^z > 0 \\ R^z/R^z_-, & \text{if } R^z < 0. \end{cases}$$

The certainty measure can be used to make a quantitative statement about the deviation of the individual relevancies from the overall relevance. Considering definitions of R^z_+, R^z_-, and Equation 7, we have $R^z_+ \in [R^z, \infty)$ and $R^z_- \in (-\infty, R^z]$. Depending on the sign of R^z, one can determine whether the input neurons as a whole had a positive or negative relevance for the decision made. We restrict ourselves to the case of $R^z > 0$. However, the results apply to $R^z < 0$, respectively. Thus, we can deduce that a value of $\nu_t(c) = 1$ means that no negative relevancies were assigned to the input neurons. A value close to zero, on the other hand, indicates a strong dispersion of the relevancies.

(iii) **Definition 4 (Diversity Measure).** $\zeta_T, \zeta^+_T, \zeta^-_T : C \to [0, \infty)$ *with*

$$\zeta_T(c) := \max_{r \in \mathcal{R}} \left| \frac{r - \mu^r_{\mathcal{R}}}{\mu^r_{\mathcal{R}}} \right| \quad \text{and} \quad \mu^r_{\mathcal{R}} := \frac{1}{|\mathcal{R}| - 1} \sum_{r' \in \mathcal{R} \setminus \{r\}} r',$$

$$\zeta^+_T(c) := \max_{r \in \mathcal{R}_+} \left| \frac{r - \mu^r_{\mathcal{R}_+}}{\mu^r_{\mathcal{R}_+}} \right| \quad \text{and} \quad \mu^r_{\mathcal{R}_+} := \frac{1}{|\mathcal{R}_+| - 1} \sum_{r' \in \mathcal{R}_+ \setminus \{r\}} r',$$

$$\zeta^-_T(c) := \max_{r \in \mathcal{R}_-} \left| \frac{r - \mu^r_{\mathcal{R}_-}}{\mu^r_{\mathcal{R}_-}} \right| \quad \text{and} \quad \mu^r_{\mathcal{R}_-} := \frac{1}{|\mathcal{R}_-| - 1} \sum_{r' \in \mathcal{R}_- \setminus \{r\}} r'$$

for a customer $c \in C$ and top prediction y_{i^*}. We additionally introduce the set of input relevancies $\mathcal{R} := R^z_{(1,\cdot)}$, which we divide as follows:

$$\mathcal{R}_0 := \{r \in \mathcal{R} : r = 0\},\ \mathcal{R}_+ := \{r \in \mathcal{R} : r > 0\},\ \text{and } \mathcal{R}_- := \{r \in \mathcal{R} : r < 0\}.$$

The diversity measure finds outliers within certain input relevancies. For example, considering $r \in \mathcal{R}$, then $(r - \mu^r_{\mathcal{R}})/\mu^r_{\mathcal{R}}$ is the proportional deviation between the values in $\mathcal{R}$ except for r. For $r \in \mathcal{R}_+$ or $r \in \mathcal{R}_-$ one proceeds analogously. Note that the calculation of diversity measures does not apply to empty sets $\mathcal{R}, \mathcal{R}_+$, and $\mathcal{R}_-$, respectively. Furthermore, the zero is always obtained for one-element sets. For two customers c_1, c_2 with $\zeta^+_T(c_1) \ll \zeta^+_T(c_2)$, we can thus state that the prediction for c_2 depends more on a single input neuron than the prediction for c_1.

5 Evaluation

In this section, we demonstrate the benefits and application of our approach in three use cases. First, our explanation approach can help in understanding fluc-

tuations in the recommender's quality. Second, TransPer can help in finding ideas on how to improve the recommender. Third, our contribution can help to improve the understanding of the customer base. In the course of this research, we kindly received permission from the e-commerce service provider econda [8] and two of its partner companies to use their customer data. These partner companies are a jewellery shop and an interior design shop.

5.1 Evaluation Setting

At this point, we show that both online shops fit the formal model from Definition 1 and are thus applicable to the TransPer framework. We assume that T^1 is the jewellery shop and T^2 the interior design shop. As shortly mentioned in Section 3, the neural network econda uses for T^1 and T^2 comply with the third neural network type with three neural subnetworks s_1, s_2, s_3 in Fig. 1.

Online Shop Models. We now illustrate how the shops satisfy Definition 1:

a) Both shops provide anonymized information about a variety of their customers $\widetilde{C}^1 \subseteq C^1, \widetilde{C}^2 \subseteq C^2$, for example shopping history,
b) and their offered products P^1, P^2.
c) The targets P^*, in our use case a subset of selected products of the offered products, define the real output space Φ^1 and Φ^2, respectively.
d) The available customer information types are based on the information sets Λ^1 and Λ^2, respectively.
e) The information from Λ^1 (Λ^2) is classified according to its characteristic properties. In our case, the disjoint subsets are the same for both shops, i.e., $\Lambda^* = \Lambda^{1*} = \Lambda^{2*}$. Especially, T^1 and T^2 have three disjunctive information types, i.e., $|\Lambda^*| = 3$, which result in three neural subnetworks s_1, s_2, s_3.
f) According to Λ^*, any customer c has therefore the associated input space denoted by $\Omega_c = \mathbb{R}^{m_1} \times \mathbb{R}^{m_2} \times \mathbb{R}^{m_3(c)}$. The first two neural subnetworks s_1, s_2 have a fixed number of input neurons independent of the customer, so in a slight abuse of notation we write m_1 and m_2 instead of $m_1(c)$ and $m_2(c)$, respectively. The third subnetwork has a number of neurons dependent on the number of interactions of c.
g) Via preprocessing, the information about a user $c \in C$ is converted into an input $\omega_c \in \Omega_c$.
h) The function f_c represents the recommender's implicit process of decision making. Given an input ω_c, the vector $f_c(\omega_c)$ contains an entry for each product in P^* and the product with the corresponding highest prediction score is recommended.

The neural networks are trained in two steps, respectively. First, the neural subnetworks s_1, s_2, s_3 are trained independently. Based on the trained weights and biases, the subnetworks are concatenated according to Fig. 1 in their hidden layers and trained again to obtain the combined decision function f_c. This also means, each of the subnetworks s_1, s_2, s_3 individually fits Definition 1 and

processes the following information types which we will further analyze in Section 5.3. (i) s_1 processes information regarding general interactions, whereby the input vector is an embedding of a user profile. For example, an input neuron can represent the purchase of a certain product or interest in a product category. This neural subnetwork is designed as a multi-layer perceptron. (ii) s_2 processes personal information not related to former product interactions. A multi-layer perceptron is used as well. (iii) s_3 processes the most recent customer interactions as sequences, whose lengths may be different for each customer. An action performed by a user is embedded and considered as a part of the interaction sequence. An RNN approach with Gated Recurrent Unit layers is used here.

5.2 Evaluation Data Set

The data set used in this work consists of the online shops T^1 and T^2 as instantiations of the model from Definition 1. For each online shop, the corresponding recommender is provided in the form of a trained neural network. Furthermore, we receive the profile stream, which contains the user information about the customers which were previously considered as training and test data. econda updates the respective recommender at regular time intervals based on current purchasing behaviour. Therefore, the data set used includes several profile streams and recommenders per online shop. In total, we use 8 (10) profile streams for T^1 (T^2). A profile stream contains on average 524 (1004) customers and per customer we have on average 33 (64) customer interactions. All recommenders were realised in Python 3.7 with Tensorflow v2.1.0.

5.3 Evaluation Results

In Section 2, we have defined the standard LRP method. However, there are also variants of this methods which outperform the standard on some architectures. To the best of our knowledge, it is not known which of these methods works best for RNNs. As a preliminary step, we therefore fill in this gap by evaluating the performance of the standard LRP and some of its most popular variants using our algorithm from Section 4.2. As a reference, we switch off each input neuron once at a time to find the neuron that is actually most relevant to the decision. This is the case, when the change of the original prediction value is maximal by leaving out this specific input. Finally, per LRP variant, we determine the relative frequency with respect to detecting the most relevant input neuron. Regarding the mentioned LRP methods, we first consider all possible parameter combinations with respect to the values $0.01, 0.1, 1, 5, 10$, and then choose the best combination. We obtained the scores *standard* [2] **0.9800**, *epsilon* [1] 0.9560, *gamma* [15] 0.9080, *alpha-beta* [16] 0.7720, and *non-negative* [16] 0.5040. Based on these and due to the fact that the standard method achieved a hit rate of 100% in the case of MLPs, we will limit ourselves to this method. In the following, we describe three use cases that can be achieved with our explanation quantity measures defined in Section 4.3.

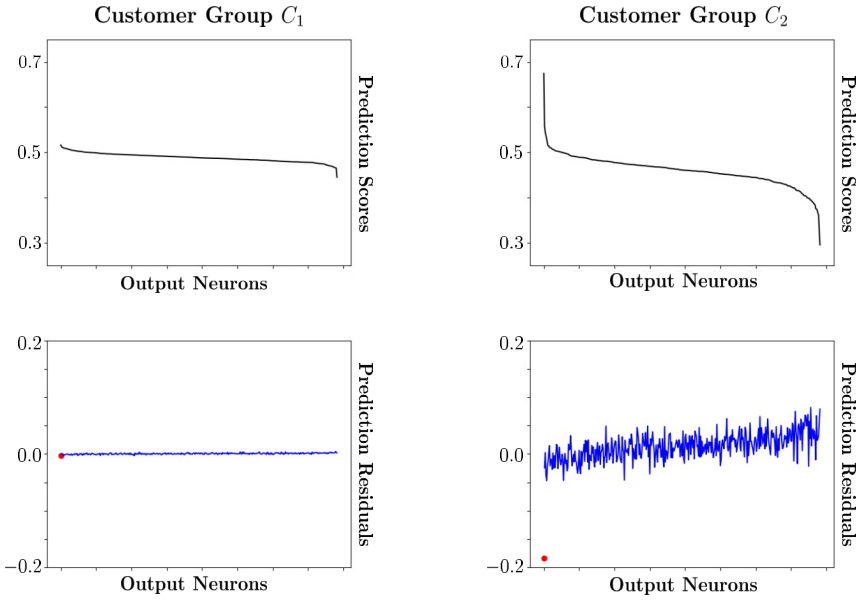

Fig. 5: Two exemplary output layers for shop T^1. Output vectors of the NN ranked in descending order for customer groups C_1 and C_2 in the upper part including corresponding residual plots after setting the most relevant input neuron to zero in the lower part. The residual of the original top prediction is marked red.

Understanding the Recommendation Quality. To tackle this, we have to examine discrepancies between prediction and input. We found one within the predictions provided by econda for the jewellery shop T^1 that could not be explained intuitively. Therefore, we apply the measures from Section 4.3 to obtain explanations regarding the recommender's decisions. The upper part of Fig. 5 shows two exemplary output layers of the neural subnetwork s_1, where $C_1, C_2 \subset C^1$ are disjoint subsets of customers C^1 of T^1. The exemplary customers were each randomly selected from 25 customers in C_1 and 29 customers in C_2, respectively. The output neurons are ranked in descending order regarding their prediction score. It can be seen that the preferred outputs for customers from C_1 are almost indistinguishable. In contrast, the scores for customers from C_2 imply clear top predictions. Considering the lower part of Fig. 5, we plot the residuals after setting the most relevant input neuron to zero to show the discrepancy. For customers from C_1, the discrepancy between the top prediction and the average prediction score is much smaller than for customers from C_2 because the entire curve hovers quite closely around its average. Thus, the product recommender s_1 of T^1 is apparently not as certain about its decisions because the predictions range over a small interval. Therefore, we consider the top predictions in each case and try to gain new insights into the decision-making of the

Table 1: Results of LRP-comparison for recurrent model

measure \ user	C_1	C_2	C^1
σ_{T^1} (individuality)	1.2668	1.0021	1.1409
ν_{T^1} (certainty)	0.7302	0.9733	0.8804
$\zeta^+_{T^1}$ (diversity)	1.5564	143.1009	65.7103

neural network via the explanation measures from Section 4.3. Table 1 shows these results including significant differences between C_1 and C_2:

(i) Comparing the results of the **individuality measure** σ_{T^1}, we can see that predictions for customers of C_1 depend more on the bias induced by the training data. Predictions for customers of C_2 are almost independent of the bias.
(ii) Regarding the **certainty measure** ν_{T^1}, customers of C_1 have more contradictory input neurons with negative relevance.
(iii) Since we are interested in the positive influence of input neurons on the overall decision, we consider the **diversity measure** $\zeta^+_{T^1}$. We can see the greatest divergence between customers of the two classes C_1 and C_2. Regarding the inputs with positive relevance, customers of C_2 have an input with a relevance that is significantly greater than the other relevancies. This means that there are inputs that speak in favour of the decision made which is not the case for customers from C_1.

All three measures reveal differences between the two customer groups. The diversity measure stands out particularly prominently. The key figures listed here reflect a well explainable prediction of the recommender for customers from C_2. This means that few input neurons had the strongest influence on the prediction made which is not the case for customers from C_1. This discrepancy can also be seen very well if we switch off the input with the highest relevance and plot the residuals of the output vectors, see the lower part of Fig. 5. The input with highest relevance is marked red. It has a significantly stronger influence on the prediction for customers from C_2 than C_1. For the latter, switching off this input causes almost no deviation in the predictions. Using the LRP approach and the explanatory measures, it has thus been possible to establish that the clear predictions for customers from C_2 are quite simple to explain. Namely, these customers have activated input neurons that contribute massively to the prediction made. For the customers from C_1 on the other hand, the decision-making is rather based on the entire interaction of the input neurons.

Ideas to Improve the Recommender. A closer look at the most relevant inputs reveals a certain pattern. We have two different types of input neurons: (a) input neurons representing the interaction with a product from P^* and (b) input neurons representing an interaction with a certain product category. In the latter case, an interaction with a category can only take place via an interaction with a product from the associated category. The activation of the categories occurs

for each product interaction, regardless of whether or not it is contained in P^*. Now, when looking at the input relevancies for customers from C_1 or C_2, the following is noticeable: Firstly, for customers from C_1 there are no activations of products. The most relevant inputs are therefore categories and the relevancies hardly differ. Secondly, customers from C_2 always have product activations. In these cases, the most relevant input is always a neuron belonging to a product interaction and these relevancies are significantly higher than those of the likewise activated categories. We were thus able to determine that the activation of products as input neurons leads to more unambiguous decision-making. In particular, these represent a better explanatory power as the neural network predicter can identify certain information that significantly influenced the decision made. It would therefore make sense to separate the user information even further and define the products or categories as separate subnetworks. In this way, the decision-making process for user profiles that only contain categories as input neurons could be given a stronger explanatory power.

Understanding the Customer Base. We also performed an evaluation on the interior design shop T^2. Our diversity measures σ_{T^2} and $\zeta^+_{T^2}$ revealed that the trained bias and outliers within the positive input relevancies of the neural subnetwork s_2 were particularly relevant for the decisions made. Thus, it was found that buying interest is based on daily trends rather than past interactions. Unfortunately, we cannot explain this in more detail here due to space constraints.

6 Conclusion

In this paper, we have presented TransPer, an explanation framework for neural networks used in online shopping.

We used the LRP method to define three explanation measures, namely the individuality measure, used to determine those parts of the input that are particularly relevant for the decision; the certainty measure, which measures how certain the system is about its prediction; and the diversity measure, which measures whether there are clear top predictions. These measures can be defined on complex neural networks which process heterogeneous input data.

We have demonstrated the usefulness of our metrics in three explanation use cases. First, we explained fluctuations in the prediction qualities. Second, TransPer explanations can help find ideas on how to improve the neural network. Third, our explanations can help online shops better understand their customer base. These explanations also play an important role in fulfilling legal requirements such as the ones mandated by GDPR.

References

1. Ancona, M., Ceolini, E., et al.: Towards better understanding of gradient-based attribution methods for Deep Neural Networks. In: ICLR (2018)

2. Bach, S., Binder, A., et al.: On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. PLoS ONE **10**(7) (2015)
3. Bettaney, E.M., Hardwick, S.R., et al.: Fashion Outfit Generation for E-commerce. In: eCom@SIGIR. CEUR, vol. 2410 (2019)
4. Bharadhwaj, H.: Layer-Wise Relevance Propagation for Explainable Deep Learning Based Speech Recognition. In: IEEE ISSPIT. pp. 168–174 (2018)
5. Borisov, A., Markov, I., et al.: A Neural Click Model for Web Search. In: WWW Conference. pp. 531–541 (2016)
6. Chen, M., Liu, P.: Performance Evaluation of Recommender Systems. International journal of performability engineering **13**, 1246 (2017)
7. Chen, W., Huang, P., et al.: POG: Personalized Outfit Generation for Fashion Recommendation at Alibaba iFashion. In: ACM SIGKDD. pp. 2662–2670 (2019)
8. econda: Personalization & Analytics. https://www.econda.de, last acc. 2021-03-29
9. Gebremeskel, G.G., de Vries, A.P.: Recommender Systems Evaluations : Offline, Online, Time and A/A Test. In: CLEF. CEUR, vol. 1609, pp. 642–656 (2016)
10. General Data Protection Regulation: Art. 12 GDPR. https://gdpr-info.eu/art-12-gdpr/, last acc. 2021-03-29
11. Khoali, M., Tali, A., et al.: Advanced Recommendation Systems Through Deep Learning. In: NISS. pp. 51:1–51:8 (2020)
12. Laenen, K., Moens, M.: A Comparative Study of Outfit Recommendation Methods with a Focus on Attention-based Fusion. Inf. Process. Manag. **57**(6), 102316 (2020)
13. Li, Y., Cao, L., et al.: Mining Fashion Outfit Composition Using an End-to-End Deep Learning Approach on Set Data. IEEE Trans. Multim. **19**(8), 1946–1955 (2017)
14. Loureiro, A.L.D., Miguéis, V.L., et al.: Exploring the use of deep neural networks for sales forecasting in fashion retail. Decis. Support Syst. **114**, 81–93 (2018)
15. Montavon, G., Binder, A., et al.: Layer-Wise Relevance Propagation: An Overview. In: Explainable AI, vol. 11700, pp. 193–209 (2019)
16. Montavon, G., Samek, W., et al.: Methods for interpreting and understanding deep neural networks. Digital Signal Processing **73**, 1–15 (2018)
17. Nelaturi, N., Devi, G.: A Product Recommendation Model Based on Recurrent Neural Network. Journal Européen des Systèmes Automatisés **52**, 501–507 (2019)
18. Nguyen, A., Oberföll, A., et al.: Right for the right reasons: Making image classification intuitively explainable. In: ECIR. vol. 12657, pp. 327–333 (2021)
19. Noh, H., Hong, S., et al.: Learning Deconvolution Network for Semantic Segmentation. In: IEEE ICCV. pp. 1520–1528 (2015)
20. Park, S.: Neural Networks and Customer Grouping in E-Commerce: A Framework Using Fuzzy ART. In: AIWoRC. pp. 331–336 (2000)
21. Rumelhart, D.E., Hinton, G.E., et al.: Learning Representations by Back-propagating Errors. Nature **323**(6088), 533–536 (1986)
22. Shrikumar, A., Greenside, P., et al.: Learning Important Features Through Propagating Activation Differences. In: ICML. vol. 70, pp. 3145–3153 (2017)
23. Simonyan, K., Vedaldi, A., et al.: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In: ICLR (2014)
24. Springenberg, J.T., Dosovitskiy, A., et al.: Striving for Simplicity: The All Convolutional Net. In: ICLR (2015)
25. Sundararajan, M., Taly, A., et al.: Axiomatic Attribution for Deep Networks. In: ICML. vol. 70, pp. 3319–3328 (2017)
26. Yuan, J., Li, Y., et al.: Leave-One-Out Cross-Validation Based Model Selection for Manifold Regularization. In: Adv. in NN ISNN. vol. 6063, pp. 457–464 (2010)