**University of Warsaw**
Faculty of Mathematics, Informatics and Mechanics

**Jacek Rutkowski**
Student no. 371580

# Interpretability and Efficiency of Sparse Transformers

**Master's thesis**
**in COMPUTER SCIENCE**

Supervisor:
**dr Marcin Wrochna**
MIMUW

Warsaw, July 1, 2024

## Abstract

The Transformer architecture, although originally designed for Natural Language Processing, gained in recent years much interest from Computer Vision practitioners due to the relatively new Vision Transformer. Since transformers became ubiquitous, there emerged numerous more efficient variants of them, so-called *sparse transformers*. They are marked by the lower (non-quadratic) cost in the tokens length of the attention mechanism, which is the core of the transformer architecture. This advantage is of extreme importance for long sequences. In the present master's thesis, we investigate how two chosen sparse transformers (T2T_ViT and Swin) differ from the vanilla vision transformer.

In order to explore the models' properties, we used several explanation techniques. We have computed Shapley values both from definition and with a much faster approximation using neural networks. As an additional explanation technique, we used the saliency method. Apart from computing explanations on their own, we perform comprehensive experiments to evaluate the explanations and check whether they really explain the models' behavior. We present a custom metric to evaluate the explanations of the classifier models, which is based on tweaked classifiers, called *surrogates*. We also use the ROAD method for explanation evaluation. We conduct experiments on the CIFAR-10 and HyperKvasir datasets. As a result of the experiments, we found out that the neural network-based approximation of Shapley values gives satisfactory results in terms of the evaluation methods used. As for the interpretability of the transofmers, it turned out that the Swin model is the least interpretable.

## Keywords

transformer, sparse transformers, Shapley values, attention, Performer, Swin, T2T_ViT, ROAD, XAI

## Thesis domain (Socrates-Erasmus subject area codes)

11.4 Sztuczna inteligencja

## Subject classification

I. Computing Methodologies
I.4 Image processing and computer vision
I.4.0 General

## Tytuł pracy w języku polskim

Wyjaśnialność i efektywność rzadkich transformerów

# Contents

# Introduction

With the emergence of the AI Act, explanation of machine learning models has become not only a desirable feature, but a necessity. There are numerous methods of explaining model predictions, like LIME, Shapley values, gradient saliency, integrated gradients, etc. They can be divided into two parts: model-agnostic and model-specific methods. Model-agnostic methods treat the model as a black box. They try to find out which features of the input were most and least important for the given prediction. Model-specific methods work only for some given classes of models and analyze their internal behavior, like gradients in neural networks. Therefore, model-agnostic methods are more universal. Among them, the most compelling method are Shapley values. It has strong intuitive axiomatic properties that fulfill reasonable requirements for an explanation.

However, there is a caveat here. Shapley values are prohibitively costly to compute. If we wanted to compute Shapley values for each pixel on the $224 \times 224$ image, then we would require $2^{224 \times 224}$ computation steps, which is much more than the number of atoms in the universe. Even if we divided an image into patches and wished to compute Shapley values for these patches instead of pixels, the exponential time of computations severely restricts the range of possible applications.

There have been several solutions to tackle this problem and compute Shapley values more efficiently, like KernelSHAP or FastSHAP. Later on, we study a method based on the FastSHAP approach. In this method, we do not compute Shapley values from scratch, but we tailor a loss function that leads in its argmin to Shapley values. This loss function is constructed in such a way that it does not require any ground-truth Shapley values. It is advantageous, insofar as it would be prohibitively costly to obtain such ground-truth values.

As we are interested in the explanations themselves and not only in the model predictions, we want to evaluate these explanations. Shapley values, as such, have strong theoretical guarantees, so if we compute them from definition, we could expect the evaluation metrics to be high. However, we may not be sure whether Shapley values computed with some other method approximate them sufficiently, and thus the evaluation becomes required in order to check whether the provided explanations are reasonable. As an evaluation of explanations, we use the ROAD metric and a similar custom metric.

Besides explanations of the models, we also investigate the models on their own. We use three transformer-based models designed for computer vision. Two of them are sparse transformers, and we are especially interested in comparisons between sparse and not-sparse architectures, performance, and explanations.

The overall structure of the work is as follows: at the beginning, we introduce and discuss two basic concepts: Shapley values and sparse transformers. Then we analyze in more detail the used architectures. Next, we explain how Shapley values can be computed for visual transformers. The last and most important chapter contains a description of the experiments we conducted.

In it, we firstly study the performance of the models on two datasets: CIFAR-10 and

HyperKvasir, a gastrointestinal dataset. Secondly, we explore the Shapley values explanations of the models on both datasets. Lastly, we evaluate the explanations and compare them with the standard saliency explanation. For the HyperKvasir dataset, we have access to segmentation masks for one class, which gives another compelling evaluation of the explanations: we can check whether Shapley values are high for the areas responsible for membership to a given class.

# Chapter 1

# Basic concepts

## 1.1. Shapley values

Shapley values were introduced by Lloyd Shapley in 1951 as a concept in cooperative game theory [22]. Using them, one can answer the question of how a reward should be "fairly" distributed between players in a cooperative game. For instance, imagine that some people want to solve a problem, and they get a reward inversely proportional to the time spent. How should we divide the money? The naive approach could be to separate each individual from the group and check how much time it will take the others to manage the issue. According to this time, we distribute the reward.

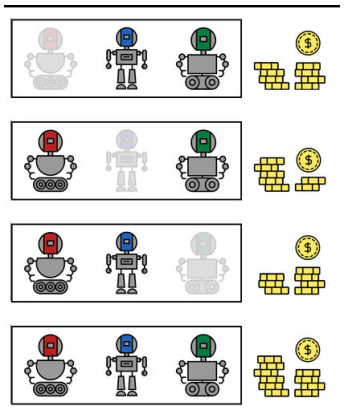


Figure 1.1: Rewards without chosen players. Source: [20].

However, such an approach does not take into account the fact that some subgroups might have an "emergent" value, i.e., they can give a much bigger (or much worse) value together than separately. It can be that the green robot vibes perfectly with the blue robot, but it is completely useless alone. If the green robot copes well only in groups, it would be reasonable to give him a large part of the final reward. That is why we should take into account all the possible coalitions (see Figure 1.2).

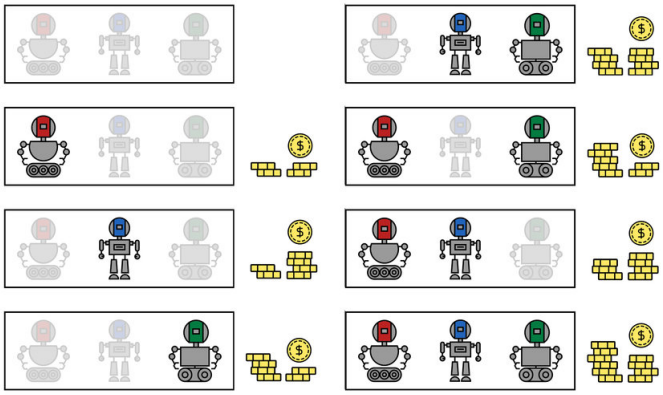


Figure 1.2: Rewards of all coalitions. Source: [20].

Shapley values for a coalitional game emerged as a unique function that satisfies some natural axioms. In order to formulate them, let us introduce some notation. Let $X$ be the set of players, and $v : 2^X \to \mathbb{R}$ be a function (called coalitional game). We interpret the value $v(S), S \subseteq X$ to be the worth of the coalition $S$. Then the four axioms of Shapley values function $\phi : X \to \mathbb{R}$ are as follows:

- **Efficiency** $\sum_i \phi(i) = v(X)$
- **Symmetry** If the players $i, j$ are symmetric (i.e. $v(S \cup \{i\}) = v(S \cup \{j\})$ for all $S \subseteq X$), then $\phi(i) = \phi(j)$
- **Nullity** If a players $i$ has only zero contributions (i.e. $v(S \cup \{i\}) = v(S)$ for all $S \subseteq X$, then $\phi(i) = 0$
- **Additivity** If there are two coalitional games $v, w$ and their corresponding Shapley value functions are $\phi, \psi$, then for every player $i$, $\phi(i) = \psi(i)$.

These axioms guarantee the above-mentioned "fairness" of this method.

It turns out that the Shapley values function not only does exist for every coalitional game, but it is also unique and can be given by an exact formula [22].

**Theorem 1.1.1** *For the set of players $X$ and the coalitional game $v$, there exists exactly one function $\phi$ that satisfies the above axioms. The value of the function $\phi$ can be given explicitly by the formula:*

$$\phi(i) = \sum_{S \subseteq X \setminus \{i\}} \frac{|S|!(|X| - |S| - 1)!}{|X|!} \left(v(S \cup \{i\}) - v(S)\right).$$

In the standard Machine Learning setup, $X$ is the set of tokens (like words in NLP or patches in Computer Vision), and $v$ is, for instance, the classification model. Shapley values show how important given features are for the model's prediction. One of the important issues with the application of Shapley values is the way of defining $v(S)$ in the above definition. It is not always straightforward how to define the value of the coalition, or, to be more specific, how to mask-out those tokens in the input that we wish to be absent. For the NLP tasks, we can simply ignore the words not belonging to $S$, but when the input is an image, it is necessary to put something in place of the "omitted" patch in order to match the dimensions of the model input. One possible solution is zeroing-out the patches not belonging to $S$. However, this

can lead to the off-manifold problem (see section 3.1 below). While the model was originally trained to fit the data on some high-dimensional manifold, putting zeros throws the input away from this manifold, which can yield unexpected behavior of the model.

If we denote the number of players by $n$ then the number of all possible coalitions is $2^n$. Therefore, it is impossible to consider all the possible coalitions, even for small $n$. It creates the need for some faster approximations with a speed-accuracy trade-off.

In the field of XAI, Shapley values are referred to as a model-agnostic method of explanation since it does not take into account any specific properties of the considered model. It takes only the outputs of the model, no matter how they are generated. However, when we use approximation techniques, Shapley values can depend on the model architecture, as we will see in Chapter 4. Here we introduce two sparse transformer models, which we compare with the vanilla transformer architecture during the experiments.

## 1.2. Sparse transformers

The Transformer [26] is a deep learning model that was originally proposed as a sequence-to-sequence model [23] for machine translation. The main advantage of transformers over previous state-of-the-art RNN or LSTM models is their ability to capture long-range dependencies in the input sequences. While RNN and LSTM process the input sequentially, transformers manage all the input sequence simultaneously [10]. Besides language related applications, transformers turned out to be successful also in computer vision [7] or audio processing [12].

The Transformer is a sequence-to-sequence model divided into encoder and decoder models. Each of them is a stack of $L$ identical blocks. The encoder consists of a multi-head self-attention module, which is the core element of transformer and a linear layer on top of it. Decoder has a similar architecture. Apart from the encoder components, it also inserts a sub-layer that performs self-attention over the output from the encoder. Another modification is masking, which prevents the decoder from attending to subsequent positions. It ensures that predictions for the $i$-th token depend only on the previous tokens. The overall architecture of the vanilla Transformer is given in Figure 1.3.

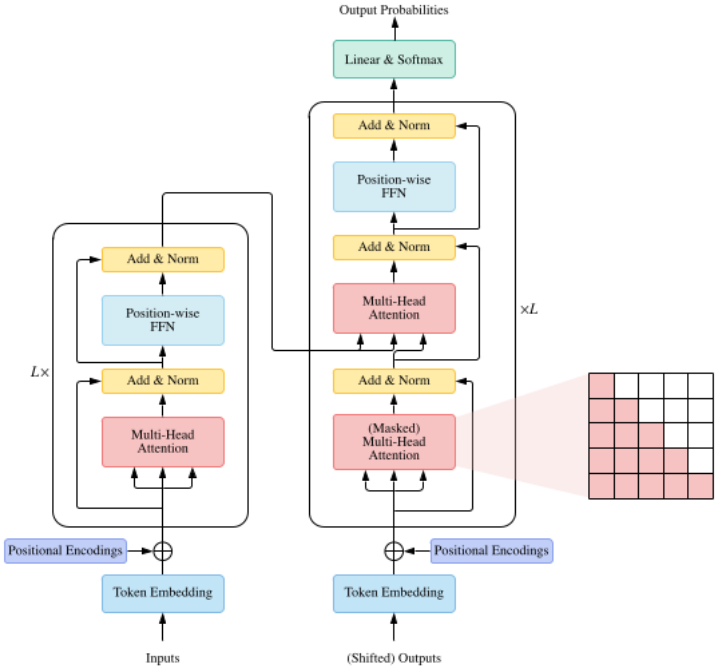



Figure 1.3: The vanilla Transformer architecture.

The core component of the Transformer architecture is the attention module, which can be defined as follows.

**Definition 1.2.1** *Let $Q \in \mathbb{R}^{N \times D_k}$, $K \in \mathbb{R}^{M \times D_k}$ and $V \in \mathbb{R}^{M \times D_v}$ be matrices. The scaled dot-product attention is given by*

$$\text{Attention}\left(Q, K, V\right) = \text{softmax}\left(\frac{QK^T}{\sqrt{D_k}}\right)V.$$

The matrices $Q, K, V$ are referred to as queries, keys, and values, respectively. Queries and keys are responsible for determining which values to attend. The most important values are those for which queries and keys are similar, i.e., have a big dot product. They are, in addition, bumped up by the softmax function. The scaling factor $\frac{1}{\sqrt{D_k}}$ serves to counteract the effect that the dot product of $Q$ and $K$ grows large, pushing the softmax function into regions where it has extremely small gradients [26].

Instead of performing a single attention, it turned out to be beneficial to stack several attentions in parallel (see Figure 1.4). Such multi-head attention allows the model to attend information from different representations of the input. Experiments conducted in [26] showed that different heads learn to perform different tasks.

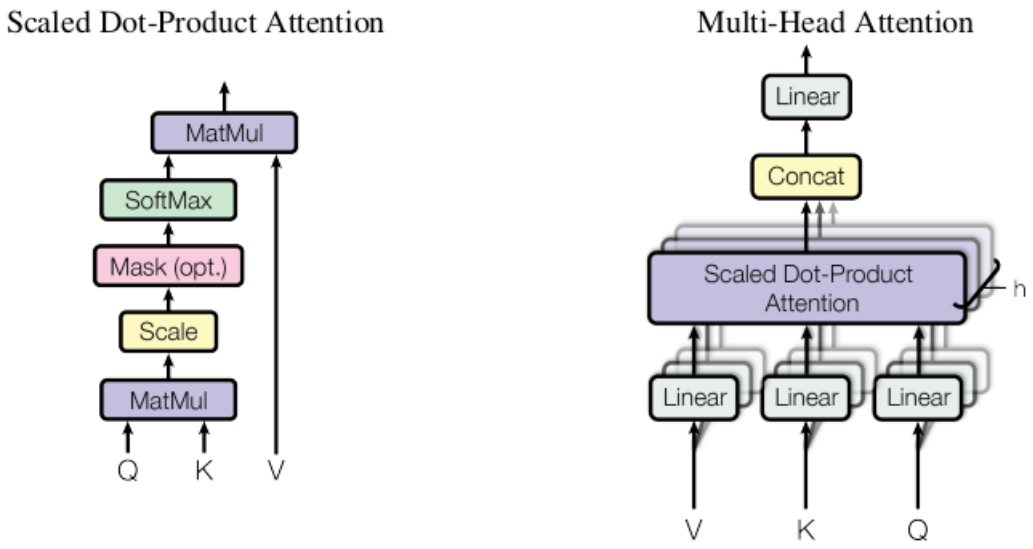



Figure 1.4: Scale-dot product attention (left) and Multi-head attention which consists of several stacked attention layers.

Since the Transformer has become the go-to architecture for numerous tasks, a variety of its variants (a.k.a. X-formers) have been proposed (see [12] where a taxonomy of X-formers was presented). One of the most important challenges of applying Transformer is its relative inefficiency. Indeed, Transformer uses all query-key pairs, which gives quadratic complexity in the length of the input. There have been numerous proposed methods to avoid this issue [25]. In our task to analyze the interpretability of sparse transformers, we have chosen two diametrally different examples of them.

# Chapter 2

# Used models

The quadratic complexity of the Transformer occurs in the matrix multiplication $QK^T$, i.e., during the comparison of each query with each key. One of the main methods to get around this problem is to avoid this heavy multiplication by forcing only restricted query-key pairs. Many approaches restrict attention only to some predefined local neighborhood [17]. Somewhat more advanced approaches limit the field of attention view to some patterns, either fixed or learnable [25]. Such an approach was used in the Swin model [14] which we will describe later 2.3.

Another popular method is to avoid costly matrix multiplication $QK^T$ and instead multiply first keys by values ($K^TV$) and only then by queries ($Q(K^TV)$). Due to the softmax operation on the result of the multiplication $QK^T$ we cannot simply change the order, but it turns out to be possible by viewing the attention mechanism through kernelization. Kernels enable us to omit the softmax operation by introducing other matrices, which allow us to represent the result of the attention mechanism as the expected value of the product of some other matrices: $Q', K'$ with the same $V$. This approach was applied in Performer [5] which we will describe in more detail now.

## 2.1. Performer

The key component of the Performer's architecture is the FAVOR+ mechanism (*Fast Attention Via positive Orthogonal Random features*), a new method for approximating softmax and Gaussian kernels. The standard attention can be presented in the following way:

$$\mathrm{Attn}(Q,K,V) = D^{-1}AV,\ A = \exp\left(QK^T/\sqrt{d}\right),\ D = \mathrm{diag}\left(A1_L\right). \tag{2.1}$$

Here, the matrix $D$ is introduced in place of the softmax denominator in order to have a more convenient form of the attention matrix $A$. The quadratic complexity comes from the fact that we have to apply the exponent function to the product of $Q$ and $K^T$. If we could replace the exponent by some linear function, we could multiply $K^T$ with $V$ first and thus reduce the multiplication cost (see Figure 2.1). That is where the kernels come in.

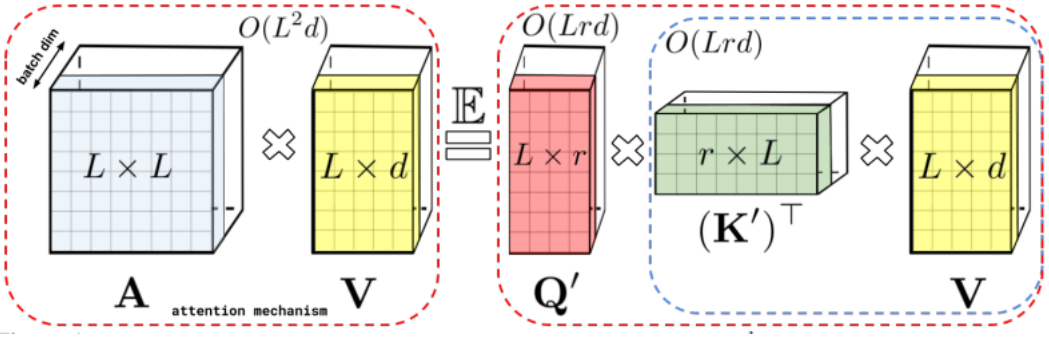



Figure 2.1: In the standard attention mechanism (on the left), we first get the $L \times L$ matrix with quadratic time complexity and then have to multiply it by $V$ also with quadratic time. The Performer adopts kernels to change the multiplication order and reduce the time complexity to linear.

Kernels are functions that are scalar products in some other vector space. More specifically, let $\phi : \mathbb{R}^d \to \mathbb{R}^r_+$ be a randomized mapping and $x, y \in \mathbb{R}^d$. Then we define the kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ as:

$$K(x, y) = \mathbb{E}\left[\phi(x)^T \phi(y)\right]. \tag{2.2}$$

In our case we have a matrix $A$ as defined in 2.1 which we would like to linearize in some way. We define the function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ as:

$$K(q_i^T, k_j^T) = A(i, j) = \exp\left(q_i k_j^T / \sqrt{d}\right). \tag{2.3}$$

If there exists such a probabilistic space and random maps $\phi$ in this space such that the equation 2.2 holds, then we could put the matrix $V$ in the attention definition 1.2.1 inside the expected value and multiply $\phi(k_j^T)$ by the matrix $V$ and only then by $q_i$. That is the "FA" (*Fast Attention*) part of the FAVOR+ acronym. Note that we can choose both the mapping $\phi$ and its output dimension $2m$ to better approximate the matrix $A$.

Not every kernel that satisfies 2.3 is suitable for approximation. If we use, for instance, a known formulation of softmax using trigonometric functions, it leads to unstable behavior and big variance, especially when kernel scores are close to zero (which is the case for $A$ because many of its entries correspond to tokens of low relevance). Specifically, this is the case if we take:

$$\phi(x) = \frac{\exp(||x||^2/2)}{\sqrt{m}}\left(\sin(\omega_1^T x), \ldots, \sin(\omega_m^T x), \cos(\omega_1^T x), \ldots, \cos(\omega_m^T x)\right), \tag{2.4}$$

where $\omega_1, \ldots \omega_m \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d)$. In Lemma 2 from [5] it is proven that although the kernel given by 2.4 is unbiased, its variance tends to infinity as approximated values tend to 0.

That is why the *positive random features* were introduced. Positive random features are given by the random map feature:

$$\phi(x) = \exp\left(\omega^T x - \frac{||x||^2}{2}\right),$$

where $\omega \sim \mathcal{N}(0, I_d)$, which is positive due to the exponent. Since the kernel is the expected value of random maps $\phi$, in practice we have to sample from the normal distribution. In order to even further reduce the variance, we can apply the Gram-Schmidt orthogonalization procedure to the sampled $\omega_1, \omega_2, \ldots, \omega_m$. This step requires $m \leq d$, that is, the number of samples has to be smaller than the embedding dimension, which was always the case in [5].

## 2.2. T2T_ViT

T2T_ViT [28] is a visual transformer that is based on the Performer backbone and uses a special tokens-to-token (T2T) tokenization module in place of convolutions. This technique can model the local structure information of surrounding tokens and reduce the length of tokens iteratively, replacing many tokens with only one. The vanilla vision transformer splits each image into a sequence of tokens without taking into account their spatial order. It was argued in [28] that this simple tokenization hinders modeling of local structure. Although ViT achieves satisfactory performance on benchmarks, it is difficult to train it from scratch, and this training requires a lot of data because ViT is not prepared to model the local structure and has to learn it somehow from scratch. The Tokens-to-Token module aims to overcome this issue by recursively aggregating neighboring tokens into one token. The main idea and the standard block of the T2T_ViT architecture are shown in Figure 2.2.

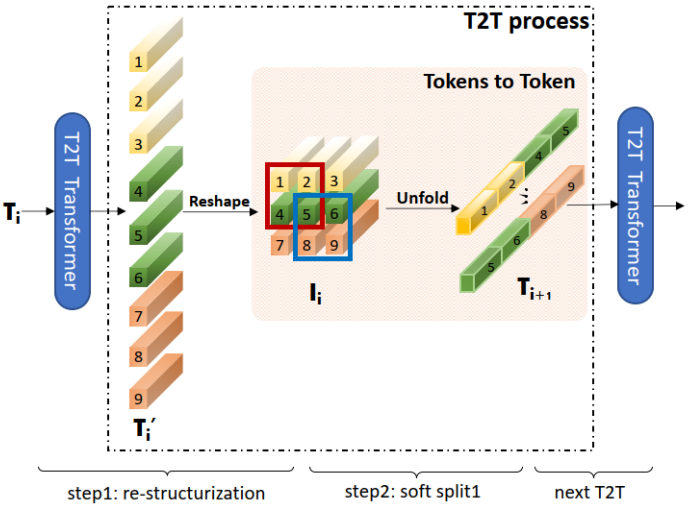



Figure 2.2: T2T_ViT basic block. Source: [28]

.

The Tokens-to-Token module is used between two transformer layers and is the core of the architecture. It consists of two phases:

- **Re-structurization** Given the output from the transformer layer, it is transformed as an image into a spatial representation.

- **Soft Split** When we have the spatial representation, we unfold so that neighbors are in one token. The unfold operation is performed after each reshaping in the Tokens-to-Token module and at the very beginning of the model, when the image itself is unfolded. In Figure 2.2, the tokens 1, 2, 4, 5 which are close to each other on the image, become one token after the Soft Split operation. With this operation, we not only aggregate the local information from surrounding pixels and patches but also reduce the number of tokens. In order to avoid information loss, the patches that are unfolded overlap. In this way is created a prior that patches close to each other have a bigger correlation.

## 2.3. Swin

The Swin model (**S**hifted **win**dow) [14] limits the computations related to the attention matrix by the window-based approach. Self-attention is calculated only on some subsets of neighboring patches (windows) rather than on the whole image. If we choose the window size to be constant, then the complexity is reduced to linear on the number of patches.

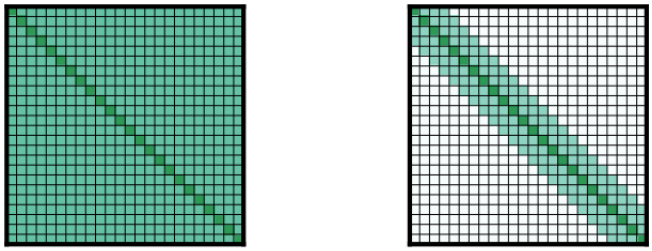


Figure 2.3: In the full self-attention (left) pattern, each token attends to each other, while for the window approach (right), tokens attend only to their neighborhoods. Source: [1]

.

A straight-forward solution in this window-based framework could be a pattern where each token attends only to a fixed number of neighboring tokens $w$. Such an approach was used in Longformer [1]. The complexity of this pattern is $O(w \times n)$, where $n$ is the number of tokens, so constant value of $w$ gives the linear complexity.

There is, however, a caveat to this method, since pixels lying far from each other do not appear together in any window. In order to provide some connections across windows in [14] a shifted window partitioning approach was proposed (see Figure 2.4).

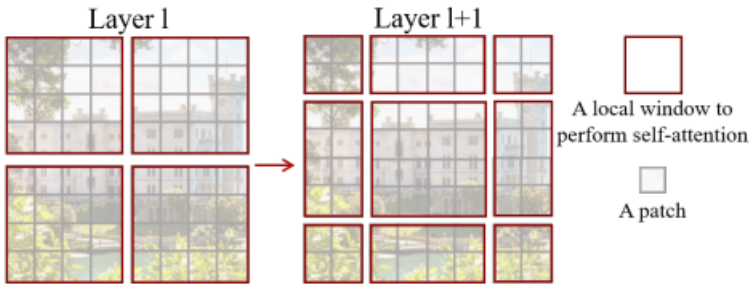



Figure 2.4: In the Swin model, self-attention is computed only within local windows, which vary in consecutive layers.

As the name suggests, this approach consists of a shifting window when attention is computed. Thus, at each layer, the tokens attend to other tokens. Moreover, since windows are shifted by half of their size, there is much space in common, which gives some connections between far tokens.

To finally improve performance, Swin Transformer constructs hierarchical feature maps like feature pyramid network [13] or U-Net [19].

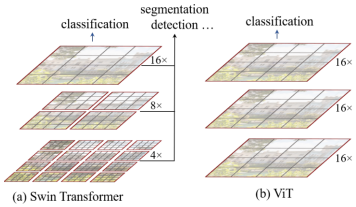



Figure 2.5: Swin model (a) computes self-attention only within each local window. On the other hand, standard ViT (b) uses self-attention on the entire image.

Swin Transformer proceeds from small windows consisting of small patches up to one big window with bigger patches. Window size increases with the size of patches that are merged from smaller patches. In this way, the number of tokens in self-attention can remain constant.

# Chapter 3

# Shapley Values for Visual Transformers

Since its first usage in computer vision in [7], the Transformer has become the state-of-the-art model also in this domain. Thus, it became urgent to provide some technique for explanaining transformers applied to images. At first glance, it might seem that attention is self-explanatory, as attention values can indicate the importance of features. However, recent works ([21], [4]) raised questions about the validity of attention values as an explanation. It was shown that this method provides an incomplete picture of a model's dependence on each token as it takes into account only the attention component in the sequence of the model's operations. The attention component taken apart provides an incomplete picture of the model's dependencies.

That is why there is a need for another solution, and Shapley values come in as a theoretically grounded explanation method. The main drawback of this method is its computational cost, since it has exponential time in the number of players, which in the context of computer vision are patches. In [6], a method was proposed to efficiently estimate Shapley values for vision transformers. This method generates Shapley value explanations via a separate, learned explainer model. In the following chapter, we will describe this approach in more detail.

## 3.1. Surrogate model

Shapley values are a removal-based explanation, i.e., they measure feature importance by removing features and checking how the prediction changes. In the context of computer vision, features are patches, so to compute the Shapley value of a computer vision model, we need to remove the chosen patches from the image.

Using the self-attention operation, we can remove patches in an elegant way. We can ignore the chosen patches by masking them in the attention operation. This approach is similar to the causal attention in transformer language models like GPT-3 [2].

Another approach could be to set some patches to zero or average the prediction across randomly sampled replacement values. As it was shown in [15], ViTs are highly robust to occlusions. It means that simply zeroing out the patches does not degrade much the model's performance. Given a ViT model $f$, we can evaluate it on a masked input $\mathbf{x_s}$. However, it leads to the off-manifold problem: the model $f$ was trained only on full images, and masked images are out of the learned data distribution [24]. To manage this problem, a surrogate model was introduced [8]. It is trained on masked images to imitate the outputs of the original model on unmasked images. If $\mathbf{x}$ is the original image and $\mathbf{x_s}$ is the image with masked patches from the set $s$, then we train the model $g(\mathbf{x_s})$ with parameters $\beta$ to have the nearest

possible output to the output of $f(x)$. More formally, we fine-tune the model $g$ using the following loss:

$$\min_\beta \mathbb{E}_{p(x)} \mathbb{E}_{p(s)} \left[ D_{\mathrm{KL}} \left( f(\mathbf{x}; \eta) || g(\mathbf{x_s}; \beta) \right) \right],$$

where $p(s)$ is the distribution over subsets. This loss has a theoretical guarantee that the optimal solution $g(\mathbf{x_s}; \beta*)$ satisfies the property:

$$g(\mathbf{x_s}; \beta*) = \mathbb{E} \left[ f(\mathbf{x}; \eta) || \mathbf{x_s} = x_s \right],$$

i.e., the optimal model $g(\mathbf{x_s})$ outputs the expected prediction given the available information about unmasked input $x$. With such a model, we are able to provide more robust and trustworthy model outputs on the masked images. It is needed both for computing the Shapley values from scratch and for the explainer model.

## 3.2. Model for Shapley values computation

Using the surrogate model, we can introduce the coalitional game $v_{xy}(s) := g_y(x_s; \beta)$, where $y$ is one of the classes and $s$ some subset of image patches. Instead of training a neural network on some ground-truth Shapley values, one can use an optimization-based characterization of the Shapley value [3].

This characterization allows us to find the Shapley values by minimizing the following objective introduced in [11]:

$$\begin{gathered} \mathcal{L}(\theta) = \mathbb{E}_{p(x,y)} \mathbb{E}_{p(s)} \left[ \Big( v_{xy}(\mathbf{s}) - v_{xy}(\mathbf{0} - \mathbf{s}^T \phi_{ViT}(\mathbf{x}, \mathbf{y}; \theta) \Big)^2 \right] \\ \text{s.t. } \mathbf{1}^T \phi_{ViT}(x, y; \theta) = v_{xy}(\mathbf{1}) - v_{xy}(\mathbf{0}) \text{ for all } x, y, \end{gathered} \tag{3.1}$$

where $p(\mathbf{s})$ is a distribution defined as $p(s) \propto (\mathbf{1}^T s - 1)!(d - \mathbf{1}^T s - 1)!$ for $0 < \mathbf{1}^T s < d$ and $p(\mathbf{1}) = p(\mathbf{0}) = 0$ and $\phi_{ViT}$ is the explainer model, which takes the image and target class as input and outputs the approximate Shapley values in a single forward pass. Compared to the definition of Shapley values 1.1.1, the probability distribution $p$ accounts for the fraction with factorials, and the crucial part is the fact that there is no sum over all possible coalitions. It is crucial because the number of coalitions ($2^n$ is what makes the computation of Shapley values so expensive. In one pass, $\phi_{ViT}$ approximates in some sense only $v_{xy}(\mathbf{s}) - v_{xy}(\mathbf{0})$, i.e., the profit of taking players from the set $s$ to the coalition. The explainer model needs for training only a pretrained surrogate model. Using it, we can define the loss 3.1.

# Chapter 4

# Experiments

We conduct experiments on CIFAR-10 and HyperKvasir, which is the largest available gastrointensinal dataset. Both sets are well-suited to the task of explanation because they have quite easily observable important patches, which we expect to be marked as such by explanation techniques. CIFAR-10 is a well-known dataset with everyday life classes such as car, bird, cat, dog, and truck. HyperKvasir is a more specialist dataset that contains gastrointestinal images with some abnormal growths. As is often the case with medical data, some findings occur more frequently than others, which makes the data classes imbalanced. To overcome this issue, we have chosen 1000 images with polyps and 1000 images without polyps. In this way, we were able to train more reliable classifier models.

A considerable advantage of the HyperKvasir dataset is that it provides some ground truth for explanation. Together with images, there are segmented images with assigned anomalies. Therefore, explanations will give us information on whether the model predictions are based on patches where anomalies indeed appear or whether the model takes random or arbitrary features, like was the case with the famous husky misclassification example (see Figure 4.1).

(a) Husky classified as wolf (b) Explanation

Figure 4.1: The explanation showed that the model did not even take into account the husky face but only the "wild" background.

We present in Figure 4.2 the example polyps together with their ground-truth segmentation masks. In further experiments, we will use these segmentation masks to analyze the explanations of the models for the polyp class (see Section 4.4 below).

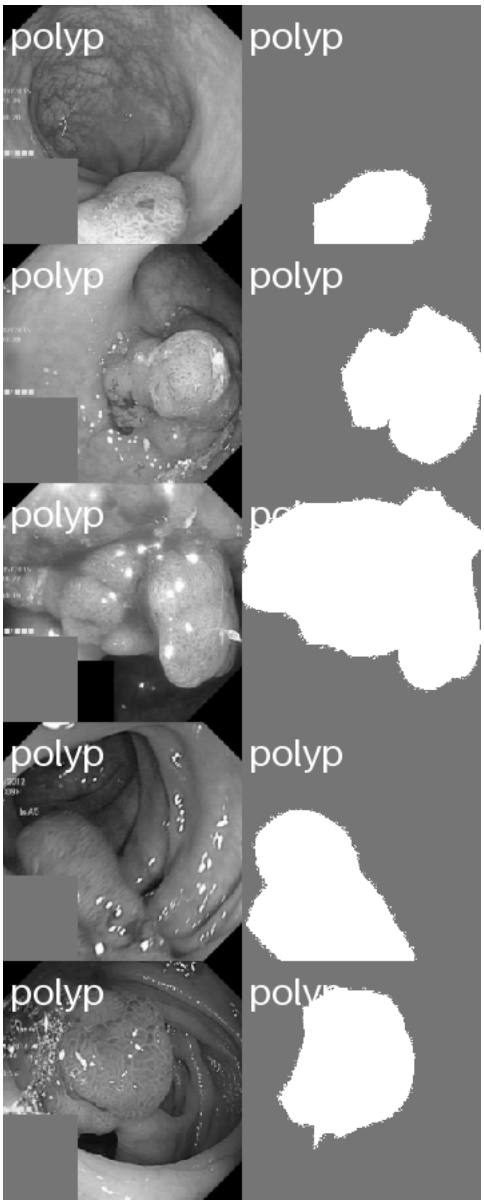



Figure 4.2: Polyps with their segmentation masks

The overall experiments were conducted in three stages. At first, we trained classification models on the unchanged data. Then we trained surrogate models, which are classification models on the masked data. The surrogate model is required for training the explainer model, which explains surrogate predictions by giving approximations of Shapley values.

For both classifier and surrogate models, we used batch size 64, learning rate $5e-5$, and the AdamW optimizer. The explainer model requires more hyperparameters. The most important of them is the freeze_backbone, which decides how many layers to freeze from the explainer backbone. It was noted in [6] that ViTs are difficult to train, and we find that fine-tuning is important to train the explainer effectively. We performed experiments to verify which freezing strategy works best. We took into consideration three options: freeze all the layers, freeze all the layers except the last two, and no freezing. We found out that the last option, although the slowest, gave the best results.

Overall, after many experiments with hyperparameters, the final setup included: 6 classifier

models (3 model types, 2 datasets), 12 surrogate models (3 model types, 2 datasets, and 2 number of players: either 16 or 196), and 48 explainer models (3 explainers to explain each of the 12 surrogate models).

## 4.1. Image classification on CIFAR-10 and HyperKvasir

For image classification, we used two sparse transformers, Swin and T2T_ViT (with Performer attention) described in Section 2, pretrained on the ImageNet dataset. We also used the plain Visual Transformer (ViT) [7]. In later sections, we will focus on differences between ViT and sparse transformers, both in terms of performance and interpretability.

For the training, we used standard augmentation techniques like RandomResizedCrop, RandomHorizontalFlip, or RandomRotation. HyperKvasir images had to be resized to fit the models' input size of $(224 \times 224)$. HyperKvasir has much less data than CIFAR-10, so we used more augmentations. That is why classifiers obtain considerably smaller results on this dataset.

The accuracy results of transfer learning are given below.

Table 4.1: Accuracy of classifiers after fine-tuning on CIFAR-10 and HyperKvasir datasets.

| Model | HyperKvasir | CIFAR-10 | Params number |
|---|---|---|---|
| ViT | 90.5 | 97.7 | 20.7 M |
| T2T_ViT | 93.7 | 97.7 | 20.2 M |
| Swin | 91.3 | 98.0 | 26.3 M |

## 4.2. Surrogate models

In Section 3.1 we briefly introduced surrogate models and explained why it is useful to additionally train them. Here we provide the results along with more details.

A surrogate model is a classification model trained on the masked inputs to resemble the outputs of another model. To train the surrogate model, we need to prepare the masked data. This preparation procedure looks as follows. For both datasets (CIFAR-10 and HyperKvasir), we resize the image to the size of $224 \times 224$. Each pixel belongs to either one of $4 \times 4$ or one of $16 \times 16$ patches (consisting of $56 \times 56$ or $16 \times 16$ pixels, respectively). We treat these patches as players. Thus, we have either $4 \times 4$ or $16 \times 16$ players.

In the first case, we have $2^{16} = 65,536$ coalitions. This number is small enough to make the computation of Shapley values from scratch feasible. It enables us to make comparisons with explainer models. For 196 players, the computation from scratch would be too long, so we have to use the explainer models only. The explanations for more players give more precise and interpretable attributions. For instance, if we have to explain the model output for the class "cat," the $56 \times 56$-pixel patches can contain many cat features, and only explaining the $16 \times 16$-pixel patches can give a fine-grained response of which features were crucial.

At the beginning, we present the performance of surrogate models. Below (Figures 4.3) are presented the images from the batch of masked CIFAR-10 images with marked predicted classes (green if correct, red for misclassification) for the ViT model. One can notice that the model makes mistakes practically only for the completely or almost completely masked images.


Figure 4.3: ViT classifier model

In order to see how the performance was improved due to the additional training on the masked data, we can view similar image grids for the ViT surrogate model (Figure 4.4) and a summary of metrics for all models in Table 4.2. We can see that Swin has the best results on the masked inputs. Note that surrogate models obtain better results than their classifier counterparts on the unmasked images. We may therefore say that masking works as data augmentation.


Figure 4.4: ViT surrogate model

Table 4.2: CIFAR-10 classification accuracy

| Model | Vanilla unmasked | Surrog unmasked | Vanilla masked | Surrog masked |
|---|---|---|---|---|
| ViT | 97.90 | 98.21 | 75.03 | 85.42 |
| T2T_ViT | 97.67 | 97.98 | 74.81 | 84.79 |
| Swin | 97.96 | 98.11 | 78.21 | 86.20 |

Table 4.3: HyperKvasir classification accuracy

| Model | Vanilla unmasked | Surrog unmasked | Vanilla masked | Surrog masked |
|---|---|---|---|---|
| ViT | 90.17 | 90.67 | 78.17 | 80.00 |
| T2T_ViT | 90.83 | 90.67 | 71.42 | 81.92 |
| Swin | 88.67 | 88.00 | 77.67 | 81.00 |

In the tabulated results, the average accuracy is reported, when each patch is masked independently with a probability of 50%.

The overall accuracy of the models on variously masked data is shown in figure 4.5. Masked % on the x axis represents how many patches were masked in the data. All six models begin at about 97% accuracy. Vanilla classifiers begin to gradually decrease performance at about 30% of masked patches, while the surrogate models' accuracy remain almost unchanged. It is only at about 70% of masked patches that surrogates' accuracy begins to significantly drop.

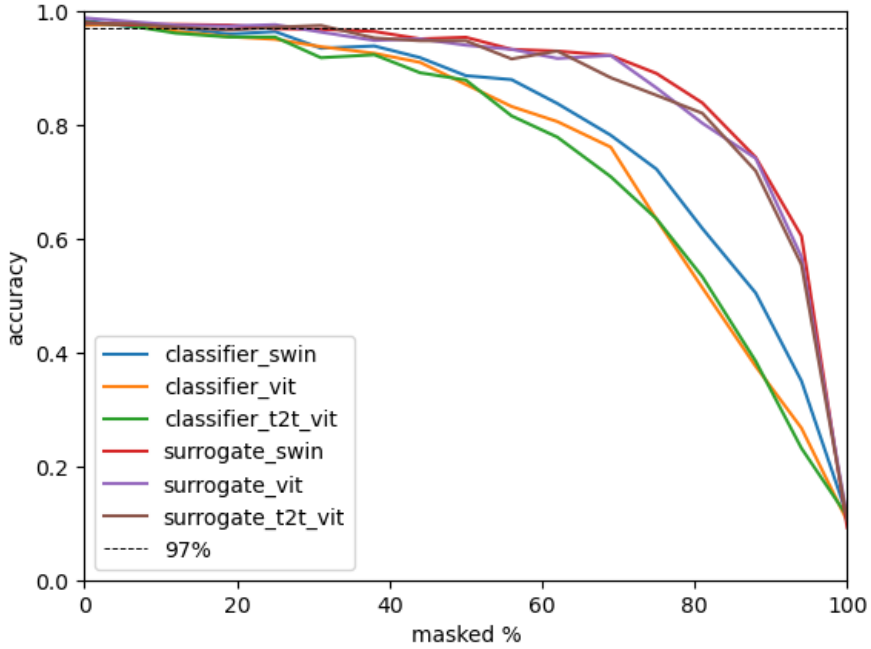



Figure 4.5: Accuracy of the models with masked patches (CIFAR-10)

For the HyperKvasir dataset, the results are much less smooth (see Figure 4.6), which is probably due to the smaller size of the dataset. It is noticeable that the vanilla classifiers are more robust and drop less significantly. We presume that it is caused by stronger augmentations used for this data.

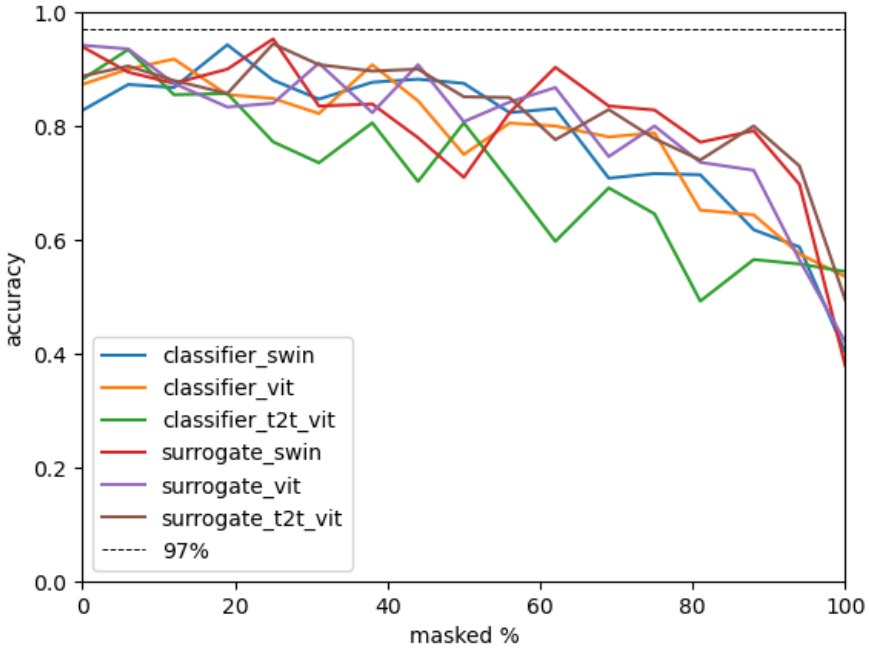



Figure 4.6: Accuracy of the models with masked patches (HyperKvasir)

## 4.3. CIFAR-10 Explanations

In this section we will describe the feature attributions obtained by Shapley values computed from scratch using the exact formula

$$\phi_v(i) = \sum_{S \subseteq X \setminus \{i\}} \frac{|S|!(|X| - |S| - 1)!}{|X|!} \left(v(S \cup \{i\}) - v(S)\right),$$

where $X$ is the set of players, $i$ is the number of players, and $v$ is the coalitional game. In our case, the coalitional game is defined by the logits of the surrogate model. Due to the strong theoretical guarantees of the Shapley values, we can refer to them as ground-truth explanations. The Shapley values computed directly from the definition indicate where the patches essential for the prediction really are. Therefore, using them, we can verify whether the model reflects commonsense presumptions about feature importance. For the CIFAR-10 dataset, we can rely on intuition, which says, for instance, that for the animals, the most important feature is their head or, for the plane, its wings. As for the HyperKvasir dataset, we have pixel-level annotations of the important features (polyps), and we can check whether the models' predictions match them (see Section 4.4 later).

All the models have a rather strong correlation with each other (see Figures 4.7, 4.8, 4.9). The strongest correlation occurs between two sparse transformers. It can also be seen in Table 4.4. We check there how many images get the biggest Shapley value for the same patch. Every pair's score is greater than 50%. It is not surprising that the consistency is not exact because, for 16 players, patches are rather small. It means that important features of the image can appear in two patches, and one model can prefer one patch while the other prefers some of its neighbors. It is mostly visible for the class "truck" (see Figure 4.10).

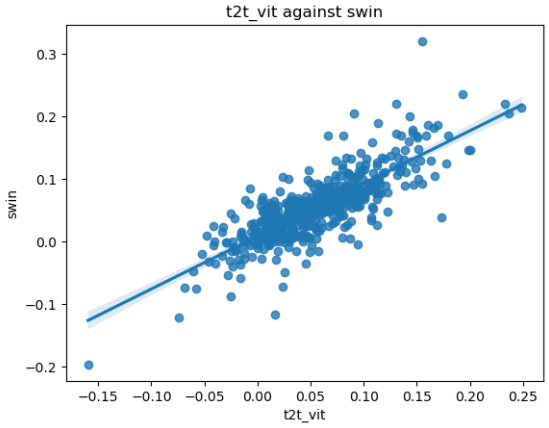



Figure 4.7: Correlation of Shapley values for Swin and T2T_ViT

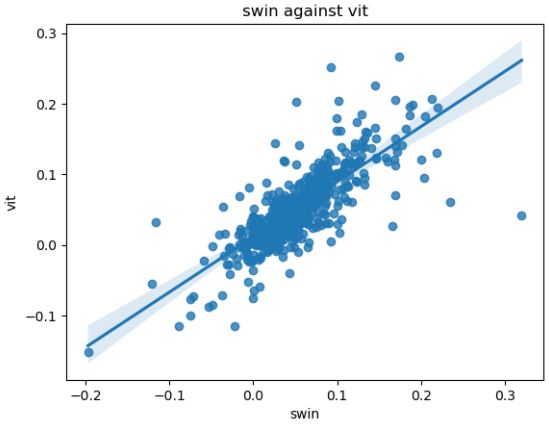



Figure 4.8: Correlation of Shapley values for ViT and Swin

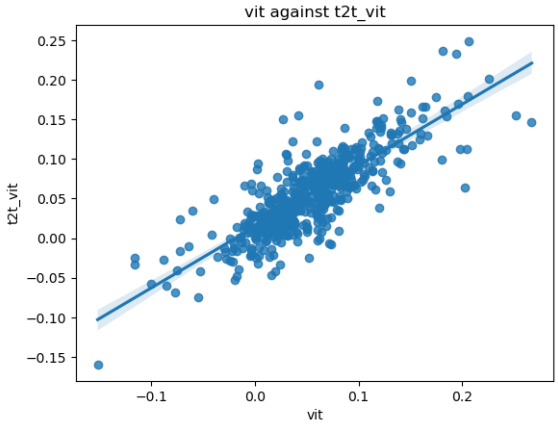
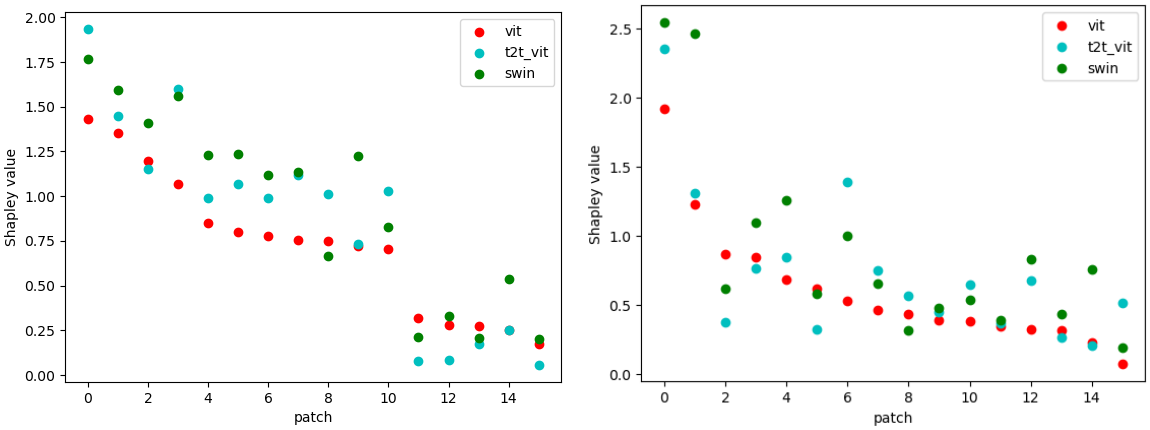



Figure 4.9: Correlation of Shapley values for ViT and T2T_ViT

Table 4.4: The percent of images with the same patch having the biggest Shapley value

| Model | ViT | T2T_ViT | Swin |
|---|---|---|---|
| ViT | 100 | 50.0 | 53.1 |
| T2T_ViT | 50.0 | 100 | 62.5 |
| Swin | 53.1 | 62.5 | 100 |




(a) truck (b) cat

Figure 4.10: Shapley values for the truck and cat class

We plotted Shapley values for one image and for all 16 patches. The truck has many features that could lead to the conclusion that there is a truck in the image: there are many indicators of "truckness." It is reflected by the fact that there are many features with a high Shapley value. On the other hand, the plots for the "cat" show one feature with a high attribution and another with relatively small attributions (see Figure 4.10). The reason for this behavior might be that cats are more definite beings than trucks. Models may learn that the most important feature of a cat is its head, while there are many other less important features like legs, trunk, and tail. Models assign to each of the features some proportionate importance. The features of the truck are much less sharp. For one model, a white patch might enhance the probability of the truck class because it learned that trucks are often white. For another model, the same white patch might work against the truck class, as it is rather an indicator of a plane.

This concentration of important features can be even better observed in Figures 4.11-4.13. For some classes, like frog or truck, the Shapley values are somehow blurred, and there are many patches with a big value (marked as yellow). On the other hand, classes like cat or bird have quite strict "positive" regions, which indicate the importance for the model output.


Figure 4.11: Shapley values on images for the ViT surrogate with 16 players


Figure 4.12: Shapley values on images for the T2T_ViT surrogate with 16 players


Figure 4.13: Shapley values on images for the Swin surrogate with 16 players

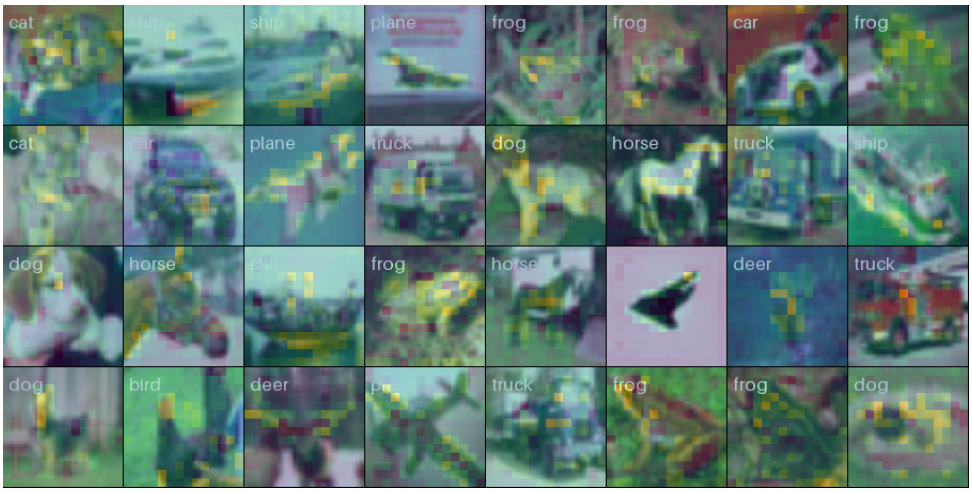



Figure 4.14: Shapley values on images for the ViT surrogate with 196 players

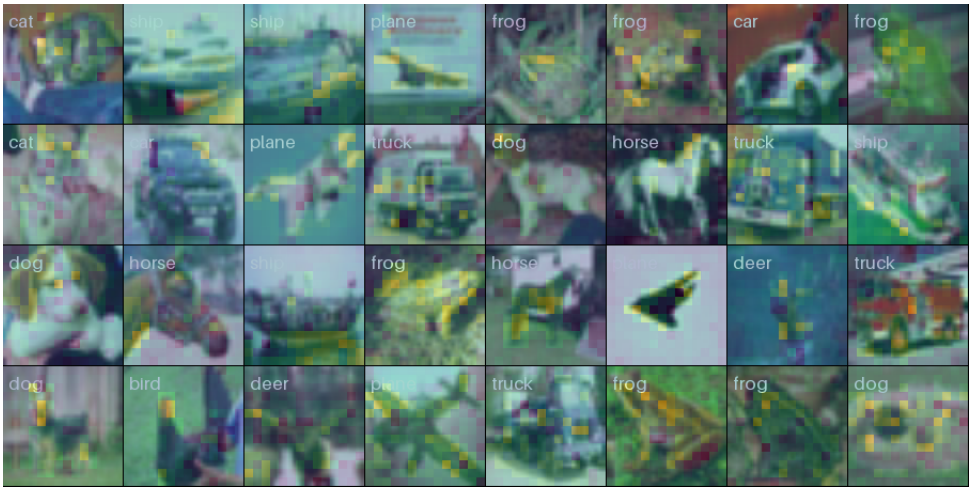



Figure 4.15: Shapley values on images for the T2T_ViT surrogate with 196 players

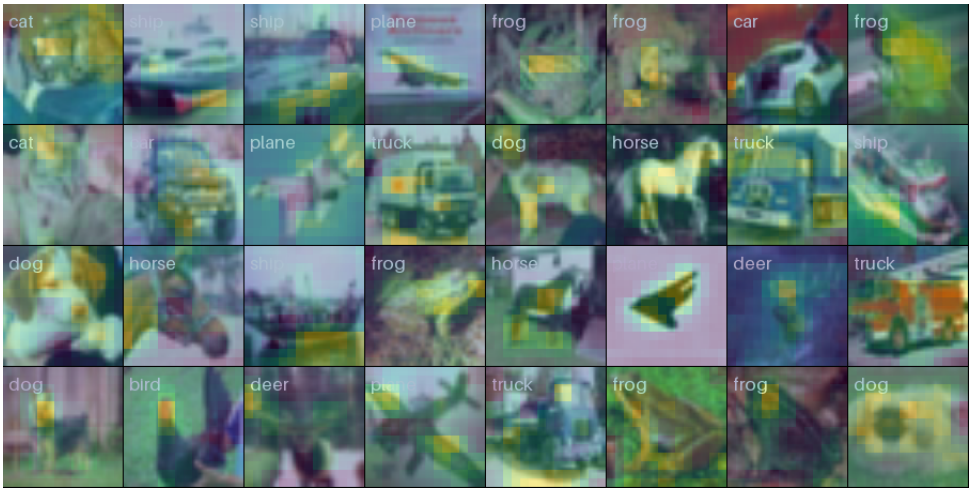



Figure 4.16: Shapley values on images for the Swin surrogate with 196 players

This behavior can be seen in more fine-grained pictures for 196 players (see Figures 4.14 - 4.16). These pictures were produced with the Shapley values computed by the explainer model (see Section 3.2), because the computation from scratch would require considering each of the $2^{196}$ coalitions. They are, for the most part, compatible with the values computed from scratch for 16 players, i.e., they indicate the same regions of images as important. In Section 4.5 we will conduct a more thorough evaluation of the explainer model.

In order to investigate the connections between the Shapley values computed by several methods, we computed their correlations and other coefficients. We plotted example correlations in Figures 4.17 - 4.18 (other pairs look very similar).

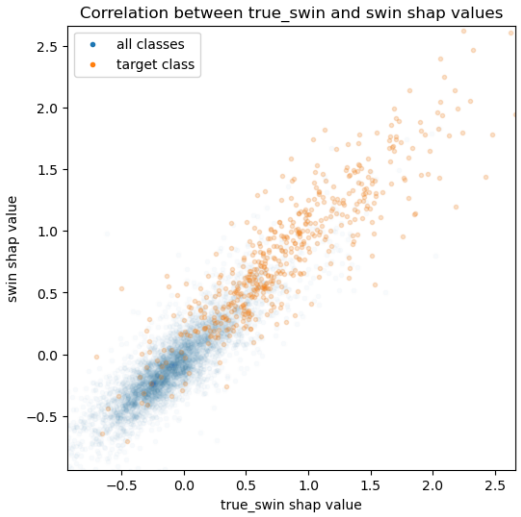



Figure 4.17: Ground-truth Shapley values vs Shapley values computed using the swin explainer, for the swin surrogate with 16 players on the CIFAR10 dataset

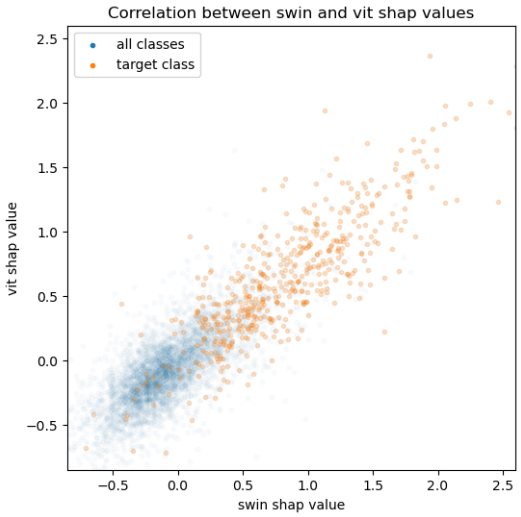



Figure 4.18: Shapley values computed by the explainer models for Swin and ViT surrogates

We can see that the Shapley values computed by the explainer model highly correlate with the ground truth. On the other hand, explainer models for other surrogates give scattered Shapley values, but they also lie around the $y = x$ line. In Table 4.5 are presented the

correlation and error metrics. The ground-truth Shapley values have a rather high correlation with those computed by the explainer models and also give smaller error rates (MAE and RMSE) than between different explainers.

Table 4.5: Agreement between ground-truth Shapley values and explainer values, as well as between different architectures; for the target class; CIFAR10, 16 players.

| Metric | Pearson's corr. | MAE | RMSE |
|---|---|---|---|
| gt-ViT vs ViT | **0.90** | **0.17** | **0.23** |
| gt-T2T_ViT vs T2T_ViT | 0.89 | 0.18 | 0.24 |
| gt-Swin vs Swin | **0.90** | 0.20 | 0.27 |
| ViT vs T2T_ViT | 0.86 | 0.21 | 0.27 |
| T2T_ViT vs Swin | 0.83 | 0.28 | 0.37 |
| Swin vs ViT | 0.85 | 0.29 | 0.37 |

## 4.4. HyperKvasir Explanations

For the HyperKvasir dataset, we have noticed an interesting issue with the models. Namely, they tended to classify images by some incidental features. Typical images with polyps and without polyps have different colors, some additional captions, and edges. The models sometimes pick up these properties. This is hard to notice even by a thorough inspection of the models' false predictions on the masked images (see Figure 4.19).

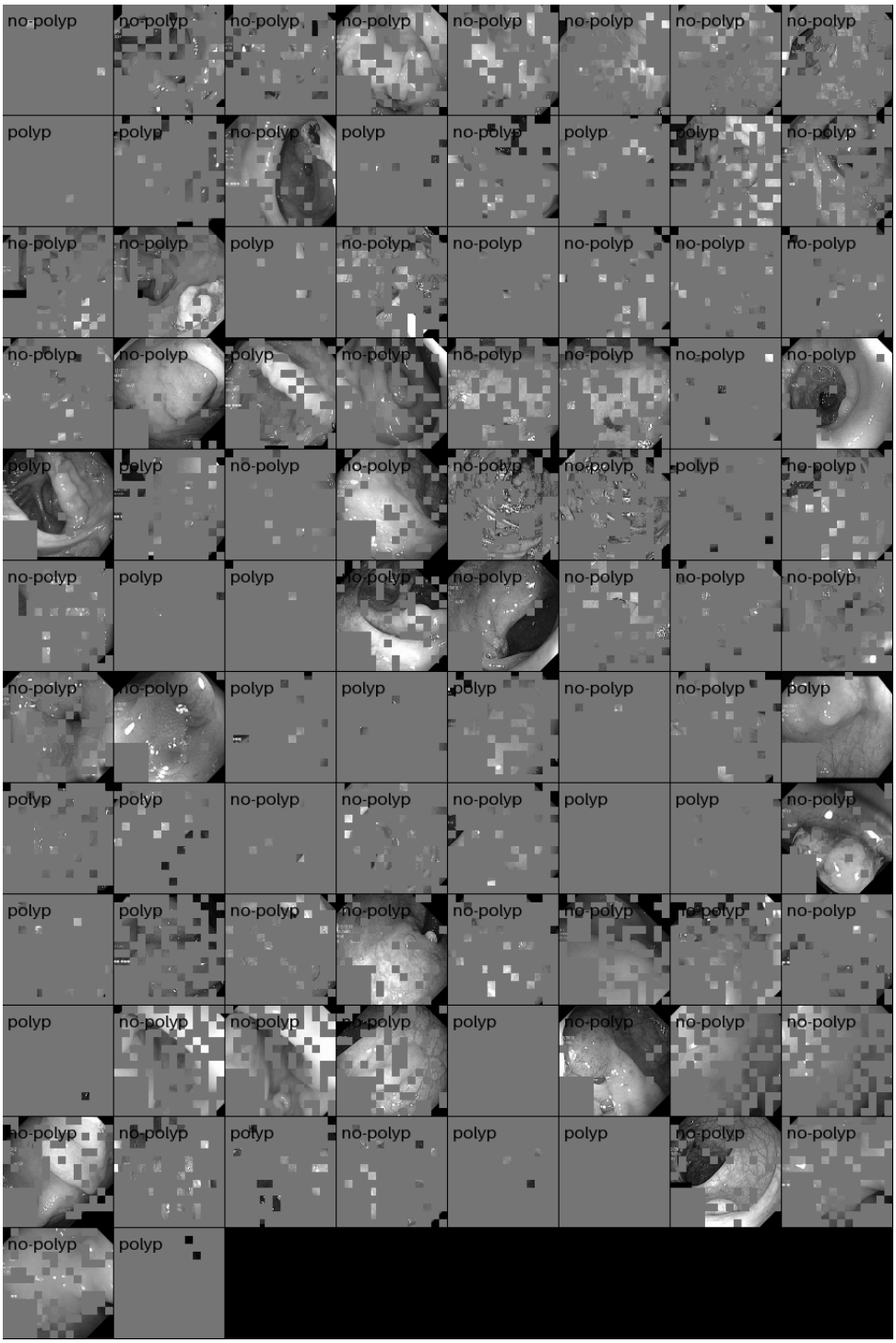



Figure 4.19: False predictions of the surrogate model

We can observe that images with fully visible polyps are misclassified when they have masked captions or edges. However, this unwanted behavior of the model can be seen much more clearly with the Shapley values, which indicate explicitly important features. (see Figure 4.20). It can be clearly seen that predictions of the surrogate model for images without polyps are often based on the edges of the image.

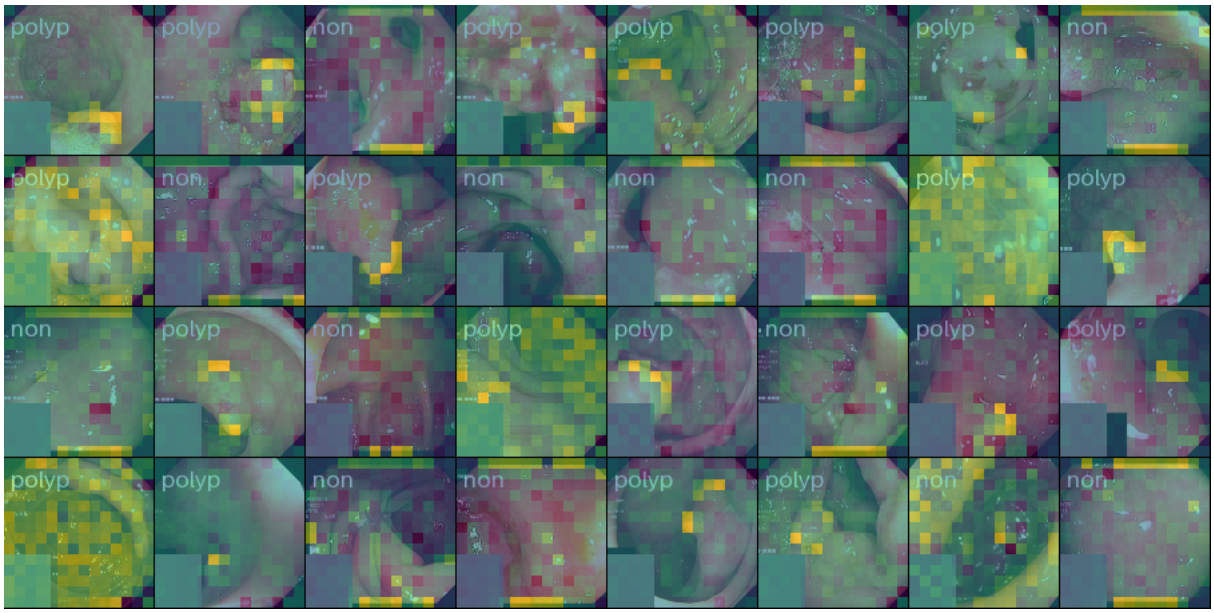



Figure 4.20: The surrogate often focuses on incidental features to classify non-polyps.

Here we can see the advantage of Shapley values, which can almost automatically show some problems with the model or with the dataset. When we noticed the above-mentioned issue with the HyperKvasir dataset, we decided to crop the images so that the unwanted additional features would disappear. Indeed, after this operation, Shapley values are the highest for the regions with polyps, and surrogate models focus on the important features, as can be seen in Figure 4.21

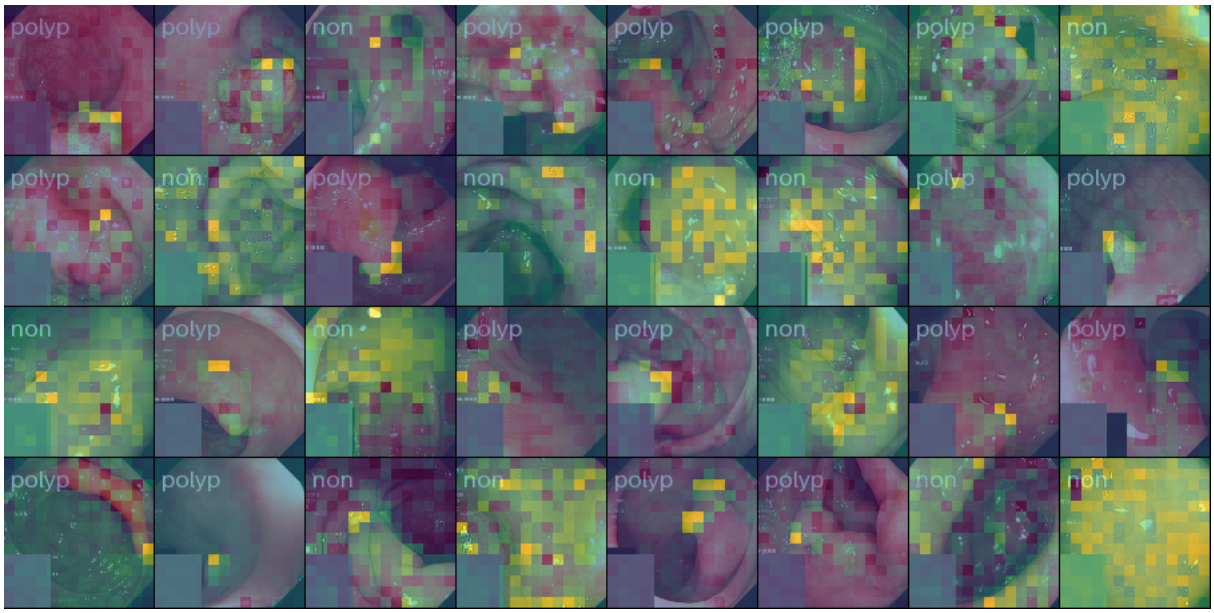



Figure 4.21: After cropping, the surrogate learns to focus on the genuine features.

Cropping of the images caused a considerable decrease in accuracy (see Table 4.6).

Table 4.6: Accuracy of different classifiers on cropped and non-cropped images.

| Model | ViT | T2T_ViT | Swin |
|---|---|---|---|
| non-cropped | 93.7 | 94.9 | 98.3 |
| cropped | 89.2 | 89.8 | 89.1 |

Just like for the CIFAR-10 dataset, we checked the relationships between the Shapley values computed in different ways. Note that Shapley values for the target class and for all classes are mixed, whereas for the CIFAR-10 dataset they were quite strictly separated (see Figures 4.22 - 4.23). In the HyperKvasir dataset, there are only two classes, and the polyp class is indicated only by a specific area on the image, so Shapley values for patches in the remaining areas might not differ from Shapley values for the non-polyp class.

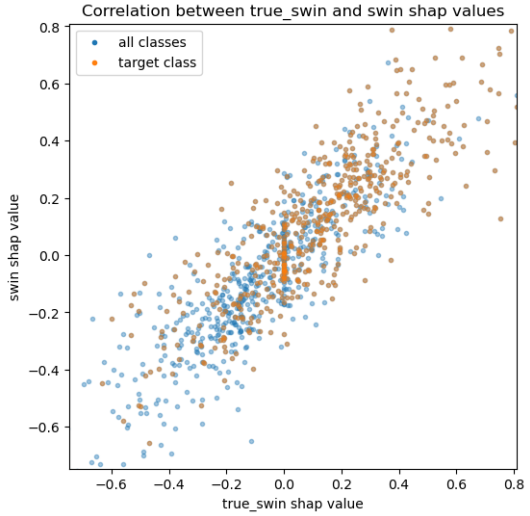



Figure 4.22: Ground-truth Shapley values vs Shapley values computed using the explainer model

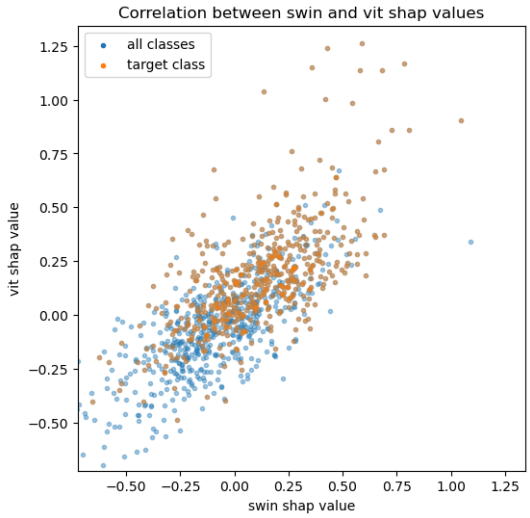



Figure 4.23: Shapley values computed by the explainer models for Swin and ViT surrogates

Table 4.7: Agreement between ground-truth Shapley values and explainer values, as well as between different architectures; for the target class; HyperKvasir, 16 players.

| Metric | Pearson's corr. | MAE | RMSE |
|---|---|---|---|
| gt-ViT vs ViT | 0.80 | 0.17 | 0.23 |
| gt-T2T_ViT vs T2T_ViT | 0.77 | 0.18 | 0.24 |
| gt-Swin vs Swin | **0.85** | **0.11** | **0.15** |
| ViT vs T2T_ViT | 0.70 | 0.17 | 0.23 |
| T2T_ViT vs Swin | 0.72 | 0.15 | 0.20 |
| Swin vs ViT | 0.66 | 0.17 | 0.24 |

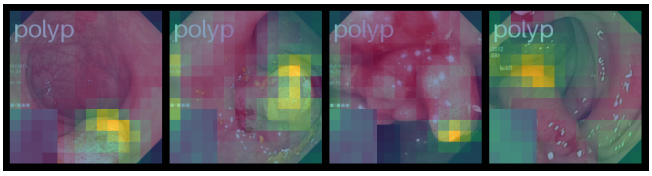
## 4.5. Evaluation of explanations

Whereas in the previous sections we were interested mainly in the model's performance and how it produced its output, here we focus rather on the explanations themselves. Before, the explanations served us as a tool for checking the models' properties; here we would like to answer whether these explanations are the *true* explanations. Namely, we would like to answer the following question: *Are the features indicated by explanations as important indeed important for the model? Are the features indicated by explanations as not important indeed not important?* These informal questions can be formalized in several ways, and many metrics were proposed to evaluate the explanation methods, like monotonicity [16], max-sensitivity [27], or ROAD [18].

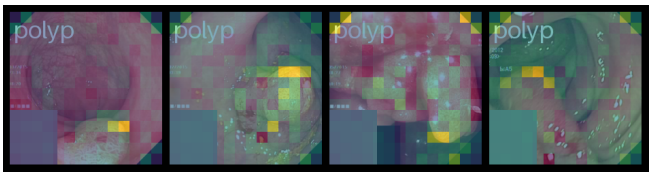
The Shapley values are well-grounded theoretically, so they can serve as a baseline for other explanation methods, but only for the case of 16 players, in which we could compute the Shapley values exactly. For 196 players, we are interested in how the explainer model performs and whether it provides a better explanation than some commonly used methods like saliency.

As it was already mentioned, the HyperKvasir dataset provides ground-truth segmentation masks that indicate where exactly the polyps are. Therefore, we can use these segmentations for the evaluation of the explanations. In Figures 4.24 - 4.25 are displayed the Shapley values computed by the explainer models together with the segmentation mask of the polyps.


(a) ViT




(b) Swin




(c) T2T

(d) Ground truth segmentation mask

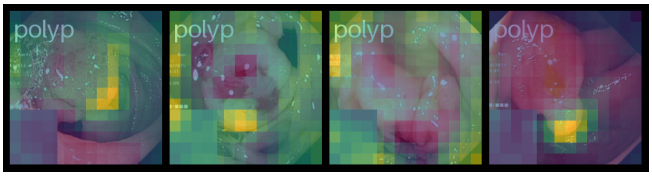
Figure 4.24: Segmentations masks together with explainer-estimated Shapley values for three models: ViT, Swin, and T2T_ViT.

We can see that the Swin model indicates the patches in the most confident manner; however, these patches do not necessarily contain the polyps. In Figure 4.25 in the second and third images for Swin, we can see that Shapley values are high for some boundary regions that do not belong to the segmentation mask.

We can build a segmentation model based on Shapley values computed by the explainer models. For each patch, we decide whether it represents a polyp using some threshold. The output of the explainer model has shape (batch size, number of players), so we have to reshape it. We do this by repeating on the whole patch the Shapley value obtained by the explainer. Then we have to choose some threshold value to decide which pixels belong to the polyp.

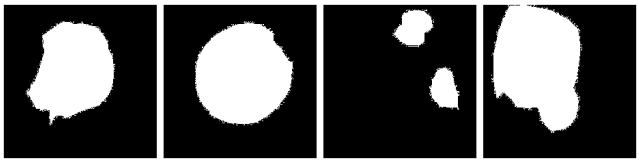
In order to find an appropriate threshold, we computed the Dice score between explainer-based segmentation and ground-truth segmentation on the training set. We choose a threshold that maximizes the Dice score on the training set. In Table 4.8, we present some metrics for this segmentation approach.


(a) ViT



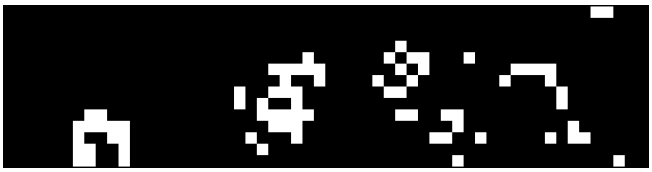

(b) Swin


(c) T2T

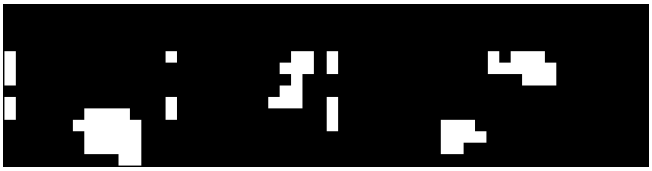


(d) Ground truth segmentation mask

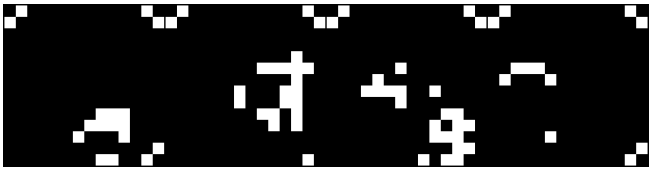
Figure 4.25: Segmentations masks together with explainer-estimated Shapley values for three models: ViT, Swin, and T2T_ViT.



(a) ViT



(b) Swin



(c) T2T

(d) Ground truth segmentation mask

Figure 4.26: Segmentations masks together with explainer-based masks for three models: ViT, Swin, and T2T_ViT.

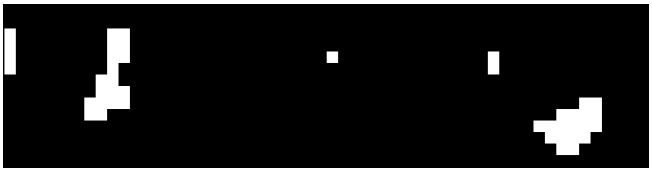
Table 4.8: Metrics for the explainer-based segmentations masks

| Model | Dice | Jaccard | Precision | Recall |
|---|---|---|---|---|
| ViT | 0.81 | 0.48 | 0.72 | 0.59 |
| T2T_ViT | 0.81 | 0.49 | 0.74 | 0.60 |
| Swin | 0.80 | 0.50 | 0.74 | 0.61 |

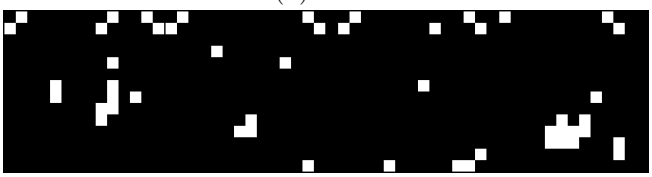
We also visualize the segmentation masks obtained using Shapley values (Figures 4.26 - 4.27). Apart from some outliers, segmentation masks indicated by Shapley values are contained in the ground-truth segmentation masks. It gives a comparatively high precision score. On the other hand, Shapley values-based segmentation masks do not succeed in locating all the polyp pixels, which gives a low recall score.

We move on now to discuss the evaluation metrics.

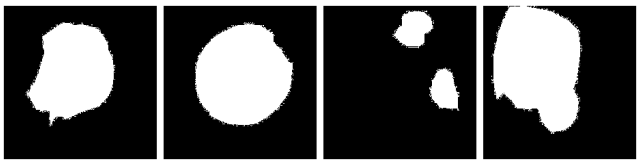
(a) ViT



(b) Swin



(c) T2T



(d) Ground truth segmentation mask

Figure 4.27: Segmentations masks together with explainer-based masks for three models: ViT, Swin, and T2T_ViT.

### 4.5.1. ROAD

The ROAD (RemOve And Debias) evaluation method was proposed in [18] as an improvement to the older method ROAR (RemOve And Retrain) [9]. The ROAR method measures how the accuracy of the model degrades when the features estimated as most important are removed. In order to avoid the out-of-distribution problem (inputs with removed features are from other distribution), ROAR proposes to retrain the model after removing important features. It is done for several percentages of important features: 10%, 20%, etc., so one should retrain the model several times. Note that the out-of-distribution problem is solved in a much less computationally expensive way via a surrogate model (see Section 3.1), which is trained only once.

The ROAD method proposes another way to reduce computations. It saves 99% of the computational costs compared to ROAR. It also measures the accuracy after removing the most important features, but instead of retraining the model, it replaces the removed features with so-called noisy linear imputations. This method is based on the observation that pixels in the image (ROAD method is designed only for Computer Vision) are highly correlated. Intuitively, green pixels of the grass are, with a high probability, near other green grass pixels. One can for example check that the correlation coefficient for direct neighbors in CIFAR-10 is $\rho = 0.89$. Therefore, we have quite strong premises to suppose that each pixel can be approximated by the weighted mean of its neighbors with constant coefficients of a weighted mean:

$$\begin{aligned} x_{i,j} = w_{\text{direct}}(x_{i,j+1} + x_{i,j-1} + x_{i+1,j} + x_{i-1,j}) + \\ w_{\text{indirect}}(x_{i+1,j+1} + x_{i-1,j+1} + x_{i+1,j-1} + x_{i-1,j-1}) \end{aligned}$$

where $x_{i,j}$ are pixels and $w_{\text{direct}}, w_{\text{indirect}}$ are constant coefficients for direct and indirect (diagonal) neighbors of a pixel, respectively. The weights $w_{\text{direct}}, w_{\text{indirect}}$ are set to $1/6, 1/12$, respectively. The resulting system of equations can be efficiently solved, even for missing pixels without neighbors, by successive interpolation. This linear interpolation of removed pixels should not be learned by a model (otherwise the model could almost exactly reproduce the input), so a small Gaussian noise is added.

In Figures 4.28 - 4.30 we present the accuracy of the models after removing the most important pixels according to the ROAD method. Note that lower accuracy indicates a better explanation. We can see that the accuracy on the batch of 32 images remains very high for the saliency method even after removing 80% of pixels. The explainer model outperforms it significantly. Removing the pixels marked as important by the explainer models causes quite an abrupt decrease in accuracy. We can also note that the accuracy for both saliency and explainer feature attributions remains high for up to 20% of removed pixels. It is probably due to the noisy linear imputations method. After removing only 20% of pixels, the images remain very similar, though somewhat blurred.

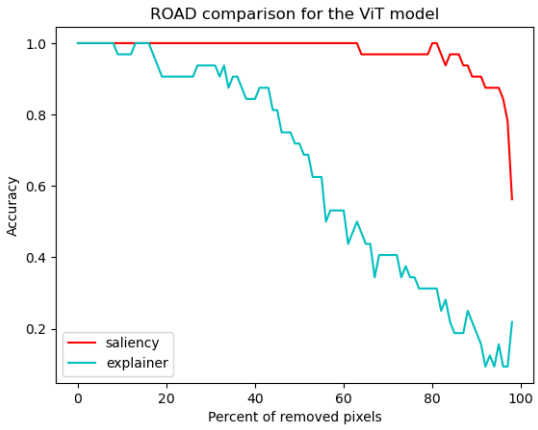



Figure 4.28: ROAD metric for Saliency and explainer on CIFAR-10 dataset (ViT model)

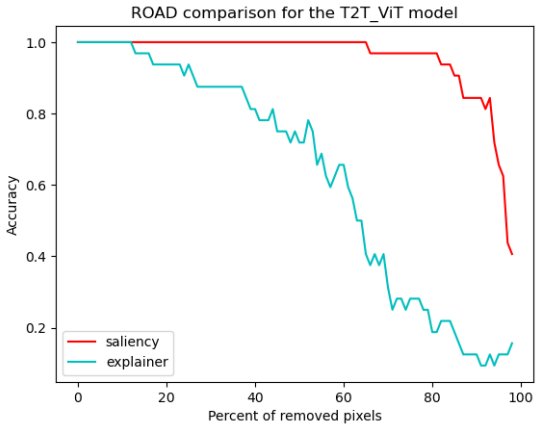



Figure 4.29: ROAD metric for Saliency and explainer on CIFAR-10 dataset (T2T_ViT model)

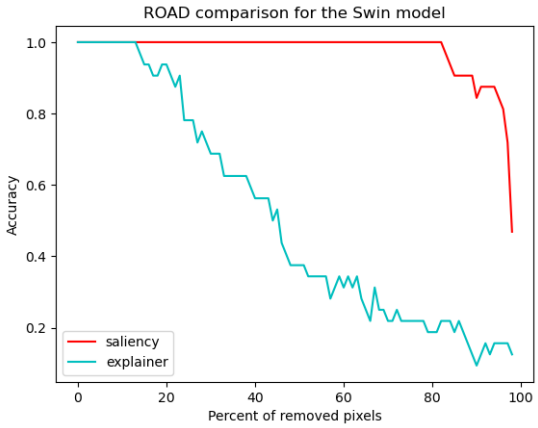



Figure 4.30: ROAD metric for Saliency and explainer on CIFAR-10 dataset (Swin model)

The advantage of the Shapley values attributions over the Saliency method is even more visible when we compare the ROAD metrics computed for the HyperKvasir dataset (see Figures 4.31 - 4.33).

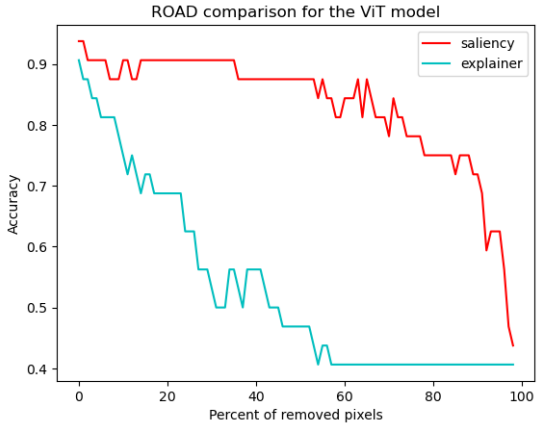



Figure 4.31: ROAD metric for Saliency and explainer on HyperKvasir dataset (ViT model)

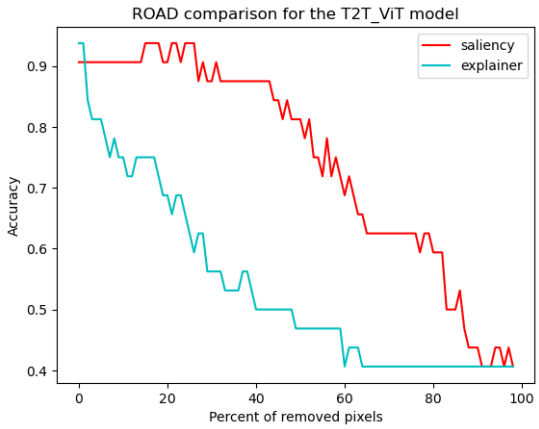



Figure 4.32: ROAD metric for Saliency and explainer on HyperKvasir dataset (T2T_ViT model)

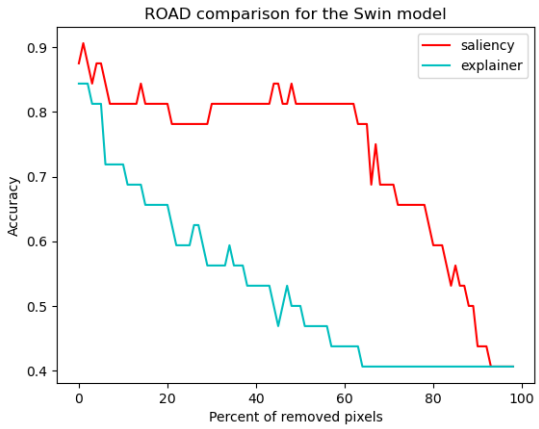



Figure 4.33: ROAD metric for Saliency and explainer on HyperKvasir dataset (Swin model)

### 4.5.2. Custom evaluation metric

As it was already mentioned, the surrogate model solves the out-of-distribution problem connected with zeroing-out the pixels on the image. Therefore, to evaluate the explanations of the surrogate model, we can simply mask the pixels of the image. We have done it for several removal strategies. Here we present the results for the CIFAR-10 dataset; later in this section we will provide the results for the HyperKvasir dataset.

We used five removal strategies: removing the best (most important) and worst (least important) patches according to Shapley values or saliency attributions, and also three independent methods: removing central, peripheral, and random patches. We can see that central features tend to be important for the model. However, the Shapley values computed by the explainer model detect even more important features, and removing them causes a radical drop in accuracy. It is also worth noting that removing the worst features even increases the accuracy of the model. The explainer model detects patches that have a negative impact for the target class, and removing them increases the surrogate confidence for the target.

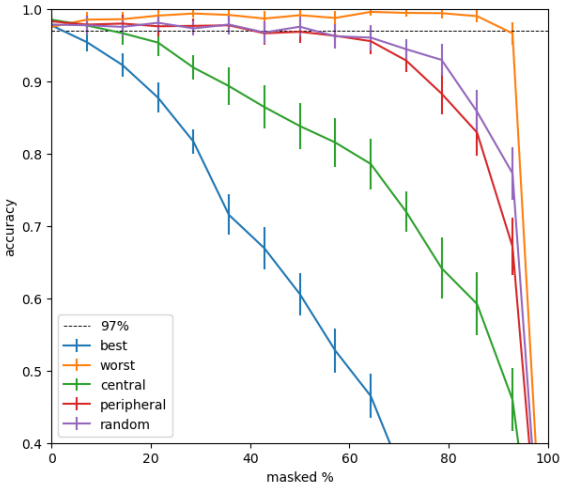
In Figure 4.34 we present the accuracies of the ViT, T2T_ViT, and Swin-based surrogate models.

We can see that the T2T_ViT model focuses the most on the local features. It strongly concentrates on the most important features, and thus it can perform well even if all patches apart from the most important are removed. On the other hand, the accuracy of T2T_ViT drops the most rapidly after removing the most important features. The model that is most robust for removing of the most important features is Swin. It is probably due to its shifting strategy: the patches close to each other share the information, and the removal of one feature is less harmful for the model's performance.

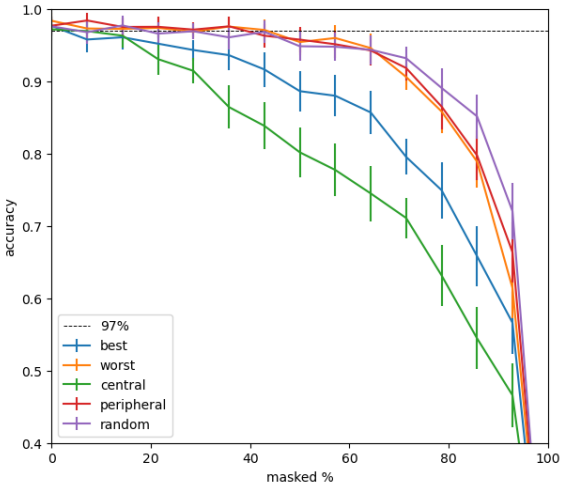
We can see that for the ViT model, the saliency attributions do not indicate the most important features with such precision as Shapley values. It confirms the results obtained for the ROAD metric. The accuracy after removing the least important features according to this metric is not only on par with the peripheral removing approach, but it is also worse than random removing. As for the most important patches according to this metric, it drops quite slowly, and the model obtains better accuracy after removing the best features than after removing simply central patches. It means that the saliency attributions do not properly recognize the importance of the patches.

The results for the HyperKvasir dataset show even better the advantages of the Shapley values computed by the explainer model over the Saliency method (see Figure 4.35. We can see that removing the patches recognized as important for the prediction by the saliency method causes a similar drop in accuracy to the random removing approach. As a matter of fact, all the curves for the saliency method look similar, and methods of removing the patches change the accuracy in a similar way.

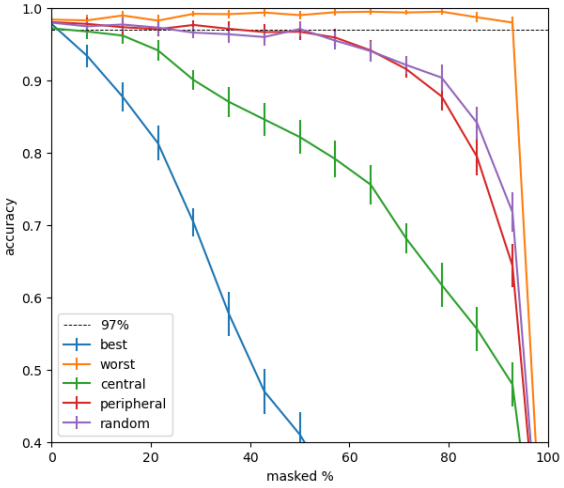
On the other hand, the explainer recognizes the important features very precisely, so that the accuracy after removing the most important features drops fastly for all three models (ViT, T2T_ViT, and Swin). Worst, peripheral, and central removing approaches of removing give the curves that lie between the curves for the worst and best patches removing approaches, as we expect from a proper feature attribution method.



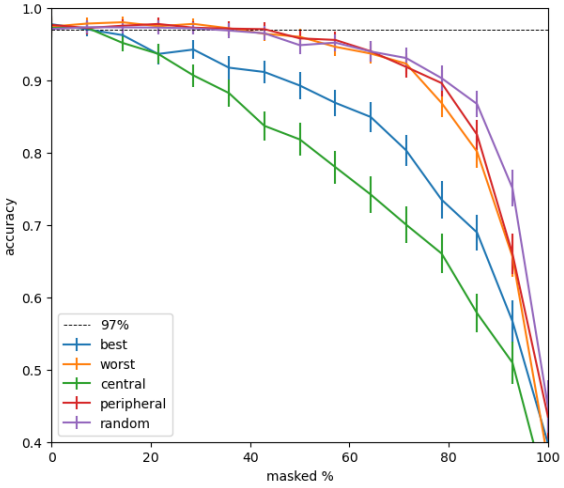

(a) Explainer (ViT)



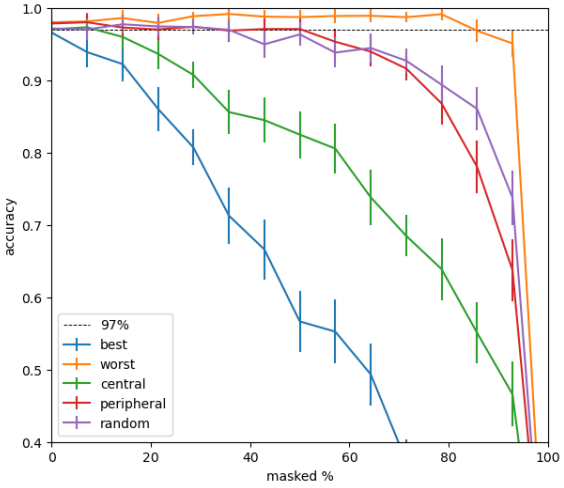

(b) Saliency (ViT)



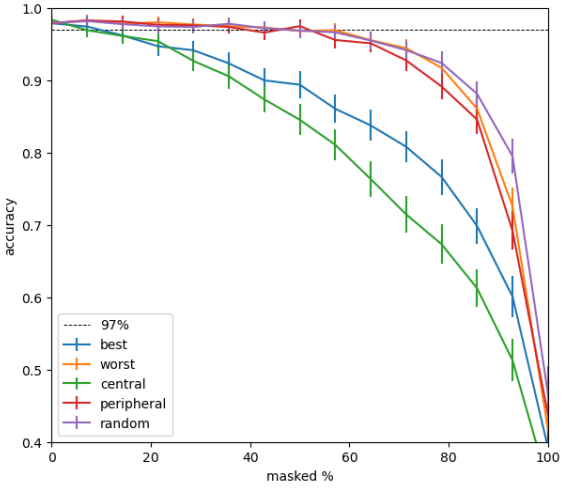

(c) Explainer (T2T)




(d) Saliency (T2T)

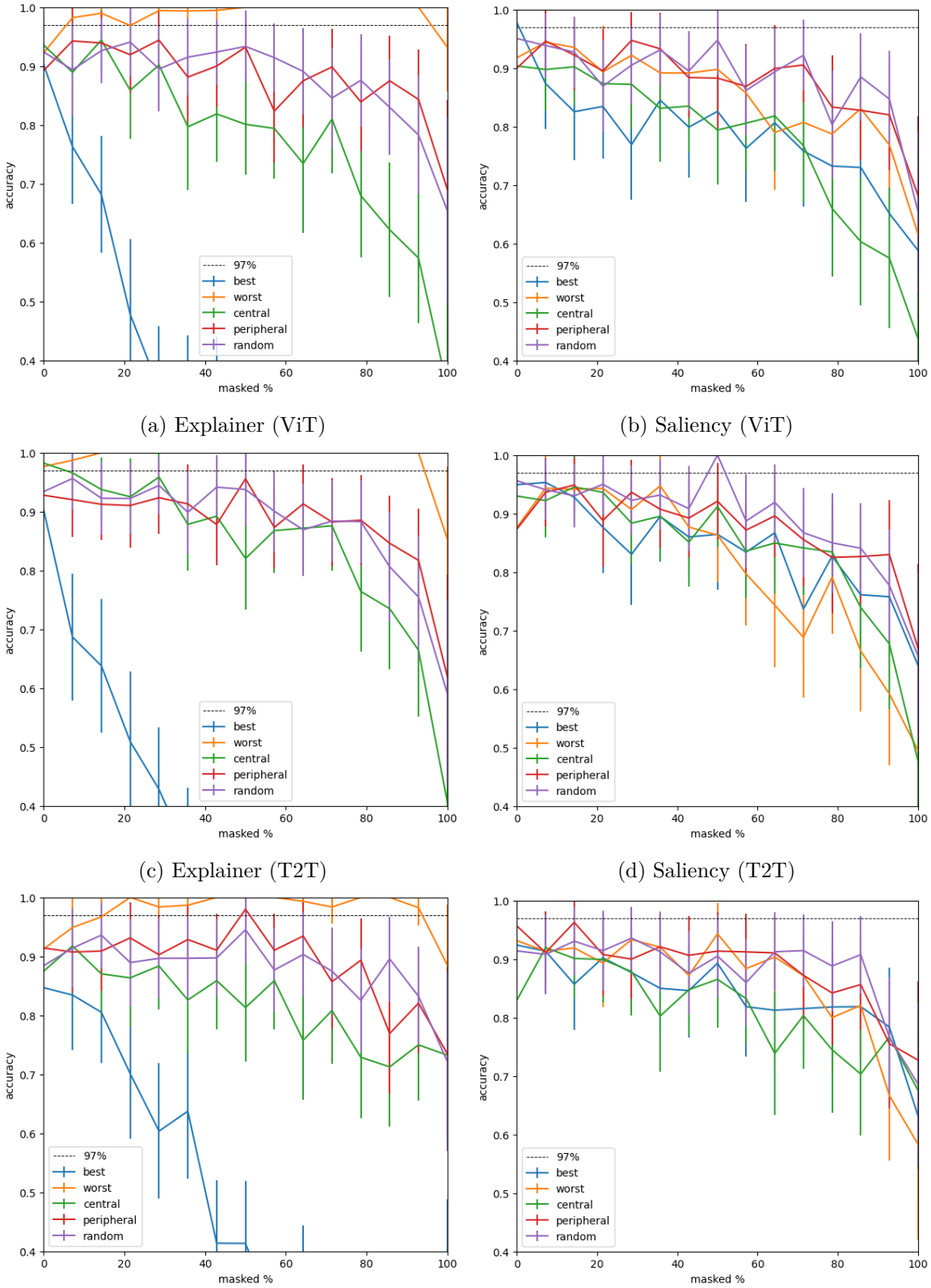



(e) Explainer (Swin)




(f) Saliency (Swin)

Figure 4.34: Accuracy of surrogate model after removing features computed by explainer or saliency method on the CIFAR10 dataset.




(a) Explainer (ViT)

(b) Saliency (ViT)

(c) Explainer (T2T)

(d) Saliency (T2T)

(e) Explainer (Swin)

(f) Saliency (Swin)

Figure 4.35: Accuracy of surrogate model after removing features computed by explainer or saliency method on the HyperKvasir dataset.

## 4.6. Explainer pairs

We performed yet another experiment that highlighted the differences between the architectures of the models. In all the other experiments, we trained the explainer model with the same backbone as the surrogate model. In contrast, here we investigate how the explanations of each one model change if we compute them with different explainers.

### 4.6.1. CIFAR-10

For the CIFAR-10 dataset, it turned out that ViT and T2T_ViT-based explainer models give similar results, while the Swin model misfits the other two.

In Figure 4.37, we computed the accuracy of the surrogate model for different removing patches approaches. The plots are the same as in Figure 4.34 but here we want to see how several explainer models differ. As in Figure 4.36, we can see that the Swin-based explainers give the worst explanations. Removing the best patches according to the Swin explainer Shapley values comparatively fast gives accuracy below 97%. Removing the worst patches gives accuracy below 40% only after about 70% removed patches, while for other explainers it is at about 60%.

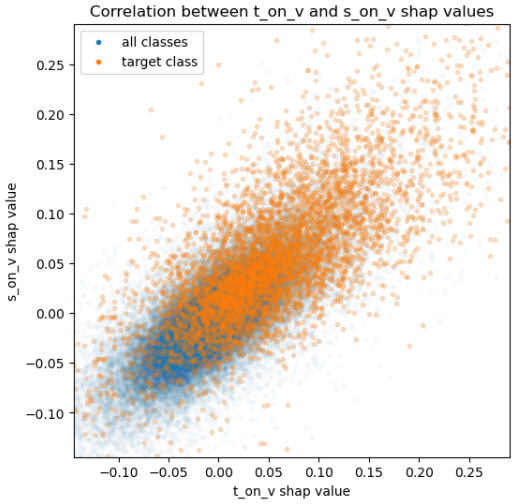
We also present the overall results in terms of accuracy after removing 50% of the best and worst patches according to Shapley values computed by several explainers for a given surrogate model (Tables 4.9, 4.10).

In Table 4.9 we can see that the Swin surrogate has the biggest accuracy after removing patches, with the best scores for all the explainers. It could mean that the Swin model is the most difficult to explain by computing Shapley values. However, we presume that such results are caused by the Swin architecture, which is more robust to masking due to its shifting mechanism. It is also worthy to note that the Swin-based explainer models manage the worst with indicating the most important patches. It obtains the highest accuracy for each surrogate model. Similarily, in Table 4.10 we can see that the Swin tackles worst with finding the worst patches; removing the patches that are worst according to the Swin explainer gives the smallest accuracy. It means that Swin explainer is unable to find the worst patches effectively.

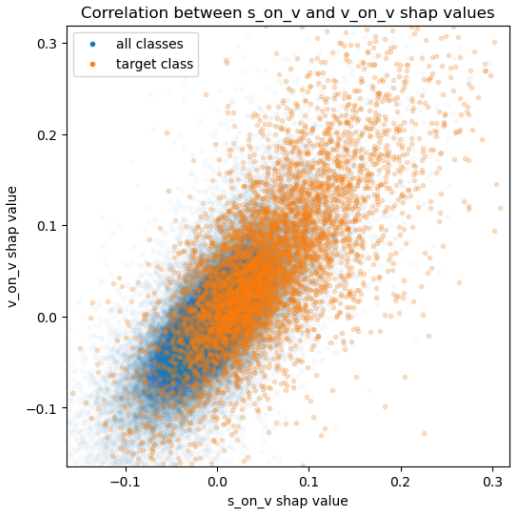
We present only the results for 196 players; the results for 16 players were insignificant; the differences between the accuracy scores were negligible.

Table 4.9: Accuracy of surrogates after removing the **best** patches according to different explainers; CIFAR-10, 196 players.

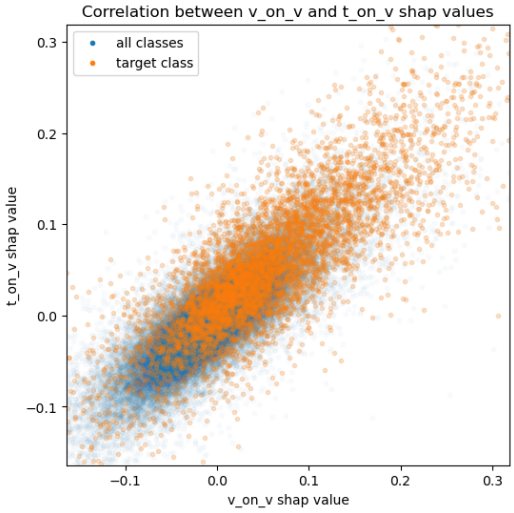
| Expl. \ Surr. | ViT | T2T_ViT | Swin | mean |
|---|---|---|---|---|
| ViT | 50.7 | 50.6 | 56.3 | 52.6 |
| T2T_ViT | 54.3 | 48.7 | 57.3 | 53.4 |
| Swin | 59.2 | 57.2 | 58.6 | 58.3 |
| mean | 54.7 | 52.2 | 57.4 | 54.8 |




(a) Swin with T2T_ViT

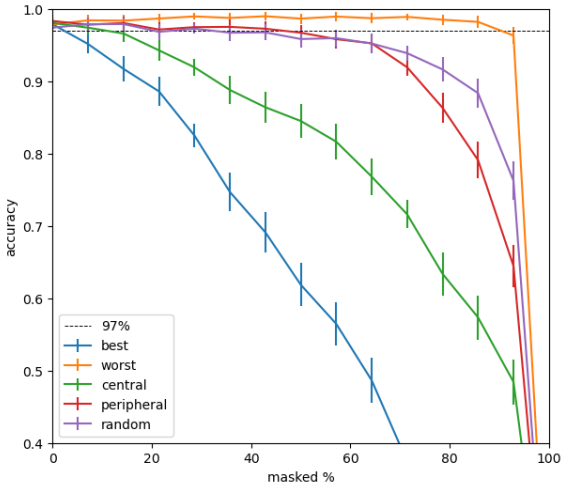



(b) Swin with ViT

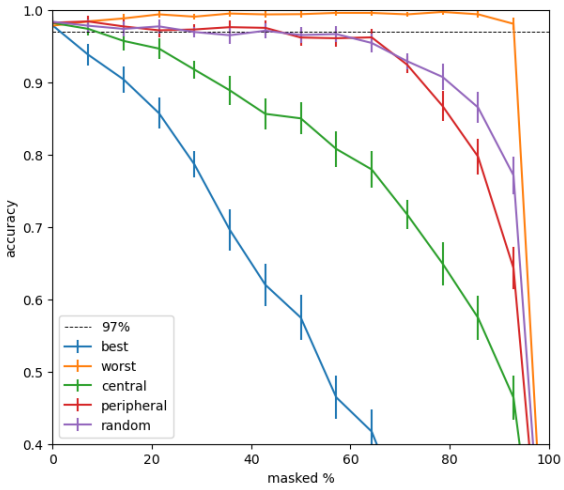



(c) ViT with T2T_ViT

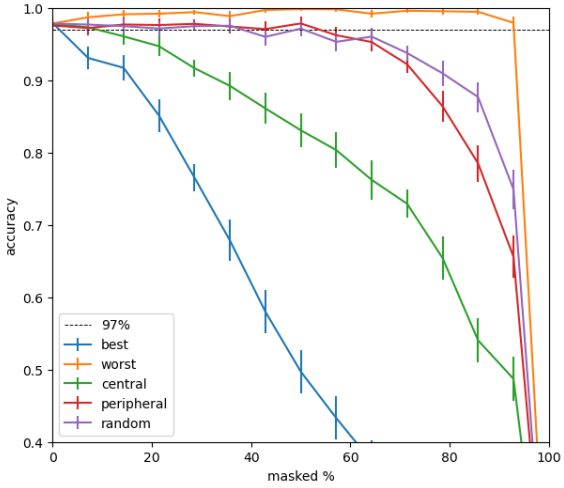
Figure 4.36: Correlation between Shapley values computed with different explainers for the ViT surrogate model




(a) Swin based masking




(b) ViT based masking




(c) T2T_ViT based masking

Figure 4.37: Accuracy of the ViT surrogate model after removing worst/best patches based on different explainers

Table 4.10: Accuracy of surrogates after removing the **worst** patches according to different explainers; CIFAR-10, 196 players.

| Expl. \ Surr. | ViT | T2T_ViT | Swin | mean |
|---|---|---|---|---|
| ViT | 98.4 | 97.9 | 98.0 | 98.1 |
| T2T_ViT | 98.0 | 98.0 | 98.1 | 98.1 |
| Swin | 97.7 | 97.1 | 97.5 | 97.5 |
| mean | 98.0 | 97.7 | 97.9 | 97.9 |

### 4.6.2. HyperKvasir

In this section, we present tables similar to the tables 4.9 - 4.10 to check how different explainers work on different surrogates for the HyperKvasir dataset. As was the case for the CIFAR-10 dataset, we might observe that the Swin explainer model significantly underperforms the other explainers. Swin explainers give the highest accuracy scores in Table 4.11 and the lowest accuracy scores in Table 4.12. We can also note that the accuracy scores in Table 4.11 are significantly lower than for the CIFAR-10 dataset. It is perhaps due to the fact that it is easier to indicate the important features of the polyp than of the truck or car, and they are less diffused. Removing these features causes the surrogate to not succeed in polyp detection.

Table 4.11: Accuracy of surrogates after removing the **best** patches according to different explainers; HyperKvasir, 196 players.

| Expl. \ Surr. | ViT | T2T_ViT | Swin | mean |
|---|---|---|---|---|
| ViT | 31.1 | 40.5 | 46.5 | 39.4 |
| T2T_ViT | 43.7 | 34.5 | 47.6 | 41.9 |
| Swin | 52.0 | 52.8 | 47.9 | 50.9 |
| mean | 42.3 | 42.6 | 47.3 | 44.1 |

Table 4.12: Accuracy of surrogates after removing the **worst** patches according to different explainers; HyperKvasir, 196 players.

| Expl. \ Surr. | ViT | T2T_ViT | Swin | mean |
|---|---|---|---|---|
| ViT | 99.0 | 99.0 | 98.4 | 98.8 |
| T2T_ViT | 98.6 | 99.3 | 98.9 | 98.9 |
| Swin | 96.8 | 97.1 | 97.5 | 97.1 |
| mean | 98.1 | 98.5 | 98.3 | 98.3 |

# Conclusions

In this master's thesis, we performed research related to both Computer Vision and Explainable Machine Learning branches of Machine Learning. On the one hand, we were interested in the models' architectures and tried to indicate the differences between them. On the other hand, we analyzed the models' explanations. Since we used the explainer model, a neural network-based explanation method, it was of interest to measure its performance. To that end, we provided a custom method of evaluating explanations.

We took into consideration two sparse transformers: Swin and T2T_ViT, and a standard Vision Transformer. We verified that the Swin model is the least interpretable. The special patterns designed in Swin to reduce the qudratic complexity of the self-attention mechanism intermingled in some way the pixels on the image. The output is produced by a mix of information from the pixels, and at the end, it was hard to find out which parts of the image influenced the prediction the most.

As for the explanations, we checked that the explainer model correctly computes the Shapley values (when the ground truth is affordable to compute) and that it gives satisfactory results in terms of the evaluation of explanation methods. Additionally, we noticed that the explainer yields, as a byproduct, a segmentation of the image. We may construct a segmentation mask for a given class based on the outputs of the explainer and a chosen threshold. This method of constructing segmentation masks seems promising as it does not require any ground truth. Another promising method is the surrogate model used to omit the out-of-distribution problem, which inevitably occurs during Shapley values computation. Future work may take advantage of this method together with the explainer model in other domains, like Natural Language Processing.

# Bibliography


[1] Iz Beltagy, Matthew E. Peters, and Arman Cohan. "Longformer: The Long-Document Transformer". In: *CoRR* abs/2004.05150 (2020). arXiv: 2004.05150. URL: https://arxiv.org/abs/2004.05150.

[2] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[3] A. Charnes et al. "Extremal Principle Solutions of Games in Characteristic Function Form: Core, Chebychev and Shapley Value Generalizations". In: *Advanced Studies in Theoretical and Applied Econometrics ((ASTA,volume 11))* (1951). arXiv: 2004.05150. URL: https://link.springer.com/chapter/10.1007/978-94-009-3677-5_7.

[4] Hila Chefer, Shir Gur, and Lior Wolf. "Transformer Interpretability Beyond Attention Visualization". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 782–791. DOI: 10.1109/CVPR46437.2021.00084. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Chefer_Transformer_Interpretability_Beyond_Attention_Visualization_CVPR_2021_paper.html.

[5] Krzysztof Marcin Choromanski et al. "Rethinking Attention with Performers". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=Ua6zuk0WRH.

[6] Ian Connick Covert, Chanwoo Kim, and Su-In Lee. "Learning to Estimate Shapley Values with Vision Transformers". In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/pdf?id=5ktFNz_pJLK.

[7] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=YicbFdNTTy.

[8] Christopher Frye et al. "Shapley explainability on the data manifold". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=OPyWRrcjVQw.

[9] Sara Hooker et al. “A Benchmark for Interpretability Methods in Deep Neural Networks”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 9734–9745. URL: https://proceedings.neurips.cc/paper/2019/hash/fe4b8556000d0f0cae99daa5c5c5a410-Abstract.html.

[10] Saidul Islam et al. “A Comprehensive Survey on Applications of Transformers for Deep Learning Tasks”. In: *CoRR* abs/2306.07303 (2023). DOI: 10.48550/ARXIV.2306.07303. arXiv: 2306.07303. URL: https://doi.org/10.48550/arXiv.2306.07303.

[11] Neil Jethani et al. “FastSHAP: Real-Time Shapley Value Estimation”. In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=Zq2G_VTV53T.

[12] Tianyang Lin et al. “A Survey of Transformers”. In: *CoRR* abs/2106.04554 (2021). arXiv: 2106.04554. URL: https://arxiv.org/abs/2106.04554.

[13] Tsung-Yi Lin et al. “Feature Pyramid Networks for Object Detection”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 936–944. DOI: 10.1109/CVPR.2017.106. URL: https://doi.org/10.1109/CVPR.2017.106.

[14] Ze Liu et al. “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”. In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 9992–10002. DOI: 10.1109/ICCV48922.2021.00986. URL: https://doi.org/10.1109/ICCV48922.2021.00986.

[15] Muzammal Naseer et al. “Intriguing Properties of Vision Transformers”. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by Marc'Aurelio Ranzato et al. 2021, pp. 23296–23308. URL: https://proceedings.neurips.cc/paper/2021/hash/c404a5adbf90e09631678b13b05d9d7a-Abstract.html.

[16] An-phi Nguyen and María Rodríguez Martínez. “On quantitative aspects of model interpretability”. In: *CoRR* abs/2007.07584 (2020). arXiv: 2007.07584. URL: https://arxiv.org/abs/2007.07584.

[17] Niki Parmar et al. “Image Transformer”. In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 4052–4061. URL: http://proceedings.mlr.press/v80/parmar18a.html.

[18] Yao Rong et al. “A Consistent and Efficient Evaluation Strategy for Attribution Methods”. In: *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 18770–18795. URL: https://proceedings.mlr.press/v162/rong22a.html.

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*. Ed. by Nassir Navab et al. Vol. 9351. Lecture Notes in Computer Science. Springer, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28. URL: https://doi.org/10.1007/978-3-319-24574-4_28.

[20] Benedek Rozemberczki et al. “The Shapley Value in Machine Learning”. In: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*. Ed. by Luc De Raedt. ijcai.org, 2022, pp. 5572–5579. DOI: 10.24963/IJCAI.2022/778. URL: https://doi.org/10.24963/ijcai.2022/778.

[21] Sofia Serrano and Noah A. Smith. “Is Attention Interpretable?” In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by Anna Korhonen, David R. Traum, and Lluís Màrquez. Association for Computational Linguistics, 2019, pp. 2931–2951. DOI: 10.18653/V1/P19-1282. URL: https://doi.org/10.18653/v1/p19-1282.

[22] Lloyd Shapley. “Notes on the n-Person Game – II: The Value of an n-Person Game”. In: *Santa Monica, Calif.: RAND Corporation* (1951). arXiv: 2004.05150. URL: https://www.rand.org/content/dam/rand/pubs/research_memoranda/2008/RM670.pdf.

[23] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks”. In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. Ed. by Zoubin Ghahramani et al. 2014, pp. 3104–3112. URL: https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html.

[24] Muhammad Faaiz Taufiq, Patrick Blöbaum, and Lenon Minorics. “Manifold Restricted Interventional Shapley Values”. In: *International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain*. Ed. by Francisco J. R. Ruiz, Jennifer G. Dy, and Jan-Willem van de Meent. Vol. 206. Proceedings of Machine Learning Research. PMLR, 2023, pp. 5079–5106. URL: https://proceedings.mlr.press/v206/taufiq23a.html.

[25] Yi Tay et al. “Efficient Transformers: A Survey”. In: *ACM Comput. Surv.* 55.6 (2023), 109:1–109:28. DOI: 10.1145/3530811. URL: https://doi.org/10.1145/3530811.

[26] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[27] Chih-Kuan Yeh et al. “On the (In)fidelity and Sensitivity of Explanations”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 10965–10976. URL: https://proceedings.neurips.cc/paper/2019/hash/a7471fdc77b3435276507cc8f2dc2569-Abstract.html.

[28] Li Yuan et al. “Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet”. In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 538–547. DOI: 10.1109/ICCV48922.2021.00060. URL: https://doi.org/10.1109/ICCV48922.2021.00060.