Warsaw University of Technology

FACULTY OF
MATHEMATICS AND INFORMATION SCIENCE

Master's diploma thesis

in the field of study Mathematics
and specialisation Mathematical Statistics and Data Analysis

Evaluation of neural networks explanations for localization and segmentation tasks, with human expert annotations

Maciej Chrabąszcz

student record book number 293343

thesis supervisor
Prof. dr hab. inż. Przemysław Biecek

WARSAW 2023

Abstract

Evaluation of neural networks explanations for localization and segmentation tasks, with human expert annotations

Deep learning techniques allow the construction of models with high performance, especially in the area of image data analysis. As a result, they are often adopted for high-stakes decisions in medical problems. An open, very important problem is an adequate explanatory model analysis to better understand the models' limitations and ensure the explainability of the models' predictions.

The aim of this work is to analyze predictive models for image data for the task of localization and segmentation. The work will use a collection of CheXpert X-ray images, and CheXlocalize containing, in addition to the images, localization, and segmentation information of significant lesions, including lesions of the disease. The performance of the models will be compared with masks developed by experts in order to evaluate both the models and the explanation methods of the models.

Keywords: deep learning, artificial intelligence, explainable model analysis, explainable artificial intelligence

Streszczenie

Analiza wyjaśnień dla sieci neuronowych w zadaniach lokalizacji i segmentacji z wykorzystaniem etykiet opracowanych przez ekspertów dziedzinowych

Techniki uczenia głębokiego (ang. deep learning) pozwalają na budowę modeli o wysokiej skuteczności, szczególnie w obszarze analizy danych obrazowych. Przez to są często adoptowane do ważnych problemów (ang. high stake decisions) w tym problemów medycznych. Otwartym, bardzo ważnym problemem, jest odpowiednia wyjaśnialna analizy modeli (ang. explanatory model analysis) pozwalająca na lepsze zrozumienie ograniczeń w działaniu modeli oraz na zapewnieniu wyjaśnialności predykcji modeli.

Celem tej pracy analiza modeli predykcyjnych dla danych obrazowych dla zadania lokalizacji i segmentacji. W pracach zostanie wykorzystany zbiór zdjęć rentgenowskich CheXpert, oraz CheXlocalize zawierający obok zdjęć również informacje o lokalizacji i segmentacji istotnych zmian, w tym zmian chorobowych. Działanie modeli zostanie zestawione z maskami opracowanymi przez ekspertów w celu ewaluacji tak modeli jak i metod wyjaśnień tych modeli.

Słowa kluczowe: uczenie głębokie, sztuczna inteligencja, wyjaśnialna analiza modelu, wyjaśnialna sztuczna inteligencja

Contents

Introduction

Artificial Intelligence (AI), the ability of machines to imitate human intelligence, has emerged as a groundbreaking area of research and development over the past few decades. Deep learning, a branch of AI, uses neural networks to model and comprehend complex patterns in data, making it possible to perform tasks that once demanded human cognition. The revolutionary force of deep learning has left its mark in many areas, from natural language processing to autonomous driving, pattern recognition, and decision-making processes.

Computer vision holds a distinguished position in this vast landscape of AI applications. This discipline aims at endowing machines with the capability to understand, interpret, and categorize visual data, essentially imitating the human visual system. By learning hierarchical representations through deep neural networks, computer vision algorithms can detect objects, recognize faces, understand scenes, and even make diagnostic recommendations based on medical images.

However, we are confronted with the profound question of understanding these AI systems' decision-making mechanisms. This has led to the emergence of another crucial field known as Explainable AI (XAI), dedicated to making AI's decision-making transparent and understandable to humans. As deep learning models have grown in complexity, they have often been criticized for their lack of interpretability, being described as 'black boxes.' XAI addresses this issue by developing tools and methodologies that try to explain the inner workings of these models.

Despite the advances in deep learning and the rising attention towards XAI, several aspects remain under-explored. One such aspect that is of particular interest to this thesis is the evaluation of these models in the context of medical imaging, a domain where accurate interpretation can be a matter of life and death. Furthermore, another question that arises is regarding the robustness of these models, particularly in their robustness to adversarial attacks.

I have formulated two central hypotheses that this thesis aims to validate. My first hypothesis is that transformer-based models outperform Convolutional Neural Networks on the CheXpert dataset [Irv+19], a large collection of chest X-rays. Concurrently, I also hypothesize that these transformer-based [Kol+21; Liu+21] models will display higher robustness against adversarial attacks.

Through this thesis, I will answer these hypotheses, carrying out exhaustive evaluations in the context of medical imaging. The goal is to assess the performance and robustness of these models, hoping to uncover insights that may drive the future of AI and XAI in medical imaging. The visual abstract of this thesis is presented in figure 0.1.

In chapter 1, "Deep Neural Networks for Computer Vision," I will introduce the concept of deep neural networks, focusing primarily on convolutional neural networks and transformer-based neural networks. The sub-sections will delve into the mechanics of convolution operations, pooling, the structure of popular architectures, and an exploration of models such as DenseNet, Vision Transformer, and others. This chapter aims to provide a comprehensive understanding of the architectures and functionalities of these deep learning models in the context of computer vision.

Chapter 2, "Explainability Methods for Neural Networks," pivots the focus to the domain of Explainable AI (XAI), explaining how the decision-making process in deep learning models can be made interpretable. I will discuss a variety of methods, including saliency maps [SVZ13], Layer-wise Relevance Propagation (LRP) [Bin+16], Integrated Gradients (IG) [STY17], and SmoothGrad [Smi+17], thereby establishing an understanding of these methods.

In chapter 3, "Evaluating Transformer Models on CheXpert," I will carry out a series of evaluations to inspect the performance of transformer-based models on the CheXpert dataset. This chapter will explore the dataset, the experimental setup, various configurations for training the models, and an analysis of models pre-trained on ImageNet [Den+09] and RadImageNet[Mei+0]. The goal here is to analyze these models' performance on a real-world medical imaging challenge. This chapter will give an answer to the first hypothesis.

Finally, chapter 4, titled "Adversarial Attacks using Feature Attribution Methods," delves into an exploration of the vulnerability of deep learning models. It uncovers two types of adversarial attacks designed to manipulate the models' attribution maps, namely In and Out mask attacks. These attacks strategically aim to either conform to or deviate from a ground truth mask crafted by a domain expert. To assess the effectiveness of the attack, two location metrics, namely rank and mass accuracy, are utilized, which help to answer the second hypothesis.

Through this thesis, I hope to provide a robust exploration of deep learning models, their explainability, performance in the medical domain, and their robustness to adversarial attacks.

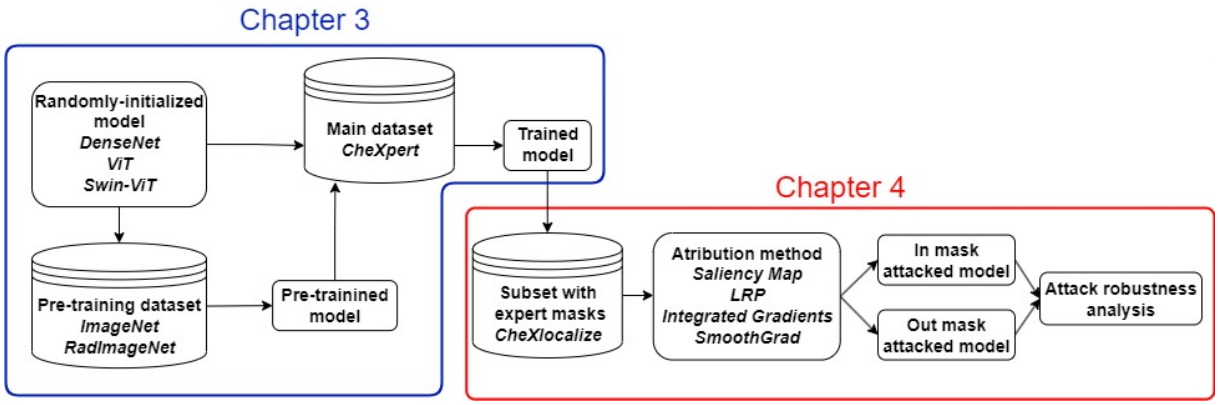

Figure 0.1: Visual abstract of experiments presented in this thesis.

1. Deep Neural Networks for Computer Vision

In this study, the focus will be on image classification, which involves the identification of a function f capable of predicting image labels l. Which can be written as:

$$f(X;\theta) = l. \tag{1.1}$$

Here, $X \in \mathbb{R}^{HwC}$ represents an image, where H denotes the image height, W refers to the image width, and C represents the number of channels. For RGB images, the number of channels C is 3. The neural network parameters are denoted by θ, while l corresponds to the possible labels predicted by the model. The domain of l varies depending on the classification task. Classification can be categorized into three types:

1. binary classification,
2. multiclass clasification,
3. multilabel classification.

In a binary classification setting $l \in \{0, 1\}$. An example of this is determining the absence or presence of a disease. On the other hand, multiclass classification aims to predict one class out of multiple possibilities, with $l \in \{0, 1, \ldots, k\}$, where k represents the number of classes. For instance, predicting whether a patient has a partial fracture, complete fracture, or fracture with misplacement of bone falls under the multiclass classification. Lastly, multilabel classification involves predicting the presence of multiple objects, then $l \in \{0, 1\}^p$, where p signifies the number of objects. An example of this is predicting which lung diseases a patient may have.

Convolutional neural networks and transformer-based models are currently considered state-of-the-art methods for solving various computer vision tasks, including image classification. This is primarily due to their ability to process inputs differently from classical densely connected networks. Utilizing a dense neural network for image processing is computationally demanding. Even for small RGB images with dimensions of 64 by 64, we get 12,288 input features when flattened.

1.1. Convolutional neural networks

Convolutional neural networks (CNNs) brought a revolutionary change in the field of computer vision. LeCun's work [Lec+98], which pioneered the use of modern CNNs for handwritten character recognition, marked the beginning of this transition. Subsequently, deep neural networks outperformed traditional computer vision algorithms such as Viola-Jones.

1.1.1. Convolution in One Dimension

Definition 1.1. The convolution of functions $f, g : \mathbb{R}^d \to \mathbb{R}$, given by:

$$(f * g)(x) = \int_{\mathbb{R}^d} f(u)g(u - x)du. \tag{1.2}$$

According to Murphy [Mur22], when we replace functions with finite-length vectors $w = (w_0, w_1, \ldots, w_L)$ (termed as weight, filter, or kernel) and $x = (x_0, \ldots, x_N)$, equation 1.2 can be rewritten as:

$$(w * x)(i) = w_0 x_i + w_1 x_{i-1} \ldots w_L x_{i-L}. \tag{1.3}$$

In the context of deep learning, the term "convolution" often means *cross correlation*, a concept closely associated with convolution. I will follow this convention in this work.

Definition 1.2. For two vectors $w = (w_0, w_1, \ldots, w_L)$ and $x = (x_0, \ldots, x_N)$, the operation of cross correlation is defined as:

$$(w * x)(i) = w_0 x_i + w_1 x_{i+1} \ldots w_L x_{i+L}. \tag{1.4}$$

The calculation of convolution is valid for i such that $i+L <= N$ - termed as valid convolution. Alternatively, for $j > N$, one may set $x_j = 0$ - known as zero-padding, enabling the computation of convolution for $i \in \{0, \ldots, N\}$.

1.1.2. Application of convolution on Images

Murphy [Mur22] showed that the convolution of an image X with a kernel $K \in \mathbb{R}^{H_k, W_k, C}$ can be expressed as:

$$(K * X)(i, j) = \sum_{u=0}^{H_k-1} \sum_{v=0}^{W_k-1} \sum_{c=0}^{C-1} k_{u,v,c} x_{i+u,j+v,c}, \tag{1.5}$$

Assuming we have input $X \in \mathbb{R}^{HWC}$ and N kernels $K_i \in \mathbb{R}^{H_k, W_k, C}$, which are learnable parameters of the layer, then the output Z of the convolutional layer is calculated as follows:

$$z_{i,j,n} = b_n + \sum_{u=0}^{H_k-1} \sum_{v=0}^{W_k-1} \sum_{c=0}^{C-1} k_{u,v,c,n} x_{i+u,j+v,c}, \tag{1.6}$$

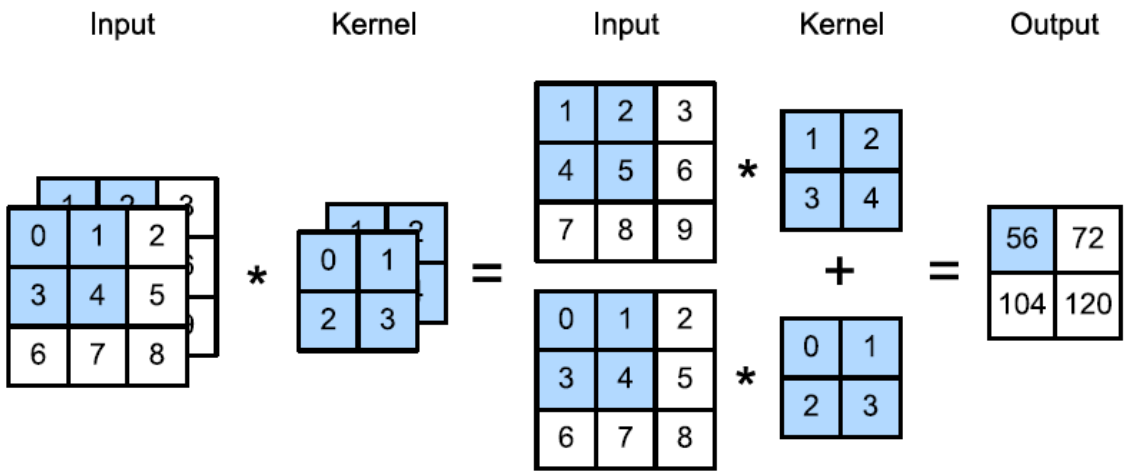

Figure 1.1: Illustration of 2d convolution applied to input with 2 channels from [Mur22].

where b_n represents the bias parameters of the convolutional layer. If the layer uses a valid convolution, then the output Z will be a three-dimensional array with dimensions $(H - H_k + 1, W - W_k + 1, N)$. When the layer uses zero-padding, the output will have dimensions (H, W, N) as described in [Mur22].

As illustrated in Figure 1.1, convolution layers "slide" kernels over inputs extracting local features from inputs. Convolutional layers in convolutional neural networks apply multiple kernels, which are learnable parameters, to the input.

The user is able to specify the *stride*, which sets a number of pixels by which a kernel moves in a single slide. For instance, a convolutional layer with a $stride = 2$ would calculate the cross-correlation of (i, j) that are multiples of 2. This can be expressed as:

$$z_{i,j,n} = b_n + \sum_{u=0}^{H_k-1} \sum_{v=0}^{W_k-1} \sum_{c=0}^{C-1} k_{u,v,c,n} x_{si+u,sj+v,c} \tag{1.7}$$

where s is the stride length.

1.1.3. Pooling

A typical convolutional layer has three stages. First, convolutions with multiple kernels are calculated in parallel. This is followed by the application of a non-linear function (e.g. ReLU). Finally, pooling is performed. Pooling layers aggregate input parts into a local output statistic. Widely used types of pooling are *average* and *max* pooling, which produce the average and the maximum of local outputs, respectively. Figure 1.2 illustrates an example of max pooling with a (2×2) *filter* and $stride = 2$.

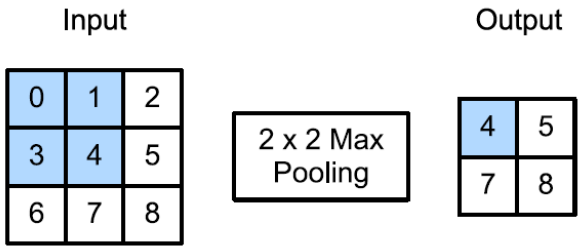

Figure 1.2: Illustration of max pooling with a 2x2 filter and a stride of 1 from [Mur22].

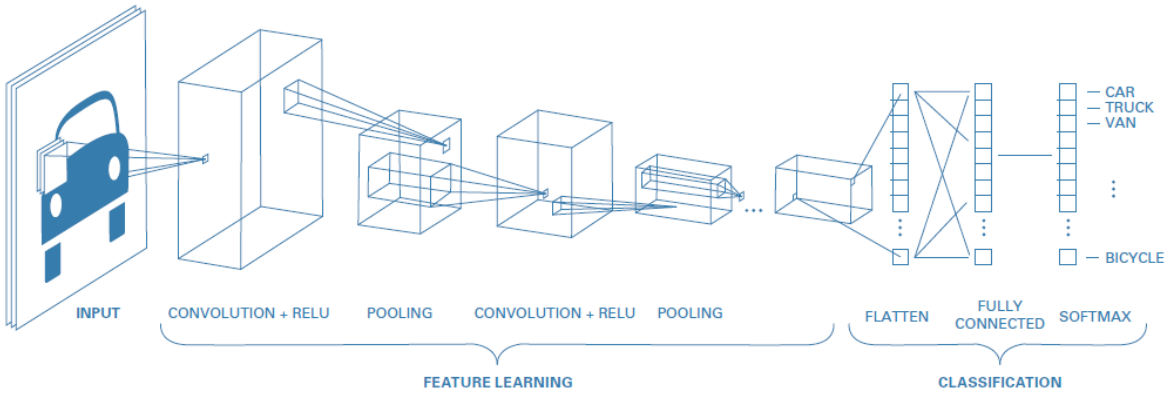

Figure 1.3: A simple CNN for classifying images. Source: [Mur22]

1.1.4. Architectural Design for Classification

A convolutional neural network designed for classification consists of multiple convolutional layers which are spatially aggregated (for example, through flattening or max/average over spatial dimensions) after the last convolutional layer. The aggregated output is then passed through a fully connected neural network that generates predictions of labels. Figure 1.3 presents a simple CNN for image classification. Convolutional neural networks offer computational efficiency due to their sparse and shared parameters. The use of convolutional layers introduces an inductive bias, implying that predictions are based on local features and are translation invariant. This feature and their lower parameter and computational requirements have enabled their successful use in image classification, as shown in [Lec+98].

1.1.5. DenseNet: A Common CNN Architecture for Computer Vision

DenseNet [Hua+17] is a commonly used CNN architecture in the field of computer vision. Unlike ResNet [He+16], which adds the output of each layer to its input (known as a residual connection), DenseNet concatenates the outputs with the inputs. This architecture will serve as the benchmark in the experiments.

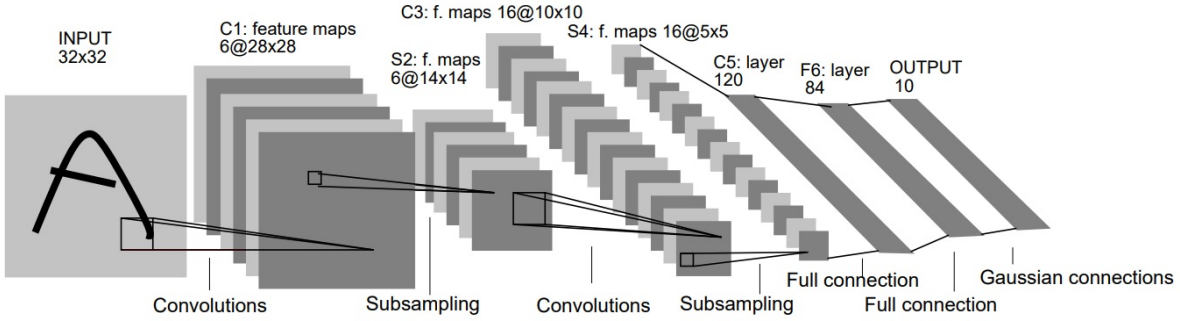

Figure 1.4: Architecture of LeNet-5 used in [Lec+98].

1.2. Transformer-based Neural Networks

The transformer-based architecture was introduced in 2017 by Vaswani [Vas+17], which was a turning point in the field of natural language processing (NLP). Initially designed for NLP, these architectures found their way into computer vision tasks by 2020, thanks to the work of Dosovitskiy [Kol+21].

1.2.1. Machine Translation

Machine translation can be written as the translation of a sequence of tokens from a source language into a sequence of tokens in a target language. Let's represent two sequences of tokens, $S = (s_1, s_2, \ldots, s_l)$ from the source language and $T = (t_1, t_2, \ldots, t_k)$ from the target language. The machine translation task is then to find a function f such that:

$$f(S, T_{:i-1}; \theta) = t_i, \quad \forall i \in 1, 2, \ldots, k, \tag{1.8}$$

where $T_{:i-1} = (t_1, \ldots, t_{i-1})$.

1.2.2. Understanding Transformer Models

Vaswani introduced the transformer model, a novel architecture that uses an encoder-decoder structure specifically designed for machine translation tasks. These transformer-based models use attention mechanisms to represent input sequences that allow them to capture long-distance relationships between input tokens.

Compared to Recurrent Neural Networks (RNNs), transformers have direct connections between all tokens, allowing easy modeling of long relationships [Mur22, p. 527]. Bahdanau [BCB14] incorporated attention to machine translation before the transformer model was introduced.

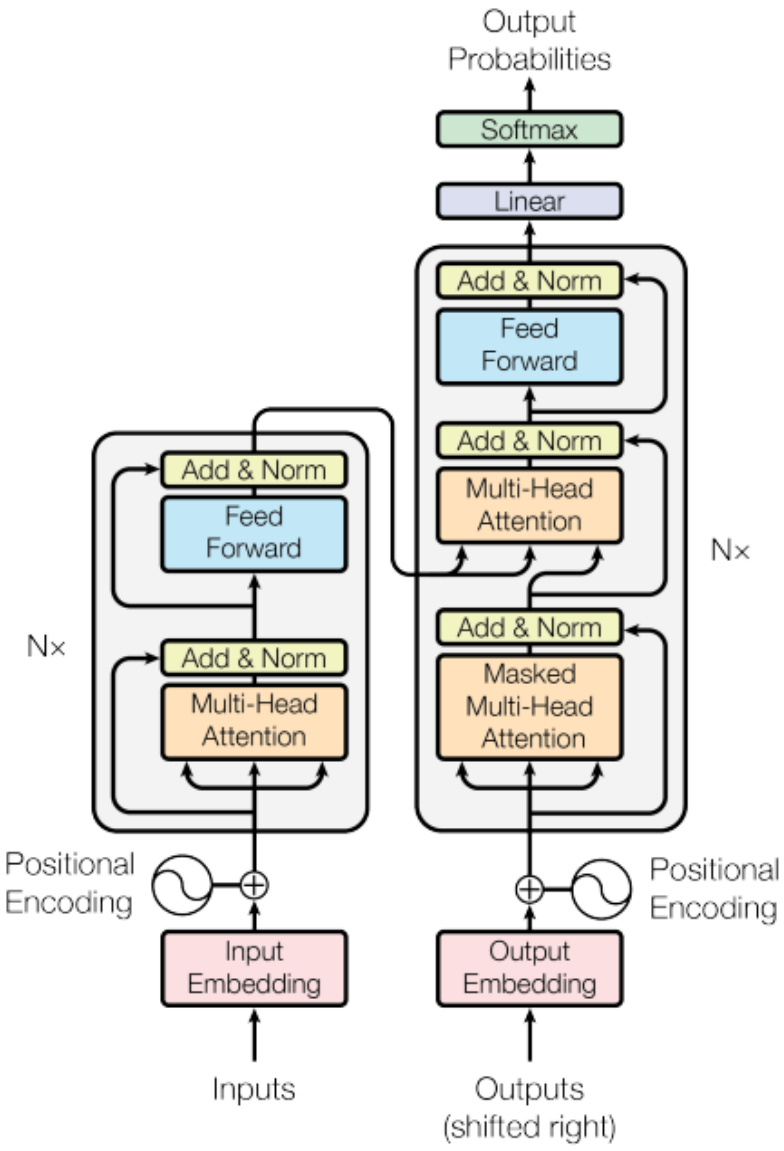

Figure 1.5: Encoder-decoder Transformer model introduced in [Vas+17].

However, this attention mechanism was used as an additional building block over RNNs to allow dynamic aggregation of encoder hidden states based on the current decoder hidden state.

Transformer-based models have allowed huge advancements in machine translation tasks. Their ability to manage long-distance relationships between tokens and efficient processing capabilities has resulted in their widespread adoption in NLP and beyond.

Understanding Attention

Attention operates based on the matrix multiplication of three two-dimensional arrays termed as *query, key,* and *value* (denoted as Q, K, and V, respectively). These dimensions represent the

sequence length and the *hidden size*. The attention operation is defined as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V, \tag{1.9}$$

where d_k = *hidden size*. Attention consists of array multiplications that can be parallelly computed, making transformer models substantially faster than RNNs. Attention assists the model in constructing a sequence representation with values V by calculating a weighted average, the weights of which are determined based on the dot product of Q and K. The output of the attention operation maintains the same dimensions as V.

While attention scores might appear as a potential tool for explaining model predictions, Jain [WP19] demonstrated that attention shouldn't be directly used for explanatory purposes.

Types of Attention

In the encoder-decoder structure, two types of attention operations are used:

1. Self-attention,
2. Cross-attention.

In self-attention, $Q = K = V$ holds, which can be interpreted as generating token representations in the context of the entire sequence. In the case of cross-attention, only $K = V$ applies, suggesting the creation of a representation of V in relation to Q (for machine translation, we aim to create a representation of the input sequence concerning the output sequence). The original Transformer paper [Vas+17] also introduced masking attention, which zeroes attention weights based on an attention mask. This technique is used to prevent attention with future tokens in the decoder by using a lower triangular attention mask.

Multi-head Attention

Transformer-based models use multi-head attention layers, enabling the model to concurrently create representation from different subspaces at various positions. Multi-head attention is defined as:

$$\begin{aligned}\text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_n) \\ \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V),\end{aligned}$$

where W_i^Q, W_i^K, and W_i^V are projection matrices into a lower dimension.

Position-wise Feed-forward Layer

Both the encoder and decoder make use of position-wise feed-forward layers placed between multi-head attention layers. These consist of two fully connected layers with a non-linearity

(Vaswani used ReLU in his work):

$$FFN(x) = \sigma(xW_1 + b_1)W_2 + b_2,$$

where σ can represent any non-linear function. These layers are applied across all sequence positions.

Incorporating positional encoding

In Figure 1.5, it is visible that *Positional Encoding* (PE) is added to input embeddings. Given the permutation invariance of attention layers, Vaswani added vectors to token embeddings that are parameterized by the token's position (pos) within the sequence. They leveraged positional encoding as sine and cosine functions of varying frequencies, defined as:

$$PE(pos, 2i) = \sin(pos/10000^{2i/d_k})$$
$$PE(pos, 2i+1) = \cos(pos/10000^{2i/d_k}),$$

where d_k is the model hidden state size. They also experimented with learnable vectors as Positional Encoding. However, the results between the two approaches were similar, but the usage of Positional Encoding based on frequencies does not limit the length of the input sequences.

Stacking Attention Layers

The transformer encoder consists of N blocks. The output X_n from block n can be expressed with the output X_{n-1} from block $n-1$:

$$X' = \text{LayerNorm}(\text{MultiHead}(X_{n-1}, X_{n-1}, X_{n-1}) + X_{n-1})$$
$$X_n = \text{LayerNorm}(FFN(X') + X'),$$

where LayerNorm is a normalization operation proposed in [BKH16]. For $n = 0$, X_0 is equal to the sum of input token embeddings and positional encoding.

The decoder also consists of N blocks. The output Y_n from block n can be expressed with the output Y_{n-1} from the preceding block and the output from the encoder X:

$$Y' = \text{LayerNorm}(\text{MultiHead}(Y_{n-1}, Y_{n-1}, Y_{n-1}) + Y_{n-1})$$
$$Y'' = \text{LayerNorm}(\text{MultiHead}(Y', X, X) + Y')$$
$$Y_n = \text{LayerNorm}(FFN(Y'') + Y'').$$

Like the encoder, Y_0 is equal to the sum of target token embeddings and positional encoding.

The main difference between the decoder and encoder blocks is an additional multi-head cross-attention layer. Figure 1.5 visualizes the complete transformer architecture.

Categories of Transformer Models

Vaswani [Vas+17] initially proposed an encoder-decoder-based model. However, transformer models can be further segmented into two additional types, resulting in three primary categories of Transformer models:

1. Encoder-decoder transformer,
2. Encoder-only transformer,
3. Decoder-only transformer.

The main distinction between encoder-only and decoder-only transformers is the method by which attention is computed. In the case of the latter, attention between tokens i, j is nullified for all tokens where $j > i$. This prevents the model from utilizing future tokens during prediction, which makes it architecture perfect for the text generation. Attention is unrestricted for the former, making it an architecture optimally designed for tasks involving natural language understanding, such as text classification, named entity recognition, etc.

BERT [Dev+19] is an example of an encoder-only transformer, while GPT-2 [Rad+19] is an example of a decoder-only transformer.

1.2.3. Vision Transformer (ViT)

Images, being three-dimensional arrays, differ from text which can be represented as two-dimensional arrays upon tokenization and conversion of tokens into embeddings. Therefore, Dosovitskiy [Kol+21] represented images as a sequence of flattened image patches (parts of an image), transforming the image into a two-dimensional array. This enabled usage of an encoder-based transformer for images for classification purposes. The significant difference between a transformer and CNN lies in the transformer's ability to generate global features of images, thereby lacking the inductive bias present in CNNs.

Partitioning of Images

Prior to processing an image through a transformer model, it must be divided into a sequence of patches. Dosovitskiy [Kol+21] proposed segmenting an image into 16x16 non-overlapping patches, which are then flattened and passed thru a linear projection layer, as illustrated in figure 1.6. Position Embeddings, which are learnable 1D position vectors, are added to the patches sequence before they are fed into the transformer layers.

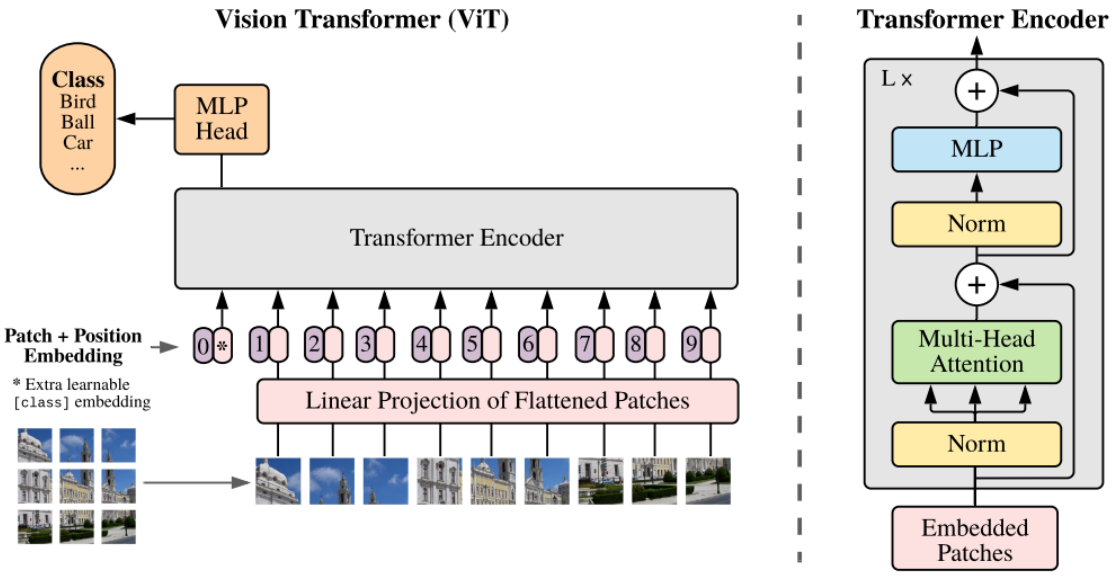

Figure 1.6: Architecture of the Vision Transformer (ViT). Source: [Kol+21]

Transformer Layers

Following the split of an image into a patch sequence, a learnable vector is concatenated to the start of the patch sequence. The output from the first token, post-transformer blocks, is used as the input to the classification head, which is a multi-layer perceptron. ViT also uses GeLU over ReLU non-linearity in the feed-forward layer.

Extensive Pre-Training

It is known that transformer architectures yield exceptional performance when pre-trained on large text datasets [Dev+19; Tou+21; Zhu+21; Rad+19]. Taking it into consideration, Dosovitskiy [Kol+21] utilized a large image dataset, JFT-300M [Sun+17], for pre-training. The ViT pre-trained on this data outperformed leading CNN architectures on several datasets, including ImageNet [Den+09], and CIFAR [Kri09]. The influence of pre-training dataset size on fine-tuning performance was also examined. Results indicated that while CNNs outperform when dataset sizes are smaller (less than 9M), they lag behind vision transformers as pre-training data begins to expand (over 90M examples). This observation aligns with the rationale that convolution inductive bias proves advantageous when data is limited, but its removal becomes beneficial as datasets enlarge.

Dosovitskiy [Kol+21] also proposed fine-tuning ViT into higher resolution. Utilizing higher-resolution images results in an increased number of patches. However, given that Positional

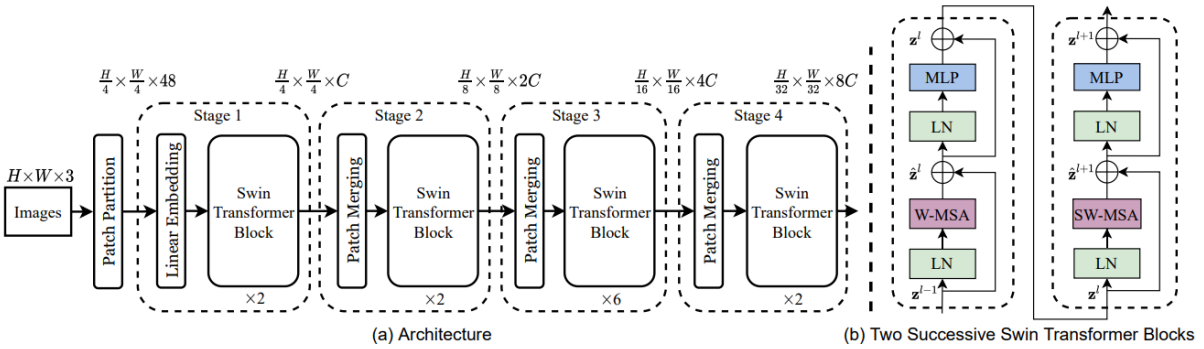

Figure 1.7: (a) Architecture of Swin Transformer (Swin-T); (b) Two consecutive Swin Transformer Blocks as illustrated in [Liu+21]

Embedding was trained on a lower resolution, vectors for new patches are not available. To resolve this issue, 2D interpolation of the pre-trained position embeddings is conducted in line with the patch's location within the image.

1.2.4. Swin Transformer: Hierarchical Vision Transformer with Shifted Window Approach

ViT model comes with several limitations. A key drawback is a challenge in processing high-resolution images due to the quadratic rise in computational complexity in line with image size. Additionally, fixed-scale patches employed in ViTs are ill-suited for vision tasks that involve visual elements with different scales. In response to these limitations, Liu [Liu+21] proposed the Swin Transformer (Swin-ViT), featuring hierarchical feature maps and a shifted window attention approach.

Hierarchical Feature Maps

Liu proposed merging patches in 2x2 neighboring groups three times to address the problem associated with elements of varying scales, as shown in figure 1.7. Instead of 16x16 patches used in ViT, a patch size of 4 was used. After **Stage 4**, it can be observed that there is a $2^3 \cdot 2^3$ reduction in input tokens entering the transformer block.

Shifted Window Attention

Another proposed innovation was shifted window attention, which reduces computational complexity relative to image size from quadratic to linear. They proposed computing attention in non-overlapping windows of size $M \times M$ patches. While this window-based attention approach doesn't allow for connections outside of windows, the authors suggested employing shifted atten-

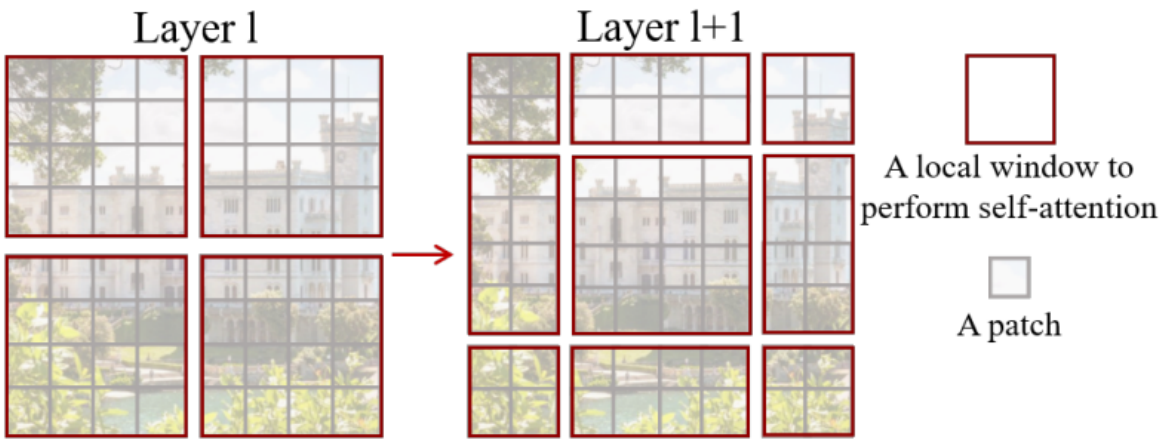

Figure 1.8: Diagram of shifted windows attention from [Liu+21].

tion in successive layers to solve this issue. Shifting attention windows by $\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor$ in spatial dimensions enables the model to compute cross-window attentions. Figure 1.8 illustrates how these attention windows look like for $M = 4$.

Relative Position Bias

In the process of computing attention, they introduced a relative position bias $B \in \mathbb{R}^{M^2 \times M^2}$ by adjusting each head attention computation as follows:

$$\text{Attention}(Q, K, V) = \text{Softmax}(QK^T/\sqrt{d} + B)V. \tag{1.10}$$

To incorporate relative patch positions in the window, a smaller matrix $\hat{B} \in \mathbb{R}^{(2M-1)\times(2M-1)}$ is introduced, as 2D relative positions fall within $[-M+1, M-1]^2$. Based on the 2D relative positions between patches within the window, $\hat{B}$ values are used to construct a B matrix.

Bringing it all together

The difference between ViT and Swin Transformer lies in the introduction of patch merging, the employment of two self-attention modules with windows and shifted windows attention of a single self-attention layer as depicted in figure 1.7 (b), and ultimately the usage of relative position bias within attention heads.

Experiments highlighted in [Liu+21] demonstrate that training the Swin Transformer model only on ImageNet [Den+09] without pretraining delivers results on par with state-of-the-art CNNs and outperforms other transformer-based architectures. Notably, the Swin Transformer operates faster than other transformer-based architectures, offering a better inference time-accuracy trade-off than CNNs.

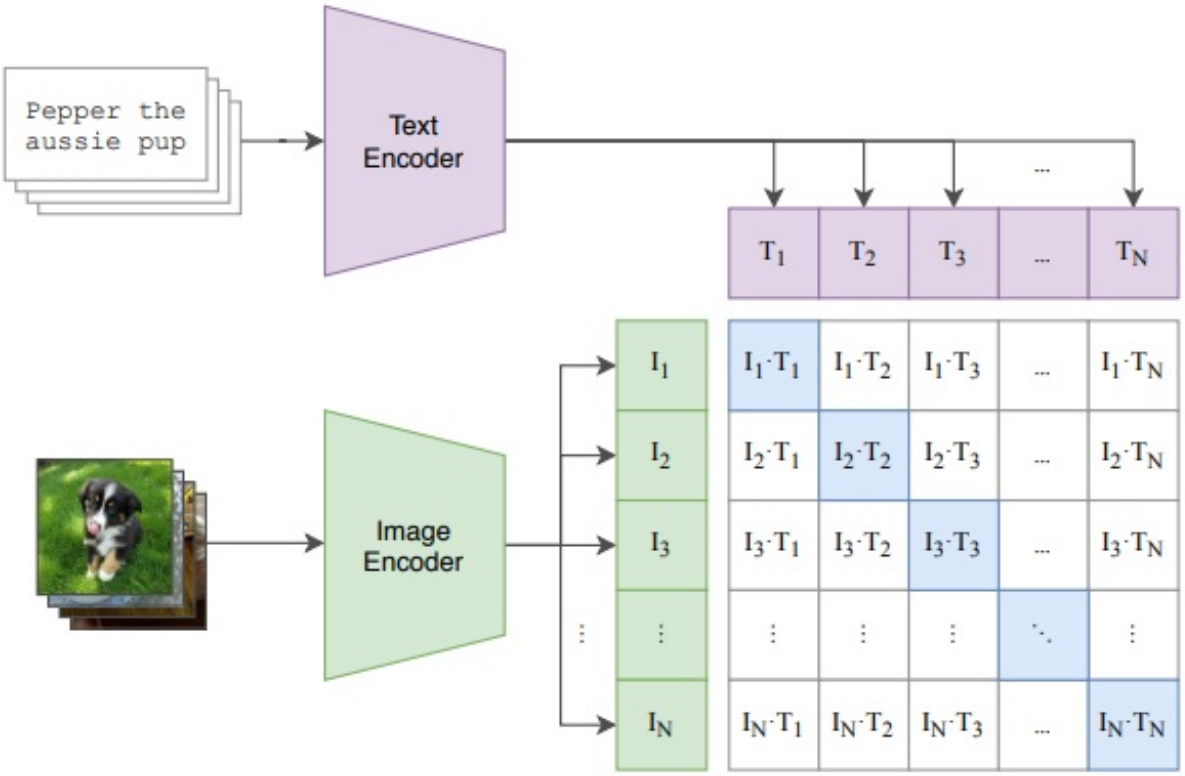

Figure 1.9: Diagram of contrastive pre-training from [Rad+21].

1.2.5. Contrastive Language-Image Pre-training (CLIP)

Articles [Kol+21; Liu+21] have shown the exceptional performance of transformer-based models when pre-trained on expansive datasets. However, large datasets with labels for pre-training aren't easily available. Because of that, Radford [Rad+21] proposed a novel approach named contrastive pre-training designed to enhance the generalization capacity and utility of models pre-trained using this method. This approach employs data comprising pairs of images and their associated captions. The task consists of predicting if a caption matches an image. They used two distinct models designated for both modalities to perform this task. During pre-training, the normalized dot product between the text and image representations is used as the logits for predicting if a caption matches an image, as shown in figure 1.9. The authors tested ResNet [He+16] and ViT as image encoders, while an encoder-based transformer model with modifications suggested by Radford [Rad+19] was utilized for the text encoder.

CLIP is pre-trained to predict if a text matches an image, so it can be conveniently used for zero-shot classification. In this regard, the authors suggested calculating the image representation and texts related to all classes of interest (for example, all classes from ImageNet), then computing the cosine similarity between image and text embeddings. This is subsequently

normalized into probability via the softmax function.

2. Interpreting Neural Networks Through Explainability Methods

Neural networks are known for their overfitting problems. Because of that, one should perform an examination beyond calculating performance metrics such as accuracy, AUC ROC, F1, etc. This kind of verification process can be done with explainability methods, which analyze the specifics of the model's prediction process. By doing so, it's possible to find the features that the model concentrates on during its prediction.

A well-known class of explainability methods are feature attributions, which highlight the input components that significantly impact the model's prediction. With these heatmaps, one can utilize test set examples to assert whether the model's predictions are influenced by spurious features.

Ribeiro's work [RSG16] serves as a compelling example of the power of explainability methods. In his study, a neural network was trained to differentiate between images of huskies and wolves. Employing explainability methods revealed that the model primarily relied on the image's background for its predictions - associating images with snowy backgrounds with wolves. This was because the dataset contained an intentional correlation - all images of wolves contained a snowy background, while those of huskies did not. As a result, the model learned to connect the presence of snow with wolves rather than distinguishing the actual animal in the image.

2.1. Saliency Map

The saliency map was first proposed by Simonyan [SVZ13] and is generated by computing the gradient of the input concerning a particular class of interest.

Definition 2.1 (Saliency Map). Consider a model $f : \mathbb{R}^p \times \mathbb{R}^m \to \mathbb{R}^k$ that provides scores $S = (s_1, \ldots, s_k)$ for $k \in \mathbb{N}$ classes, given an input $X \in R^p$ and model parameters $\theta \in \mathbb{R}^m$. The saliency map M of model f for the input X relative to class i is defined as

$$M(X;\theta) = \frac{\partial f(X;\theta)_i}{\partial X}.$$

This method employs the first-order Taylor expansion to approximate model f into a linear

model with respect to the features X. It can also be interpreted as assessing which feature needs the least change to affect the output of the model for class i, as represented by the gradient.

2.2. Layer-wise relevance propagation (LRP)

LRP, introduced by Bach et al. [Bac+15; Bin+16], uses the hierarchical structure of neural networks to compute input attributions. Suppose we have a model $f : \mathbb{R}^p \times \mathbb{R}^m \to \mathbb{R}^k$, an input $X \in R^p$ and model parameters $\theta \in \mathbb{R}^m$, LRP aims to determine relevance scores $R_i^{(1)}$ satisfying

$$f(X;\theta) \approx \sum_{i=1}^{p} R_i^{(1)}.$$

Assume that our model f is composed of $L \in \mathbb{N}$ layers. Let (l) denote a set of all outputs from layer $l \in L$. Given the relevance of neuron j, $R_j^{(l+1)}$ at layer $l+1$, we aim to decompose $R_j^{(l+1)}$ into messages $R_{i\leftarrow j}^{(l,l+1)}$ from neuron i at layer l sent to neuron j of layer $l+1$ so that the following equations are true:

$$R_j^{(l+1)} = \sum_{i\in(l)} R_{i\leftarrow j}^{(l,l+1)} \tag{2.1}$$

$$R_i^{(l)} = \sum_{j\in(l+1)} R_{i\leftarrow j}^{(l,l+1)}. \tag{2.2}$$

Relevance at the last layer L is set to $R_i^{(L)} = f(X)_i$. If we wish to determine the relevance relative to a specific class, we set the relevance of other classes at layer L to zero.

Two methods for computing messages $R_{i\leftarrow j}^{(l,l+1)}$ are proposed in [Bin+16]. Let's define:

$$z_{ij} = a_j \cdot w_{jk}$$
$$z_j = \sum_{i\in(l)} z_{ij},$$

where a_j are activations of the lower–layer l and w_{jk} are weights of lower–layer l. The first formula, known as the ϵ–rule, introduced in [Bin+16], is expressed as:

$$R_{i\leftarrow j}^{(l,l+1)} = \frac{z_{ij}}{z_j + \epsilon \cdot \text{sign}(z_j)} R_j^{(l+1)}, \tag{2.3}$$

where $\epsilon \in \mathbb{R}$ is a small number (e.g., 1e–9) introduced to prevent zero division. The second formula, the β–rule, is given by:

$$R_{i\leftarrow j}^{(l,l+1)} = \left((1+\beta)\frac{z_{ij}^+}{z_j^+} - \beta\frac{z_{ij}^-}{z_j^-} \right) R_j^{(l+1)}. \tag{2.4}$$

Using these relevance propagation rules, we can calculate the relevance of inputs with respect to the output. A negative relevance implies that the feature provides evidence against a class, while a positive relevance suggests that the feature supports the presence of class i.

Several additional propagation rules that can be used to compute LRP are outlined in [Mon+19]. Shrikumar [Shr+17] illustrated that in certain conditions, LRP [Bac+15] simplifies to element-wise multiplication of the saliency map [SVZ13] with the input.

2.3. Integrated Gradients (IG)

The Integrated Gradient method, introduced by Sundararajan [STY17], was designed to satisfy two essential axioms: Sensitivity and Implementation Invariance. Sensitivity is achieved when, for every input and baseline differing in one feature and prediction, that feature has a non-zero attribution. An attribution method is considered Implementation Invariant if attributions remain consistent for two networks that only differ in their implementation, meaning they return identical outputs for all inputs. IG's authors noted that both LRP and saliency maps fail to satisfy these axioms.

Definition 2.2 (Integrated Gradients). Suppose we have a function $f : \mathbb{R}^p \times \mathbb{R}^m \to \mathbb{R}$, an input $X = (x_1, \ldots, x_p) \in R^p$, a "baseline" input $X' = (x'_1, \ldots x'_p) \in R^p$ and model parameters $\theta \in \mathbb{R}^m$. The Integrated Gradients for feature i is then computed as follows:

$$\text{Integrated Gradients}_i := (x_i - x'_i) \cdot \int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha(x - x'); \theta)}{\partial x_i} d\alpha.$$

Integrated gradients satisfy both axioms as described in [STY17]. The authors also established that

$$\sum_{i=1}^{p} \text{Integrated Gradients}_i = f(X; \theta) - f(X'; \theta).$$

Given a baseline X' such that $f(X'; \theta) \approx 0$, IG can be interpreted as a distribution of the output into input features.

Computing the integral in IG is typically infeasible in real-world scenarios. The authors suggest approximating it as follows:

$$\text{Integrated Gradient}_i \approx \left(x_i - x'_i\right) \cdot \frac{1}{m} \sum_{k=1}^{m} \frac{\partial f\left(x' + \frac{k}{m}\left(x - x'\right)\right); \theta\Big)}{\partial x_i}.$$

In [STY17], the authors selected black images (all zeros) as the baselines. While IG is defined for one-dimensional outputs, if we have a neural network with $k > 1$ dimensional output, we would need to select an output to calculate the attribution with respect to.

2.4. SmoothGrad

"SmoothGrad: Removing Noise by Adding Noise" is an article by Smilkov [Smi+17] that presents the SmoothGrad technique. This method involves adding noise to input features multiple times and then calculating the average of the saliency map across these perturbed features. This straightforward procedure results in sharper, less noisy attribution maps.

Definition 2.3. Assuming a saliency map M, input X and model parameters θ, the SmoothGrad attribution M_S is defined as follows:

$$M_S(X, n; \theta) = \sum_{i=1}^{n} M(X + \epsilon_i; \theta), \tag{2.5}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

Smilkov suggested the use of $\sigma = \sigma'/(x_{max} - x_{min})$, where σ' represents the *noise level*. The article mentions that a 10%-20% noise level yielded the best results on ImageNet data. One can substitute the saliency map M with any preferred attribution method. Smilkov observed that the combination of SmoothGrad and Integrated Gradients enhances the visual consistency of IG maps.

3. Evaluating Transformer models on CheXpert

To evaluate transformer models, I trained them in 3 scenarios: training from scratch, fine-tuning models pre-trained on ImageNet, and fine-tuning models pre-trained on RadImageNet. These experiments will allow me to test how Transformer models behave in the X-Ray domain.

3.1. RadImageNet

RadImageNet [Mei+0] is a medical image dataset. The RadImageNet database includes 1.35 million annotated CT, MRI, and ultrasound images. The RadImageNet database contains medical images of 3 modalities, 11 anatomies, and 165 pathologic labels. Pre-training models on this dataset make more sense in case of later fine-tuning them on medical images. ImageNet has a much different data distribution than X-Ray images because it contains colorful images with common objects, whereas X-Rays are grayscale.

3.2. CheXpert dataset

I will use CheXpert [Irv+19] dataset in my experiments. This dataset contains 224 316 chest radiographs of 65 240 patients and 14 possible labels related to chest pathologies. Unfortunately, training data labels were created with automatic labelers, and because of that, some observations have a special value "U" (uncertain) for some labels, which means that this observation label is uncertain. Table 3.1 shows the distribution of labels in CheXpert training data. The authors propose multiple ways to handle uncertainty. I followed the "U-Ones" model in my experiments, replacing the "U" labels with 1.

3.2.1. CheXlocalize subset

Saporta [Sap+22] introduced the CheXlocalize dataset. The validation and test sets consist of 234 chest X-rays from 200 patients and 668 chest X-rays from 500 patients. It contains 11 labels out of 14 in CheXpert: Atelectasis, Cardiomegaly, Consolidation, Edema, Enlarged

Table 3.1: Labels distribution in CheXpert training data.

Pathology	Positive (%)	Uncertain (%)	Negative (%)
No Finding	16627(8.86)	0(0.0)	171014(91.14)
Enlarged Cardiom.	9020(4.81)	10148(5.41)	168473(89.78)
Cardiomegaly	23002(12.26)	6597(3.52)	158042(84.23)
Lung Lesion	6856(3.65)	1071(0.57)	179714(95.78)
Lung Opacity	92669(49.39)	4341(2.31)	90631(48.3)
Edema	48905(26.06)	11571(6.17)	127165(67.77)
Consolidation	12730(6.78)	23976(12.78)	150935(80.44)
Pneumonia	4576(2.44)	15658(8.34)	167407(89.22)
Atelectasis	29333(15.63)	29377(15.66)	128931(68.71)
Pneumothorax	17313(9.23)	2663(1.42)	167665(89.35)
Pleural Effusion	75696(40.34)	9419(5.02)	102526(54.64)
Pleural Other	2441(1.3)	1771(0.94)	183429(97.76)
Fracture	7270(3.87)	484(0.26)	179887(95.87)
Support Devices	105831(56.4)	898(0.48)	80912(43.12)

Cardiomediastinum, Lung Lesion, Lung Opacity, Pleural Effusion, Pneumothorax, No Findings, and Support Devices.

The validation set of CheXlocalize is the same as CheXpert validation data, but the test set contains newly labeled observations. All these observations were labeled by radiologists, and additionally, every positive label has a corresponding mask related to the pathology.

3.3. Experimental setup

All models were trained using images resized to a size of (224 x 224). The images were normalized to the range [-1, 1]. Every model was trained with random rotations in the range of (-15, 15) degrees. The optimizer used was AdamW [LH17a]. Transformer models were trained using a cosine learning rate schedule with warmup [LH17b] with 2000 warmup steps. In each experiment, the base learning rate was set to 1e–4.

Given the limited amount of validation data available in CheXpert (234 images), a subset of 1200 patients was randomly selected from the training data to create a separate validation dataset. The performance of the models was then evaluated on this validation dataset. The training process was implemented using PyTorch Lightning [Ft19] which is a library built on top

Table 3.2: Maximal ROCAUC for each class for one channel input models trained on CheXpert.

Class	DenseNet	Swin-ViT	ViT
Atelectasis	**0.684**	0.679	0.668
Cardiomegaly	**0.826**	0.808	0.811
Consolidation	**0.704**	0.702	0.699
Edema	**0.831**	0.818	0.803
Enlarged Cardiom.	**0.617**	0.584	0.574
Fracture	**0.629**	0.521	0.577
Lung Lesion	**0.612**	0.498	0.597
Lung Opacity	**0.711**	0.700	0.688
No Finding	**0.881**	0.797	0.869
Pleural Effusion	**0.863**	0.857	0.838
Pleural Other	**0.563**	0.507	0.522
Pneumonia	**0.749**	0.697	0.672
Pneumothorax	**0.816**	0.734	0.760
Support Devices	0.823	**0.828**	0.753
Macro	**0.729**	0.688	0.694

of [Pas+19]. All data was logged using the Weights and Biases framework [Bie20].

3.4. Training models from scratch

3.4.1. One input channel

Firstly, I trained models using grayscale X-ray images, which have only one input channel. Table 3.2 shows the macro ROCAUC values for these three models. It reveals that DenseNet achieved the highest macro ROCAUC. On the other hand, Figure 3.1 illustrates that the DenseNet model starts to overfit quicker than the transformer-based models.

3.4.2. Three input channels

Upon increasing the number of channels in the models to three, we observe improved performance on the validation dataset. This improvement is particularly noticeable for the transformer-based models. Specifically, the Swin ViT achieves a maximum macro ROCAUC of approximately 0.76, while the ViT achieves a ROCAUC of around 0.73, compared to a ROCAUC of approxi-

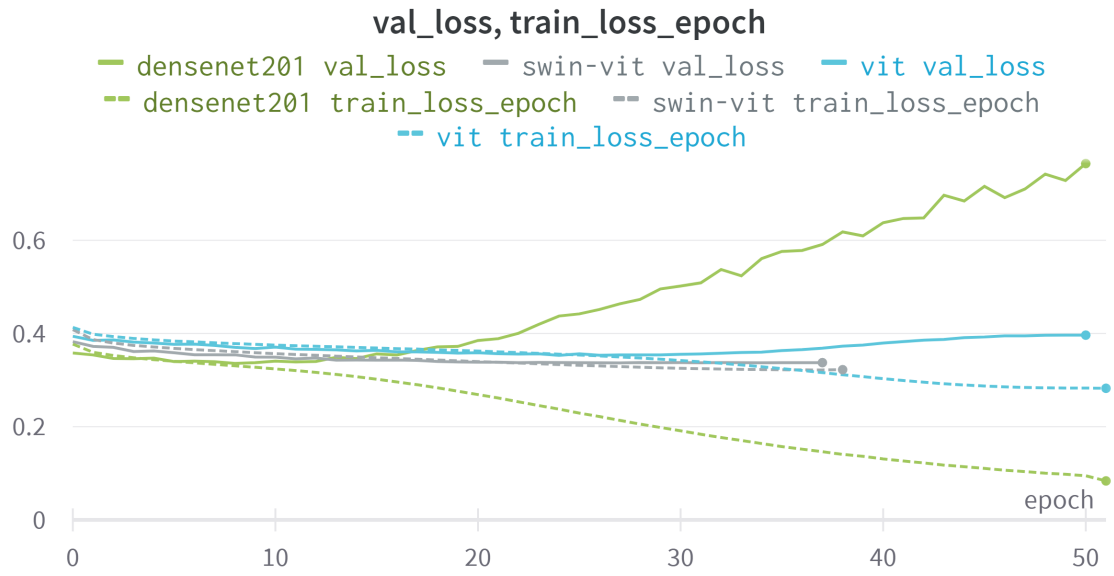

Figure 3.1: Train and validation losses of models with one input channel on CheXpert data.

Table 3.3: Maximal ROCAUC for each class for three channel inputs models trained on CheXpert.

Class	DenseNet	Swin-ViT	ViT
Atelectasis	0.684	**0.686**	0.674
Cardiomegaly	**0.844**	0.834	0.824
Consolidation	0.694	**0.712**	0.683
Edema	**0.829**	0.828	0.814
Enlarged Cardiom.	**0.667**	0.666	0.635
Fracture	**0.760**	0.743	0.720
Lung Lesion	0.732	**0.740**	0.709
Lung Opacity	0.708	**0.709**	0.703
No Finding	**0.881**	0.876	0.875
Pleural Effusion	**0.863**	0.860	0.851
Pleural Other	**0.712**	0.687	0.671
Pneumonia	0.748	**0.749**	0.692
Pneumothorax	**0.825**	0.803	0.763
Support Devices	**0.833**	0.830	0.773
Macro	**0.761**	0.758	0.734

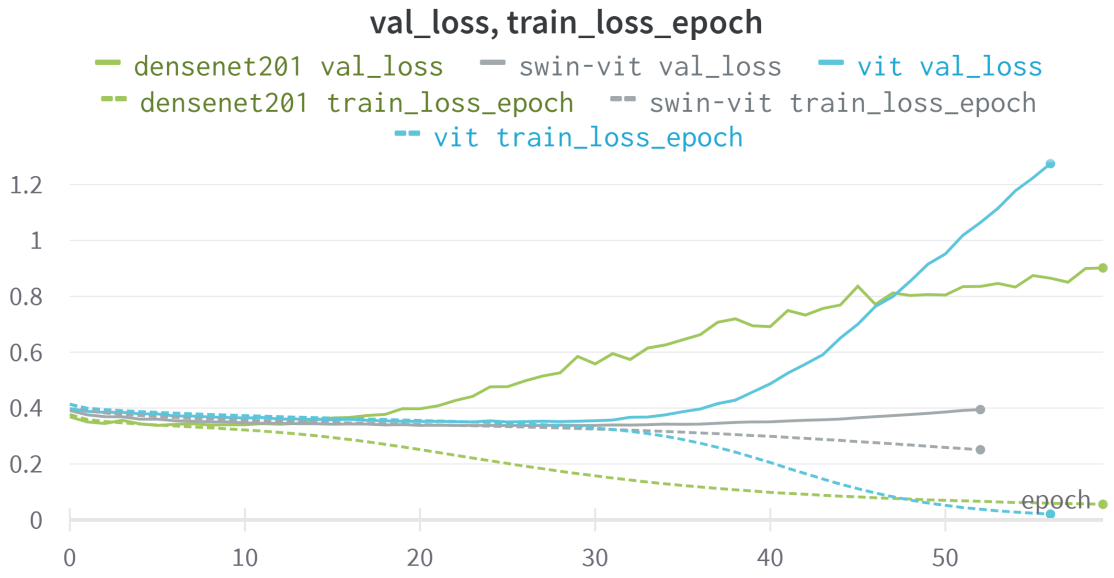

Figure 3.2: Train and validation losses of models with three input channels on CheXpert data.

mately 0.69 for the one-channel setup (see table 3.3).

We can make several observations by examining both figure 3.1 and figure 3.2. Firstly, the ViT model starts to overfit later than DenseNet. However, its validation loss quickly rises and catches up to the loss of DenseNet. Secondly, we notice that the training and validation losses start to diverge for the Swin ViT, although much slower than for the other architectures.

3.5. Models pre-trained on ImageNet

During the training process of models pre-trained on other ImageNet data, it is evident from figure 3.3 that these models show quicker signs of overfitting compared to models initialized randomly. Furthermore, by comparing the macro ROCAUC scores (figure 3.4), we observe that the pre-trained DenseNet and Swin ViT have worse performance than when trained from scratch.

This performance can be attributed to the significant difference between the data used for pre-training these models and the chest X-ray images. Consequently, when applied to X-ray images, these pre-trained models may attempt to utilize features learned from "ordinary" images, which could lead to classification based on irrelevant features unrelated to the presence of pathology.

Moreover, by examining the loss function in figure 3.3, we notice that the training and validation losses begin to diverge rapidly for all models, even for the Swin ViT, which displayed higher robustness to overfitting when initialized with random weights.

Table 3.4: Maximal ROCAUC for each class for models pre-trained on ImageNet and fine-tuned on CheXpert.

Class	DenseNet	CLIP	Swin-ViT	ViT
Atelectasis	**0.706**	0.689	0.702	0.694
Cardiomegaly	0.826	0.812	0.808	**0.834**
Consolidation	**0.712**	0.704	0.703	0.704
Edema	**0.845**	0.823	0.836	0.835
Enlarged Cardiom.	0.624	0.618	0.624	**0.628**
Fracture	**0.659**	0.611	0.550	0.624
Lung Lesion	**0.616**	0.599	0.494	0.588
Lung Opacity	**0.719**	0.702	0.709	0.710
No Finding	**0.891**	0.876	0.806	0.882
Pleural Effusion	**0.871**	0.856	0.865	0.864
Pleural Other	**0.570**	0.541	0.522	0.546
Pneumonia	**0.784**	0.754	0.728	0.772
Pneumothorax	**0.852**	0.816	0.798	0.835
Support Devices	0.863	0.837	**0.864**	0.846
Macro	**0.747**	0.727	0.710	0.738

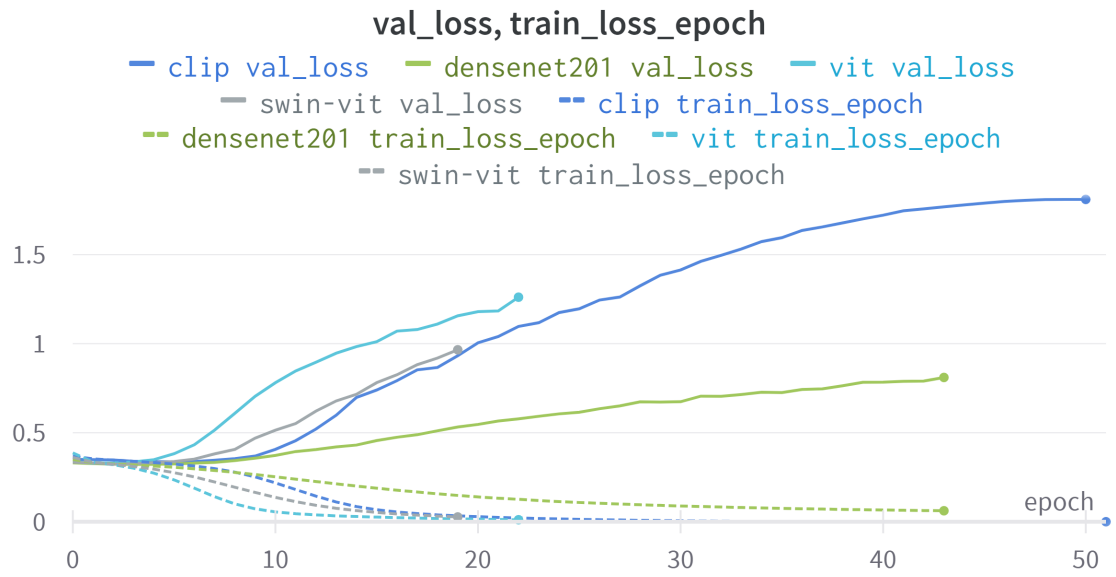

Figure 3.3: Train and validation losses of models pre-trained on ImageNet on CheXpert data.

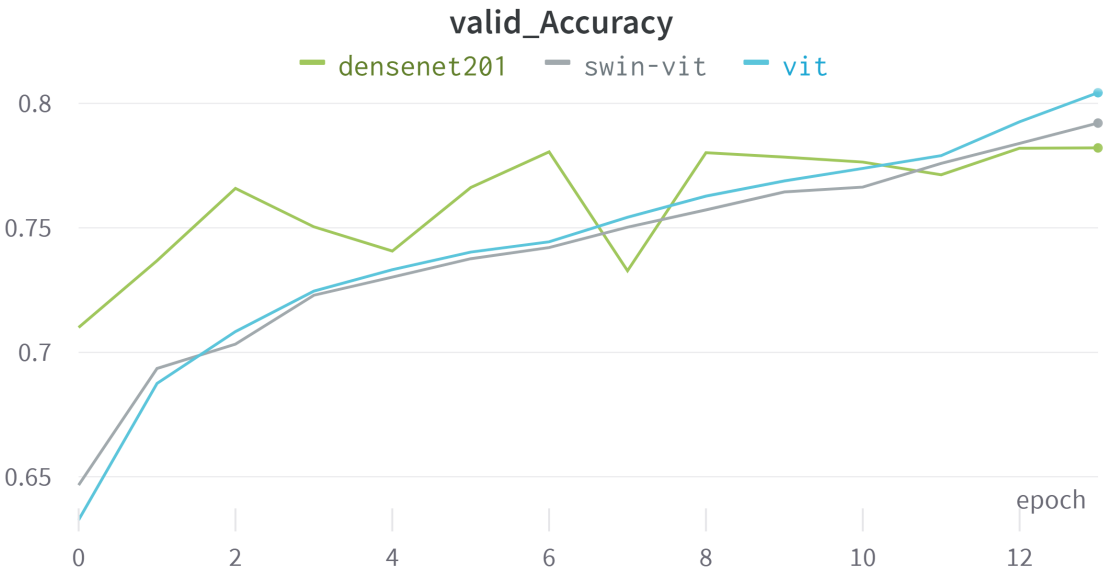

Figure 3.4: Accuracy on validation data during pre-training models on RadImageNet data.

3.6. pre-training models on RadImageNet

After observing the lower performance of models pre-trained on ImageNet and considering that Transformer-based models tend to perform better when trained on large datasets, I decided to utilize pre-training on RadImageNet. This dataset has a distribution that closely resembles CheXpert, making it more suitable for our purposes. The training of these pre-trained models revealed some intriguing behavior.

Figure 3.4 shows that the accuracy of Transformer-based models on the validation data steadily increases, displaying a smooth trend. On the other hand, DenseNet's accuracy exhibits some fluctuations. The Transformer-based models achieve a Top5 accuracy of approximately 0.98, while DenseNet achieves a Top5 accuracy of around 0.97. It is worth noting that in figures 3.4 and 3.5, the accuracy of the Transformer-based models does not appear to have reached a saturation point, indicating a potential for further improvement when allowing them to train longer.

3.7. Models pre-trained on RadImageNet

The performance of models pre-trained on RadImageNet and later fine-tuned on CheXpert is summarized in table 3.5. By analyzing the macro ROCAUC scores, we observe that utilizing

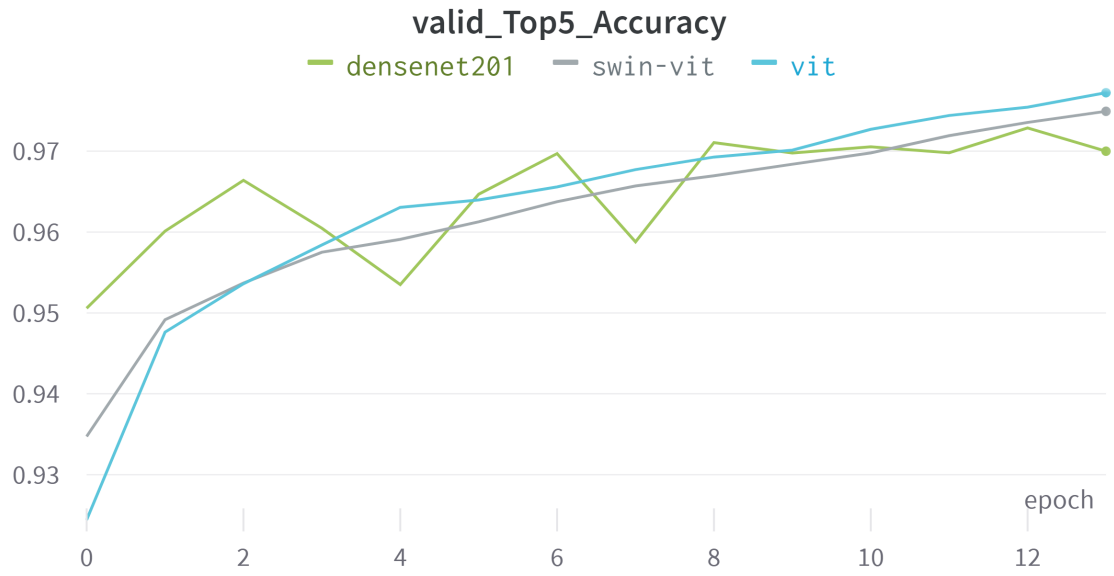

Figure 3.5: Top-5 accuracy on validation data during pre-training models on RadImageNet data.

models pre-trained on RadImageNet leads to the highest performance across all model types, which aligns with our expectations.

An interesting finding is that DenseNet continues to achieve the best performance among the models. However, it is important to note that this could be due to the potentially stopping to early the pre-training process for Transformer-based models, as indicated by the accuracy trends shown in figure 3.4. The accuracy of the Transformer models suggests that there is still room for improvement, implying that their performance may not have reached a saturation point.

3.8. Conclusions

The experiments conducted in this chapter demonstrate that, despite pre-training transformer-based models on large datasets with images from a similar domain, they do not outperform CNN architectures. It can be argued that the limited size of the RadImageNet dataset may not fully unlock the potential of transformer models. It is worth noting that Dosovitskiy's [Kol+21] paper demonstrated improved performance over CNNs when the pre-training dataset size exceeded 90 million images. Considering the ongoing advancements in transformer-based models for computer vision tasks, it is plausible that pre-training larger models on billions of medical images could enable these models to outperform CNN architectures significantly. Future research with larger-scale pre-training efforts may lead to substantial advancements in the performance of transformer models in the medical imaging domain.

Table 3.5: Maximal ROCAUC for each class for models pre-trained on RadImageNet.

Class	DenseNet	Swin-ViT	ViT
Atelectasis	**0.690**	0.680	0.678
Cardiomegaly	**0.840**	0.824	0.825
Consolidation	0.698	**0.707**	0.702
Edema	**0.839**	0.826	0.813
Enlarged Cardiom.	**0.635**	0.624	0.628
Fracture	0.775	**0.776**	0.762
Lung Lesion	**0.763**	0.740	0.738
Lung Opacity	**0.726**	0.712	0.704
No Finding	**0.875**	0.870	0.867
Pleural Effusion	**0.873**	0.855	0.848
Pleural Other	**0.729**	0.704	0.691
Pneumonia	**0.767**	0.737	0.714
Pneumothorax	**0.865**	0.828	0.811
Support Devices	**0.842**	0.814	0.791
Macro	**0.772**	0.758	0.749

4. Adversarial Attacks Using Feature Attribution Methods

In this chapter, I will address two additional hypotheses:

- It is harder to alter the explanations of models pre-trained on medical data.
- These three model types exhibit varying robustness to location loss.

To calculate the attribution methods, I utilized the Zennit Python library [And+21].

4.1. Evaluation of Model Explanations

To evaluate all models, I used the *mass accuracy* and *rank accuracy* metrics proposed by Arras [AOS20]. These metrics are calculated using a relevance map and a ground truth mask. Let us assume that we have a mask M that represents a region of interest selected by a human expert and an attribution map A. Without loss of generality, we can assume that M and A are one-dimensional arrays. If they are not, we can flatten them into one-dimensional arrays. Let $M_{\text{interest}} = \{i : M_i = 1\}$ be the set of all indices for which the mask M has a value of 1. The mass accuracy is then defined as:

$$\text{mass accuracy}(M, A) = \frac{\sum_{i \in M_{\text{interest}}} A_i}{\sum_i A_i}. \tag{4.1}$$

The mass accuracy measures the ratio of the sum of attributions A that lie within the region of interest M over the sum of all attributions across the entire input.

Let M sum up to k, and A_{top_k} be the set of indices that represent the k highest values in A. The rank accuracy is then defined as:

$$\text{rank accuracy}(M, A) = \frac{|\{i : i \in M_{\text{interest}} \text{ and } i \in A_{\text{top}_k}\}|}{k}. \tag{4.2}$$

The rank accuracy measures how many of the highest values of attributions A lie within the region of interest M.

Since our inputs have multiple channels C, I process the raw attributions by summing the non-negative attributions over the channels, as given below:

$$A_{\text{processed}} = \sum_{c=0}^{C-1} \max(A_c, 0). \tag{4.3}$$

Table 4.1: Attribution metrics calculated on CheXlocalize test set for models without adversarial attack.

Attribution method	Models training	Accuracy ↑	DenseNet	Swin-ViT	ViT
Saliency map	from-scratch	Mass	0.165	**0.180**	0.178
		Rank	**0.221**	0.209	0.192
	pre-trained	Mass	0.168	0.159	**0.173**
		Rank	**0.191**	0.174	0.175
Integrated Gradients	from-scratch	Mass	0.149	**0.173**	0.172
		Rank	0.171	**0.188**	0.168
	pre-trained	Mass	**0.164**	0.153	0.161
		Rank	**0.174**	0.163	0.160
LRP	from-scratch	Mass	0.099	**0.127**	0.097
		Rank	0.115	**0.149**	0.096
	pre-trained	Mass	**0.122**	0.119	0.081
		Rank	**0.122**	0.107	0.084
SmoothGrad	from-scratch	Mass	0.128	0.139	**0.163**
		Rank	0.170	0.166	**0.181**
	pre-trained	Mass	0.132	0.117	**0.151**
		Rank	0.160	0.133	**0.165**

Table 4.1 presents the metrics calculated for models pre-trained on RadImageNet and models trained from scratch on the CheXpert dataset.

4.2. Attacking models

I based my adversarial attacks on the concept of adding a location loss, which Heo introduced [HJM19]. The location loss L_{loc} is defined as:

$$L_{\text{loc}} = \left\| \frac{A - \min(A)}{\max(A) - \min(A)} - M \right\|^2 \tag{4.4}$$

The attribution values are normalized because attribution methods are not restricted, while masks take values of 0 and 1.

To perform the attacks, the models were fine-tuned to minimize a loss function that consisted of the sum of the location loss and the cross-entropy loss. Two types of attacks were performed:

- In Mask attack: In this attack, the mask created by the expert was used in the location

Table 4.2: Macro AUCROC calculated on the validation set of CheXlocalize after a full attack that consisted of 25 epochs fine-tuning.

Model training	Attribution method	Attack	DenseNet	ViT	Swin-ViT
From scratch	LRP	In mask	0.707	0.600	**0.778**
		Out mask	**0.762**	0.503	0.752
	Saliency Map	In mask	**0.723**	0.722	0.713
		Out mask	0.730	0.681	**0.744**
	SmoothGrad	In mask	0.646	**0.744**	0.724
		Out mask	0.616	0.663	**0.723**
	Integrated Gradients	In mask	**0.759**	0.742	0.741
		Out mask	0.605	0.679	**0.734**
pre-trained	LRP	In mask	**0.764**	0.611	0.719
		Out mask	**0.735**	0.490	0.697
	Saliency Map	In mask	**0.802**	0.686	0.705
		Out mask	**0.762**	0.690	0.756
	SmoothGrad	In mask	**0.793**	0.677	0.748
		Out mask	0.680	0.675	**0.704**
	Integrated Gradients	In mask	**0.749**	0.679	0.728
		Out mask	**0.732**	0.713	0.701

loss.

- Out Mask attack: In this attack, the inverse of the mask created by the expert was used in the location loss.

To fine-tune the models, I selected a single label at a time. The location loss was set to zero for examples that did not contain the specified label. The fine-tuning process was performed on models trained on the CheXpert training data, which were pre-trained on RadImageNet, as well as on models trained from scratch with three-channel inputs. This resulted in a total of 432 different scenarios (3 model types, 2 pre-training types, 2 attack types, 9 classes, and 4 attribution methods).

4.3. Analysis of effects on attacks

To investigate whether there are differences in the robustness of models and their pre-training to adversarial attacks, I conducted a linear regression analysis using the differences in rank and

Table 4.3: Relevance metrics results, calculated on CheXlocalize test set, grouped by base model training and attack type for the Saliency map attribution method.

Models training	Attack type	Accuracy ↑	DenseNet	Swin-ViT	ViT
from–scratch	Out Mask	Mass	0.164	**0.172**	0.148
		Rank	**0.226**	0.195	0.152
	No attack	Mass	0.165	**0.180**	0.178
		Rank	**0.221**	0.209	0.192
	In mask	Mass	0.200	**0.202**	0.187
		Rank	**0.257**	0.229	0.186
pre-trained	Out Mask	Mass	**0.182**	0.155	0.149
		Rank	**0.209**	0.166	0.159
	No attack	Mass	0.168	0.159	**0.173**
		Rank	**0.191**	0.174	0.175
	In mask	Mass	0.202	**0.210**	0.209
		Rank	**0.216**	0.208	0.203

Table 4.4: Relevance metrics results, calculated on CheXlocalize test set, grouped by base model training and attack type for the Integrated Gradients method.

Models training	Attack type	Accuracy	DenseNet	Swin-ViT	ViT
from–scratch	Out Mask	Mass	**0.177**	0.100	0.136
		Rank	**0.211**	0.104	0.146
	No attack	Mass	0.149	**0.173**	0.172
		Rank	0.171	**0.188**	0.168
	In mask	Mass	0.184	**0.213**	0.181
		Rank	0.210	**0.237**	0.177
pre-trained	Out Mask	Mass	**0.191**	0.108	0.114
		Rank	**0.195**	0.114	0.115
	No attack	Mass	**0.164**	0.153	0.161
		Rank	**0.174**	0.163	0.160
	In mask	Mass	**0.201**	0.199	0.180
		Rank	**0.195**	0.193	0.172

Table 4.5: Relevance metrics results, calculated on CheXlocalize test set, grouped by base model training and attack type for the LRP attribution method.

Models training	Attack type	Accuracy	DenseNet	Swin-ViT	ViT
from–scratch	Out Mask	Mass	**0.119**	0.046	0.060
		Rank	**0.123**	0.083	0.059
	No attack	Mass	0.099	**0.127**	0.097
		Rank	0.115	**0.149**	0.096
	In mask	Mass	**0.114**	0.065	0.082
		Rank	**0.120**	0.076	0.084
pre-trained	Out Mask	Mass	**0.135**	0.108	0.078
		Rank	**0.123**	0.106	0.082
	No attack	Mass	**0.122**	0.119	0.081
		Rank	**0.122**	0.107	0.084
	In mask	Mass	**0.119**	0.109	0.084
		Rank	**0.117**	0.109	0.085

Table 4.6: Relevance metrics results, calculated on CheXlocalize test set, grouped by base model training and attack type for the SmoothGrad attribution method.

Models training	Attack type	Accuracy	DenseNet	Swin-ViT	ViT
from–scratch	Out Mask	Mass	**0.153**	0.085	0.134
		Rank	**0.209**	0.084	0.145
	No attack	Mass	0.128	0.139	**0.163**
		Rank	0.170	0.166	**0.181**
	In mask	Mass	0.177	**0.213**	0.187
		Rank	0.224	**0.270**	0.197
pre-trained	Out Mask	Mass	**0.165**	0.096	0.143
		Rank	**0.195**	0.102	0.147
	No attack	Mass	**0.132**	0.117	0.151
		Rank	0.160	0.133	**0.165**
	In mask	Mass	**0.206**	0.169	0.205
		Rank	**0.237**	0.177	0.199

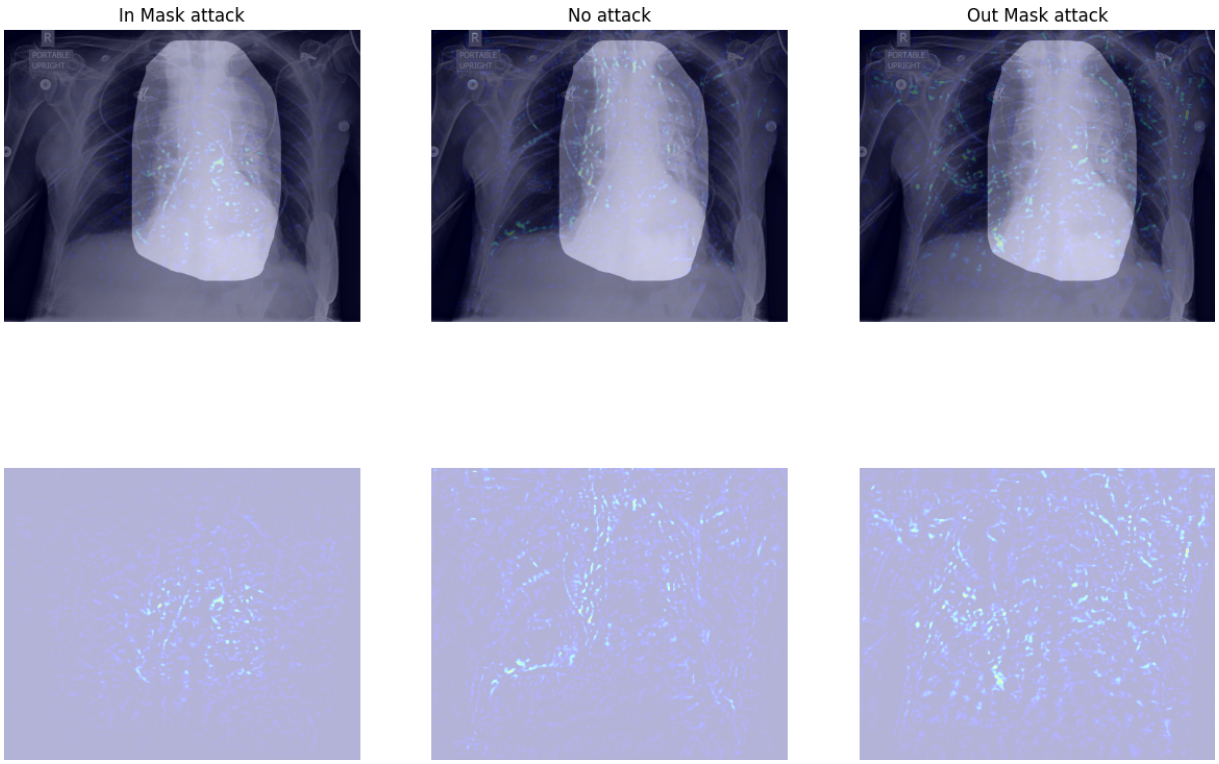

Figure 4.1: Attribution maps for three types of DenseNet: In Mask attacked, not attacked, Out Mask attacked. At the top row, you can see attributions plotted on top of the X-Ray and radiologist mask. At the bottom row, only attributions are plotted.

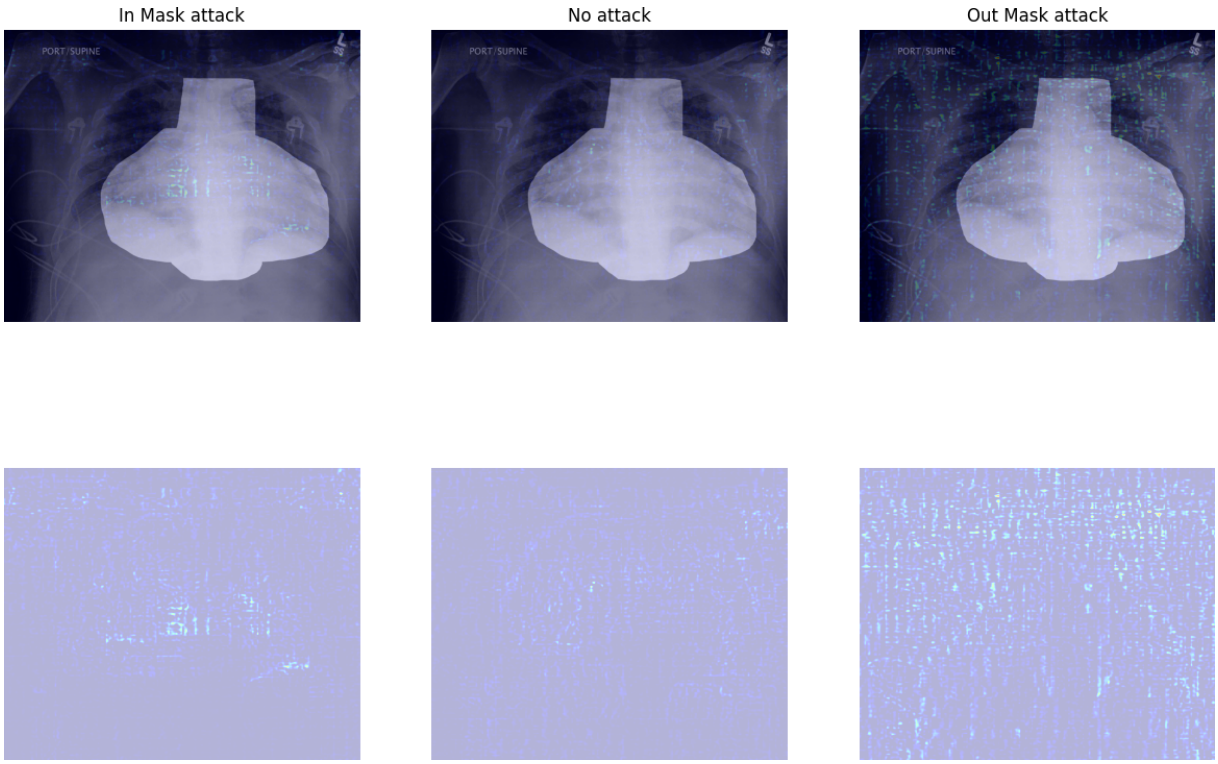

Figure 4.2: Attribution maps for three types of Swin-ViT: In Mask attacked, not attacked, Out Mask attacked. At the top row, you can see attributions plotted on top of the X-Ray and radiologist mask. At the bottom row, only attributions are plotted.

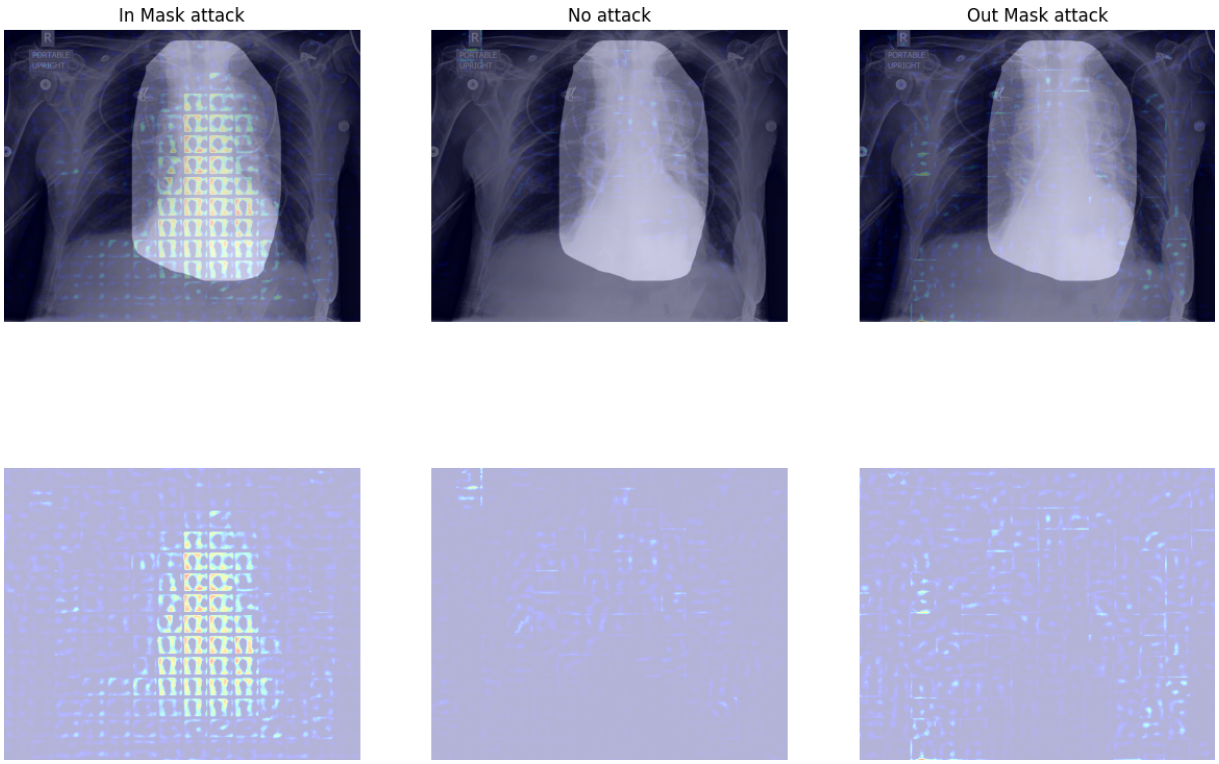

Figure 4.3: Attribution maps for three types of ViT: In Mask attacked, not attacked, Out Mask attacked. At the top row, you can see attributions plotted on top of the X-Ray and radiologist mask. At the bottom row, only attributions are plotted.

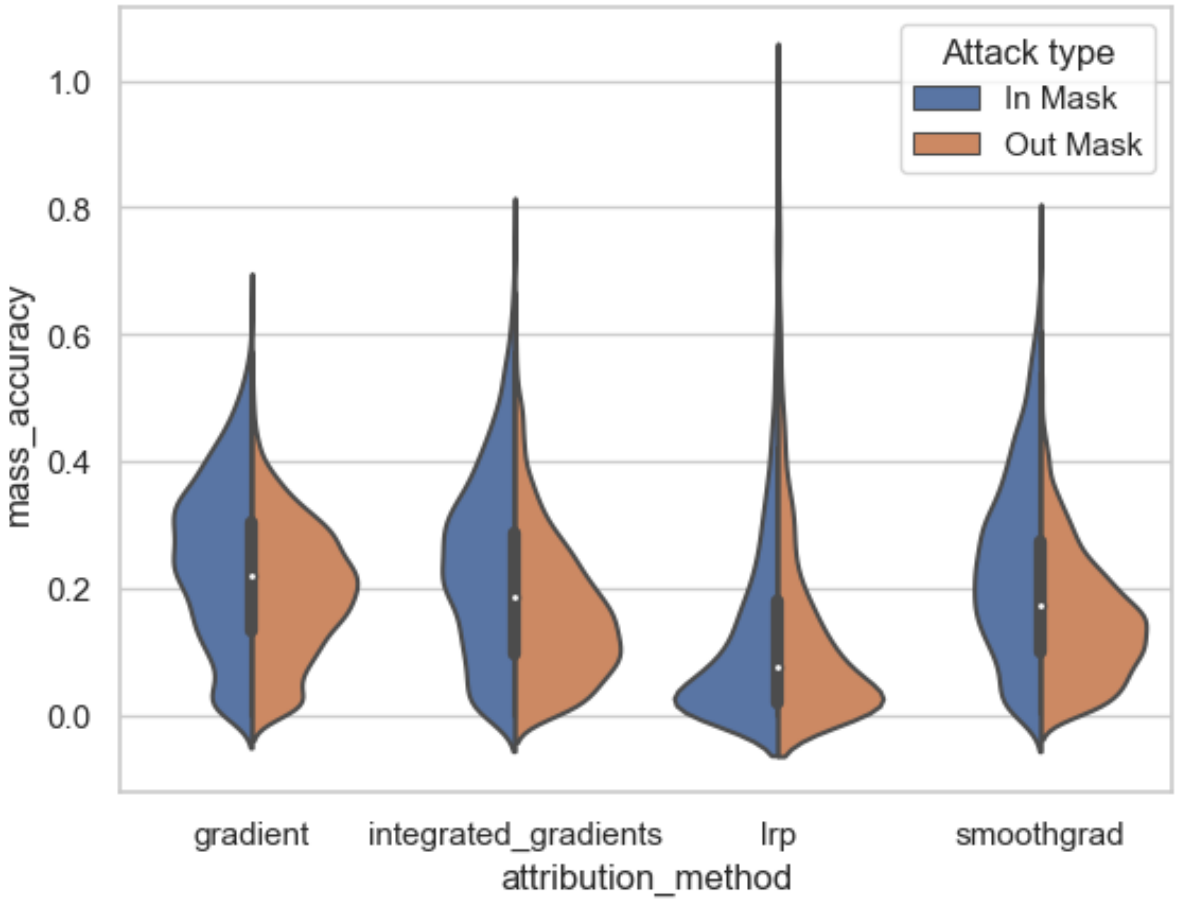

Figure 4.4: Violin plot of mass accuracy per attribution methods grouped by attack type.

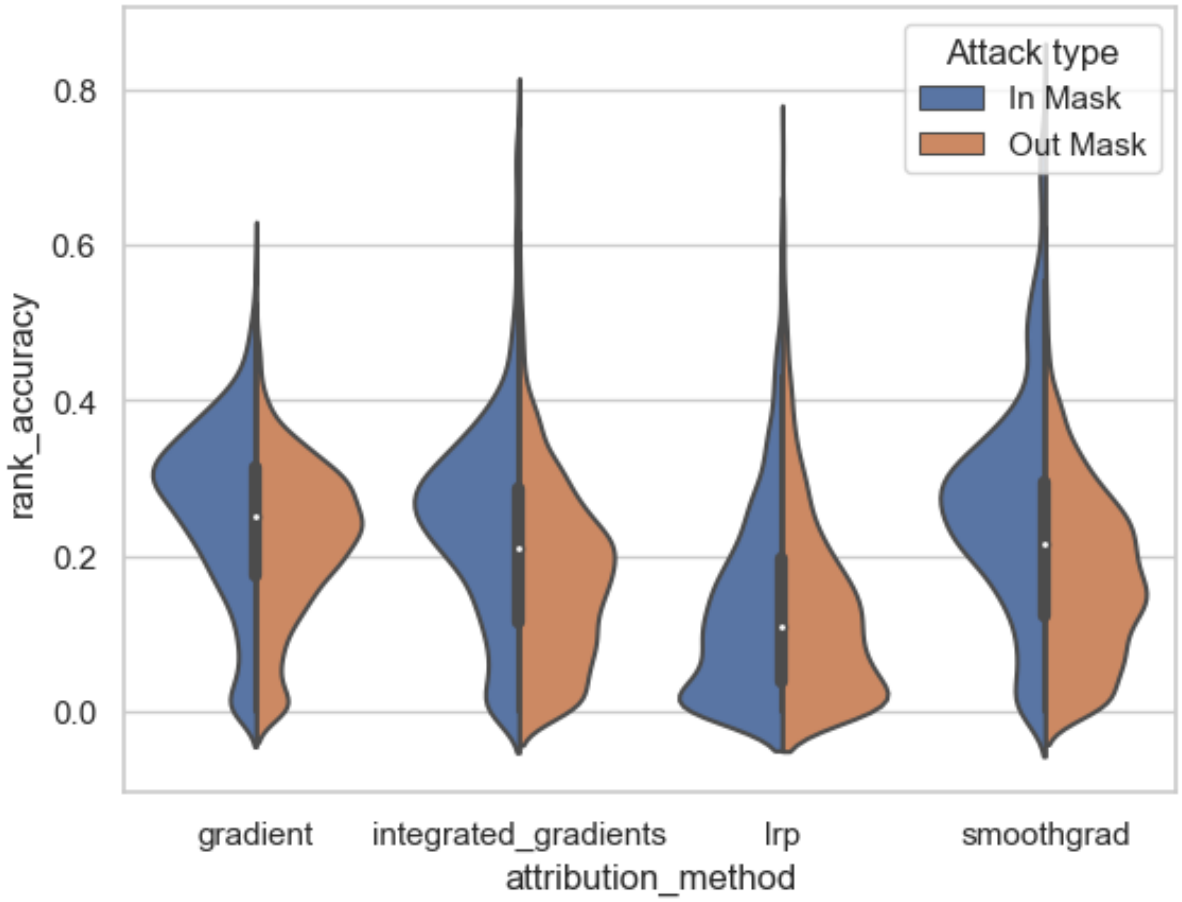

Figure 4.5: Violin plot of rank accuracy per attribution methods grouped by attack type.

Table 4.7: Linear regression coefficients fitted to the difference of mass accuracy between In Mask and Out Mask fine-tuning.

	Coef.	Std.Err.	P>|t|
Intercept	0.024	0.0027	8.1e-19
model[T.swinvit]	0.058	0.0026	5.4e-110
model[T.vit]	0.031	0.0025	5.2e-36
pretraining[T.pretrained]	-0.003	0.0021	1.5e-01
attribution_method[T.integrated_gradients]	0.026	0.0028	1.2e-19
attribution_method[T.lrp]	-0.044	0.0031	1.9e-44
attribution_method[T.smoothgrad]	0.038	0.0028	1.0e-39

mass accuracy metrics as the dependent variables. The model type, attribution method type, and pre-training type were used as regressors. The coefficients fitted to the models, along with the p-values from the Wald test, are presented in Tables 4.7 and 4.9. Additionally, the results of the ANOVA conducted on these linear models can be found in Tables 4.8 and 4.10.

Examining the coefficients, we observe that for mass accuracy, only the coefficient associated with the pre-training of the model is statistically insignificant, with a p-value of approximately 0.15. In contrast, all coefficients are significant for rank accuracy difference, with very small p-values (maximal p-value is equal to 1.6e-16). It's worth noting that ANOVA returned the p-value of pre-training being influential is below 0.05 for both linear models (the maximum from both linear models was 1.4e-6).

Analyzing the coefficients, we find that both Swin-ViT and ViT models have positive coefficients in both linear models. This suggests that the attacks had a greater impact on transformer-based models, providing insight into the second hypothesis proposed in the introduction of this thesis. Conversely, in terms of pre-training, both linear models yielded negative coefficients. This indicates that models pre-trained on RadImageNet were more robust to attacks compared to models trained from scratch. Figures 4.6 and 4.7 display the means of the differences grouped by the effects analyzed. The analysis of the coefficients and the plotted means aligns with the first hypothesis proposed in this chapter: it is more challenging to attack models pre-trained on RadImageNet. Furthermore, Wald's test p-values associated with the coefficients provided by the models confirm that these models indeed differ in their robustness to location loss, thus addressing the second hypothesis proposed at this chapter's beginning.

Table 4.8: Result of ANOVA used on mass accuracy difference model.

	df	sum_sq	mean_sq	F	PR(>F)
model	2.0	8.44	4.22	324.9	3.6e-138
pretraining	1.0	0.30	0.30	23.30	1.4e-06
attribution_method	3.0	10.2	3.41	262.38	5.7e-165
Residual	12059.0	156.62	0.013		

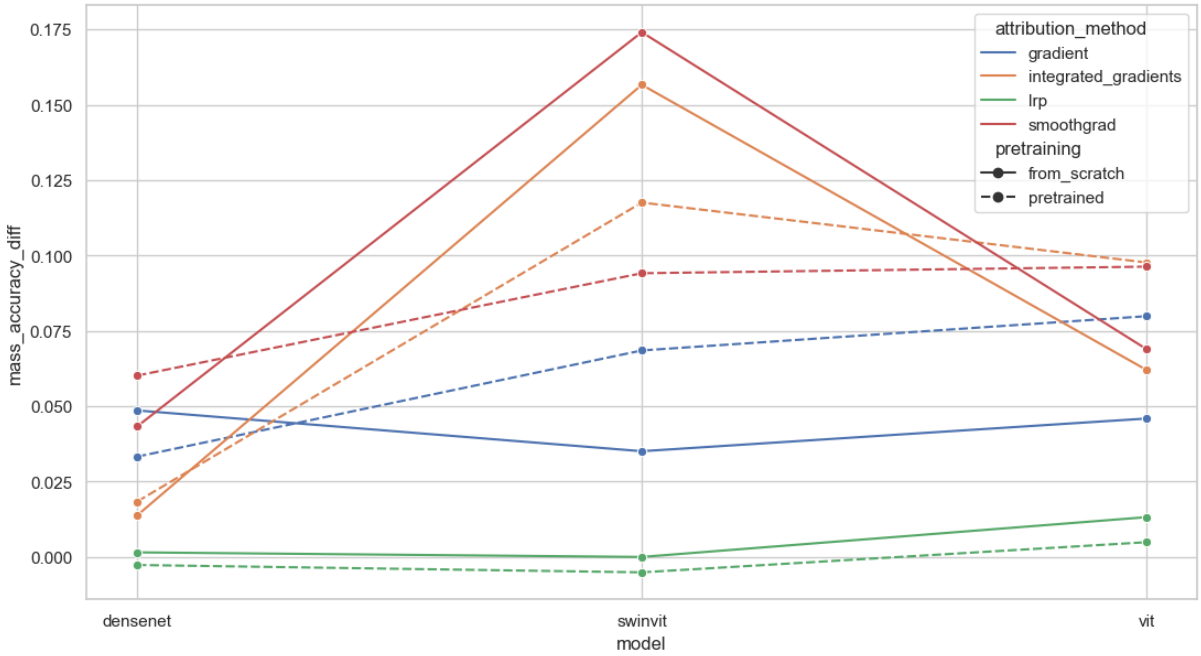

Figure 4.6: Means of differences of mass accuracy grouped by attribution method, model, and pre-training type.

Table 4.9: Linear regression coefficients fitted to the difference of rank accuracy between In Mask and Out Mask fine-tuning.

	Coef.	Std.Err.	P>|t|
Intercept	0.021	0.0023	4.0e-19
model[T.swinvit]	0.072	0.0022	1.0e-221
model[T.vit]	0.025	0.0021	3.8e-31
pretraining[T.pretrained]	-0.019	0.0018	1.6e-26
attribution_method[T.integrated_gradients]	0.029	0.0024	3.4e-32
attribution_method[T.lrp]	-0.030	0.0027	2.2e-29
attribution_method[T.smoothgrad]	0.052	0.0025	1.1e-97

Table 4.10: Result of ANOVA used on rank accuracy difference model.

	df	sum_sq	mean_sq	F	PR(>F)
model	2.0	12.1	6.06	622.7	4.2e-258
pretraining	1.0	1.97	1.97	202.6	1.32e-45
attribution_method	3.0	10.6	3.52	361.4	9.5e-225
Residual	12059.0	117.4	0.0097		

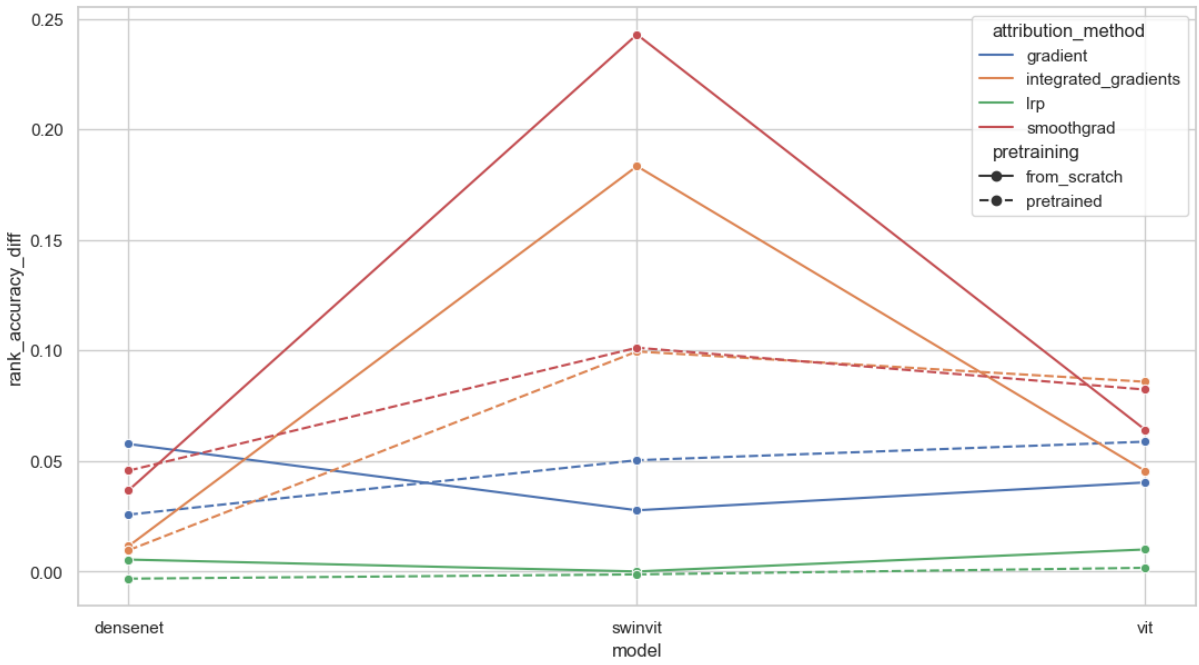

Figure 4.7: Means of differences of rank accuracy grouped by attribution method, model, and pre-training type.

4.4. Conclusions

This chapter evaluated the model's ability to accurately identify regions of interest using various attribution methods. Rank and mass accuracy metrics were calculated based on these attributions and masks to assess the models' localization performance.

Furthermore, the robustness of the models to adversarial attacks on their explanations was investigated. In mask and Out mask attacks were conducted by altering the model's explanations to focus on the radiologist's region of interest or everything outside of it, respectively. The attacked models' location metrics were then recalculated to evaluate their localization performance. The difference in localization performance between the In mask and Out mask attacks was analyzed to determine how well we were able to attack these models. The consistent coefficients for both metrics indicate that Transformer-based models are less robust to these attacks, while models pre-trained on RadImageNet exhibit greater robustness.

It's worth discussing whether using such a metric (difference in In and Out mask attack) makes sense. In my opinion, this metric allows us to check how far apart we are able to pull the models' explanations without changing their performance much.

It's also worth noting that basic LRP (with ϵ rule) for transformer-based models led to numerical instability resulting in having NaNs as the output of this attribution method. These examples were omitted from the analysis.

Summary

This thesis focused on evaluating DenseNet, ViT, and Swin-ViT models in the context of chest X-Ray images. I tested these models' performance and their attributions. Experiments led me to the conclusion that transformer-based models underperform CNNs in this scenario. Also, they are less robust to attacks that focus on altering their explanations.

Further research in this area can be conducted. Firstly, RadImageNet is a "small" dataset compared to current state-of-the-art datasets used for pre-training large language and vision models, so creating a bigger pre-training dataset could lead to transformer-based models over-performing CNNs in the CheXpert dataset. It could be hard to create a much bigger dataset with labels related to medical images, but many unsupervised methods are emerging for pre-training vision models without labels. So the problem of labels could possibly be overcome with such a pre-training. It's also worth checking bigger transformer and CNN models. One could also try to train all these models in higher resolution.

When it comes to attacks, one could try to perform them with different attribution methods, maybe model-specific ones, but this wouldn't allow performing a comparison of transformer-based models with CNNs. It would also be interesting to see how these attacks would look if we had another dataset, even a dataset with "normal" images. Because it's hard to have a classification dataset with regions of interest created for each label, one could use any segmentation dataset for the model training in multi-label classification tasks and later use segmentation masks as regions of interest while performing such an attack.

Bibliography

[And+21] Christopher J. Anders et al. "Software for Dataset-wide XAI: From Local Explanations to Global Insights with Zennit, CoRelAy, and ViRelAy". In: *CoRR* abs/2106.13200 (2021).

[AOS20] Leila Arras, Ahmed Osman, and Wojciech Samek. "Ground truth evaluation of neural network explanations with clevr-xai". In: *arXiv preprint arXiv:2003.07258* (2020).

[Bac+15] Sebastian Bach et al. "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation". In: *PLOS ONE* (2015).

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* abs/1409.0473 (2014).

[Bie20] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: `https://www.wandb.com/`.

[Bin+16] Alexander Binder et al. "Layer-Wise Relevance Propagation for Neural Networks with Local Renormalization Layers". In: *Springer International Publishing eBooks* (2016).

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: `1607.06450 [stat.ML]`.

[Den+09] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255.

[Dev+19] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: `10.18653/v1/N19-1423`. URL: `https://aclanthology.org/N19-1423`.

[Ft19] William Falcon and The PyTorch Lightning team. *PyTorch Lightning*. Mar. 2019. DOI: `10.5281/zenodo.3828935`. URL: `https://www.pytorchlightning.ai`.

[He+16] K. He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.90.

[HJM19] Juyeon Heo, Sunghwan Joo, and Taesup Moon. “Fooling Neural Network Interpretations via Adversarial Model Manipulation”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 32. 2019, pp. 2925–2936.

[Hua+17] Gao Huang et al. “Densely Connected Convolutional Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2261–2269.

[Irv+19] Jeremy Irvin et al. “Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 590–597.

[Kol+21] Alexander Kolesnikov et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: 2021.

[Kri09] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. In: 2009.

[Lec+98] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[LH17a] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *International Conference on Learning Representations*. 2017.

[LH17b] Ilya Loshchilov and Frank Hutter. *SGDR: Stochastic Gradient Descent with Warm Restarts*. 2017. arXiv: 1608.03983 [cs.LG].

[Liu+21] Ze Liu et al. “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 9992–10002.

[Mei+0] Xueyan Mei et al. “RadImageNet: An Open Radiologic Deep Learning Research Dataset for Effective Transfer Learning”. In: *Radiology: Artificial Intelligence* 0.ja (0), e210315. DOI: 10.1148/ryai.210315. eprint: https://doi.org/10.1148/ryai.210315. URL: https://doi.org/10.1148/ryai.210315.

[Mon+19] Grégoire Montavon et al. “Layer-Wise Relevance Propagation: An Overview”. In: Sept. 2019, pp. 193–209. ISBN: 978-3-030-28953-9. DOI: 10.1007/978-3-030-28954-6_10.

[Mur22] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. URL: probml.ai.

[Pas+19] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[Rad+19] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: 2019.

[Rad+21] Alec Radford et al. "Learning Transferable Visual Models From Natural Language Supervision". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8748–8763. URL: https://proceedings.mlr.press/v139/radford21a.html.

[RSG16] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 97–101. DOI: 10.18653/v1/N16-3020. URL: https://aclanthology.org/N16-3020.

[Sap+22] Adriel Saporta et al. "Benchmarking saliency methods for chest X-ray interpretation". In: *Nature Machine Intelligence* (2022).

[Shr+17] Avanti Shrikumar et al. *Not Just a Black Box: Learning Important Features Through Propagating Activation Differences*. 2017. arXiv: 1605.01713 [cs.LG].

[Smi+17] Daniel Smilkov et al. *SmoothGrad: removing noise by adding noise*. 2017. arXiv: 1706.03825 [cs.LG].

[STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *International conference on machine learning*. PMLR. 2017, pp. 3319–3328.

[Sun+17] Chen Sun et al. "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 843–852.

[SVZ13] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”. In: *CoRR* abs/1312.6034 (2013).

[Tou+21] Hugo Touvron et al. “Training data-efficient image transformers amp; distillation through attention”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 10347–10357. URL: https://proceedings.mlr.press/v139/touvron21a.html.

[Vas+17] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[WP19] Sarah Wiegreffe and Yuval Pinter. “Attention is not not Explanation”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 11–20. URL: https://aclanthology.org/D19-1002.

[Zhu+21] Liu Zhuang et al. “A Robustly Optimized BERT Pre-training Approach with Post-training”. English. In: *Proceedings of the 20th Chinese National Conference on Computational Linguistics*. Huhhot, China: Chinese Information Processing Society of China, Aug. 2021, pp. 1218–1227. URL: https://aclanthology.org/2021.ccl-1.108.

List of symbols and abbreviations

ANOVA	analysis of variance
AUC	area under curve
CLIP	Contrastive Language-Image Pre-training
CNN	Convolutional neural network
DenseNet	Densely Connected Convolutional Network
IG	Integrated Gradients
LRP	Layer-wise relevance propagation
ROC	Receiver operating characteristic
ViT	Vision transformer
$\|\| \cdot \|\|$	L^2 norm

List of Figures

List of Tables