Contents lists available at ScienceDirect

Graphics and Visual Computing

journal homepage: www.elsevier.com/locate/gvc

Technical section

Efficient structuring of the latent space for controllable data reconstruction and compression[☆]

Elena Trunz [a,*], Michael Weinmann [b], Sebastian Merzbach [a,c], Reinhard Klein [a]

[a] University of Bonn, Germany
[b] Delft University of Technology, Netherlands
[c] X-Rite Europe GmbH, Switzerland

A R T I C L E   I N F O

Article history:
Received 4 March 2022
Received in revised form 25 June 2022
Accepted 1 November 2022
Available online 5 November 2022

Keywords:
Autoencoder
Encoder–decoder architectures
Data compression
Data representation
Deep learning
Shapley values

A B S T R A C T

Explainable neural models have gained a lot of attention in recent years. However, conventional encoder–decoder models do not capture information regarding the importance of the involved latent variables and rely on a heuristic a-priori specification of the dimensionality of the latent space or its selection based on multiple trainings. In this paper, we focus on the efficient structuring of the latent space of encoder–decoder approaches for explainable data reconstruction and compression. For this purpose, we leverage the concept of Shapley values to determine the contribution of the latent variables on the model's output and rank them according to decreasing importance. As a result, a truncation of the latent dimensions to those that contribute the most to the overall reconstruction allows a trade-off between model compactness (i.e. dimensionality of the latent space) and representational power (i.e. reconstruction quality). In contrast to other recent autoencoder variants that incorporate a PCA-based ordering of the latent variables, our approach does not require time-consuming training processes and does not introduce additional weights. This makes our approach particularly valuable for compact representation and compression. We validate our approach at the examples of representing and compressing images as well as high-dimensional reflectance data.

1. Introduction

The rapid progress in deep learning has led to huge improvements in numerous application areas including data reconstruction, compression and streaming, where explainability of the model's behavior and decisions is of particular importance. Respective approaches including autoencoders rely on the core idea of encoding the information extracted from the input data in another *latent* space, from where it can be decoded back to the original domain. Powerful and compact representations can then be obtained based on the combination of an information bottleneck, i.e. choosing the dimensionality of the latent space to be lower than the input dimensionality so that only the most salient features are preserved, and appropriate loss functions.

In this paper, we put our attention to the challenging problem of the efficient specification of a convenient dimensionality of the latent space that preserves model accuracy for respectively considered tasks.

So far, most approaches for designing encoder–decoder schemes have been based on a heuristic specification of the number of latent dimensions, i.e. without an actual explanation why the respective dimensionality has been chosen and without an analysis regarding the dimensionality that best suits the particular application. In contrast, determining a suitable number of latent variables has also been addressed for some encoder–decoder approaches (e.g. [1,2]) that are trained with different numbers of latent variables, where finally the dimensionality leading to the best trade-off between small dimensionality of the latent space and reconstruction quality is chosen. However, this procedure is very time-consuming, since the encoder–decoder needs to be trained anew for each number of latent variables, and, for high-dimensional latent spaces, such a repeated training may even be infeasible. Instead, we propose a novel approach for the specification of a suitable dimensionality of the latent space by analyzing the contribution of the individual latent dimensions and their respective ranking in encoder–decoder schemes based on their Shapley values [3]. The concept of Shapley values has originally been introduced in cooperative game theory for feature attribution, and we leverage this concept similar to the computation of a natural ordering of the components regarding their contribution based on principal component analysis (PCA) [4,5], but instead for the more general non-linear relationship that

☆ This article was recommended for publication by D. Bommes.
* Correspondence to: Visual Computing Department, FriedrichHirzebruch-Allee 5, 53115 Bonn, Germany.

E-mail addresses: trunz@cs.uni-bonn.de (E. Trunz), M.Weinmann@tudelft.nl (M. Weinmann), merzbach@cs.uni-bonn.de (S. Merzbach), rk@cs.uni-bonn.de (R. Klein).

https://doi.org/10.1016/j.gvc.2022.200059

typically allows autoencoders to find more flexible and more powerful latent spaces. While the latent space of an autoencoder exhibits compactness with respect to the original data domain, it does not allow gaining structural information of the latent space as in the case of a PCA. More sophisticated autoencoder variants [6,7] allow the ordering of the dimensions of the latent codes according a decreasing importance with respect to the input data while preserving statistically independent components. However, these approaches are based on progressively increasing the dimensionality of the latent space, i.e. learning one new dimension per step. Instead, our approach avoids such a progressive adaption of the required dimensionality by the direct use of the contribution of individual latent dimensions according to their Shapley values. The computation of the contributions of individual latent dimensionalities and their ordering in terms of the Shapley value based analysis can be applied at different times during training as long as the training loss does not significantly change over the epochs anymore. In the scope of our experiments, we will compare the results of applying the Shapley value based analysis in the middle of the training and at the end of the training. We investigate the beneficial combination of Shapley values and encoder–decoders regarding the choice of the dimensionality of the latent space, the ordering of the involved latent variables according their importance and the respective capability for reconstruction and compression. This is motivated by the fact proven in the paper that in case of a linear model the ordering based on Shapley values is the same as the one after the singular value decomposition and, hence, optimal. In summary, the key contributions of this paper are:

- We present novel method for ranking the latent variables in an encoder–decoder, based on their contribution to the result, and the subsequent specification of a suitable dimensionality of the latent space.
- We demonstrate the benefits of our approach by evaluations on various different application scenarios.
- We provide the theorem and the proof of the optimality of the Shapley ordering in the linear case.

2. Related work

The complexity of the concept of interpretability [8,9] makes a general interpretation regarding a model's behavior/decisions intractable. The key objective of feature attribution methods that focus on local interpretability is the identification of relevant features based on a scalar attribution score, relevance score [10] or contribution [11] that defines how much each input feature contributes to a model's behavior. However, a limited theoretical understanding as well as the lack of reliable quantitative metrics for evaluating explanations in case on ground truth is available [12] may lead to unreliable or even misleading results that may still appear visually appealing [12–15]. This problem has been addressed based on incorporating desirable axioms into the attribution method [13,16–19]. These axioms have to be fulfilled by any explanation obtained from the respective attribution methods and allow the design of attribution methods with theoretical guarantees [17]. In the context of deep learning, *backpropagation-based attribution methods* rely on the idea of computing the attribution based on backward passes through the network. Examples include the computation of attributions by exploiting gradients that carry the information regarding the local perturbations of features that mostly influence the output. Here, attributions can be obtained in terms of saliency maps [20], that refer to the gradient of the class score with respect to the input image, or by elementwise multiplications of input data and signed gradients (Gradient×Input) [21]. However, this approach only provides local information in case of highly non-linear functions and, hence, not suitable to compute marginal contributions of features. Therefore, other approaches such as Layer-wise Relevance Propagation (LRP) [10,18], DeepLIFT variants [11,21] and Integrated Gradients [17] make use of different propagation rules in comparison to the use of the instant gradient in the Gradient×Input approach. However, perceptually similar inputs with the same predicted labels may be interpreted differently as even small random perturbations affect the feature importance and systematic perturbations may change the interpretation while keeping the label [14]. In contrast, perturbation-based approaches rely on the computation of the relevance of input features by analyzing the behavior of a neural network in case of feature removal or perturbation [22–24].

A related classical concept developed in the domain of cooperative game theory in order to distribute the contribution of individual players in a cooperative game while fulfilling desirable axioms is given by the Shapley values [3] and resulting feature attributions even seem to agree to human intuition [19]. As discussed in literature, computing exact Shapley values remains an NP-hard problem [25] and, in practice, can only be performed for less than 20 to 25 players (i.e. input features in our case respectively). For this reason, much effort was spent on finding adequate approximations of Shapley values such as in terms of sampling-based methods [26–29] as well as on investigating new classes of additive feature importance measures for particular predictions as denoted by SHapley Additive exPlanations (SHAP) [19]. To avoid the rapidly increasing number model evaluations for increasing numbers of input features, additional lasso regression has been used in KernelSHAP [19] and its respective extensions towards global interpretability [30], different importance metrics and feature packing [31], handling dependent features [32,33] and producing additional types of explanations [34] such as explaining whether samples are likely to a certain class, why prediction differ depending on the observations and when the model has a bad performance. Furthermore, approximations based on the assumption of model linearity have been proposed (such as DeepSHAP) [19] and extended to mixed model types [35] in terms of a layerwise propagation of Shapley values built upon DeepLIFT [11,21]. This also enables computational tractability for obtaining exact Shapley values for certain model types like tree-based models [36], such as random forests or gradient boosted trees, and allows attributing stacks of mixed models such as the feature extraction of neural networks into a tree model and also attributing loss functions. Polynomial approximations for specific games such as voting games [37] allow a polynomial-time approximation of Shapley values as shown with Deep Approximate Shapley Propagation (DASP) [38]. Further work includes extensions towards global explainability (by combining the Shapley value concept with Lorenz zonoids [39] to combine the advantages of the local Shapley value based approach with the properties of the Lorenz Zonoids), the generalization of Shapley values to the Shapley–Taylor index that reflects attributions of subsets of features [40] and the exploitation of assumptions regarding the underlying data structures [41]. As a counterpart, Shapely Residuals [42] have been introduced to capture the information not preserved by Shapley values.

However, most of the approaches for calculating Shapley values rely on post-hoc explanations. Therefore, the explanation approach cannot be used for designing and training models. Generalized Additive Models (GAM) based on tree boosting [43,44] or neural networks [45] allow the simultaneous prediction and computation of the corresponding exact SHAP explanation, but their representational is power inherently limited. Instead, Shapley Explanation Networks [46] rely on directly incorporating Shapley values as the learned latent representations in deep neural networks. SHapley Additive exPlanations connect local explanations with optimal credit allocation and has been used to rank input variables for identification and prediction of failure modes [47]. Furthermore, Shapley value based error apportioning (SVEA) [48] has been introduced where the key idea is the apportioning of the total training error among the features and Ghorbani and Zou [49] focus on equitable data valuation in the context of supervised learning, where data Shapley values are used to rate the contribution of each training sample to the prediction performance. Covert et al. [50] focused on explainability in terms of simulating the effect of feature removal to determine the influence of individual features. Their framework analyzes how features are removed for different methods, the respectively explained model behavior, and how methods summarize the features' contributions. In addition, by assigning contribution scores to edges instead of nodes within a causal graph structure, Shapley Flow [51] generalizes the Shapley value axioms to directed acyclic graphs. Ghorbani and Zou [52] used Shapley values for quantifying the importance of individual neurons for network predictions and performance. This allows a more efficient identification of important filters in comparison to the use of activation patterns. Furthermore, Ma et al. [53] considered Shapley values in the scope of Bayesian networks and showed a relation between Shapley values and conditional independence.

We exploit feature attribution based on Shapley values in encoder–decoder frameworks to efficiently structure the latent space by ordering the latent variables according to their importance and exploit this strategy for explainable data reconstruction and compression.

Aside from feature attribution, several works specifically focus on data compression to allow efficient storage and transmission of contents through constrained channels, where in particular neural image compression has gained a lot of attention in recent years. The targeted tradeoff of determining an as-compact-as-possible binary representation (i.e. lowest rate bitstream) while preserving a certain level of fidelity (i.e. minimum distortion) of the data has been investigated in terms of autoencoder architectures with quantization and entropy coding. Such compressive autoencoders [54–56] rely on also minimizing the combination of rate and distortion during training and have been improved by multi-scale extensions of the encoder and/or decoder [57–59] or adding generalized divisive normalization (GDN) layers [56,60]. Furthermore, end-to-end training can be achieved based on replacing the non-differentiable quantization by differentiable proxies [55,61,62]. In addition, hyperpriors [63] and contextual models [64–68] have been used to improve entropy coding. Several works also focus on adversarial training schemes to achieve very low rates [58,69,70].

Targeting variable rate image compression, traditional compression methods were based on quantizing Discrete Cosine Transform (DCT) coefficients according to the target rate. Further techniques include the learning of rate-specific bottleneck scaling (i.e. scaling the bottleneck features before quantization) [55], the modulation of intermediate features based on modulated autoencoders (MAEs) [71] and conditional autoencoders (cAEs) [72], the use of recurrent neural networks [54] and the use of a multi-scale decomposition network where each scale targets a different rate. Furthermore, increasing the efficiency of deep learning for resource-limited scenarios as occurring for tablets or smartphones has been generally addressed in terms of searching lightweight architectures [73–75], integer and binary networks [76–78], automatic architecture search [79], or adjusting the width of layers to achieve a trade-off between computational efficiency and accuracy based on slimmable neural networks [80]. In the scope of neural image compression, network architecture search [81] or progressive ecoding [82] have been used to address runtime and latency respectively. However, memory requirements and the computational burden do not change significantly and only a single rate–distortion tradeoff is considered which prohibits flexibility regarding rate, memory or the computational burden. To increase the practicality of neural image compression, slimmable compressive autoencoders [83] also allow controlling computation, memory and rate. While these approaches have shown great potential in the context of image compression, our approach represents a powerful alternative that can be combined with these approaches and can be applied to any compression method, which utilizes an encoder–decoder.

3. Structuring and pruning of latent space representations

Data reconstruction and compression techniques often rely on the transformation of the input data into a lower-dimensional latent space that describes the most salient features. Here, the dimensionality of the latent space has to be chosen to adequately represent the distribution of the input data while allowing an as-compact-as possible latent representation. This trade-off between reconstruction accuracy and compression rate has to be carefully considered depending on the underlying task and its implications. For this purpose, we have to focus on the following central questions:

1. What is a suitable choice for the dimensionality of the latent space, i.e. how many latent variables are required?
2. Can we relate the lossy compression induced by discarding dimensions to individual features' importance?
3. Does the structure of the latent space exhibit insights on how much we gain by taking less or more latent variables, thereby allowing the efficient control for the specification of a suitable dimensionality of the latent space without the need of having to train different autoencoders for each dimensionality like in the approach by Rainer et al. [1]?

Gaining control over latent variables and the ability to detect the contribution of each dimension to the overall performance of the model is an important step towards explainable models. In fact, we would even wish to determine the contribution of sets of important dimensions, i.e. we would like to get insights regarding which subsets of k dimensions exhibit the largest importance instead of taking the k individually most contributing features. That way, we can order the dimensions according their contribution and, hence, structure the latent space, which, in turn, allows taking the first most important k dimensions to get a rank-k approximation of the data for lossy compression.

In the following, we first provide an overview on how these questions have been handled in the scope of linear approaches. In this regard, we will demonstrate that, in the linear case, where the Eckart–Young–Mirsky theorem states how low-rank approximation can be approached, the ordering of the eigenvectors according to their Shapley values is equal to the ordering of the corresponding singular values. Then, we motivate why these questions become much more challenging in the case of non-linear models such as autoencoders and finally devise a strategy towards an efficient model that helps answering the aforementioned questions. Due to the properties of Shapley values that directly measure contributions of individual components, we aim at analyzing whether these could also serve in these non-linear scenarios (where the Eckart–Young–Mirsky theorem would not be applicable) and experimentally address these questions.

3.1. Latent space representations in linear models

Before analyzing whether the properties of Shapely values regarding their capability to directly measure contributions of individual components make them suitable for non-linear scenarios as given for encoder–decoder architectures, we first demonstrate that in the linear case, where the Eckart–Young–Mirsky theorem states how low-rank approximation can be approached, the ordering of the eigenvectors according to their Shapley values is equal to the ordering of the corresponding singular values.

In case of a linear relationship $Ax = y$ between inputs x and model outputs y, we can compute the best rank-k approximation A_k to A in the L_2-norm in terms of an analytical solution, i.e. by performing a singular value decomposition of $A = U\Sigma V^*$, as postulated by the Eckart–Young–Mirsky theorem [84]. This low-rank approximation A_k is given by

$$A_k = \sum_{i=1}^{k} (\sigma_i u_i v_i^*), \tag{1}$$

where σ_i are the singular values, u_i are the columns of U and v_i are the columns of V. Classical low-rank approximations such as SVD-based approaches and respective robust variants [4,5,85] have been proven to work for matrix-based data. However, the extension of these methods to higher-dimensional data is not straight-forward and comes at the loss of some of their underlying unique properties. For this reason, higher-order tensor decomposition models such as multilinear SVD [86], higher-order orthogonal iteration (HOOI) and the higher order power-method (HOPM) [87] as well as the CANDECOMP/PARAFAC (CP) model [88,89] and the Tucker tensor model [90] have been widely applied. Tensor decomposition can be interpreted as a generalization of the SVD approach to higher-order tensor data and the original data can be approximated based on a rank-reduced tensor decomposition [91,92]. Furthermore, latent variable models such as factor analysis [93] and probabilistic principal component analysis [94–96] exploit low-dimensional features to define powerful generative models.

3.2. Latent space representations in non-linear models

In contrast to linear models, deep neural networks enable non-linear mappings [97,98] and have demonstrated their power in modeling high-dimensional data. Autoencoders focus on the reconstruction of the inputs by first using an encoder e to project the input data to a latent space, which is followed by a decoder d that transfers the latent code back into the original domain to get a reconstruction of the input.

Therefore, their objective consists of minimizing reconstruction errors given by the reconstruction loss, where specific performance metrics v such as the L_2-norm are widely used. To prevent autoencoders from directly copying the inputs and instead force the encoder to learn useful properties of the data, autoencoders typically constrain the latent representation to be lower-dimensional than the input, thereby enforcing them to instead capture the most salient features of the input. In addition, if the latent space has lower dimensionality than the input space $\mathcal{X}$, the latent vector $e(x)$ can be regarded as a compressed representation of the input $x \in \mathcal{X}$. Ideally, the dimension of the latent space should be chosen according to the complexity of the considered problem.

Unfortunately, the latent space is not represented in terms of independent components that can be ordered according to a decreasing relevance for the data as in the case of the PCA. Hence, there is no analytical solution for rank-k approximation problems equivalent to the case of PCA that provides insights regarding the top-k latent components that together most contribute to the reconstruction quality. This induces the initial prerequisite to design a neural network architecture that suits the requirements of a particular task by manually selecting the number of latent variables which, a-priori, is non-trivial. A reasonable and widely followed approach is choosing a sufficiently high number of dimensions of the latent space and adding a respective regularization term.

In fact, we would instead wish to rate the contributions of latent dimensions based on a function ϕ that fulfills the following three axioms:

1. **Zero-player**: Latent variables that do not contribute to the resulting output should be assigned the weight 0 (or a *baseline value*).
2. **Symmetry**: The loss should not depend on the ordering of the latent variables but instead only on their presence.
3. **Efficiency**: Contributions of individual latent variables sum up to the contribution of all latent variables.

For the sake of explainability in terms of getting insights on the structure of the latent space, it would be useful to have a respective ordering of components according to an importance score and the respectively resulting error induced by rank-k approximation similar to the ordering in terms of singular values in the linear case. A naive way to approach this goal is the training of several autoencoders with different numbers of latent variables [2]. As a result, the corresponding error for each number of dimensions is obtained and, hence, the dimensionality that results in the best trade-off between dimensionality of the latent space and reconstruction quality can be selected. However, the multitude of involved training procedures make this procedure time- and resource-demanding.

In contrast, the PCA-like autoencoder [6] and the principal Component Analysis Autoencoder (PCAAE) [7] organize the dimensions of the latent space in decreasing importance with respect to the input data while preserving statistically independent components. For this purpose, these approaches rely on progressively increasing the dimensionality of the latent space and learning one new dimension per step as well as extending the standard autoencoder reconstruction loss by an additional covariance loss applied to the latent codes to enforce statistically independent latent space components. However, this procedure is very time-consuming, as the decoder needs to be re-trained with every additional dimension. Instead, we propose to leverage the ordering of the latent variables according to their contribution defined in terms of Shapley values [3], a concept of game theory for computing the contribution of players in a cooperative game.

3.3. Shapley value guided latent representation

In the scope of a cooperative game, Shapley values [3] assign the participating players, in our case latent variables, their respective contribution to the overall task, in our case the reconstruction of the latent code through the decoder. More formally, the Shapely value $\phi_i(v, N)$ of a latent dimension i is calculated as

$$\phi_i(v, N) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} (v(S \cup \{i\}) - v(S)), \tag{2}$$

where v is a coalition function that maps each subset $S \subseteq N$ of the players to real numbers, which represent the outcome of the game when players in S participate in it. In our scenario, N is the set of $n = |N|$ latent dimensions and the function v can be adapted for a decoder function d by defining the baseline as discussed by Ancona et al. [38]. This way we replaced $v(S)$ in (2) by $d(\mathbf{z}_S)$ and $\mathbf{z}_S$ denotes the original latent vector $\mathbf{z}$ where all entries not included in S are replaced with the baseline value, which is zero in our case. Since we have to process more than one latent vector in order to calculate the contribution of one latent dimension, we randomly choose a set of m example latent vectors from the space of all possible ones and average the resulting contributions.

Shapely values fulfill several desirable axioms and in particular the three axioms described in Section 3.2, which are of great relevance for our envisioned objective of ranking individual latent variables according to their contributions and specifying which set of them to keep for an as-compact-as-possible but still powerful latent representation. Since we are interested in the correct ordering of the latent dimensions according their descending contribution, we need to validate that the ordering according to the Shapley values exhibits this property. Therefore, we first prove that in the case of a linear model the ordering according to the Shapley values is optimal:

Theorem 1. *Let $A \in \mathbb{R}^{m\times n}$ be a real matrix with $m \geq n$ and $A = U\Sigma V^T$ be the singular value decomposition of A, where Σ is an $m\times n$ diagonal matrix with entries $\sigma_1, \dots, \sigma_n$, such that $\sigma_1 \geq \dots \geq \sigma_n$. Furthermore, let $N = \{u_1, \dots, u_n\}$ be the set of left-singular vectors, i.e. the columns of U. Let v be a function, which assigns a set $S \subseteq N$ the corresponding reconstruction error, i.e. $v(S) = \|A - A_S\|_2$, where $A_S = \sum_{i\in S} \sigma_i u_i v_i^T$ denotes the approximation of the matrix A with the vectors from the set S. Then for all pairs i, j with $i \neq j$ the following holds: $\sigma_i \geq \sigma_j \Leftrightarrow \phi_i(v, N) \leq \phi_j(v, N)$, where $\phi_i(v, N)$ is defined according to (2).*

We provide the proof of this theorem in the supplemental. Note that in our case v is not a function of the reconstruction error but a function of the actual reconstruction, i.e. the decoder. We can still apply the theorem, if we change $(v(S \cup \{i\}) - v(S))$ in (2) to the absolute value, as used in the remainder of our paper. As a direct corollary to this theorem we conclude that the first k elements according to the Shapley values constitute the best rank-k approximation of a linear model A.

In other words, the aforementioned Theorem 1 demonstrates that, in the linear case, where the Eckart–Young–Mirsky theorem states how low-rank approximation problem can be approached, the ordering of the eigenvectors according to their Shapley values is equal to the ordering of the corresponding singular values. In our work, we additionally (experimentally) investigate whether the properties of Shapley values in terms of being a measure for the contribution of individual components also brings benefits for non-linear scenarios, where the Eckart–Young–Mirsky theorem would not be applicable.

Unfortunately, the computation of exact Shapley values remains an NP-hard problem [25] and is feasible only for a very limited number of less than 20 to 25 players or, in our case, latent dimensions respectively. Recently, Deep Approximate Shapley Propagation (DASP) [38] has been introduced as an approach that allows incorporating desirable axioms in the scope of a polynomial-time approximation of Shapley values which makes them suitable for being used in deep neural networks. We use this approach to approximate the Shapley values of the latent dimension for our purposes. The Shapley values are then approximated by the average of the expected contribution to a random coalition according to

$$\mathbf{E}[\phi_i] = \frac{1}{n}\sum_{j=0}^{n-1} \mathbf{E}_j[\phi_{i,j}]. \tag{3}$$

Here the expectations $\mathbf{E}_j$ are calculated over the distribution of sets of size j and $\mathbf{E}_j[\phi_{i,j}]$ denotes the contribution of the latent entry z_i to any random coalition of size j. $\mathbf{E}_j[\phi_{i,j}]$ is then calculated as follows:

$$\mathbf{E}_j[\phi_{i,j}] = |\underset{S\subseteq N\setminus\{i\}, |S|=j}{\mathbf{E}}[d(\mathbf{z}_{S\cup\{i\}})] - \underset{S\subseteq N\setminus\{i\}, |S|=j}{\mathbf{E}}[d(\mathbf{z}_S)]| \tag{4}$$

As already described, we use the absolute value instead of the difference. If our decoder function outputs values with more than one dimension, as it does for example in the case of RGB values, then the difference in (4) is computed componentwise and then we sum over all dimensions of the output.

3.4. Choice of the latent space dimensionality

In order to choose an appropriate dimensionality for the latent space that allows capturing the most salient features in an as-compact-as-possible latent representation, we start the training of the network with a sufficiently (i.e. typically too) large number of dimensions of the latent space, thereby following the intuition that a too large size of the latent space can be determined quite easily. To be sure that the initial "big drop-off" of the loss is passed and we reach a plateau-like behavior, we let the training progress for half the number of the epochs of a full training. We then compute the contribution of each latent dimension in terms of their Shapley values and order the dimensions in a descending order according to these contributions. Note that in earlier epochs of the training the adaptions to the network weights, and hence also the adaptions of the distribution of data in the latent space, are strongly changing. Therefore, Shapley value based contribution assignments to individual dimensions of the latent space would provide less insights there while requiring a computational overhead, and, hence, we apply the Shapley value based analysis only when approaching a plateau-like behavior of the loss. Subsequently, we compute the loss for each set of the h first dimensions. Based on the cumulative contribution and cumulative loss we choose the dimensionality k of the latent space and also specify which of the latent variables to take for the continuation of the training by taking a reference of the coverage percentage of the contributions, i.e. according to the percentage of contribution that should be covered. As a result, we prune the latent space custom-tailored according to the complexity of the considered application scenario. The training continues with the same autoencoder as before but without the discarded latent dimensions and the corresponding neurons in the input layer of the decoder. The overall training is finished in the same total amount of epochs as the full training with the only overhead of the computation of Shapley values. Hence, this strategy does not require a time-consuming training process based on successively adding one more dimensions for iteratively conducted trainings. Instead, it only requires a single training to identify less relevant latent variables, and several of these latent variables with low importance can be discarded in a single step. The individual steps of our approach are presented in Algorithm 1.

Algorithm 1 Shapley value based pruning of latent dimensions

```
1: Train model for a certain number of epochs with initially
   specified number of latent dimensions
2: Select a subset of $m$ samples of the latent codes (obtained for
   training examples)
3: Compute approximate Shapley values for the latent variables
   based on the selected $m$ samples and the decoder function
   (i.e., coalition function), e.g. based on DASP
4: Order the latent variables in descending order w.r.t. the
   Shapley values and compute the cumulative contribution and
   visualizations
5: Decide from the orderings, cumulative contribution and
   visualizations how many dimensions $k$ will be kept
6: Modify the last layer encoder layer and the first decoder layer
   by only keeping the connections/neurons belonging to the
   first $k$ latent variables
7: Resume the training
```

Note that the Shapley value based analysis to compute the contributions of individual latent dimensionalities and their ordering can be applied at different times during training. The only requirement is that the training loss does not significantly change over the epochs, but instead is (almost) approaching a flat decreasing behavior. In the scope of our experiments, we will compare the effects of conducting the Shapley value based analysis in the middle of the training as well as at the end of the training. We observed that the importance of latent dimensions still changes until the end of the training and performing the Shapley analysis early might overestimate the number of dimensions that are needed. Instead, to get the best results we should perform the analysis in the end. However, in the scope of our experiments we show that for a good estimation it is sufficient to perform the analysis in the middle of the training to save time, since the changes of the importance of latent dimensions are only small afterwards.

4. Experiments

In the following, we validate our approach for determining a suitable dimensionality of the latent space and the compression of models. To demonstrate the versatile applicability of our approach, we focus on a set of different exemplary application scenarios that differ in the type of data and their respective complexity. We validate the benefits of incorporating Shapley values within autoencoder frameworks regarding the choice of the latent code size and the respective ordering of the dimensions according their importance at the examples of representing and compressing images as well as high-dimensional reflectance data. All experiments were performed on a desktop computer with an Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40 GHz and an Nvidia Titan XP GPU with 12 GB of RAM.

4.1. Reflectance representation and compression

First, we demonstrate the potential of our approach for the task of representing and compressing reflectance data. Bidirectional texture functions (BTFs) $f(\mathbf{x}, \lambda, \omega_i, \omega_o)$ have been proven to accurately capture local material appearance at surface positions $\mathbf{x}$ of a material sample under varying viewing conditions ω_o and lighting conditions ω_i and possibly also depending on the wavelength λ [99], however, at the cost of massive memory consumption. In the scope of our experiments, we used publicly available BTF datasets provided by Weinmann et al. [100] and particularly focused on leather, carpet and fabric materials due to their complex reflectance behavior. These measurements come at a high angular resolution (i.e. $151 \times 151 = 22801$ light/view configurations with approximately identical samplings of the light and view configurations) and a spatial resolution of 400×400 texels.

The measurements for individual surface positions $\mathbf{x}$ are stored as 4D reflectance functions $f_{\mathbf{x},\lambda}(\omega_i, \omega_o)$ that are denoted as *apparent bidirectional reflectance distribution functions (ABRDFs)*. In contrast to bidirectional reflectance distribution functions (BRDFs), ABRDFs also capture non-local effects of light exchange at the surface such as local subsurface scattering, self-masking or self-shadowing. Finally, material samples are represented in terms of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, where the columns represent the ABRDFs of the m considered surface texels. Recent work on BTF compression and interpolation, which is particularly required for efficiently storing and rendering such data, includes the neural approach by Rainer et al. [1]. In contrast to matrix factorization techniques, that may cause blurring or ghosting artefacts in case of coarse angular resolution, and the fitting of analytic models with a reduced representation capability regarding complex non-local lighting effects, Rainer et al. leveraged the concept of autoencoders to introduce a neural network-based BTF representation. Here, the local surface appearance under different viewing and lighting conditions is first compressed to a latent representation by an

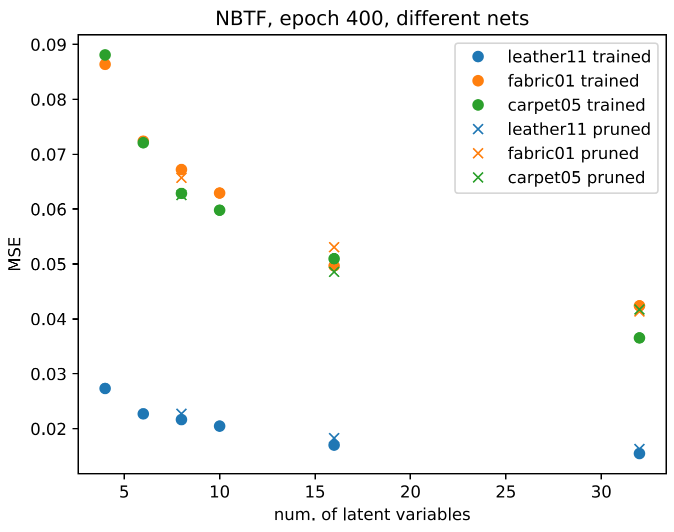

Fig. 1. MSE errors observed for different BTFs when training with different numbers of latent variables from scratch (points) vs. when starting training with 64 dimensions and pruning after 200 epochs to a different number of (most important) latent variables (crosses). We observe that both methods converge to nearly the same results. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

encoder component, and the decoder decodes the latter with additional light and view specifications to render the color of a particular surface point, thereby overcoming limitations based on a linear interpolation between measurements. However, for each BTF a new autoencoder needs to be trained. To make a reasonable choice regarding the dimensionality of the latent space, Rainer et al. [1] separately trained autoencoders for different numbers of dimensions of the latent space up to $n = 32$ and finally selected 8 latent variables, based on the trade-off between the compression rate and the reconstruction error. Besides the resulting high computational burden for all these trainings, the selection of an actually optimal trade-off can only be reached for the single, considered BTF as the network has been trained for each BTF separately. From the observations for a single BTF or a very limited set of BTFs, Rainer et al. concluded 8 dimensions to be a suitable size of the latent space. However, this chosen dimensionality may not be adequate for other materials, e.g. with different or more complex appearance characteristics that had not been investigated. Indeed, the resulting mean squared error for the fully trained networks for different numbers of latent dimensions for 3 BTFs depicted in Fig. 1 reveal that the curves deviate, i.e. the reconstruction based on the same number of latent dimensions will result in different quality levels for different materials/BTFs. Instead, a separate analysis of a suitable trade-off choice of dimensions even further increases the computational effort to also address these materials. This demonstrates that the provided trade-off value is not an optimal choice in general, despite the high computational burden.

In the scope of our experiments, as suggested by Rainer et al. [1], we applied a log transform as well as a whitening to the input ABRDFs and used 400 epochs for the full training. On one GPU, one full training took approximately 4 h. We used DASP to approximate Shapley values after training for the first 200 epochs. In order to calculate the Shapley values of the latent dimensions according to the coalition function, which is the decoder function d in our case, we need to generate some latent vector examples $Z = \{\mathbf{z}_1, \dots, \mathbf{z}_m\}$. For this we randomly sampled 2000 pairs of view and light directions (out of the 151×151 view-light sampling) for 100 randomly pixels (within the 100×100 available ones), resulting in 200 000 latent vectors in total. The

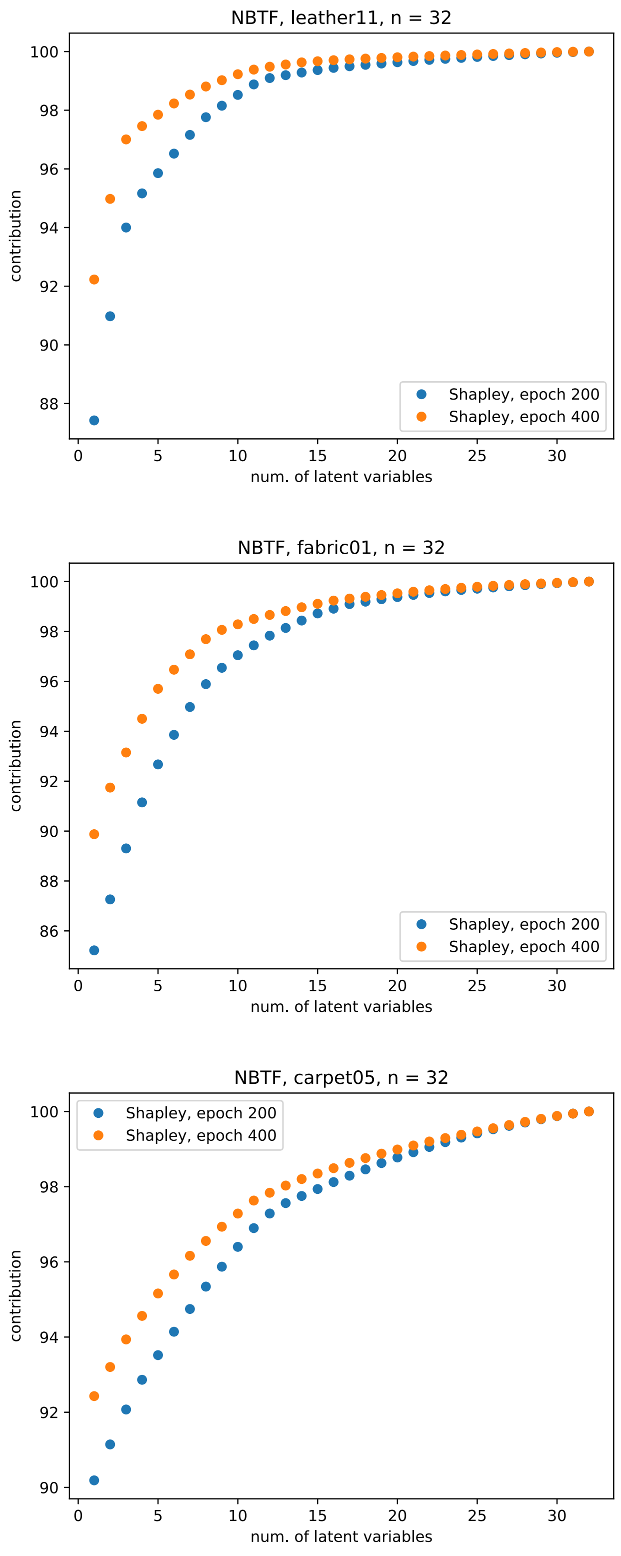

Fig. 2. Cumulative contributions observed when performing Shapley analysis after half of the training vs. after the full training for leather11 BTF (top), fabric01 BTF (middle) and carpet01 (bottom).

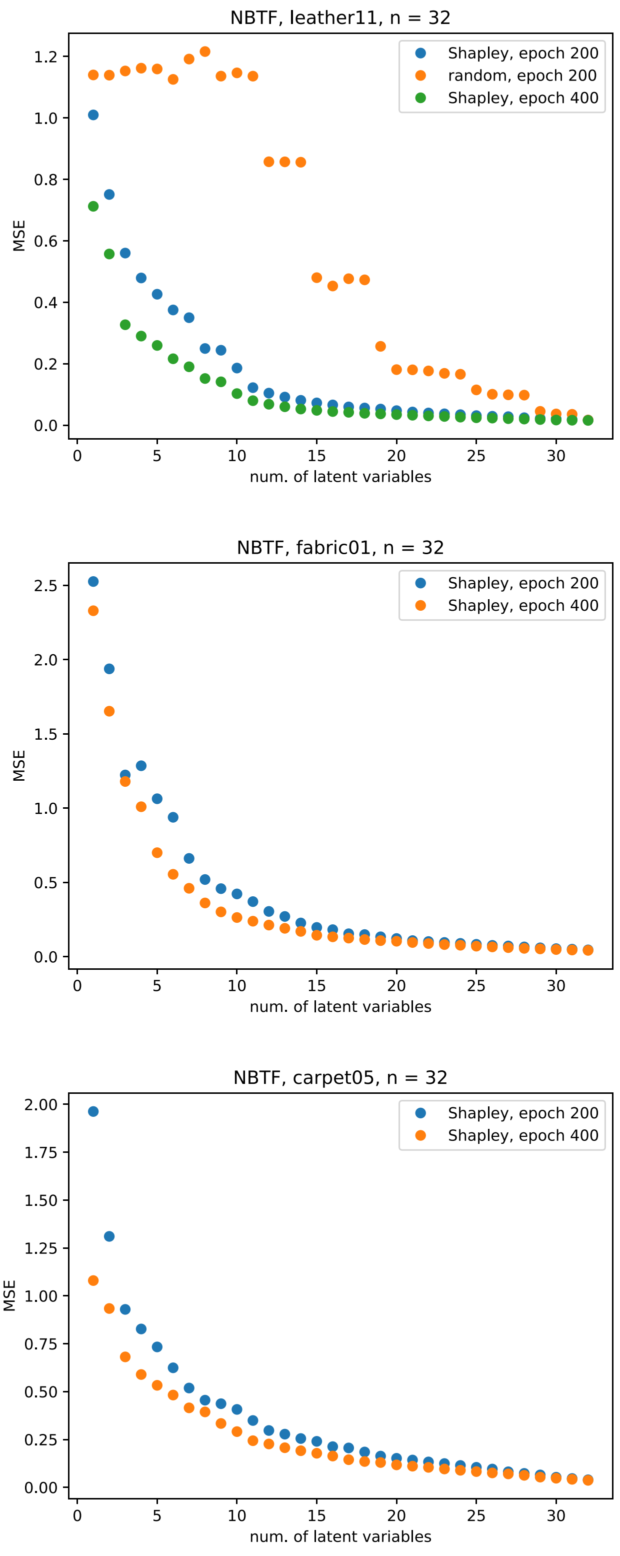

Fig. 3. Cumulative errors with respect to the full net, observed when performing Shapley analysis after half of the training vs. after the full training for leather11 BTF (top), fabric01 BTF (middle) and carpet01 (bottom).

Shapley value based analysis took about 8 min and Figs. 2 and 3 depicts the resulting contributions and MSE plots, whereas Figs. 4, 5 and Fig. 1 in supplemental provide respective visualizations. As a reference, we also show plots and visualizations obtained for a full training. In Table 1, we present a detailed analysis of the relationship between the number of latent variables and the time required to compute the Shapley values with the DASP method. In Fig. 2, we observe that different BTFs need different numbers of latent dimensions to achieve the same cumulative contribution percentage.

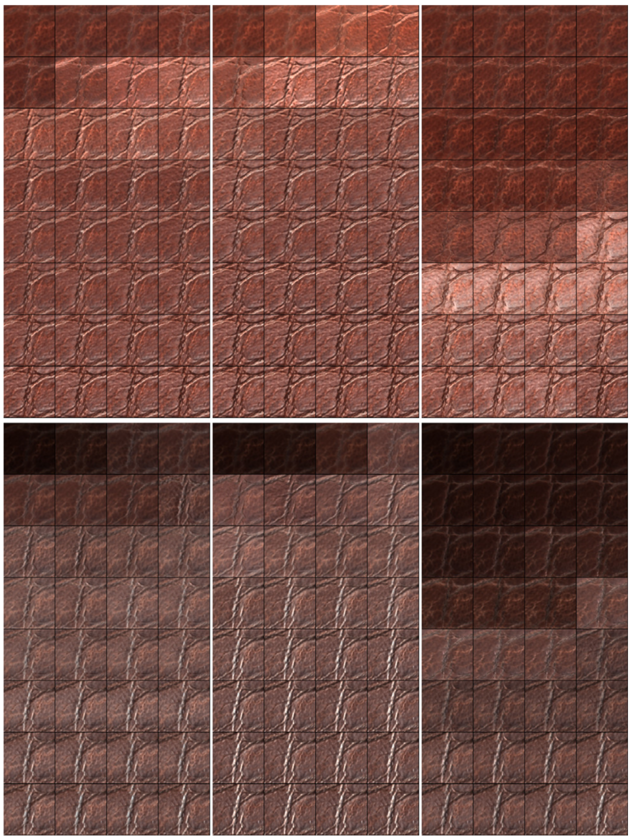

Fig. 4. Visualizations of the material appearance reconstructed based on different numbers of latent variables (for $h = 1, \ldots, 32$) for two different pairs of light and view directions ($L = 3, V = 3$ (top) and $L = 35, V = 100$ (bottom), see Fig. 8) for the material leather11. On the left we see the visualizations after 200 epochs, in the middle the visualizations after 400 epochs and on the right the visualization according to a random ordering after 200 epochs.

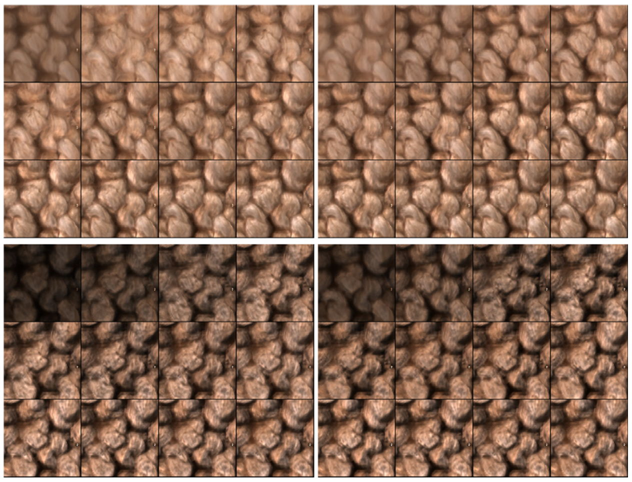

Fig. 5. Visualizations of the material appearance reconstructed based on different numbers of latent variables for two different pairs of light and view directions for carpet05. Top: $L = 3, V = 3$ and bottom $L = 35, V = 100$ (see Fig. 8). On the left we see the visualizations after 200 epochs, on the right the visualization after 400 epochs, each with the first $h = 1, 4, 7, 9, 10, 13, 16, 19, 22, 25, 28, 32$ latent variables according to the Shapley ordering.

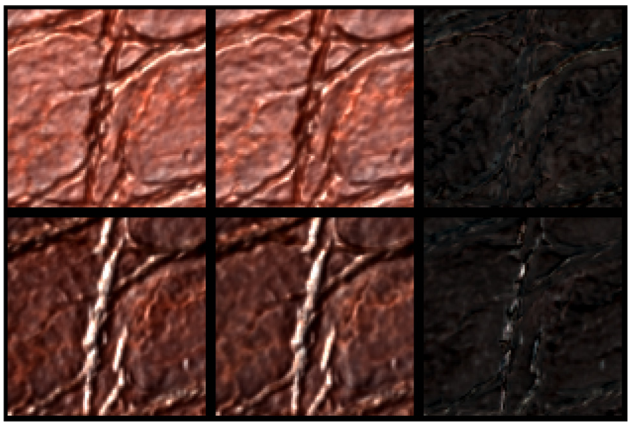

Fig. 6. Visualizations of the material appearance reconstructed based on different numbers of latent variables for two different pairs of light and view directions for leather11 after training with 32 latent variables (left) and 5 latent variables (right) and their difference. Top: $L = 3, V = 3$ and bottom $L = 35, V = 100$ (see Fig. 8).

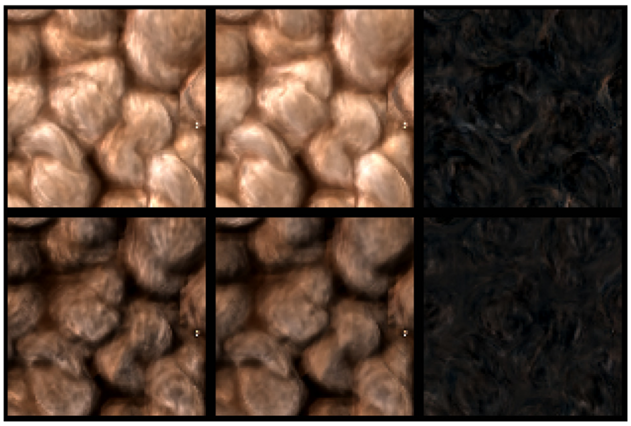

Fig. 7. Visualizations of the material appearance reconstructed based on different numbers of latent variables for two different pairs of light and view directions for carpet05 after training with 32 latent variables (left) and 9 latent variables (right) and their difference. Top: $L = 3, V = 3$ and bottom $L = 35, V = 100$ (see Fig. 8).

Table 1
Relation between the time (in minutes) required to compute the Shapley values and the number of latent variables. When the number of latent variables doubles, the time for the computation of Shapley values with DASP increases by a factor of about four.

	8	16	32	64	128
BTF	0,43	1,73	7,2	30,3	141,6
IC	2,5	10,5	41	171	695

For instance, taking 8 latent variables for the material leather11 results in capturing about 98% of the contribution of all latent variables. In contrast, matching this reconstruction quality in terms of 98% of the overall contributions requires a dimensionality of 12 for the latent space for the fabric01 BTF and a dimensionality of 17 for the latent space for the carpet05 BTF. Besides these significant variations of the cumulative contributions over the number of latent variables, we get evidence that the fixed choice of 8 latent dimensions independent of the considered material as used by Rainer et al. [1] can be suboptimal for different materials.

When analyzing the accumulated contributions (see Fig. 2), the MSE behavior depending on the number of used dimensions (see Fig. 3) as well as the corresponding visual depictions (see Figs. 4, 5 and Fig. 1 in supplemental), we observe that we can take the first ordered latent variables that contribute to 95%–96% of the overall reconstruction and continue training with

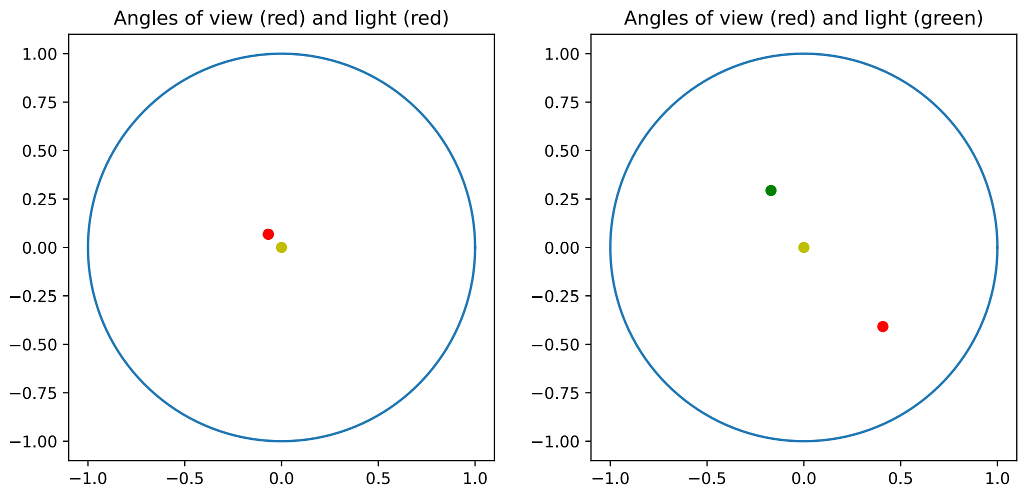

Fig. 8. Angles displayed on the unit disk: view direction V (green), light direction L (red). LE eft: $L = 3, V = 3$ and right: $L = 35, V = 100$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the dimensions 5, 7 and 9 for leather11, fabric01 and carpet05 respectively. Figs. 6, 7 and Fig. 2 in supplemental in show the visualizations of these pruned trainings in comparison to the full training with the initial 32 dimensions. Hence, we achieve a guidance of the compression that allows using different dimensionalities of the latent spaces to represent different materials depending on the complexity of their appearance characteristics which allows a more suitable reconstruction and compression than taking a single fixed number of dimensions for all materials as done by Rainer et al.. To analyze whether the ordering of latent variables according to their Shapley values is reasonable, we provide a comparison of the ordering of the latent dimensions according to their Shapley values to a random ordering (Figs. 3, 4). We observe that the random ordering does not allow insights regarding the choice of a plausible dimensionality of the latent space in contrast to our approach based on Shapley values.

4.2. Image compression

As already discussed in Section 2, there has been significant progress in neural image compression and in particular advanced autoencoder architectures have been demonstrated to be promising. The targeted trade-off of determining an as-compact-as-possible binary representation (i.e. lowest rate bitstream) while preserving a certain level of fidelity (i.e. minimum distortion) of the data has been investigated in terms of autoencoder architectures with quantization and entropy coding. One popular approach for image compression is the lossy compression approach based on compressive autoencoders by Theis et al. [55]. We used the publicly available implementation of this approach [https://github.com/alexandru-dinu/cae], which combines the proposed network architecture with the idea of compressing the code with stochastic binarization [54] instead of rounding it. The benefits of such stochastic binarization for image compression as described by Toderici et al. [54] also include the advantage that bit vectors are trivially serializable/deserializable which helps in efficient data transmission. The latent code then exhibits the form of a binary matrix, where the matrix dimensions directly correspond to the number of bits stored per image and the MSE loss is used for the optimization. Each image is pre-processed in terms of an arrangement of 10 × 6 non-overlapping patches of size 128 × 128 so that an image is represented based on 60 patches and, hence, based on 60 latent codes. For the subsequent processing, we took one of the models provided with the implementation with dimensions of 32 × 32 × 32 (i.e. 32 channels of size 32 × 32 resulting from the used encoder architecture) and analyzed, whether and how much we can reduce the dimension of channels based on the Shapley value based analysis.

Furthermore, for our respective experiments, we used the YouTube-8M dataset that was taken from [https://research.google.com/youtube8m/] for training. Due to the missing information regarding a suitable explicit specification of the number of epochs during training by Theis et al. [55], we analyzed the behavior of training and validation losses and concluded 50 epochs to be a suitable choice for a complete training process. After training for 25 epochs, i.e. after half of the overall training, we perform an analysis of the latent space based on Shapley values. For this purpose, we used the code provided along the DASP approach [38] and extended their implementation of Lightweight Probabilistic Deep Networks [101] by the remaining probability layers required for the underlying architecture that have not been included in the DASP framework.

Even though using DASP for the approximation of the Shapley values is quite fast, it still depends on the number m of examples used for the computation and the time which is used to process a batch, i.e. forward pass of the decoder. So since the decoder of this network is a lot more complex than the one used for the BTF compression, we took less latent codes for Shapley computations for this application. When using sample sizes of six randomly selected patches from 21 randomly chosen images, i.e. 126 latent codes in total, the DASP computation took about 43 min, while the overall training took about 12 h, i.e. the total time is only moderately influenced by the DASP computation.

Figs. 9 and 10 show the contributions and MSE curves obtained by our approach, while Fig. 11 provides a depiction of qualitative results. We can observe that the first 25 ordered latent variables contribute to 95% of the reconstruction but after observing the error plot and some visualizations, we conclude that for this application taking 22 dimensions, which correspond to 92% of contribution, is still a reasonable choice for the subsequent training. Table 2 shows the test error we get if we continue training with different numbers of latent dimensions. Fig. 12 shows the results after the continued training with 22 latent dimensions vs. the full training with 32 channels. Furthermore, Figs. 9 and 10 provide the plots we obtain if we perform the Shapley analysis

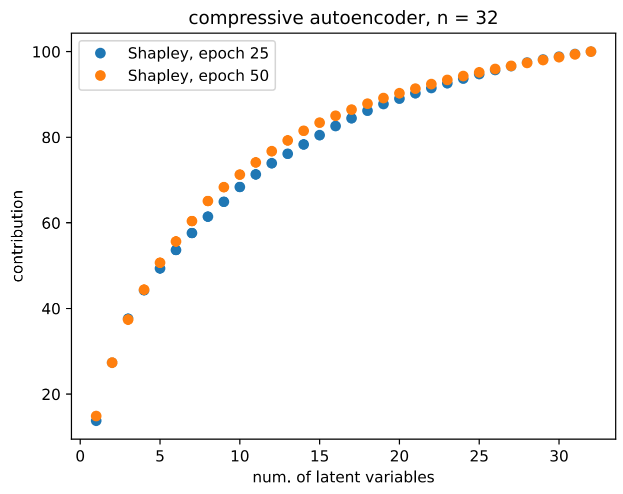

Fig. 9. Cumulative contributions observed when performing Shapley analysis after half of the training vs. after the full training for image compression.

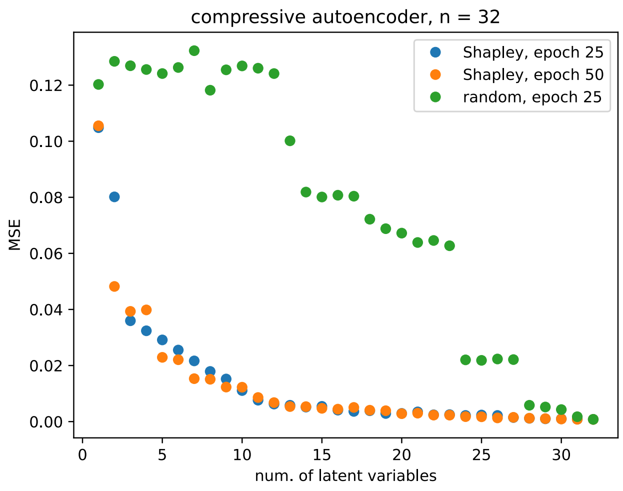

Fig. 10. Illustration of the reconstruction error depending on the number of latent variables that are used for the reconstruction. Note, how the error decays much quicker when variables are added in the order of their importance compared to a random selection of the same number of variables (blue vs. green plot). At the same time, there is no significant difference between applying the Shapley value based analysis after the full or after half of the training (blue vs. orange plot). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

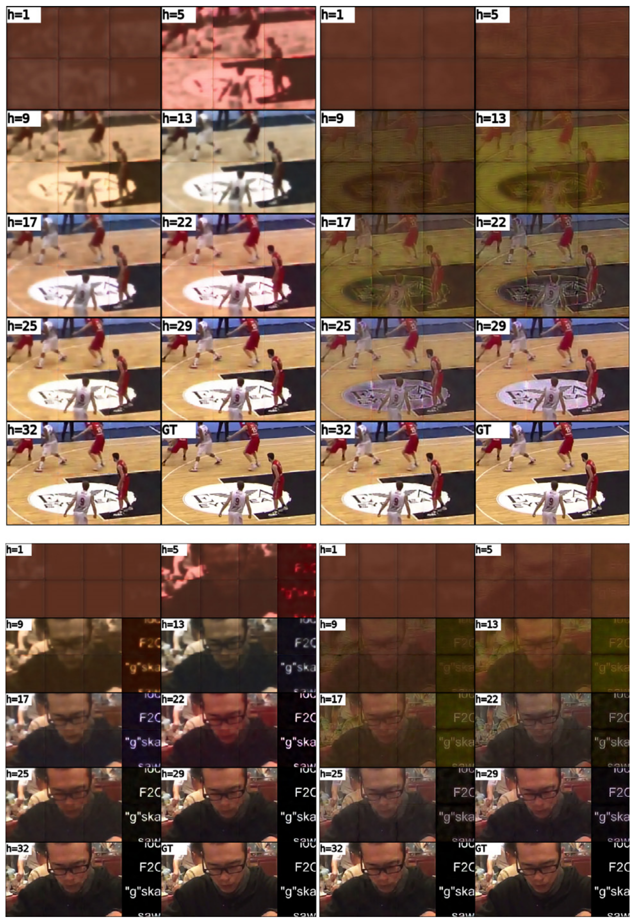

Fig. 11. Two different images reconstructed with different numbers of latent variables and applying the Shapley value based analysis after half of the training. Left: The latent variables are added in accordance to their estimated importance. Right: The latent variables are selected randomly. Note how the random selection leads to a significantly slower convergence. (The blocking artefacts result from the implementation of tiling the original image into patches and are not originating from our approach.)

Table 2
Correspondence between the estimated importance from the Shapley analysis to the final image reconstruction error. While Fig. 13 suggest, that 7 latent variables are sufficient for good quality (a-priori estimation), this analysis shows that more variables should be used (a-posteriori analysis).

% contribution	Dim of net	MSE	MSE increase compared to full net (in %)	MSE increase per pruned dimension compared to full net (in %)
58	7	0.00119	75	3
92	22	0.00075	10,3	1,3
95	25	0.00073	7,35	1,5
98	29	0.00070	3	1
100	32	0.00068	–	–

after a full training. As demonstrated, the results do not significantly change already after 25 epochs, which indicates that applying the analysis regarding relevant dimensions to be used for the subsequent training based on Shapley values after half of the overall training epochs seems a reasonable choice.

To validate that the ordering of latent variables according to their Shapley values is a good choice, we again compared it to the random ordering. Fig. 10 depicts the error for the ordering of the dimensions according to their contribution as computed by the Shapley values in comparison to a random ordering and respective qualitative results are provided in Fig. 11.

Since the main effect of our method is the compression of the latent space, in order to compare to other methods, we studied the effect of alternatively compressing the latent space of the autoencoder based on PCA. Fig. 13 shows the error plot which results if we perform the PCA on the latent matrix and then reconstruct the latent matrix using different numbers p of principal components. From this plot and some image series like in 14 we observed that 7 principal components already suffice to reconstruct the image quite well. So if we link the principal components to the idea of the latent variables and conclude that we only need to train with 7 latent dimensions, we would see that PCA under-estimates the required number of latent dimensions. Fig. 12 shows the reconstructions obtained with the full network, with a pruned network with 22 channels resulting from our choice after the Shapley analysis and with a pruned network with 7 channels, as chosen after a PCA analysis. Moreover, Table 2 also reports the procentual increase of the error. We observe that

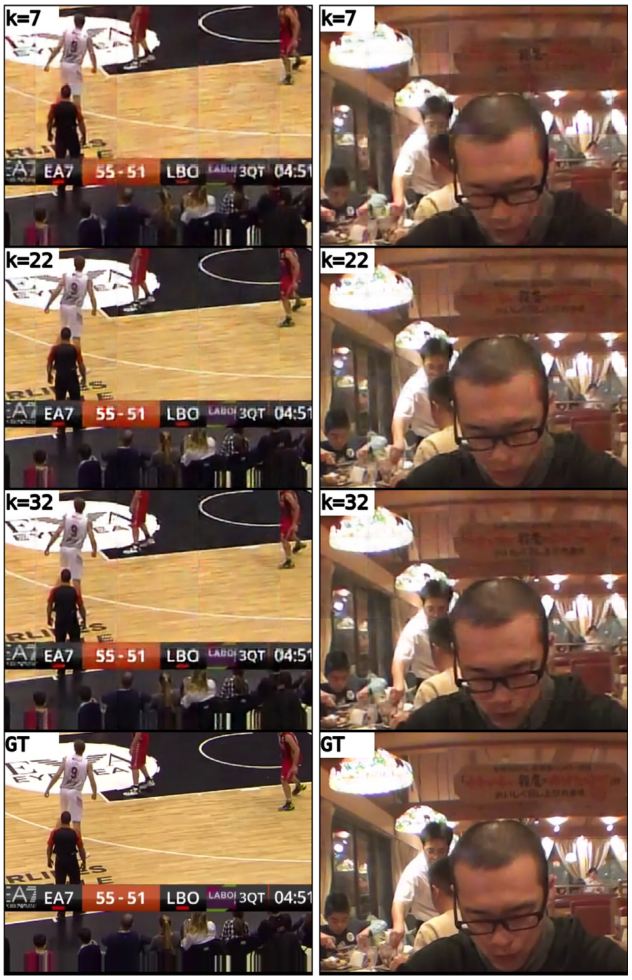

Fig. 12. Final image reconstructions achieved after the full training for different numbers of latent variables.

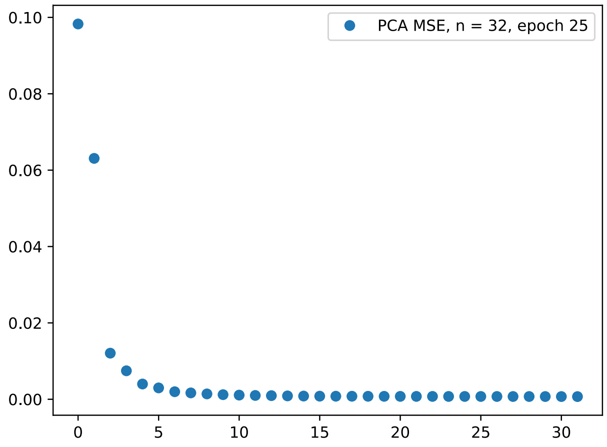

Fig. 13. Resulting error if we perform the PCA on the latent matrix and then reconstruct the latent matrix using different numbers of principal components. While this plot suggests that a low number of variables may be sufficient for the final reconstruction, we find that this does not translate to the number of latent variables required during the training.

with 7 dimensions the quality of the reconstruction is significantly reduced as opposed to what we expected when performing the reconstruction using the PCA. The explanation is that even though we can use only 7 components, the PCA was performed

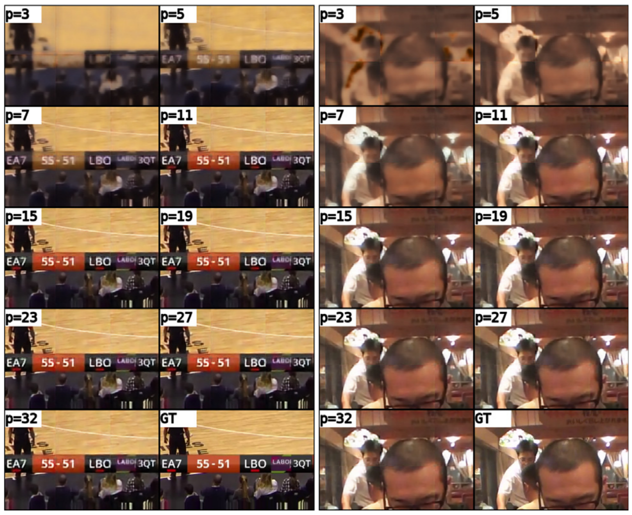

Fig. 14. Images corresponding to the PCA selection strategy shown in Fig. 13.

on the whole set of 32 dimensions which means that this full dimensionality is still indirectly included and not reduced as in the case of retraining the network with less dimensions.

Discussion. We experimented with leather11, fabric01 and carpet05 by training with different initial number of latent variables (32, 64 and 128), performing Shapley analysis in the middle of the training and observed that we get the same results if we prune the latent space of a model with 32 and 64 dimensions. If we prune a model with 128 dimensions, it tends to overestimate the number of latent dimensions (12 in case of leather, 15 in the case of fabric and 17 in the case of carpet). One way to overcome this, when choosing a very large number of dimensions beforehand, would be to perform the Shapley analysis again in the end of the training, prune again and continue training (in our case, training for another 100 epochs was sufficient) to converge to the same result as the one when starting with 64 or 32 dimensions.

Limitations. While DASP allows a better handling of larger numbers of dimensions, the computation of the Shapley values directly involves the decoder function. As a result, the computational burden increases with the complexity of the decoder structure, which results in increasing processing times.

5. Conclusions

We presented a novel approach for efficiently structuring the latent space for explainable data reconstruction and compression in a single training process. In particular, we have demonstrated that leveraging Shapley values to determine the contribution of the latent variables on the model's output which, in turn, allows organizing the latent variables according to a decreasing importance, discarding several latent variables at the same step and, finally, specifying a reasonable size of the latent codes. The truncation obtained when discarding latent variables after the first k latent variables with most importance results in an effect similar to a rank-k approximation as achieved when applying PCA in the linear case. We have demonstrated the relevance of this approach for compact representation and compression for images and high-dimensional material appearance.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.gvc.2022.200059.

References

[1] Rainer G, Jakob W, Ghosh A, Weyrich T. Neural btf compression and interpolation. In: Computer graphics forum, Vol. 38. Wiley Online Library; 2019, p. 235–44.

[2] Rainer G, Ghosh A, Jakob W, Weyrich T. Unified neural encoding of btfs. In: Computer graphics forum, Vol. 39. Wiley Online Library; 2020, p. 167–78.

[3] Shapley LS. A value for n-person games. Contrib Theory Games 1953;2(28):307–17.

[4] Karl Pearson F. LIII. On lines and planes of closest fit to systems of points in space. Lond Edinb Dublin Philos Mag J Sci 1901;2(11):559–72. http://dx.doi.org/10.1080/14786440109462720, arXiv:https://doi.org/10.1080/14786440109462720.

[5] Hotelling H. Relations between two sets of variates. Biometrika 1936;28(3/4):321–77, URL: http://www.jstor.org/stable/2333955.

[6] Ladjal S, Newson A, Pham C-H. A PCA-like autoencoder. 2019, arXiv preprint arXiv:1904.01277.

[7] Pham C-H, Ladjal S, Newson A. PCAAE: Principal component analysis autoencoder for organising the latent space of generative networks. 2020, arXiv preprint arXiv:2006.07827.

[8] Doshi-Velez F, Kim B. Towards a rigorous science of interpretable machine learning. 2017, arXiv preprint arXiv:1702.08608.

[9] Lipton ZC. The mythos of model interpretability. In: ICML workshop on human interpretability in machine learning (WHI). 2016.

[10] Bach S, Binder A, Montavon G, Klauschen F, Müller K-R, Samek W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS One 2015;10(7):e0130140.

[11] Shrikumar A, Greenside P, Kundaje A. Learning important features through propagating activation differences. In: International conference on machine learning. PMLR; 2017, p. 3145–53.

[12] Adebayo J, Gilmer J, Muelly M, Goodfellow I, Hardt M, Kim B. Sanity checks for saliency maps. Adv Neural Inf Process Syst 2018;31:9505–15.

[13] Kindermans P-J, Hooker S, Adebayo J, Alber M, Schütt KT, Dähne S, Erhan D, Kim B. The (un)reliability of saliency methods. In: Explainable AI: Interpreting, explaining and visualizing deep learning. Cham: Springer International Publishing; 2019, p. 267–80.

[14] Ghorbani A, Abid A, Zou J. Interpretation of neural networks is fragile. In: Proceedings of the AAAI conference on artificial intelligence, Vol. 33. 2019, p. 3681–8.

[15] Nie W, Zhang Y, Patel A. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In: International conference on machine learning. PMLR; 2018, p. 3809–18.

[16] Sun Y, Sundararajan M. Axiomatic attribution for multilinear functions. In: Proceedings of the 12th ACM conference on electronic commerce. 2011, p. 177–8.

[17] Sundararajan M, Taly A, Yan Q. Axiomatic attribution for deep networks. In: International conference on machine learning. PMLR; 2017, p. 3319–28.

[18] Montavon G, Lapuschkin S, Binder A, Samek W, Müller K-R. Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognit 2017;65:211–22.

[19] Lundberg SM, Lee S-I. A unified approach to interpreting model predictions. Adv Neural Inf Process Syst 2017;30:4765–74.

[20] Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: Visualising image classification models and saliency maps. In: Bengio Y, LeCun Y, editors. 2nd international conference on learning representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, workshop track proceedings. 2014, URL: http://arxiv.org/abs/1312.6034.

[21] Shrikumar A, Greenside P, Kundaje A. Learning important features through propagating activation differences. In: Precup D, Teh YW, editors. Proceedings of the 34th international conference on machine learning. Proceedings of machine learning research, Vol. 70, International Convention Centre, Sydney, Australia: PMLR; 2017, p. 3145–53, URL: http://proceedings.mlr.press/v70/shrikumar17a.html.

[22] Ribeiro MT, Singh S, Guestrin C. "Why should i trust you?" Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016, p. 1135–44.

[23] Zintgraf LM, Cohen TS, Adel T, Welling M. Visualizing deep neural network decisions: Prediction difference analysis. 2017, CoRR, abs/1702.04595 arXiv:1702.04595 URL: http://arxiv.org/abs/1702.04595.

[24] Fong RC, Vedaldi A. Interpretable explanations of black boxes by meaningful perturbation. In: IEEE international conference on computer vision, ICCV 2017, Venice, Italy, October 22-29, 2017. IEEE Computer Society; 2017, p. 3449–57. http://dx.doi.org/10.1109/ICCV.2017.371.

[25] Matsui Y, Matsui T. NP-completeness for calculating power indices of weighted majority games. Theoret Comput Sci 2001;263(1):305–10. http://dx.doi.org/10.1016/S0304-3975(00)00251-6, URL: https://www.sciencedirect.com/science/article/pii/S0304397500002516, Combinatorics and Computer Science.

[26] Castro J, Gómez D, Tejada J. Polynomial calculation of the Shapley value based on sampling. Comput Oper Res 2009;36(5):1726–30.

[27] Strumbelj E, Kononenko I. An efficient explanation of individual classifications using game theory. J Mach Learn Res 2010;11:1–18.

[28] Maleki S, Tran-Thanh L, Hines G, Rahwan T, Rogers A. Bounding the estimation error of sampling-based Shapley value approximation. 2013, arXiv preprint arXiv:1306.4265.

[29] Datta A, Sen S, Zick Y. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In: 2016 IEEE symposium on security and privacy (SP). IEEE; 2016, p. 598–617.

[30] Tan S, Caruana R, Hooker G, Koch P, Gordo A. Learning global additive explanations for neural nets using model distillation. 2018, arXiv preprint arXiv:1801.08640.

[31] Nohara Y, Matsumoto K, Soejima H, Nakashima N. Explanation of machine learning models using improved Shapley Additive Explanation. In: Proceedings of the 10th ACM international conference on bioinformatics, computational biology and health informatics. 2019, p. 546.

[32] Aas K, Jullum M, Løland A. Explaining individual predictions when features are dependent: More accurate approximations to Shapley values. 2019, arXiv preprint arXiv:1903.10464.

[33] Sellereite N, Jullum M. Shapr: An R-package for explaining machine learning models with dependence-aware Shapley values. J Open Source Softw 2019;5(46):2027. http://dx.doi.org/10.21105/joss.02027.

[34] Bowen D, Ungar L. Generalized SHAP: Generating multiple types of explanations in machine learning. 2020, arXiv preprint arXiv:2006.07155.

[35] Chen H, Lundberg S, Lee S-I. Explaining models by propagating Shapley values of local components. In: Explainable AI in healthcare and medicine. Springer; 2021, p. 261–70.

[36] Lundberg SM, Erion G, Chen H, DeGrave A, Prutkin JM, Nair B, Katz R, Himmelfarb J, Bansal N, Lee S-I. Explainable AI for trees: From local explanations to global understanding. 2019, arXiv preprint arXiv:1905.04610.

[37] Fatima SS, Wooldridge M, Jennings NR. A linear approximation method for the Shapley value. Artificial Intelligence 2008;172(14):1673–99.

[38] Ancona M, Oztireli C, Gross M. Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. In: International conference on machine learning. PMLR; 2019, p. 272–81.

[39] Giudici P, Raffinetti E. Shapley-Lorenz explainable artificial intelligence. Expert Syst Appl 2020;114104.

[40] Sundararajan M, Dhamdhere K, Agarwal A. The Shapley taylor interaction index. In: International conference on machine learning. PMLR; 2020, p. 9259–68.

[41] Chen J, Song L, Wainwright MJ, Jordan MI. L-Shapley and C-Shapley: Efficient model interpretation for structured data. In: International conference on learning representations. 2019.

[42] Kumar IE, Scheidegger C, Venkatasubramanian S, Friedler S. Shapley residuals: Quantifying the limits of the Shapley value for explanations. In: ICML workshop on workshop on human interpretability in machine learning (WHI). 2020.

[43] Lou Y, Caruana R, Gehrke J. Intelligible models for classification and regression. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining. 2012, p. 150–8.

[44] Lou Y, Caruana R, Gehrke J, Hooker G. Accurate intelligible models with pairwise interactions. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining. 2013, p. 623–31.

[45] Wang X, Chen H, Yan J, Nho K, Risacher SL, Saykin AJ, Shen L, Huang H, ADNI. Quantitative trait loci identification for brain endophenotypes via new additive model with random networks. Bioinformatics 2018;34(17):i866–74.

[46] Wang R, Wang X, Inouye DI. Shapley explanation networks. In: International conference on learning representations. 2021, URL: https://openreview.net/forum?id=vsU0efpivw.

[47] Mangalathu S, Hwang S-H, Jeon J-S. Failure mode and effects analysis of RC members based on machine-learning-based Shapley Additive explanations (SHAP) approach. Eng Struct 2020;219:110927.

[48] Tripathi S, Hemachandra N, Trivedi P. Interpretable feature subset selection: A Shapley value based approach,". In: Proceedings of 2020 IEEE international conference on big data, special session on explainable artificial intelligence in safety critical systems. 2020.

[49] Ghorbani A, Zou J. Data shapley: Equitable valuation of data for machine learning. In: International conference on machine learning. PMLR; 2019, p. 2242–51.

[50] Covert I, Lundberg S, Lee S-I. Explaining by removing: A unified framework for model explanation. 2020, arXiv preprint arXiv:2011.14878.

[51] Wang J, Wiens J, Lundberg S. Shapley flow: A graph-based approach to interpreting model predictions. 2020, arXiv preprint arXiv:2010.14592.

[52] Ghorbani A, Zou J. Neuron shapley: Discovering the responsible neurons. 2020, arXiv preprint arXiv:2002.09815.

[53] Ma S, Tourani R. Predictive and causal implications of using shapley value for model interpretation. In: Proceedings of the 2020 KDD workshop on causal discovery. PMLR; 2020, p. 23–38.

[54] Toderici G, O'Malley SM, Hwang SJ, Vincent D, Minnen D, Baluja S, Covell M, Sukthankar R. Variable rate image compression with recurrent neural networks. 2015, arXiv preprint arXiv:1511.06085.

[55] Theis L, Shi W, Cunningham A, Huszár F. Lossy image compression with compressive autoencoders. 2017, arXiv preprint arXiv:1703.00395.

[56] Ballé J, Laparra V, Simoncelli EP. End-to-end optimized image compression. In: 5th international conference on learning representations, ICLR 2017, Toulon, France, April 24-26, 2017, conference track proceedings. 2017.

[57] Cai C, Chen L, Zhang X, Gao Z. Efficient variable rate image compression with multi-scale decomposition network. IEEE Trans Circuits Syst Video Technol 2018;29(12):3687–700.

[58] Rippel O, Bourdev L. Real-time adaptive image compression. In: International conference on machine learning. PMLR; 2017, p. 2922–30.

[59] Nakanishi KM, Maeda S-i, Miyato T, Okanohara D. Neural multi-scale image compression. In: Asian conference on computer vision. Springer; 2018, p. 718–32.

[60] Ballé J, Laparra V, Simoncelli EP. Density modeling of images using a generalized normalization transformation. In: 4th international conference on learning representations, ICLR 2016. 2016.

[61] Ballé J, Laparra V, Simoncelli EP. End-to-end optimized image compression. 2016, arXiv preprint arXiv:1611.01704.

[62] Agustsson E, Mentzer F, Tschannen M, Cavigelli L, Timofte R, Benini L, Van Gool L. Soft-to-hard vector quantization for end-to-end learning compressible representations. 2017, arXiv preprint arXiv:1704.00648.

[63] Ballé J, Minnen D, Singh S, Hwang SJ, Johnston N. Variational image compression with a scale hyperprior. 2018, arXiv preprint arXiv:1802.01436.

[64] Mentzer F, Agustsson E, Tschannen M, Timofte R, Van Gool L. Conditional probability models for deep image compression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, p. 4394–402.

[65] Lee J, Cho S, Beack S-K. Context-adaptive entropy model for end-to-end optimized image compression. 2018, arXiv preprint arXiv:1809.10452.

[66] Li M, Ma K, You J, Zhang D, Zuo W. Efficient and effective context-based convolutional entropy modeling for image compression. IEEE Trans Image Process 2020;29:5900–11.

[67] Minnen D, Ballé J, Toderici G. Joint autoregressive and hierarchical priors for learned image compression. 2018, arXiv preprint arXiv:1809.02736.

[68] Minnen D, Singh S. Channel-wise autoregressive entropy models for learned image compression. In: 2020 IEEE international conference on image processing (ICIP). IEEE; 2020, p. 3339–43.

[69] Tschannen M, Agustsson E, Lucic M. Deep generative models for distribution-preserving lossy compression. 2018, arXiv preprint arXiv:1805.11057.

[70] Agustsson E, Tschannen M, Mentzer F, Timofte R, Gool LV. Generative adversarial networks for extreme learned image compression. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019, p. 221–31.

[71] Yang F, Herranz L, Van De Weijer J, Guitián JAI, López AM, Mozerov MG. Variable rate deep image compression with modulated autoencoder. IEEE Signal Process Lett 2020;27:331–5.

[72] Choi Y, El-Khamy M, Lee J. Variable rate deep image compression with a conditional autoencoder. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019, p. 3146–54.

[73] Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. 2017, arXiv preprint arXiv:1704.04861.

[74] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 4510–20.

[75] Tang Y, You S, Xu C, Han J, Qian C, Shi B, Xu C, Zhang C. Reborn filters: Pruning convolutional neural networks with limited data. In: Proceedings of the AAAI conference on artificial intelligence, Vol. 34. 2020, p. 5972–80.

[76] Khan A, Hines E. Integer-weight neural nets. Electron Lett 1994;30(15):1237–8.

[77] Rastegari M, Ordonez V, Redmon J, Farhadi A. Xnor-net: Imagenet classification using binary convolutional neural networks. In: European conference on computer vision. Springer; 2016, p. 525–42.

[78] Jacob B, Kligys S, Chen B, Zhu M, Tang M, Howard A, Adam H, Kalenichenko D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 2704–13.

[79] Tan M, Chen B, Pang R, Vasudevan V, Sandler M, Howard A, Le QV. Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 2820–8.

[80] Yu J, Yang L, Xu N, Yang J, Huang T. Slimmable neural networks. 2018, arXiv preprint arXiv:1812.08928.

[81] Johnston N, Eban E, Gordon A, Ballé J. Computationally efficient neural image compression. 2019, arXiv preprint arXiv:1912.08771.

[82] Cai C, Chen L, Zhang X, Lu G, Gao Z. A novel deep progressive image compression framework. In: 2019 picture coding symposium (PCS). IEEE; 2019, p. 1–5.

[83] Yang F, Herranz L, Cheng Y, Mozerov MG. Slimmable compressive autoencoders for practical neural image compression. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021, p. 4998–5007.

[84] Eckart C, Young G. The approximation of one matrix by another of lower rank. Psychometrika 1936;1(3):211–8.

[85] Jolliffe IT, Cadima J. Principal component analysis: a review and recent developments. Phil Trans R Soc A 2016;374(2065):20150202.

[86] De Lathauwer L, De Moor B, Vandewalle J. A multilinear singular value decomposition. SIAM J Matrix Anal Appl 2000;21(4):1253–78.

[87] De Lathauwer L, De Moor B, Vandewalle J. On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors. SIAM J Matrix Anal Appl 2000;21(4):1324–42.

[88] Carroll JD, Chang J-J. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. Psychometrika 1970;35(3):283–319.

[89] Harshman RA, Lundy ME. PARAFAC: Parallel factor analysis. Comput Statist Data Anal 1994;18(1):39–72.

[90] Tucker LR. Some mathematical notes on three-mode factor analysis. Psychometrika 1966;31(3):279–311.

[91] Kolda TG, Bader BW. Tensor decompositions and applications. SIAM Rev 2009;51(3):455–500.

[92] Pajarola R, Suter SK, Ruiters R. Tensor approximation in visualization and computer graphics. In: Eurographics 2013 - tutorials, Vol. t6. Girona, Spain: Eurographics Association; 2013, http://dx.doi.org/10.2312/conf/eg2013/tutorials/t6, URL: http://diglib.eg.org/EG/DL/conf/EG2013/tutorials/t6.pdf.

[93] Bartholomew DJ, Knott M, Moustaki I. Latent variable models and factor analysis: a unified approach, Vol. 904. John Wiley & Sons; 2011.

[94] Tipping ME, Bishop CM. Probabilistic principal component analysis. J R Stat Soc Ser B Stat Methodol 1999;61(3):611–22.

[95] Yu S, Yu K, Tresp V, Kriegel H-P, Wu M. Supervised probabilistic principal component analysis. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining. 2006, p. 464–73.

[96] Guan Y, Dy J. Sparse probabilistic principal component analysis. In: Artificial intelligence and statistics. PMLR; 2009, p. 185–92.

[97] Rezende DJ, Mohamed S, Wierstra D. Stochastic backpropagation and approximate inference in deep generative models. In: International conference on machine learning. PMLR; 2014, p. 1278–86.

[98] Kingma DP, Welling M. Auto-encoding variational bayes. 2013, arXiv preprint arXiv:1312.6114.

[99] Weinmann M, Langguth F, Goesele M, Klein R. Advances in geometry and reflectance acquisition. In: Sousa A, Bouatouch K, editors. EG 2016 - tutorials. The Eurographics Association; 2016, http://dx.doi.org/10.2312/egt.20161032.

[100] Weinmann M, Gall J, Klein R. Material classification based on training data synthesized using a BTF database. In: European conference on computer vision. Springer; 2014, p. 156–71.

[101] Gast J, Roth S. Lightweight probabilistic deep networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 3369–78.