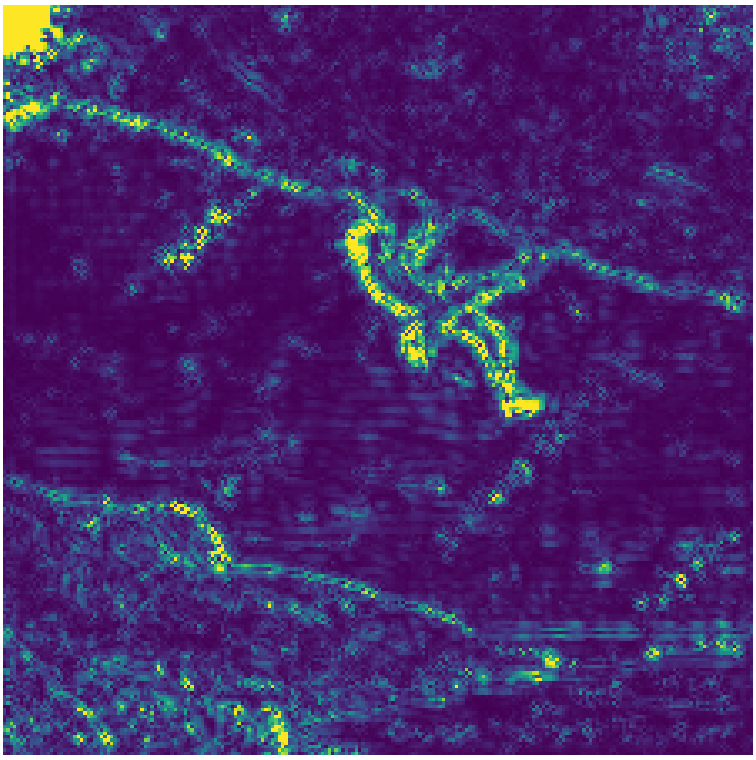
CHALMERS



# Distinguishing Real from AI-Generated Images

A Study on Model Performance and Visualisation Techniques

Andersson. J, Danckwardt. S, Gurusinghe. D, Lauri. P, Lindberg. O, Sjödahl. M

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2024

www.chalmers.se

Bachelor's thesis 2024

# Distinguishing Real from AI-Generated Images

A Study on Model Performance and Visualisation Techniques

Andersson. J, Danckwardt. S, Gurusinghe. D, Lauri. P,
Lindberg. O, Sjödahl. M


CHALMERS

Department of Electrical Engineering
*Division of Signal Processing*
Project Group EENX16-VT24-31
Chalmers University of Technology
Gothenburg, Sweden 2024

Distinguishing Real from AI-Generated Images
A Study on Model Performance and Visualisation Techniques
Andersson. J, Danckwardt. S, Gurusinghe. D, Lauri. P,
Lindberg. O, Sjödahl. M



Supervisor: Fredrik Kahl, E2
Examiner: Ida Häggström, E2

Bachelor's Thesis 2024
Department of Electrical Engineering
Division of Signal Processing
Project Group EENX16-VT24-31
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Guided backpropagation output on an image of a climber in a steep overhang

Typeset in LaTeX
Printed by Chalmers Library
Gothenburg, Sweden 2024

# Abstract

With the rapid advancement of Artificial Intelligence (AI), the capability to generate highly realistic images has emerged, creating social and ethical challenges. This study aims to construct neural network-based classification models of three different architectural types to distinguish AI-generated images from real ones. The three architectures used are Convolutional Neural Network (CNN), Fully Convolutional Network (FCN), and Vision Transformer (ViT). These models, which take images as input and output a classification score between 0 and 1, were trained on the GenImage dataset and tested on various datasets to evaluate their performance. The CNN and FCN model performs best on the GenImage dataset with and accuracy of 99.9% and 95.6% respectively, while the ViT model gets moderate success with an accuracy of 62.4% when handling images from unseen more modern generators. Additionally, visualisation techniques such as Gradient Class Activation Mapping (grad-CAM), Guided Backpropagation (GBP), and Attention Heatmaps provided insights into what parts of an image the models focus on for classification. The visualisations provide useful data to show what parts of images are more complex, but show no clear pattern that makes it easy to determine what distinguishes an AI image from a real one. To make these models accessible to the public, a web application was developed, allowing users to upload images and receive a classification score and useful visualisations of model activity.



Keywords: AI, FCN, CNN, ViT, Generative, Classification, Visualisation

# List of Acronyms

| | |
|---|---|
| AI | Artificial Intelligence |
| CNN | Convolutional Neural Network |
| GAN | Generative Adversarial Network |
| ViT | Vision Transformer |
| FCN | Fully Convolutional Network |
| CAM | Class Activation Map |
| grad-CAM | Gradient Class Activation Map |
| SD1.4 | Stable Diffusion version 1.4 |
| GBP | Guided Backpropagation |

# Contents

# List of Figures

# List of Tables

# 1
# Background

Artificial Intelligence, or AI, has advanced dramatically in the last few years. Currently, it is capable of many things, from generating complete essays to being able to synthesize voices similar to people in the real world. One of these new capabilities of AI is the capability to create generative images or so-called "AI- images". Due to the nature of these images being almost indistinguishable from man-made images, there is an increasing threat that they may be used to spread misinformation in the form of fake news [5] or political slander [6].
The social issues revolving around AI-generated images are connected to misinformation and fields such as photography and art. Creators of traditional art voice their concern that the increasing volume of AI-generated art could lead to the homogenization of photography and art, reducing the originality and diversity within the practice [7]. Moreover, if the amount of AI-generated art desaturates the existing pool of human-made art, the audience may find it hard to differentiate between original art pieces and AI-generated art. The lack of differentiation and the ease of producing these AI images could negatively affect the art and photography market since availability and supply would vastly outweigh the demand. This would cause traditional and original artwork's cultural and economic value to decrease [7].
Besides the social implications of generative models, there is also the Non-consensual Use of Likeness, which refers to the ethical issue of how AI images force people to relinquish the control and ownership of individual likeness [8]. With this comes an opportunity to, for example, use fraudulent advertisements to promote goods through AI-generated images and deepfakes of celebrities, government officials, and political leaders [7]. Furthermore, the ethics of deepfakes and whether they should be permitted circles back to the spread of misinformation and the arguably worst application of AI images, which is for military use. The creation of distrust and civil unrest within another nation with the intention to destabilize the region could quickly be done through inexpensive and readily available generated images.
With the issues presented in mind, it has become apparent that finding techniques to determine the authenticity of images could be very beneficial, as they could minimize the misuse and harm caused by this type of technology.

## 1.1 Related Work

Several solutions for AI image detection have been proposed recently [9], with a high-quality image dataset limiting their performance [10]. GenImage [10] attempts to solve the lack of quality data with its proposed massive dataset of real and AI-

generated images, then trains a single architecture model (CNN, see Section 2) on each image-generator individually. But does not attempt to train a generalised network architecture to detect AI-generated images. This paper uses the dataset proposed by GenImage to create several generalised state-of-the-art models based on different architectures for AI detection and visualises where the models choose to focus. This is to better understand the progress in making a generalised model to detect AI images in as-of-yet-unseen image generators and to demystify the neural networks' image detection process.

## 1.2 Aim of Study

This study aims to construct neural network-based classification models from three different architectural structures: Convolutional Neural Network (CNN), Fully Convolutional Network (FCN), and Vision Transformer (ViT), further explained in Section 2.
Each classification model constructed will take an image as an input and generate a decimal value between 0 and 1 as the output, classifying the input as a man-made image (closer to 1) or an AI-generated image (closer to 0). In addition to the binary image classification of either man-made or AI-generated, the FCN will be used for individual pixel classification on the whole image. This will ensure that partially AI-generated images will get a more accurate visual representation of which parts have been edited. The intention is also to visualize what led to the model's decision as clearly as possible. The idea is that alongside providing as reliable a classification result as possible to the user, we also want to highlight what parts of the image seem important to each model, aiming to empower human users to make well-informed decisions.
Additionally, a web application will be implemented to utilize the optimized models' functionalities and make them available and useful for the public.

## 1.3 Limitations

The construction of several models during a short time frame with limited computational resources has put some restrictions on the project.

### 1.3.1 Dataset

The creation of an entirely new dataset containing hundreds of thousands of images in order to give a satisfying result would take too much focus off the main aim of creating models for classification and visualisation. Due to limited time and resources, we have not constructed the main image dataset for training and testing on our own, instead relying primarily on the GenImage dataset [10]. The contents of which are clarified in Section 3.1.
However, we have constructed much smaller auxiliary data sets to complement GenImage. More on this in Section 3.1.

### 1.3.2 Computational Resources

Model training has been done on personal computers, mainly on two personal computers, PC1 and PC2, housing an Nvidia RTX 3080 and an Nvidia RTX 4080 graphics card, respectively. Whilst powerful enough to get good results, the training time on these machines is too long to replicate the findings in the paper from the GenImage team [10], with specific models for each individual generative model in the dataset. Thus, the project only involves a generalized and a singular specific model, trained on Stable Diffusion V1.4 (SD1.4) for each of the three model types used. This generative model was chosen due to having the best results according to the GenImage research team [10].

### 1.3.3 Further development

For the classification models to remain valid, they would have to be retrained on newer generative models whenever they are released. Otherwise, the risk is that newer generative models differ from older ones to the point that it negatively impacts model performance. The result proves this hypothesis, with models having a weak performance on images from the generator Dall-e 3 [11], which was released during the project. However, retraining is not within the project's scope, and users are advised to use the tool with caution if a long time has passed since its release.

## 1.4 Social and Ethical Aspects

Even though there is an argument to be made that developing a tool that accurately detects AI involvement in imagery would be ethically positive, as mentioned in the introduction, we must also limit the possibility that the development of such a tool doesn't cause harm. We must ensure all images used for training do not infringe on copyright law or any individual's integrity. Something that poses a problem when working with datasets consisting of millions of images, such as GenImage. However, we trust the methodology used when creating the GenImage dataset, as described in the paper "GenImage: A Million-Scale Benchmark for Detecting AI-Generated Image" [10] is sufficiently rigorous and does not cause any ethical problems. When constructing our own additional datasets, described more in detail in Section 3.1, we have taken care not to infringe on these ideals either.
It is also important to clarify the limitations of the models as clearly as possible to avoid a scenario where our tool is used to support a false attack on artists for using AI-generated imagery. We do not support any such use of our classification tool. We are simply interested in the possibility of empowering individuals to make their own assessments of an image's validity with higher certainty.
Recently, the issues with AI as an environmental hazard have garnered attention among the general public. For this project, we determined that our computational resources do not consume enough energy to have a considerable impact. To support this claim, we disclose the estimated energy consumption for creating each model in Section 4.5.

# 2
# Technical background

Understanding the technical foundation of network architecture and visualisation of network activity is essential for understanding the project's result. The technical background provides a short summary of the most important topics.

## 2.1 Neural Networks

Neural networks (NNs) are inspired by the biological neural networks in human brains [12]. They consist of layers of connected nodes or weighted functions called neurons [13]. Each layer is designed to process specific types of information. The fundamental components include the input, hidden, and output layers.
The input layer is where the network receives its data [13]. Each neuron in this layer represents one unit of input data information. To further illustrate this, if the network was designed to decide if a handwritten digit is either a 1, 9, or any other digit in between and was given an image of the handwritten digit. The input layer could be designed to have one neuron for every pixel in the input image.
The hidden layers perform most of the computation through a network of neurons that process inputs from the previous layer. Each neuron in a hidden layer transforms its inputs into something the new layer can use. This is done by receiving an input from the previous layer and then applying a constant, called a "weight", and then a function, referred to as an "activation function", which is the resulting output of that node. The complexity of an NN is partly determined by the number of hidden layers and the neurons within them.
The output layer is the final layer that produces the network's output. The structure of this layer depends on the task the network is designed to perform. In the previous example, with a network deciding what digit a handwritten digit is, the output layer would contain ten neurons [12]. One for each different outcome. If the network concludes that the handwritten digit is a five, the neuron representing a five will fire up.
When training the network, the most important component is the weights. The weights are located in the connections of the neurons and can be compared to a nob adjusting the volume on a stereo. When a Neuron fires up, it will likely result in a connected neuron being fired up, and how much this connected neuron will fire up is decided by the weight of this specific connection.
Training an NN involves adjusting all the weights within the network to give the desired output when a specific input is given. To achieve this, a method called back-propagation is used. The method operates by initially giving the weights a

random value and then inputting a labelled example and passing it through the entire network, called a "forward pass". This will give an output that will either match the label, which is a positive example or be different, which is a negative example. After inputting a batch of labelled examples, a loss function gives a value based on the number of positive and negative examples. Using this loss function, a gradient of all weights can be calculated by going backwards and calculating the chain rule to a specific weight called back-propagation. Calculating the gradient and then changing the weights according to the gradient can minimize the loss function, which will generally lead to increased accuracy [13]. While back-propagation is the base method used to train NNs, many variations of this method have been developed, which can train networks faster and better. Some of these methods are stochastic gradient descent and Adaptive moment estimation(ADAM) [14].

## 2.2 Convolutional Networks

Convolutional Neural Networks (CNNs) are a category of deep neural networks that are highly effective for tasks like image recognition, image classification, and object detection. They mimic how human vision analyzes visuals, focusing on local patterns and structures [15]. They are named after a principle from linear algebra, the convolution operation.

A typical CNN architecture consists of several types of layers. The layers in which the convolution operand is used are called convolutional layers. These layers are the backbone of CNNs. A convolutional layer takes input as a matrix representing pixels in an image. For each pixel, a function called a filter or a kernel is applied that takes the RGB values of the pixel and its surrounding pixels to compute a new value for that pixel. When all this is done for every pixel, the output becomes much like the input, a two-dimensional matrix [16].

The depth of the CNN is defined by the number of layers in the network. Each layer typically consists of multiple filters (kernels) that extract features from the input data, creating a feature map for each as output. Deeper into the network, the layers capture increasingly abstract and complex features. The width of the CNN is determined by the number of filters (kernels) in each layer. Each filter learns to detect different patterns or features from the input image, such as edges or textures. With a higher width, layers can capture more diverse features, but it also increases the computational cost of the network. The dimensions of the layers are set by the dimension of the input, the number of kernels applied in each layer, and the size of the kernels [17]. Small kernels may only use the values of the adjacent pixels, one pixel in each direction, creating a 3x3 kernel. Still, there are also larger kernels that will calculate the value inputs in a larger circumference.

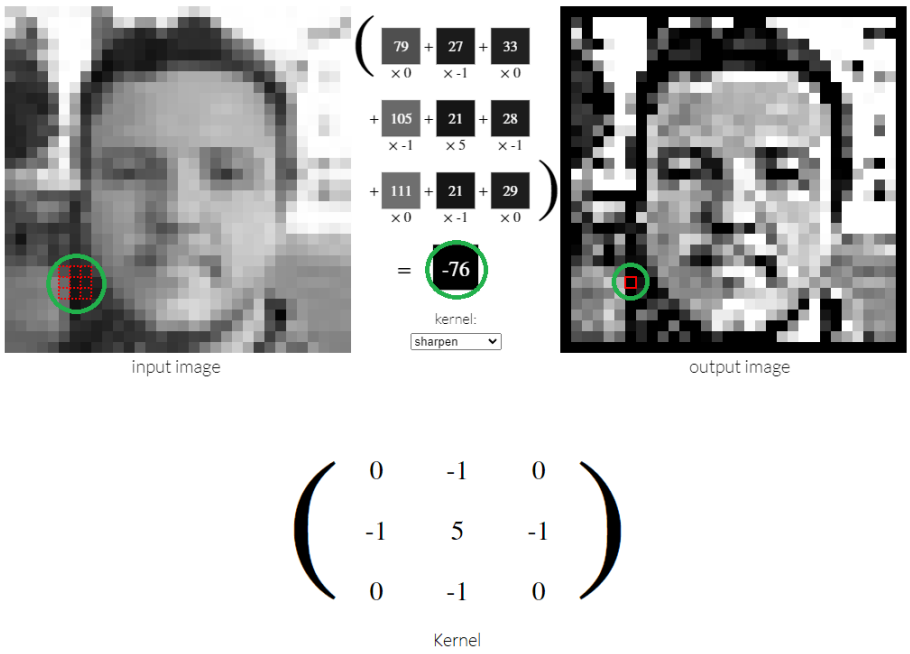



**Figure 2.1:** Visualisation of a kernel being applied to an image [1]

The second type of layer is called the pooling layer. The purpose of this layer is to reduce the size of spatial dimensions such as height and width of images that, in turn, reduces computational efforts [16, 18]. The technique of max pooling, choosing the largest value in a filter region, is illustrated in Figure 2.2. In contrast to this illustration, the filter of a pooling layer has a set width and height, which will "slide" across a given image and summarize the most essential features of the image. This will result in a smaller and more condensed map of the image with a focus on essential features [16]. There are also other pooling layers, such as average pooling, which calculates the average value of a filter, and global average pooling, which calculates the average of an entire feature map.

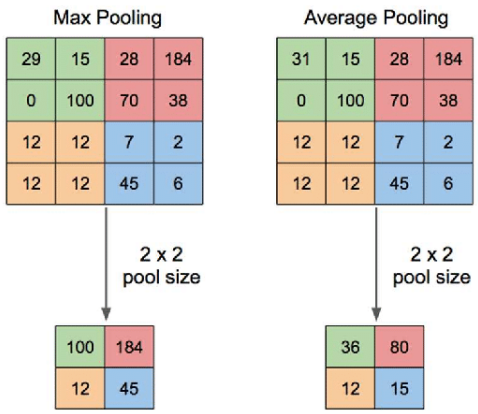



**Figure 2.2:** Illustration of Average and Max pooling[2].

A problem that can occur when training deep neural networks is that the network eventually stops improving no matter how long it is trained. This can be caused

by the "vanishing gradient problem", which essentially states that back-propagation stops working well when there are too many layers. A way to solve this problem is to add a "Skip Connection", which gives information from the previous layers to the later layer without processing the information. A convolutional network with "Skip Connections" adds the previous layer's output to the current layer. This is called a Residual Network or "ResNet". The "ResNet", compared to traditional convolutional networks, can have a substantially larger amount of layers, leading to better results [19].

## 2.3 Fully Convolutional Networks

Fully Convolutional Neural Networks (FCNs) were a solution Jonathan Long et al. proposed to perform image segmentation [20]. The FCN architecture is similar to the CNN architecture, with convolutional and pooling layers. However, instead of eventually flattening the images and feeding those values as input to a fully connected network, FCN models utilize a deconvolutional layer to upscale the output back to the size of the input image [20].
The layer's name is somewhat confusing since it doesn't perform the mathematical operation of deconvolution. Performing a deconvolution wouldn't be desirable since you would get the input back. Instead, deconvolutional layers utilize transposed convolutions, with the idea of inserting zeros between the output matrix values to increase its size and then convolving the upscaled output with a kernel [20]. This operation is illustrated in Figure 2.3.

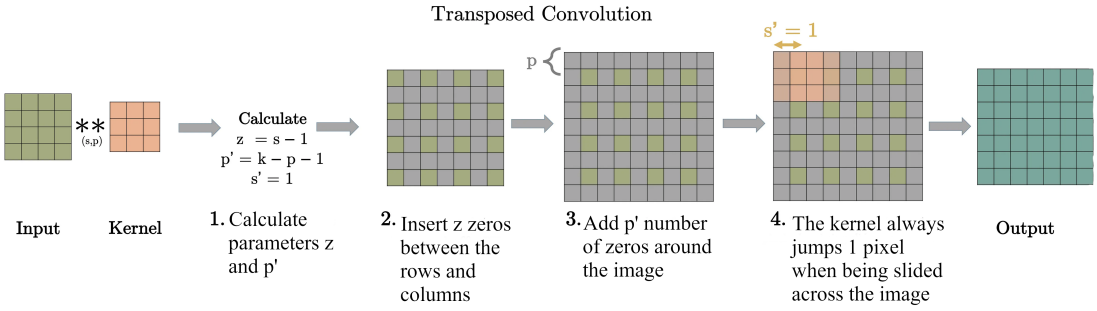



**Figure 2.3:** Illustration of the deconvolutional layer[3]

One of the major advantages of FCN models compared to CNN models is that since there is no fully connected component, they allow arbitrary input size [20]. In contrast, a CNN requires inputs of a specific size. This effectively eliminates the need for image preprocessing when using an FCN model.

## 2.4 Vision Transformers

The Vision Transformer proposed by Alexey Dosovitskiy et al. [21], used in this project, relies on the original Transformer component, which has had successful

results in natural language processing [21]. The Transformer model architecture is entirely dependent on self-attention. Self-attention is an attention mechanism that can capture positional dependencies in a single specific sequence and, using those dependencies, compute a representation of that specific sequence [22].
Transformers used for "Natural Language Processing" take sentences as inputs and convert them into so-called "Embeddings", which is done by taking each word in the sentence and as a "token". Each token is then converted into a vector representing that specific token. The vectors generated by this process are called "Embeddings" [22]. These embeddings are then used as input for the rest of the network. The Vision Transformer, which has images as inputs, can similarly construct Embeddings by splitting the image into patches of varying sizes, such as 3x3 or 4x4 and then taking each patch as a token. Embeddings representing the image can be constructed and fed into the rest of the network [21] using a similar process as the original transformer.
An attention function maps a Query, Key and Value pair to an output. The Query is used as the context in the attention function and is computed by multiplying the embedding vector from each token with a matrix with learned parameters. This creates a Query for each token. The same procedure is used to compute the Keys. However, the matrix of learned parameters differs. The Query and Key match each other if they closely align. In the Query/Key space, the Key "answers"/"attends to" the Query. The Value is the vector describing the relation between two given Embeddings. Hence, the values can be described as the weights in the network and are calculated using the query function, such as a dot-product and the key. The output is then computed as a weighted sum of the values [22].
The following is the mathematical representation of the attention calculation:

$$Attention(Q, K, V) = Softmax(\frac{Q \cdot K^T}{\sqrt{d_K}}) \cdot V \text{ [22]} \tag{2.1}$$

However, the Transformer usage of self-attention employs Multi-head self-attention (MHSA), allowing the model to attend information from different representation subsidiary spaces at different positions. This is impossible with a regular single attention function because of averaging, which inhibits it [22]. Another way of describing MHSA is that several single self-attention functions are running in parallel, and the result from attention is linearly projected as the output array with a specified dimension [23].
The following is the mathematical representation of the MHSA:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \text{ [22]} \tag{2.2}$$

where

$$head_i = Attention(Q \cdot W_i^Q, K \cdot W_i^K, V \cdot W_i^V)$$

The Transformer comprises an encoder and a decoder, the usual practice in most neural sequence transduction models. The encoder computes a set of input representations while the decoder uses these representations to compute an output [22]. The Vision transformer only derives the encoder block from the transformer described by Vaswani et al. [22]. The sequence representations computed by the

encoder are then run through a "Multi-layer-Perceptron". The "Multi-Layer Perceptron"(MLP) is several linear layers at the end of the encoder block. When the input has traversed the encoder block, it goes through a final linear layer called the "MLP Head" which outputs the resulting class of the model, based on the input [21].

## 2.5 Class Activation Mapping

Class Activation Mapping (CAM) is a technique that highlights the input regions important for a convolutional neural network (CNN) to identify a particular class in an image. This method is particularly useful in our work, where understanding the model's focus is one of the core challenges we aim to tackle.

CAM utilises the global average pooling (GAP) layers in CNNs. Instead of fully connected layers, GAP layers are employed towards the network's end, significantly reducing the parameters and helping combat overfitting [24]. The key idea behind CAM is to use these GAP layers to directly relate the output feature maps of the last convolutional layer to the final class prediction scores.

### 2.5.1 The General Process of CAM

After the CNN has processed the input image through several convolutional and pooling layers and extracted a set of feature maps from the last convolutional layer, these feature maps, containing spatial information about various aspects of the input image, are fed into a GAP layer. This GAP layer computes the average value of all the activations in each feature map, resulting in a single number for each feature map.

The output from the GAP layer is then used to compute the class prediction, in our case, to determine whether the image belongs to the class of AI-generated images. This is achieved by feeding the pooled features from the GAP layer through a simple linear classifier. The weights in this linear classifier are crucial as they determine the contribution of each feature map to the final class prediction.

To visualize what CNN focuses on in making its decision, we scale the feature maps by their corresponding weights (learned during training) for the target class and then sum them up. This results in the Class Activation Map for that specific class[25]. The CAM can be displayed on top of the original image to highlight the most influential areas in the model's decision. This whole process is illustrated in Figure 2.5 below.

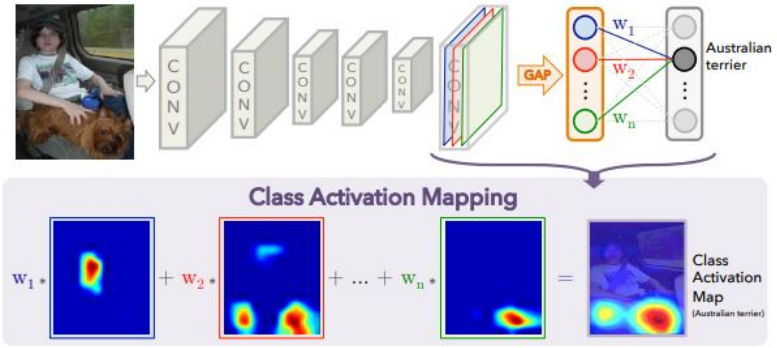



**Figure 2.4:** Zhou, B., Khosla, A., et al. (2016). Learning Deep Features for Discriminative Localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

However, this methodology only works when using a linear classifier. When adding complexity to the classifier, such as using a fully connected neural network (NN) as a classifier, a grad-CAM is created using the NN gradient instead.

### 2.5.2 Grad-CAM

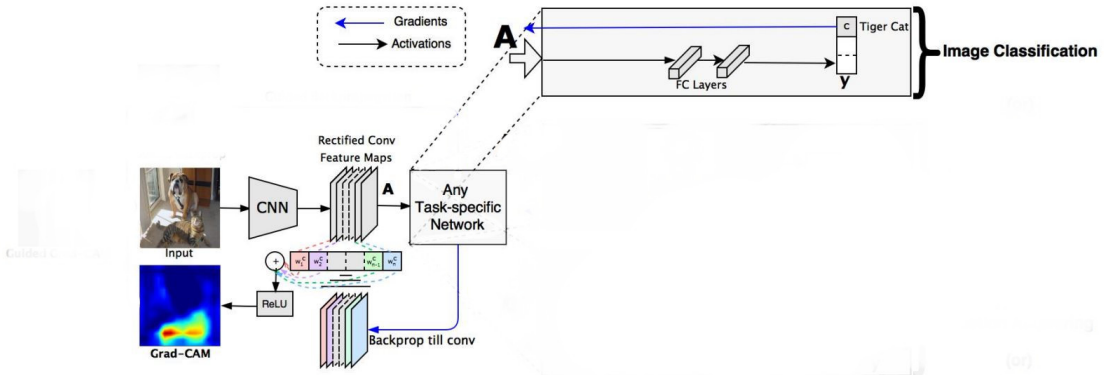



**Figure 2.5:** Selvaraju, Ramprasaath R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." Proceedings of the IEEE international conference on computer vision. 2017. [4]

When the linear classifier is replaced with a fully connected NN, the direct linear relationship between the feature maps and the different classes is obscured. This means each feature map's contribution to the final class decision is not as straightforward to compute.
When using the original CAM methodology it is possible to just scale the feature maps by their corresponding weight from the linear classifier. We must calculate the network's gradient to find these weights from a NN. After the NN has classified the input, we use backpropagation to calculate the gradient. The NN, in its essence, acts as a function that takes many variables as input in the form of feature maps

and yields a single output variable, which is a class prediction. The gradient will be a collection of derivatives in the form of feature maps.
These feature maps are then fed into a GAP layer. This GAP layer uses the outputting numbers like the weight in the original methodology. In other words, the original feature maps are scaled with their corresponding "weight" from the GAP layer. They are then summed up and form a so-called grad-CAM.

## 2.6 Guided Backpropagation

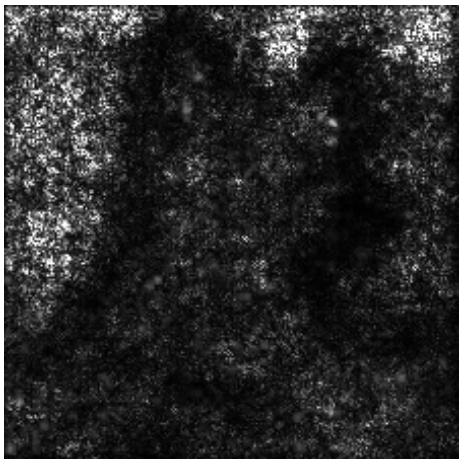
Guided Backpropagation is another technique that will be used to visualize what the CNN is looking at in an image. When trying to visualize a CNN's behaviour, the most straightforward approach would be to calculate the output gradient with regard to each input pixel and display these gradients with a heatmap. However, this method can often result in a messy visualisation. This is because an image contains significant and negligible features that tend to blend together in this visualisation, making it difficult to determine which features are the most important to the model's decision-making process.
Guided Backpropagation is a slightly altered version of this basic idea of visualisation. It recalibrates the traditional gradient calculation to focus only on paths in the network where the gradients are positive[26]. This filters out negative or neutral features, highlighting the features that truly matter to the model's outcome.
As Figure 2.6 shows, Guided Backpropagation is a significant improvement. Selectively computing the gradient containing only positive derivatives gives a much more accurate visualisation of what features from the input image influence the model's predictions.

## 2.7 Difference between Guided Backpropagation and CAM

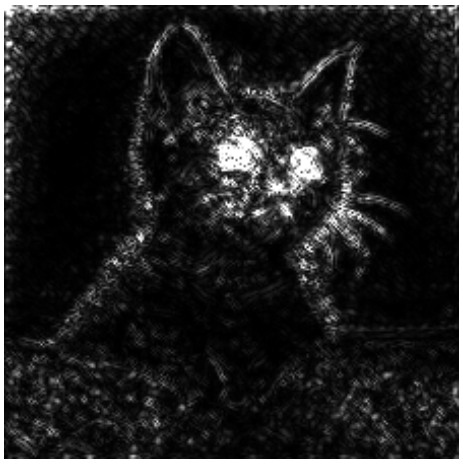
CAM and Guided backpropagation differ mainly in the parts of the network they examine. CAM primarily examines the last few layers, making its highlights coarser-grained and more region-based. This makes it great for analysing the spatial relevance of features. In contrast, Guided Backpropagation propagates through all layers, highlighting even small textures and patterns with pixel-level detail. This allows us to see how the network acts more granularly. We chose to use both as we are interested in how regions of the input affect the output decision and how specific visual features lead to those outputs.

## 2.8 Calibration

When creating a binary classification model, the output will be a confidence score between 0 and 1, with a score closer to 1 signifying more confidence that the given input belongs to class 1, in this paper, that is equivalent to the image being real. However, simply using the confidence score as a proxy for accuracy, turning 1 into

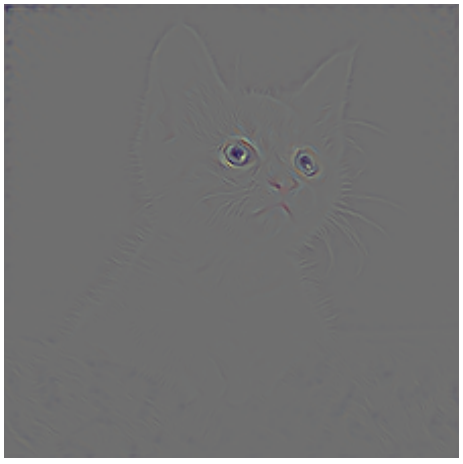


**(a)** Backpropagation.



**(b)** Guided Backpropagation.

**(c)** Backpropagation.



**(d)** Guided Backpropagation.

**(e)** Original Image.

**Figure 2.6:** Difference between guided and non-guided backpropagation

100% probability of being real, 0.8 into 80%, and so on, does not accurately describe the model's output. Therefore, calibration is needed to convert the confidence score to an accuracy [27].

### 2.8.1 Platt Scaling

Platt scaling is a calibration method described by John Platt [28], where a sigmoid curve is fit to the relation between the confidence score and the correct answer. The specific equation used for Platt scaling is [28]

$$P_{calibrated}(Y = 1) = \frac{1}{1 + e^{A \cdot P_{uncalibrated}(Y=1)+B}} \tag{2.3}$$

with the two parameters $A$ and $B$ to be determined by calibration, described further in section 3.3.
The idea behind Platt scaling is to convert the uncalibrated estimator, in this case, the confidence score, into a calibrated probability by assessing the actual accuracy of inputs with similar confidence scores taken from a validation data set. Introducing a sigmoid curve to achieve this is a quick and reasonably accurate simplification.

# 3
# Method

The project's technical work was divided into five major parts, mostly dedicated to developing three different kinds of detection models and corresponding visualisation components. The collection of data, including downloading and managing a pre-existing dataset and creating a few small datasets for testing. The last part was the construction of the user interface and website.

## 3.1 Data Collection and Preparation

All models were trained on the GenImage dataset [10]. The GenImage dataset consists of 2.68 million images divided into 1000 different classes, the same classes as the ImageNet[1] dataset. The real images are initially from ImageNet, and the AI-generated images are generated by one of the following generators: Midjourney, Stable Diffusion (v1.4 and v1.5), ADM, GLIDE, Wukong, VQDM, BigGAN. Each generator is a sub-directory of the entire GenImage dataset consisting of two directories, 'train' and 'val' folders, which in turn consist of another two directories, 'ai' and 'nature' folders of images. The images from the 'train' folders are split (0.9, 0.1) into a training and validation dataset, respectively. The entire GenImage testing dataset consists of all images from the 'val' folders of all generators. Some modifications were made due to file corruption, remov- ing 17 files from the dataset, all of which were AI-generated. The list of these files can be found in appendix B. Due to the limited amount of removed images, we conclude that this change does not affect the dataset's quality.
The models were also tested on a dataset of 5000 images from the subreddit r/dalle2[2], most of these images are assumed to be generated with Dall-e 3 [11][3], a generative model which was not present in the training data. Thus, this dataset can be used to get a performance metric on generative models on which the classification model has not been trained. Dall-e 3 is also much more modern than all generative models in the GenImage dataset.
Additionally, to allow fine-tuning of FCN models, a small dataset of 1940 images that are part AI-generated and part real was constructed. These images were created with the help of masks made from real images and stable diffusion's inpainting functionality [29]. The entire process looks like this: 970 real images are selected.

[1] https://www.image-net.org/index.php
[2] https://www.reddit.com/r/dalle2/
[3] The name of the subreddit being "dalle2" might confuse readers in this regard, but the name is outdated, and the community there seems to have moved on completely to Dall-e 3

Then, the background of each image is removed with third-party software [30]; this is the first step of creating the masks. After this, the images are manipulated so that the transparent parts (background) become black and the rest (object) white. The masks are now finished, and an image can be seen in Figure 3.1 after each step in the process. After this, random prompts are generated. These prompts are necessary since they describe what Stable diffusion's image generator should generate. When done, all components are ready to be input into the stable diffusion image generator. A Python script was written to automate the process of inputting a random prompt, a real image with its corresponding mask, and then to retrieve the result from stable diffusion, namely a partly AI-generated and partly real image. This resulted in 970 images with a real background and an AI-generated object. To equal the amount of AI-generated pixels and real pixels in the dataset, a second batch of 970 images was generated. This batch was made with inverted masks and slightly different prompts, making 970 images with real objects and AI-generated backgrounds. Due to using both a mask in its original and inverted version when generating the images, exactly 50% of the total pixels in the data set will be AI modified, thus not introducing any bias when finetuning the FCN models.

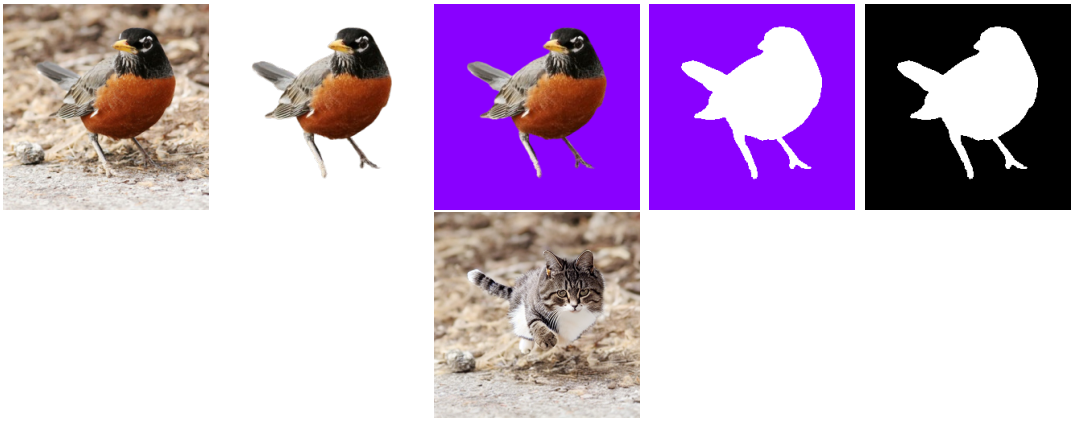


**Figure 3.1:** An image shown after each step in the process of creating the inpainting images

## 3.2 Model Construction

The project involved constructing detection models relying on three different architectures and training them on either the specific generator SD1.4 or the entire GenImage data set. The aim was to make the construction and training as similar as reasonably possible to increase the quality of comparative analysis, but some compromises were made to increase performance.

### 3.2.1 CNN

The CNN model is a slightly modified version of the ResNet50 model, initialized with pre-trained weights from classification training on the ImageNet dataset. The last fully connected layer was changed from 1000 output nodes to just one output

node. A sigmoid function was applied to the output to scale it from 0 to 1, where 0 corresponds to AI-generated and 1 corresponds to real images.
The training loop runs for the number of chosen epochs, which was selected as 10 for SD1.4 and 5 for the entire GenImage dataset. For each epoch, the model iterates through the entire training set by iterating through the training data loader with a batch size of 32 images. First, in the forwardpass, the model gets the input images and applies its forward function to output the predictions, a vector of size 1x32, with each element in the range 0 to 1. The true label is a vector of size 1x32 containing only 0's and 1's. After that, the loss is calculated with the Binary Cross-Entropy Loss (BCELoss) function. Then, in the backward pass, the gradients of the loss with respect to the model parameters are computed, and the Stochastic Gradient Descent (SGD) optimizer updates the model weights based on these gradients.
Following the training phase, the model is set to evaluation mode at the end of each epoch. Then, without computing gradients, the model is evaluated on the validation dataset. Similar to the training loop, the validation dataset is iterated over, and the model applies its forward function to the input images and outputs the predictions as a vector of size 1x32 with each element in the range 0 to 1. It computes the loss by applying the BCELoss function with the predictions and true labels as arguments. After that, it rounds all values in the vector to the closest integer value, i.e. 0 or 1, and calculates the accuracy by taking the ratio of the number of correct predictions divided by the total number of images. If the model's evaluation loss decreases after an epoch is executed, its state and parameters are saved.
As previously stated, BCELoss was used and the model weights were optimized with stochastic gradient descent. A constant learning rate of $10^{-2}$ was used.

### 3.2.2 FCN

The FCN was constructed based on the paper "Fully Convolutional Networks for Semantic Segmentation"[20] with the last layer of the model changed to reflect the change from the 21 classes to 2. The FCN utilizes the same architecture backbone as the CNN model (ResNet50), with the major difference that the fully connected component has been replaced by a series of deconvolutional layers, with the output being an image of equal size to the input, but only in one channel. Effectively returning a grayscale image, classifying each pixel as AI or Real.
The model weights were initialized with pre-trained weights from the classification of images on the COCO dataset [31]. Then, two different models were trained, training only on GenImage images for ten epochs on SD1.4 or five epochs on all generative models, respectively. The second model was also fine-tuned by training it on images produced by stable diffusion inpainting as described in 3.1. The model was saved before and after inpainting training to measure performance differences.
During training, all images were randomly cropped to 224-pixel squares and flipped horizontally and vertically with 50% probability for each direction. All pixel values were normalized with mean value 0.5 and standard deviation 0.5. During testing, both cropped images and full images[4] were considered. The loss function was mean

[4]Up to 1920x1080 resolution due to memory restrictions

squared error, the optimizer was Adam and, the learning rate was static at $10^{-3}$. The learning rate was lowered to $10^{-5}$ during fine-tuning.

As stated in Section 2.3, in theory, no preprocessing is required when working with FCNs. However, cropping the images was still necessary during training due to weak computational resources. Analysing larger images would have required more powerful machines to train the model, with VRAM capacity being the main bottleneck.

### 3.2.3 ViT

The Vision Transformer is based on the scientific paper "An Image Is Worth 16X16 Words: Transformers For Image Recognition At Scale [21], see section 2.4. The backbone of the model uses the pre-trained torch-vision ViT model (vit_l_16) trained on the ImageNet dataset[32].

However, the pre-trained model is constructed to be able to differentiate between 1000 classes. This required a change of heads in the model. This was done by replacing the number of classes, originally 1000, in the linearization step of the model heads with only 2 (Real or AI). We then use this model and teach it to differentiate between the 2 classes by training the model on the GenImage dataset[10]. The model is trained on the entire GenImage dataset for 10 epochs and individually on SD1.4 for 10 epochs.

To ensure a better accuracy representation during testing, we use the torch softmax function to get the output values in a range between 0 and 1; the function also relates the values so that the sum is equal to 1. Therefore, only one of the two values needs to be considered.

When training the ViT models, we used Cross Entropy Loss and the Adam optimizer, the learning rate started at 0.05 for the first 5 epochs, and was then lowered to $10^{-5}$ for the last 5 epochs.

## 3.3 Calibration

Calibration for all models was realized by Platt scaling, as described in section 2.8.1. The two parameters $A, B$ were initialized to the values $-10, 5$ and then changed to fit the sigmoid to the given model's output on 3 epochs through the relevant validation data set, with decreasing learning rate for each epoch, starting at 0.1, then 0.01 and finally 0.001.

## 3.4 Model Visualisation

Visualisation of models from the three different architectures is achieved using different methods. For CNN, Guided Backpropagation and Grad-CAM based visualisations are produced, for FCN, the pixel classifications are shown as an image, and for the ViT an attention heatmap based visualisation is provided.

### 3.4.1 CNN Visualisation

The visualisation components, Guided Backpropagation and Grad-CAM were developed in parallel with the CNN model to provide insights into what the models focus on when making predictions.

Grad-CAM was the first visualisation technique that was attempted. It combines weighted feature maps with the gradients flowing back from the specific class output at the final fully connected layer, offering a visual explanation for the model's predictions.

Guided Backpropagation was the second visualisation component that was implemented. As explained earlier in Section 2.6, this method modifies the traditional backpropagation process by only allowing positive gradients to backpropagate through the network. This approach highlights the features that contribute positively to the model's decision, offering clearer insights into the neural network's functioning.

### 3.4.2 FCN Pixel Classification Image

The output from the FCN is already considered a visualised result, and is shown as is, no further visualisation is provided from the FCN models.

### 3.4.3 Attention Heatmaps

Understanding the attention in a transformer-based model is necessary when trying to gain insights into how the model predicted the output. In the context of a Vision Transformer model, the attention heatmap visually represents where the model directs its attention within the input image. Since our model employs a multi-head attention mechanism, each attention head produces its own heatmap. These individual heatmaps reveal which tokens in the input sequence receive the most attention from each head [33]. This offers a detailed insight into the model's hierarchical visual information processing.

Each pixel in the combined heatmap corresponds to a spatial location in the image, and its intensity reflects the collective attention assigned by all attention heads to that particular region. By stacking the individual heatmaps and analyzing them collectively, we can discern which areas of the image the model deems most crucial for making predictions or classifications. This insight aids in understanding the model's decision-making process and identifying which visual features contribute most significantly to its output.

We accomplish this by dividing the image into patches, concatenating these patches with the class tokens, and adding the position embeddings later. This tensor is then fed to the first linear layer in the first multi-layer perceptron to be split into query, key and value matrices. These matrices are used for the multi-head attention, and since this is done step-by-step, the resulting attention matrix is added to an identity matrix to account for the residual connections. The next step is multiplying the weights recursively and using bilinear interpolation to plot the heatmap.

## 3.5 Interface Development

The network interface was developed to allow public access to our network. To facilitate this as easily as possible, it was decided to host it online. Specifically, a website using Next js is connected to a Python server, which hosts the neural networks and receives POST requests containing pictures that users upload to the website. The result of the Python server's analysis is then returned to the user and displayed on the website.

# 4

# Results

The results will be presented separately for each model, alongside a combined resource comparison. This section includes some analysis and discussion of performance and visualisation results for the individual models. Comparative discussion is provided in Section 5.

The suffix (ALL, SD1.4, or FINETUNED) following the model name (CNN, FCN, or ViT) shows what training dataset the model has been trained on. For ALL, the training dataset includes all AI and real images in the training folders of all generators in the GenImage dataset. SD1.4 contains only the AI and real images from the training folder of the SD1.4 generator. The FCN-FINETUNED model is trained both on the ALL training dataset and the training images from the Inpainting dataset.

Several test sets are used to measure the results, the contents of which are given as an itemized list below. If not explicitly stated otherwise, the images in each test set have been randomly cropped to a 224x224 pixel image before being processed by the relevant model.

- GI ALL - Contains all test images from the GenImage dataset.
- GI SD1.4 - Contains all test images in the GenImage dataset that were generated using Stable Diffusion v1.4. This is a subset of GI ALL.
- Reddit - Contains all images from the Reddit dataset. These images are assumed to come from Dall-e 3 but without guarantee. See Section 3.1 for more information. The most important feature of the Reddit dataset is that all of the images are AI-generated, which has caused some problems in analyzing the results for this dataset. These quandaries are further discussed when relevant.
- Inpainted - Contains all test images from the Inpainting dataset. See Section 3.1 for more information. When used with CNN or ViT models that give a singular label for the entire image, the true label for all of these images is defined as "AI", even though they are not fully AI-generated. For the more nuanced FCN models, it doesn't make as much sense to label these images. This is discussed further in Section 4.2.

For all models, a positive answer is stated as the label "AI", and a negative answer as the label "Real." Thus, a false positive means a model labels a real image as "AI," and a false negative means a model labels an AI-generated image as "Real".

## 4.1 CNN

The accuracies of the CNN-SD1.4 and the CNN-ALL models on four different test datasets are shown in Table 4.1 and 4.2, respectively. The test datasets are the GI ALL, GI SD1.4, Reddit and the Inpainted dataset, which are described in the itemized list above. The CNN-SD1.4 model is trained on only the SD1.4 generator training dataset, a subset of the GenImage dataset. CNN-ALL is trained on the entire GenImage training dataset, including all 8 generators.

**Table 4.1:** Accuracies for CNN model trained on SD1.4 training set for 10 epochs (CNN-SD1.4)

| | GI ALL | GI SD1.4 | Reddit | Inpainted |
|---|---|---|---|---|
| Accuracy | 76.6% | 99.9% | 4.7% | 53% |

**Table 4.2:** Accuracies for CNN model trained on the entire training set for 5 epochs (CNN-ALL)

| | GI ALL | GI SD1.4 | Reddit | Inpainted |
|---|---|---|---|---|
| Accuracy | 99.9% | 99.9% | 12.6% | 98.2% |

**Table 4.3:** Confusion matrix for CNN-SD1.4, tested on GI ALL

| | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | 26640 | 23360 |
| Was Real | 43 | 49957 |

**Table 4.4:** Confusion matrix for CNN-ALL, tested on GI ALL

| | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | 49926 | 74 |
| Was Real | 64 | 49936 |

**Table 4.5:** Results for CNN-SD1.4, tested on Inpainted dataset

| | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | 208 | 177 |

**Table 4.6:** Results for CNN-ALL, tested on Inpainted dataset

| | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | 378 | 7 |

**Table 4.7:** Results for CNN-SD1.4, tested on Reddit dataset

| | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | 286 | 4356 |

**Table 4.8:** Results for CNN-ALL, tested on Reddit dataset

| | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | 586 | 3976 |

As shown in Table 4.1, the CNN-SD1.4 model performs very well with an accuracy of 99.9% when tested on the SD1.4 test set, which makes sense since it should learn to identify typical characteristics in images created by the generator. However, the accuracy of the entire GenImage test set is naturally lower at 76.6%. As seen in the confusion matrix in Table 4.3, CNN-SD1.4 has a high rate of false negatives, classifying about 46.7% of the AI images as real images. For the real images, however, the model has an accuracy of $> 99.9\%$. This is probably due to the model's bias, predicting about 73.3% of all images as real, but also that the model has learned typical features in real images. This is still arguably a satisfactory result given that the model is trained on 1 generator and tested on all 8, which means that 7/8 of the AI images in the test dataset are from never-before-seen generators, one could argue that SD1.5 is similar enough to SD1.4 to impact the result further, but even with 6/8 images from unseen generators, this is still very impactful. As shown in Table 4.2, the CNN-ALL model has an accuracy of 99.9% when tested on all 8 generators and the SD1.4 generator. This indicates that the model has learned to identify features from all generators effectively and accurately. The exact predictions are shown in Table 4.4.

There is a significant difference in accuracy on the Inpainted dataset, where CNN-ALL has an accuracy of 98.5% and CNN-SD1.4 has an accuracy of only 53.0%, shown in Table 4.1 and 4.2. Both models behave quite similarly on this dataset compared to the GI ALL dataset regarding accuracy on AI-generated images. What explains the difference in accuracies is that the Inpainting dataset only contains AI-generated images. CNN-SD1.4 predicts about 53% of the images as AI-generated, while CNN-ALL predicts roughly 98% as AI-generated, shown in tables 4.5 and 4.6. This causes the accuracy to be lower than that of the GI ALL dataset, which contains the same number of real and AI-generated images since the models have a very high true negative rate.

Both models perform poorly on the Reddit dataset consisting of only AI-generated images, with an accuracy of 12.6% for CNN-ALL and 4.7% for CNN-SD1.4, shown in tables 4.2 and 4.1. This is probably due to a few different reasons. Firstly, the Reddit dataset consists of only AI-generated images, and with a high rate of false negative predictions of 93.8% for CNN-SD1.4 and 87.2% CNN-ALL, shown in tables 4.7 and 4.8, the accuracy will naturally be lower than for a dataset with a balance between real and AI-generated images. Secondly, the Reddit dataset consists almost entirely of images generated by Dall-e 3, which is a more advanced and modern generator compared to all 8 generators in GenImage that generate more

realistic images. Since the models are not trained on images from this generator, they seem to make very poor predictions, probably because they haven't learned the generator's typical characteristics.

### 4.1.1 Visualisation

Figure 4.1 displays three images that help to visualize how the CNN model analyzes a real image. The first image shows the original photograph of a dog. The second image visualizes the grad-CAM with a heatmap, highlighting the areas in warmer colours with more contribution to the model's classification. The third image shows the result of applying guided backpropagation to the image. The brighter areas are pixels from the input image with a positive output gradient, making up features that the model deems important.
Figure 4.2 follows the same structure as Figure 4.1 but uses an AI-generated image instead. The AI-generated image in question is an image generated by the image-generator Midjourney.

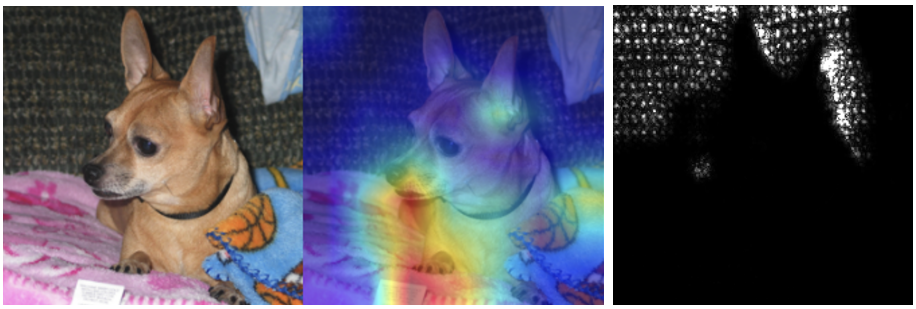


**Figure 4.1:** Real image, grad-CAM followed by Guided Backpropagation

**Figure 4.2:** AI generated image, grad-CAM followed by Guided Backpropagation

grad-CAM and Guided Backpropagation (GBP) highlight important areas for the model's decision-making process. As mentioned in the technical background, grad-CAM examines the last few layers, highlighting broader regions, while GBP focuses on the flow through the network, giving visualisation for specific details. As seen in Figure 4.2, grad-CAM and GBP highlight different regions of the image. This indicates that the last few layers show more activity at the bottom of the image.

Meanwhile, the overarching flow through the network has more activity in the upper parts of the image. This is seen in both Figure 4.1 and Figure 4.2. However this isn't always the case, some images give similar visualisation with grad-CAM and GBP and others like this one get different regions highlighted. This will be discussed later in section 5 aswell as showcased in appendix A.

## 4.2 FCN

The performance of the FCN models was measured across six different data sets. The four main datasets were from GenImage, alongside GI ALL and GI SD1.4, using the same transforms that were applied during training, cropping the images down to the size of 224x224 pixels, GI ALL (Full) and GI SD1.4 (Full) were used, containing the same images, but without any transforms, effectively allowing the model to analyze the entire image and classify each pixel. Additionally, the models were tested on images from more modern generative models, with the "Reddit" dataset, and partially AI-generated images, with the "Inpainted" dataset. For these datasets, only full images were considered.
Additionally, two metrics are used for each dataset. The first metric is the classification accuracy for classifying each individual pixel. The second metric is the classification accuracy of the label for the entire image, obtained by calculating the mean value of the pixel classification for all pixels in a given image. In other words, if most pixels in the image are classified as AI-generated, the entire image is labelled as AI.
As previously mentioned, no full image classification accuracy is given when measuring FCN performance for the "Inpainted" dataset. Instead, the data point for that table entry is used to show how many images the model considers to be AI-generated, from here on, called the "AI classification rate".

**Table 4.9:** Accuracies for FCN model trained on SD1.4 training set for 10 epochs (FCN-SD1.4)

| Classification | GI ALL | GI ALL (Full) | GI SD1.4 | GI SD1.4 (Full) | Reddit | Inpainted |
|---|---|---|---|---|---|---|
| Pixel classification accuracy | 78.6% | 85.7% | 99.0% | 98.5% | 10.4% | 52.0% |
| Image classification accuracy | 78.4% | 80.8% | 99.1% | 99.4% | 5.5% | 98.7% AI |

**Table 4.10:** Accuracies for FCN model trained on entire training set for 5 epochs (FCN-ALL)

| Classification | GI ALL | GI ALL (Full) | GI SD1.4 | GI SD1.4 (Full) | Reddit | Inpainted |
|---|---|---|---|---|---|---|
| Pixel classification accuracy | 99.0% | 98.4% | 99.4% | 99.5% | 8.81% | 51.2% |
| Image classification accuracy | 99.1% | 95.6% | 99.4% | 99.7% | 7.40% | 100.0% AI |

**Table 4.11:** Accuracies for FCN model trained on the entire training set for 5 epochs and on the inpainting training set for 100 epochs (FCN-FINETUNED)

| Classification | GI ALL | GI ALL (Full) | GI SD1.4 | GI SD1.4 (Full) | Reddit | Inpainted |
|---|---|---|---|---|---|---|
| Pixel classification accuracy | 60.1% | 58.1% | 66.1% | 71.4% | 29.5% | 70.9% |
| Image classification accuracy | 61.3% | 64.2% | 67.6% | 72.5% | 19.9% | 36.9% AI |

**Table 4.12:** Confusion matrix for FCN-SD1.4, tested on GI ALL

| Pixel Classification | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | $1.47 \cdot 10^9$ | $1.04 \cdot 10^9$ |
| Was Real | $3.74 \cdot 10^7$ | $2.47 \cdot 10^9$ |

**Table 4.13:** Confusion matrix for FCN-SD1.4, tested on cropped images from GI ALL except Stable Diffusion (both v1.4 and v1.5 are removed)

| Pixel Classification | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | $7.76 \cdot 10^8$ | $1.03 \cdot 10^9$ |
| Was Real | $2.7 \cdot 10^7$ | $1.78 \cdot 10^9$ |

**Table 4.14:** Confusion matrix for FCN-ALL, tested on GI ALL

| Pixel Classification | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | $2.48 \cdot 10^9$ | $2.59 \cdot 10^7$ |
| Was Real | $2.64 \cdot 10^7$ | $2.48 \cdot 10^9$ |

**Table 4.15:** Confusion matrix for FCN-FINETUNED, tested on GI ALL

| Pixel Classification | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | $1.90 \cdot 10^9$ | $6.12 \cdot 10^8$ |
| Was Real | $1.39 \cdot 10^9$ | $1.12 \cdot 10^9$ |

**Table 4.16:** Confusion matrix for FCN-SD1.4, tested on the Inpainted testset

| Pixel Classification | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | $5.11 \cdot 10^7$ | $4.97 \cdot 10^5$ |
| Was Real | $4.79 \cdot 10^7$ | $1.39 \cdot 10^6$ |

**Table 4.17:** Confusion matrix for FCN-ALL, tested on the Inpainted testset

| Pixel Classification | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | $5.16 \cdot 10^7$ | $3.95 \cdot 10^3$ |
| Was Real | $4.93 \cdot 10^7$ | $2.56 \cdot 10^3$ |

**Table 4.18:** Confusion matrix for FCN-FINETUNED, tested on the Inpainted testset

| Pixel Classification | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | $3.45 \cdot 10^7$ | $1.72 \cdot 10^7$ |
| Was Real | $1.22 \cdot 10^7$ | $3.71 \cdot 10^7$ |

As seen in Table 4.9, FCN-SD1.4 gets very good results on the partial test set, which is to be expected. However, it still scores surprisingly well on the entire dataset, even after removing the increase in accuracy due to the SD1.4 and SD1.5 test images (closely related to the training data) having $> 99\%$ accuracy. The model has a 70.7% accuracy for pixel classification on the 6 remaining generators in the dataset, as seen in Table 4.13. Further analysis of the model shows that false positives are rare, with most misclassifications being false negatives; see Table 4.12.
It is also remarkable that we observe a 7.1%pt increase in pixel classification accuracy by letting the model analyze the entire image compared to a 224x224 pixel cropped image, even though the model has been trained on solely cropped images. However, the same effect is not observed when testing on the SD1.4 subset.
Using FCN-ALL, the accuracy is very good for all metrics related to the GenImage dataset, showing that the model seems to learn identifying features from all 8 generators in the dataset well.
The Reddit dataset, composed of only AI-generated images, shows very poor performance from both models. This is unsurprising considering the findings in Table 4.12. With few false positives compared to false negatives, a data set composed of only AI-generated images is bound to be problematic for the model. However, compared to the 43.0% accuracy on AI-generated images from not-before-seen generators (calculated from the top row of Table 4.13), the result is still poor enough to show that a more modern generator like Dall-e 3 seems to differ from the older generators used in training enough to impact the detection results.
For the inpainted images, both models perform quite badly. Investigating the confusion matrices in Table 4.16 and Table 4.17, we see that both models seem to classify a significant majority of pixels as AI, signifying detection of AI involvement in the image cascading out to the classification of all pixels in the images. After fine-tuning, this problem is alleviated with FCN-FINETUNED learning features of the inpainted images, increasing the accuracy of inpainted images. However, the performance on all GenImage-based test sets drops quite dramatically. Surprisingly, the performance increases on the Reddit dataset. This doesn't seem to be a sign of increased performance but rather a consequence of that (in contrast to the non-fine-tuned models) the fine-tuned model favours classifying pixels as AI. This leads to more false positives than false negatives, as shown in Table 4.15. This would, of course, increase accuracy on a dataset of only AI-generated images. As an additional point, it might seem counterintuitive that the AI classification rate of inpainted images drops after finetuning, but this is to be expected, considering that due to the construction methodology of the inpainted data set, 50% of images in the dataset have more AI generated pixels than real pixels, the optimal AI classification rate for an FCN model should be 50%.

Even though the accuracy of FCN-FINETUNED is quite poor, the output images are deemed useful to show what parts of an image might be more indicative of AI involvement, compared to the non-fine-tuned images that mostly output wholly AI-generated or Real classifications for all pixels in an image, even if only parts of the image are AI-generated. Therefore, the classification tool will show the image classification result from FCN-ALL (but not the pixel classification image, which is almost always monocoloured anyway) alongside the pixel classification image from FCN-FINETUNED. Example images from FCN-ALL and FCN-FINETUNED are provided below in Figure 4.3.

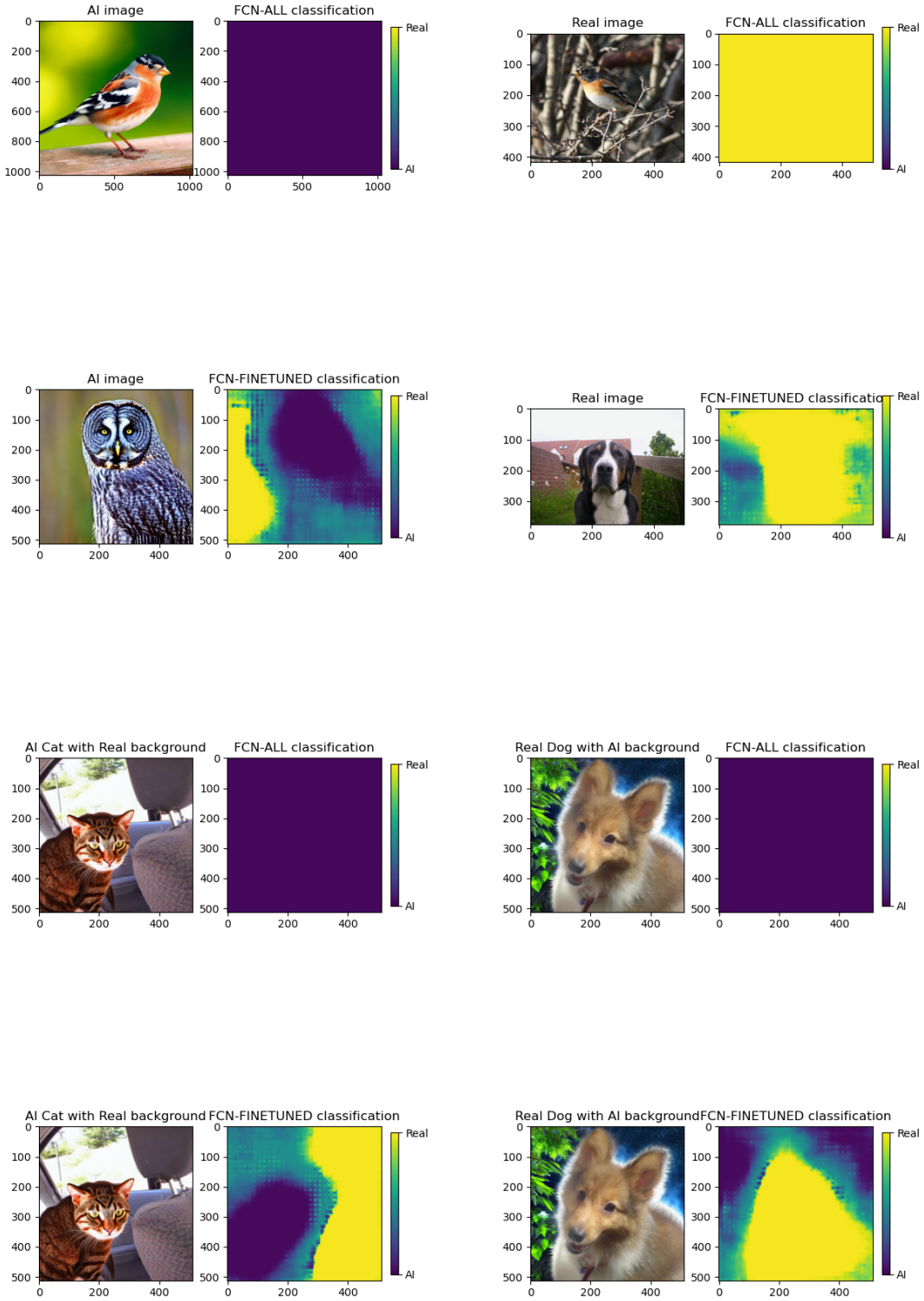



**Figure 4.3:** Example images and the classification outputs from FCN-ALL and FCN-FINETUNED

## 4.3 ViT

The tests are conducted on two separate partitions of the ViT model. One is trained on all of the generators in the GenImage dataset (GI ALL), and another is only trained on the SD1.4 generator from the GenImage dataset (GI SD1.4). However, even more sophisticated generators, such as Dall-e 3, have been published. Hence, we test the model's robustness on the "Reddit" and "Inpainted" datasets. The model is judged based on the accuracy of the image classification.
The following nomenclature describes the attributes of the datasets used in the testing. We distinguished between cropped and resized images, "Crop (GI ALL)" means that the dataset uses the images from all the generators and, during the transformation, takes a random crop of the image with dimensions 224x224. "Resize (GI ALL)" refers to the images being resized to 224x224 and that all generators were used in testing. This general pattern, with the prefix "Resize" or "Crop" and the consequent generator or dataset, is true for all the datasets described. Since the partitions were trained on cropped images, we wanted to see if this significantly impacted the accuracy of the same images but just resized. Hence, using "Resize" and "Crop" is necessary to differentiate between the two dataset instances. The following table, 4.19, shows the resulting accuracy achieved by the entirely trained model:

**Table 4.19:** Accuracies for ViT model trained on GI ALL for 10 epochs (ViT-ALL).

| | Crop (GI ALL) | Resize (GI ALL) | Crop (GI SD1.4) | Resize (GI SD1.4) |
|---|---|---|---|---|
| Accuracy | 81.8% | 68.0% | 84.8% | 69.6% |

| | Crop (Reddit) | Crop (Inpainted) |
|---|---|---|
| Accuracy | 62.4% | 90.4% |

**Table 4.20:** Confusion matrix for ViT-ALL, tested on the GI ALL dataset.

| | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | 41516 | 8484 |
| Was Real | 9672 | 40328 |

**Table 4.21:** Confusion matrix for ViT-ALL, tested on the GI SD1.4 dataset.

| | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | 5320 | 680 |
| Was Real | 1144 | 4856 |

**Table 4.22:** Confusion matrix for ViT-ALL, tested on the Reddit dataset.

| | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | 2847 | 1715 |
| Was Real | 0 | 0 |

**Table 4.23:** Confusion matrix for ViT-ALL, tested on the Inpainted dataset.

| | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | 348 | 37 |
| Was Real | 0 | 0 |

and Table 4.24 below presents the accuracy achieved by the model, which was trained exclusively on the GI SD1.4 dataset for 10 epochs:

**Table 4.24:** Accuracies for ViT model trained on GI SD1.4 for 10 epochs (ViT-SD1.4).

| | Crop (GI SD1.4) | Crop (GI ALL) | Crop (Reddit) | Crop (Inpainted) |
|---|---|---|---|---|
| Accuracy | 91.6% | 72.9% | 38.2% | 54.3% |

**Table 4.25:** Confusion matrix for ViT-SD1.4, tested on the GI SD1.4 dataset.

| | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | 5574 | 426 |
| Was Real | 585 | 5415 |

**Table 4.26:** Confusion matrix for ViT-SD1.4, tested on the GI ALL dataset.

| | Guessed AI | Guessed Real |
|---|---|---|
| Was AI | 27739 | 22270 |
| Was Real | 4850 | 45142 |

Based on the results shown in Table 4.19, the resulting accuracy of the ViT model trained on GI ALL are moderately high, having an accuracy of at most 81.8% when tested on GI ALL and at most 84.4% when tested on GI SD1.4. However, FCN and CNN have notably higher accuracy than ViT when trained on the same dataset, with both models reaching an accuracy of around 99%. By observing the confusion matrix for the results on GI ALL, which can be seen in Table 4.20, there seems only to be a slight bias towards classifying an image as AI, leading to a slight bias towards false positives. The small difference between the number of false negatives and false positives indicates that the bias present in the model when trained on GI All is minor. A similar inference can be made from the confusion matrix from testing the model on GI SD1.4 seen in Table 4.21 where a slight bias can be observed similarly to the previous matrix. The accuracy difference between Crop and Resize shown in Table 4.19 indicate a clear decrease from cropping the images compared to resizing the images. From an accuracy of 81.8% when testing on the Crop instance to an accuracy of 68.0% when testing on the Resize instance of GI ALL and with GI SD1.4, from an accuracy of 84.8% with Crop to 69.6% with Resize.
The results of the "Reddit" dataset, shown in Table 4.19 are quite noteworthy. The resulting accuracy is quite high, with the "Reddit" dataset giving an accuracy of

62.4%. In comparison, the FCN model had at most an accuracy of 19.9% accuracy on "Reddit", and the CNN had at most 12.6%. These results indicate that the ViT model is somewhat robust and can classify images from newer generative models.

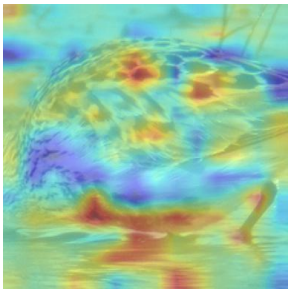
Compared to the previous ViT model trained on GI ALL, the model ViT model trained on SD1.4 exhibits lower accuracy on most dataset instances, as shown in Table 4.24. The only exception is when the model was tested on GI SD1.4, where it achieved a higher accuracy of 91.6% compared to the model trained on GI ALL, which had an accuracy of 84.8%. This is expected as the GI ALL model was trained on the SD1.4 dataset and other datasets, which generally results in lower accuracy than a model trained solely on the tested dataset. Conversely, the model performed notably worse when tested on GI ALL, where it achieved an accuracy of 72.9%, which is lower than the other model, which reached an accuracy of 81.8%. Regarding the "Reddit" and "Inpainted" datasets, the model performed worse than the ViT-ALL model. However, it still performed surprisingly well with an accuracy of 38.2% on "Reddit" and 54.3% on "Inpainted".

The confusion matrix of the model tested on GI SD1.4, shown in Table 4.25 seems to be somewhat similar to the confusion matrix of the model trained on GI ALL when tested on GI ALL, shown in Table 4.20. The only difference is the distributions of True and False values, which might reflect each model's resultant accuracy for that confusion matrix. A notable quirk of the SD1.4 model can be observed when constructing a confusion matrix of the results from testing on the GI ALL model, presented in Table 4.26. The model had a significantly higher amount of false negatives than false positives. Consequently, this would mean that, given an AI image, the model would have a probability of 55.4% of classifying the image correctly.

### 4.3.1 Visualisation

To further interpret the results from the ViT model, we utilize Attention Heatmaps to visualize the potential difference between generated and real images; this method is described in section 3.4.3. The following is the resulting Attention Heatmaps, using the ViT-ALL partition, on real and AI-generated images:

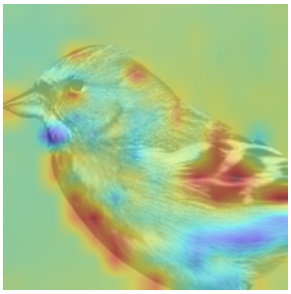
**(a)** Real image



**(b)** Attention Heatmap

**Figure 4.4:** Subfigure 4.4a and its heatmap in subfigure 4.4b

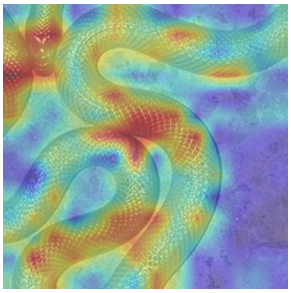
**(a)** AI image (SD1.4)



**(b)** Attention Heatmap

**Figure 4.5:** Subfigure 4.5a and its heatmap in subfigure 4.5b

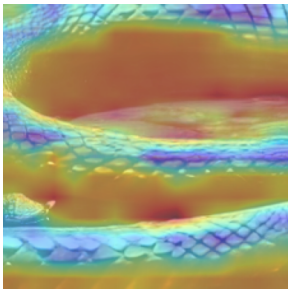
**(a)** Real image



**(b)** Attention Heatmap

**Figure 4.6:** Subfigure 4.6a and its heatmap in subfigure 4.6b

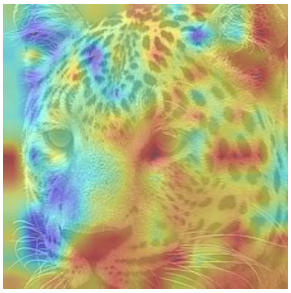
**(a)** AI image (SD1.4)



**(b)** Attention Heatmap

**Figure 4.7:** Subfigure 4.7a and its heatmap in subfigure 4.7b

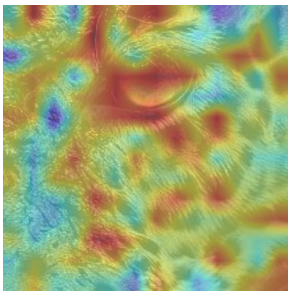
**(a)** Real image



**(b)** Attention Heatmap

**Figure 4.8:** Subfigure 4.8a and its heatmap in subfigure 4.8b

(a) AI image (Reddit)



(b) Attention Heatmap

**Figure 4.9:** Subfigure 4.9a and its heatmap in subfigure 4.9b

The results from the Attention Heatmaps in Figure 4.4 trough 4.9 highlight that the ViT-ALL partition gives more attention towards parts of the image which contain more variation, perhaps linked with the colours where the darker tones are more represented in the attention. Since the ViT model uses spatial information across the whole input image, see section 2.4 for a more thorough explanation, assuming that the attention would not be localized is reasonable.

## 4.4 Website

The website works as intended. However, images with a resolution higher than 512x512 pixels are cropped to accommodate the server's low RAM. The results from analyzing the image are returned to the user and displayed on the website. Figure 4.11 is an example of the data from the analysis shown to the user. This analysis includes the FCN-Finetuned pixel classification, ViT attention heatmap, GradCAM heatmap, and a guided backpropagation gradient map in addition to the image's predicted class from CNN-ALL, FCN-ALL and VIT-ALL, which is Platt-scaled to show probability. The website can be accessed at AI Detection Marvel[1].

[1]https://www.aidetectionmarvel.com

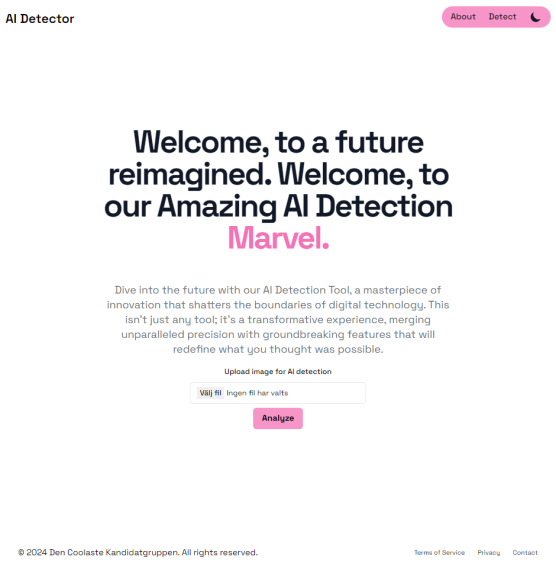



**Figure 4.10:** Landing page of website

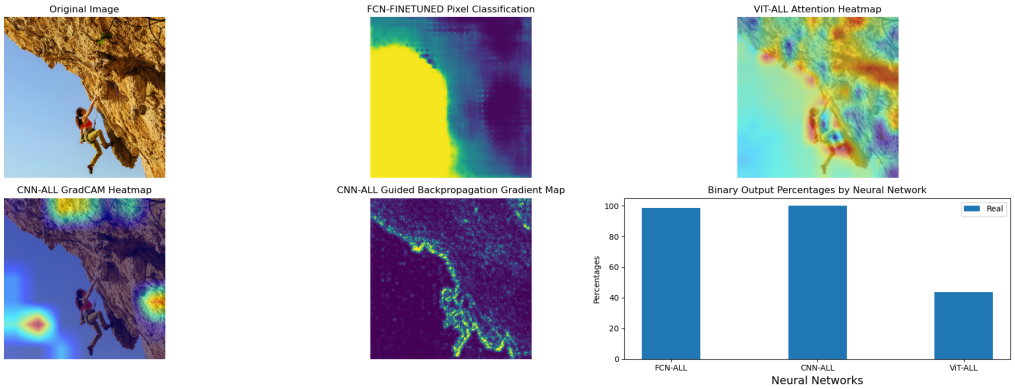



**Figure 4.11:** Example output from analysing an image on the website

## 4.5 Resource Comparisons

The models vary dramatically in size and training duration. To illustrate this, table 4.27 shows the size of the weight file, the training duration and the estimated power consumption for the largest model in each category (CNN-ALL, FCN-ALL, ViT-ALL). Note that all of these values are provided as if the models would have been trained on the same computer (PC1), housing an Nvidia RTX 3080 graphics card and an AMD Ryzen 5 5600x processor. In reality, ViT-ALL was trained on the more powerful computer (PC2), but we present the data in this manner to enable comparative analysis. The estimated energy cost is calculated based on the average power draw of the first 1000 training batches, which was the same for all models at 365 W.

**Table 4.27:** Resource comparison for CNN-ALL, FCN-ALL and ViT-ALL

| Model | Model size (Mb) | Training duration (hours) | Estimated energy cost (kWh) |
|---|---|---|---|
| CNN-ALL | 90 | 11 (2,2/epoch) | 4.01 (0.802/epoch) |
| FCN-ALL | 135 | 31,5 (6,3/epoch) | 11.5 (2.30/epoch) |
| ViT-ALL | 1130 | 108 (10,8/epoch) | 39.4 (3.94/epoch) |

Unsurprisingly, the larger the model, the more energy is needed when training. FCN-ALL takes more energy than expected for its size, most likely due to handling $224 \cdot 224 = 50176$ labels per input, compared to 1 label per input in the CNN and ViT models.

## 4.6 Reproducibility

All code used for the project will be uploaded to GitHub[2] with an MIT license. Additionally, the weights for the trained models discussed in the paper will be uploaded to a public OneDrive folder[3] to make result reproduction as easy as possible.

[2]https://github.com/EENX16-VT24-31/AI-Image-Detector

[3]https://chalmers-my.sharepoint.com/:f:/g/personal/joelande_chalmers_se/Ei3nqDWShtdKqZZWQ_2ihbMBX3z_SPxnL6P9zy6oNirD8w?e=9GSPi3
OneDrive available until 07/06-2024 due to Chalmers Policy, after that, request access by email to joelande(at)chalmers.se

# 5
# Discussion

The discussion presents two analyses: classification, which delves into the models' performance, and visualisation, which aims to represent the patterns and relationships within the different graphical model outputs. These complementary approaches offer a comprehensive understanding of the different model architectures, blending structured categorization with intuitive representation and fostering deeper insights and informed decision-making.

## 5.1 Analysis of Classification Results

With the results from the three different model architectures, it is quite clear that the CNN and FCN models behave quite similarly, the only major difference being that the CNN models generally outperform the FCN models on test set related to the training data, while the FCN handles unseen generators from GenImage better. It is possible that this is due to the FCN models being able to analyze the entire image since even when testing the FCN models themselves, as previously stated, the performance increases significantly if you allow the model to analyze the entire image instead of just a crop.
CNN and FCN struggle heavily with false negatives when analyzing images from generators, which they haven't seen previously. This seems to indicate that the classification models are better at understanding the features of the generative models than those of real images. This becomes problematic when new generative models are released, as demonstrated by the poor performance on the Reddit dataset for both architecture types, even when trained on all 8 generators in the GenImage dataset.
The ViT architecture is a major outlier in this regard, having significantly worse accuracy on test images related to its training data without a massive dropoff when testing on the Reddit dataset. This could result from the ViT models having many more parameters and, therefore, being capable of understanding what a real image looks like instead of analyzing specific features from the generators. It is also possible that the architecture plays a part, enabling the model to process images based on global features instead of relying solely on local and textural features.
One thing to note regarding the ViT model is how it responds to new data when the model is not given enough training examples. This was observed when the ViT SD1.4 model was tested on the GI ALL dataset, where the number of false negatives was equal to the true positives. Essentially, the model had a probability of around 50% correctly classifying an AI image. This is most likely caused by a lack

of training examples, which has decreased the model's robustness. It is also worth noting that the model could still classify real images with high precision even with fewer training examples of real images. This suggests that rather than a general lack of training examples, a more varied set of training examples, ideally from many different sources, could improve the model significantly.
The ViT model responded differently when the image processing of the input images was changed, from cropping the images to resizing the images. This change resulted in a significant decrease in accuracy. The cause of this decrease is not very clear. One plausible reason for this decrease is that the model was trained with cropped images. When the input images are resized instead of cropped when testing, the model might see these images as completely new images. The resulting accuracy of the resized dataset is also closer to the Reddit dataset, which contains mostly new images. It might be ideal to train these models with resized images compared to cropped images since images of extremely large pixel sizes might lose most, if not all, of their points of interest if cropped compared to resizing.

### 5.1.1 Inpainting Classification Results

The most striking difference between models in the results is the AI classification rate of images from the Inpainted data set. The rate is very similar for the models trained on the entire training dataset (CNN-ALL, FCN-ALL and ViT-ALL). Still, when trained on the SD1.4 subset, the FCN architecture has a significantly higher AI classification rate at 98.7%. It possibly shows the importance of understanding the context of the entire image when only parts of it are AI-generated. However, after training on the entire GenImage training data set, the discrepancy between the models, as previously stated, goes away. A possible explanation is that with enough training, the CNN and ViT-based models better learn what to focus on in an image, making the additional context from analysing the entire image less useful. This is supported by comparing grad-CAM visualisations from CNN-SD1.4 and CNN-ALL, where most heatmaps show smaller focus areas when using the latter model. See Figure 5.1 for an example.

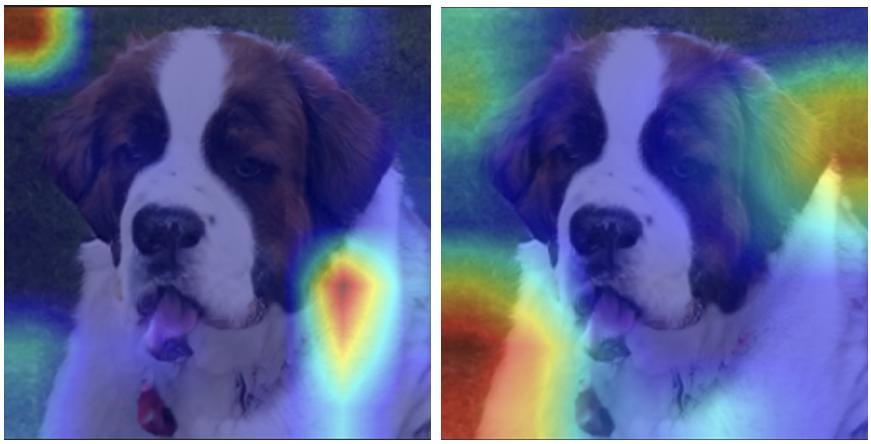


**Figure 5.1:** grad-CAM heatmaps for CNN-ALL (left) and CNN-SD1.4 (right)

CNN-ALL, FCN-ALL and ViT-ALL all classify $> 90\%$ of inpainted images as AI-

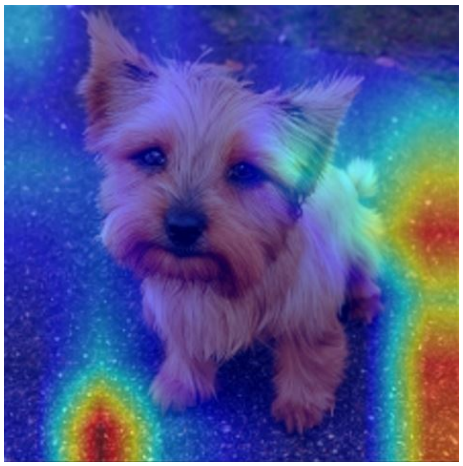
generated, even though 50% of the pixels in the images were not altered by AI. This shows that there isn't a lot of context needed to determine whether an image is AI-generated, and the models do not rely on some global statistics of the image; instead, they focus on conspicuous details in an image. This somewhat supports our idea of using the models and relevant visualisation techniques to highlight these areas to a user to enable a more informed decision.
The process of fine-tuning FCNs to increase accuracy on inpainted images did not yield the results we were hoping for, with very poor performance on the original data sets. It is possible that with a larger inpainting data set, this problem could have been solved, producing a model that can do pixel classification reliably. The method's validity seems plausible, considering the major increase in pixel classification on inpainted images. Some attempts at constructing a more heavily fine-tuned model by increasing the learning rate were made and yielded better accuracy on the inpainted dataset but with even worse performance on the GenImage data sets and less informative classification images.

## 5.2 Analysis of Visualisations

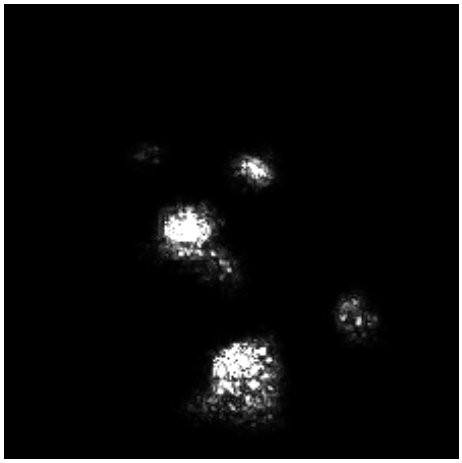
After having produced the visualisations of the different models, see Figure 5.2 and Figure 5.3, we can see that the outputs share similarities. The most prevalent similarities lay between the grad-CAM, Guided Backpropagation and the Attention Heatmap across the real and AI images.
However, the FCN-FINETUNED pixel classification produced a close overlap with the Attention Heatmap, with the most notable resemblance by the head of the AI-generated dog and in between the front legs, see Figures 5.2a, 5.2e, 5.2d. The ViT-ALL model had lower attention by the head and higher between the legs in Figure 5.2d, which reflects the FCN-FINETUNED AI pixel cluster and Real pixel cluster in Figure 5.2e. In Figure 5.3, we can see the different visualisations of a real dog image. The biggest difference from the AI-generated image is that the Attention Heatmap shows much lower attention to the image with fewer red regions. However, there are still overlapping shapes between the FCN-FINETUNED and the ViT-ALL Attention Map. FCN-FINETUNED still classifies the lower left corner as AI, and the Attention Heatmap shows less attention to that corner.
Furthermore, by comparing the grad-CAM and Attention Heatmap, we find that the grad-CAM seems to focus much more on a localized area of the image than how the ViT model's attention highlights attention spread more throughout the image. Take Figure 5.2b and 5.2d as an example. To human perception, the grad-CAM image does not follow any patterns or colour gradients that differ from the Attention Heatmap.
However, there are some similarities between the grad-CAM and Attention Heatmaps. To a human, some underlying structure may seem to be visualized in the attention heatmap 5.2e; the ViT model is much more interested in the background than the object, the dog, in the image. Although not as clear, this can also be seen in the grad-CAM image 5.2d. The similarity is shown further when inspecting the grad-CAM and Attention Heatmap for the image of the real dog, Figure 5.3a. Here, one can observe that both the grad-CAM and Attention Heatmap highlight the back-

ground once again and not the dog. The similarity is even more prevalent in this case because both significantly highlight a dark spot to the right more than other areas. This can be further exemplified by the results in section 4.3.1, where darker areas gain more attention from the ViT model.
This implies that the CNN and ViT examine similar parts of the image when classifying the image. This behaviour is further implied when comparing the attention heatmap with the GBP. In Figure 5.3c, we find the same area of attention, the darker spot in the original image 5.3a, as for the grad-CAM and Attention Heatmap. The similarity is even more prevalent when looking at the AI-generated image. Both GBP and the Attention Heatmap highlight the space between the dog's front legs, nose, right eye, and left back paw. For further examples see Appendix A.
In summary, all the models show that there are inherent similarities between them in which areas are deemed relevant to the image classification process even though the architectures of the models are different, as explained in section 3.2.

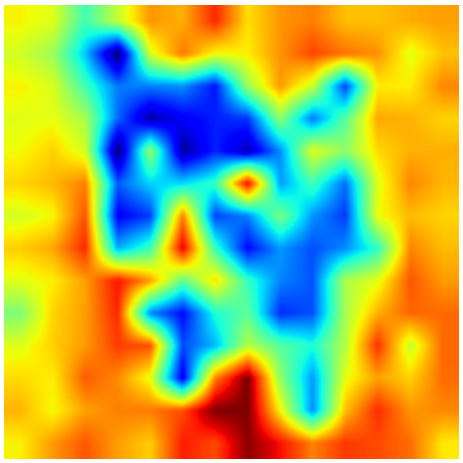
(a) Original Image



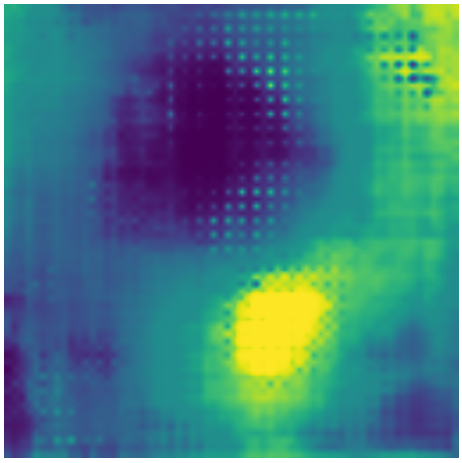
(b) CNN-ALL grad-CAM



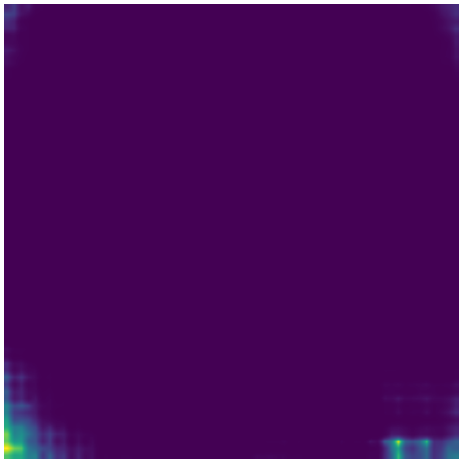
(c) CNN-ALL Guided Backpropagation



(d) ViT-ALL Attention Heatmap



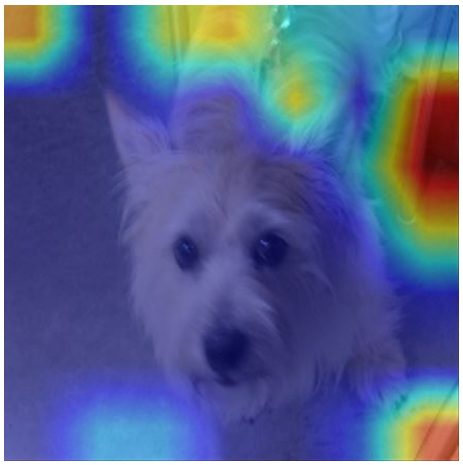
(e) FCN-FINETUNED pixel classification



(f) FCN-ALL pixel classification

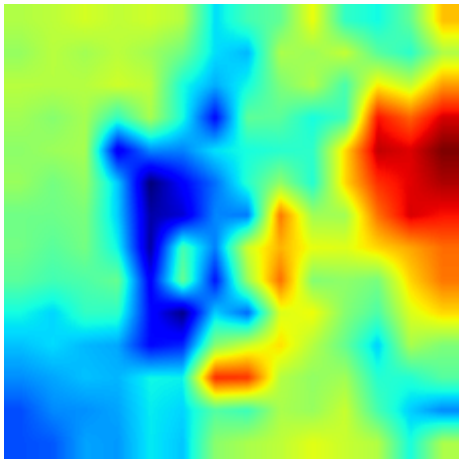
**Figure 5.2:** Model visualisations for AI images

(a) Original Image

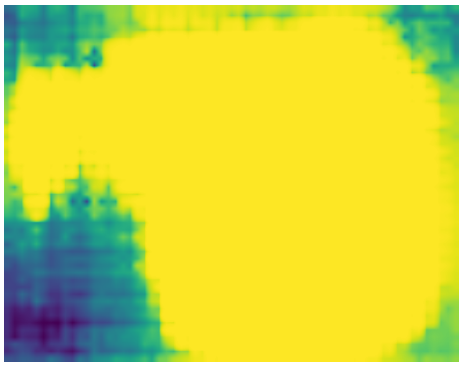


(b) CNN-ALL grad-CAM

(c) CNN-ALL Guided Backpropagation



(d) ViT-ALL Attention Heatmap



(e) FCN-FINETUNED pixel classification

(f) FCN-ALL pixel classification

**Figure 5.3:** Model visualisations for Real images

## 5.3 Applicability in the Real World

One could claim that the models in this study are trained on a dataset with images from rather outdated generators. This fact makes it troublesome for the models to

keep up with the more modern, up-to-date generators, but as seen in the results, the performance is very weak when testing on never-before-seen models. This, of course, significantly impacts how applicable the models are in the real world. In the long term, a model with insufficient performance on modern and popular generators is not applicable in "the real world". However, this issue could be handled by continuously updating the dataset with images from the latest AI generators and retraining the models on the updated dataset. This way, the models would be more robust and reliable and thus more relevant for users in the long term.
While the models constructed may not have accuracy high enough to classify random images on the internet reliably, some applications can use the model to label an image as "Suspected to be AI", which can have uses in social media platforms. In these applications, the model would also ideally favour false negatives over false positives, as false positives in large amounts would generally lead to a bad reputation for the model. Systems can also be built around this kind of labelling, such as flagging users who have had multiple images to be given the label.

# 6
# Conclusion

The project has been a partial success, the CNN and FCN models got much better accuracies on test sets relevant to training data than was expected, and is at a level where the models can reliably give a classification of if an image was generated by any of the generative models it was trained on. However, with arbitrary generative models, the results do not seem good enough to promote to a user as trustworthy. However, with the construction of more exhaustive and modern training data sets, this problem might be avoidable.
The ViT models suffer from a completely different problem: too low overall accuracy but without the deterioration of accuracy when testing on new generative models. We believe that with better computational resources and perhaps even more training data, this is also a good approach for creating a good classification model without retraining the model on newly released generative models.

## 6.1 Further Work

The most interesting question that remains after this project is: "Could the visualisation techniques in our tool help a human assess the validity of an image?" We do not have time to carry out this study during the limited time of the project. Still, it could be interesting to answer in the future by testing human AI classification accuracy for users with and without access to the tool.

# Bibliography


[1] V. Powell, "Image kernels, setosa.io," (Accessed on 2024-05-08). [Online]. Available: https://setosa.io/ev/image-kernels/

[2] Y. Muhamad, S. Irawan, and C. Setianingsih, "Application of transfer learning using convolutional neural network method for early detection of terry's nail," *Journal of Physics: Conference Series*, vol. 1201, p. 012052, 05 2019, the image is licensed under CC BY 3.0.

[3] M. Xiang. (2020) Convolutions: Transposed and deconvolution, medium.com. Accessed on 2024-03-16. [Online]. Available: https://medium.com/@marsxiang/convolutions-transposed-and-deconvolution-6430c358a5b6

[4] R. R. Selvaraju *et al.*, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, p. 336–359, Oct. 2019, (Accessed on 2024-05-08). [Online]. Available: http://dx.doi.org/10.1007/s11263-019-01228-7

[5] P. Verma. (2023) The rise of ai fake news is creating a 'misinformation superspreader, washingtonpost.com'. Accessed on 2024-01-22. [Online]. Available: https://www.washingtonpost.com/technology/2023/12/17/ai-fake-news-misinformation/

[6] T. Acres. (12,13,2023) Fake ai images keep going viral - here are eight that have caught people out, news.sky.com. Accessed on 2024-01-22. [Online]. Available: https://news.sky.com/story/fake-ai-images-keep-going-viral-here-are-eight-that-have-caught-people-out-13028547

[7] R. Verma. (2023) Ai-generated images and videos: A game-changer or a threat to authenticity? Accessed on 2024-01-22. [Online]. Available: https://www.businessinsider.in/tech/news/ai-generated-images-and-videos-a-game-changer-or-a-threat-to-authenticity/articleshow/99560443.cms

[8] aiornot, "How can ai-generation photos can harm each of us, aiornot.com," 2023, accessed on 2024-01-25. [Online]. Available: https://www.aiornot.com/blog/how-can-ai-generation-photos-can-harm-each-of-us

[9] M. Zhu *et al.*, "Gendet: Towards good generalizations for ai-generated image detection," 2023, arXiv:2312.08880 [cs.CV], (Accessed on 2024-05-08). [Online]. Available: https://arxiv.org/abs/2312.08880

[10] ——, "Genimage: A million-scale benchmark for detecting ai-generated image," 2023, arXiv:2306.08571 [cs.CV], (Accessed on 2024-05-08). [Online]. Available: https://arxiv.org/abs/2306.08571

[11] J. Betker *et al.*, "Improving image generation with better captions," *openai*, 2023.

[12] L. F. S. Scabini and O. M. Bruno, "Structure and performance of fully connected neural networks: Emerging complex network properties," 2021, arXiv:2107.14062 [cs.LG], (Accessed on 2024-05-08). [Online]. Available: https://arxiv.org/abs/2107.14062

[13] M. Kirk, *Thoughtful machine learning with python*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, Feb. 2017.

[14] S. Doshi, "Various optimization algorithms for training neural network, towardsdatascience.com," 2019, (Accessed on 2024-05-02). [Online]. Available: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6#:~:text=Optimizers%20are%20algorithms%20or%20methods,help%20to%20get%20results%20faster

[15] L. Craig. (2024) convolutional neural network (cnn). Accessed on 2024-01-28. [Online]. Available: https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network

[16] F. Chollet, *Deep Learning with Python*, 1st ed. Shelter Island, NY, USA: Manning Publications, 2017.

[17] E. Stevens, L. Antiga, and T. Viehmann, *Deep Learning with Pytorch*, 2nd ed. Shelter Island, NY, USA: Manning Publications, 2020.

[18] B. B. Traore, B. Kamsu-Foguem, and F. Tangara, "Deep convolution neural network for image recognition," *Ecological Informatics*, vol. 48, pp. 257–268, 2018.

[19] N. Adaloglou, "Intuitive explanation of skip connections in deep learning, theaisummer.com," 2020, (Accessed on 2024-04-24). [Online]. Available: https://theaisummer.com/skip-connections/#u-nets-long-skip-connections

[20] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," 2015, arXiv:1411.4038 [cs.CV], Accessed on 2024-05-07. [Online]. Available: https://doi.org/10.48550/arXiv.1411.4038

[21] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021, arXiv:2304.13023 [cs.AI],(Accessed on 2024-05-08). [Online]. Available: https://arxiv.org/abs/2304.13023

[22] A. Vaswani *et al.*, "Attention is all you need," 2023, arXiv:1706.03762 [cs.CL], (Accessed on 2024-05-08). [Online]. Available: https://arxiv.org/abs/1706.03762

[23] R. E. Turner, "An introduction to transformers," 2024, arXiv:2304.10557 [cs.LG],(Accessed on 2024-05-08). [Online]. Available: https://doi.org/10.48550/arXiv.2304.10557

[24] L. Min, C. Qiang, and Y. Shuicheng, "Network in network," 2013, arXiv:1312.4400 [cs.NE],(Accessed on 2024-05-08). [Online]. Available: https://arxiv.org/abs/1312.4400

[25] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," June 2016, arXiv:1512.04150 [cs.CV], Accessed on 2024-05-07. [Online]. Available: https://doi.org/10.48550/arXiv.1512.04150

[26] S. Mostafa *et al.*, "Leveraging guided backpropagation to select convolutional neural networks for plant classification," *Frontiers in Artificial Intelligence*,

vol. 5, May 2022. [Online]. Available: http://dx.doi.org/10.3389/frai.2022.871162

[27] J. Vaicenavicius *et al.*, "Evaluating model calibration in classification," in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and M. Sugiyama, Eds., vol. 89. PMLR, 16–18 Apr 2019, pp. 3459–3467. [Online]. Available: https://proceedings.mlr.press/v89/vaicenavicius19a.html

[28] J. Platt, "Probablistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in Large Margin Classifiers*, 2000.

[29] R. Rombach and P. Esser, an image generator from Stable diffusion, fine-tuned for inpainting. [Online]. Available: https://huggingface.co/runwayml/stable-diffusion-inpainting

[30] Software used to remove backgrounds from images: https://www.remove.bg/.

[31] T. Lin *et al.*, "Microsoft coco: Common objects in context," *arXiv*, 2015, arXiv:1405.0312 [cs.CV], (Accessed on 2024-05-08). [Online]. Available: https://arxiv.org/abs/1405.0312

[32] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[33] N. Malingan. (2023) Transformer visualization and explainabilitys, scaler.com. Accessed on 2024-03-31. [Online]. Available: https://www.scaler.com/topics/nlp/transformers-interpret/

# A

# Output from Models

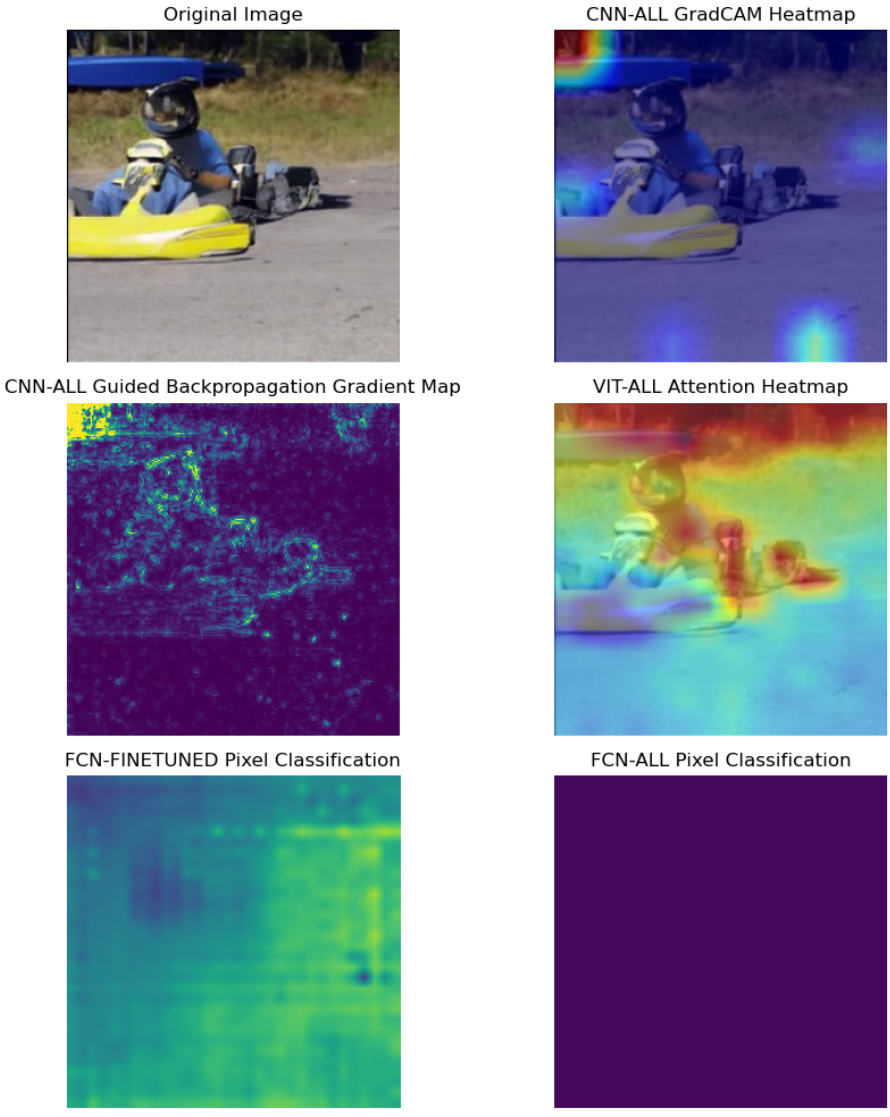



image generated with ADM

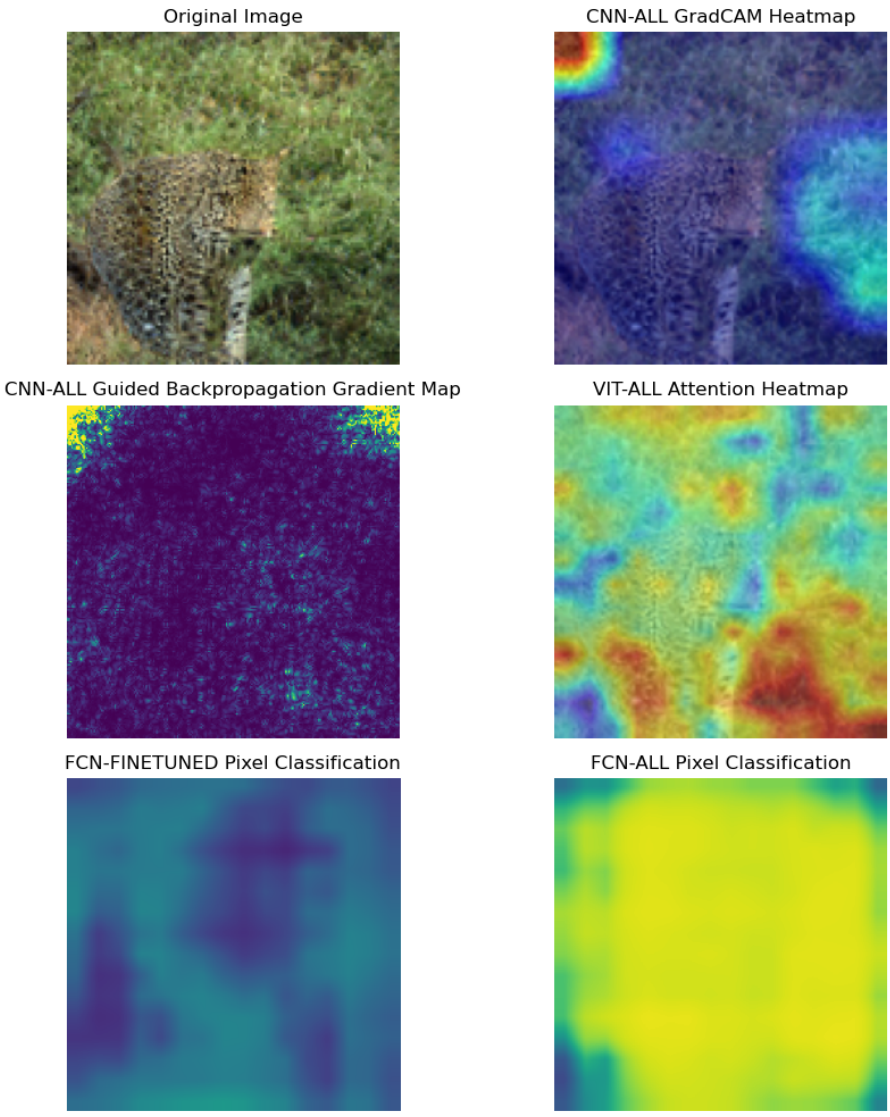



image generated with BIGGAN

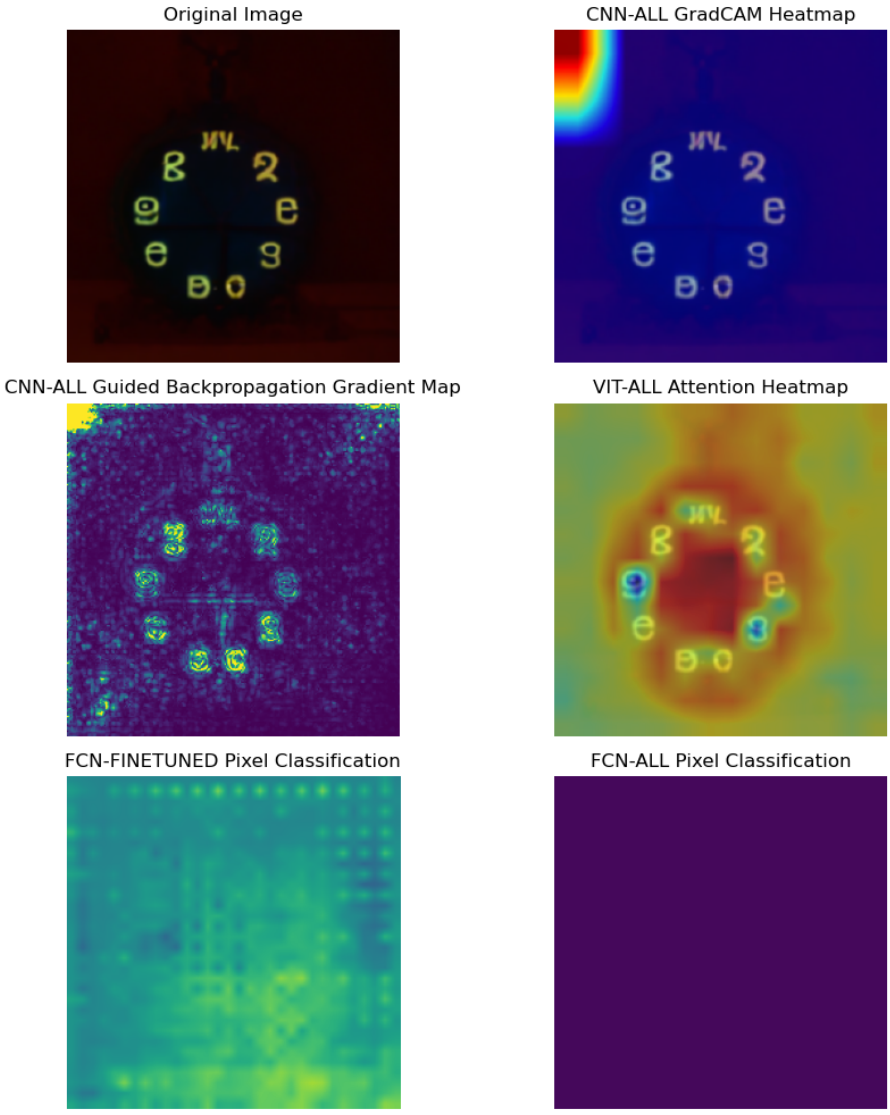
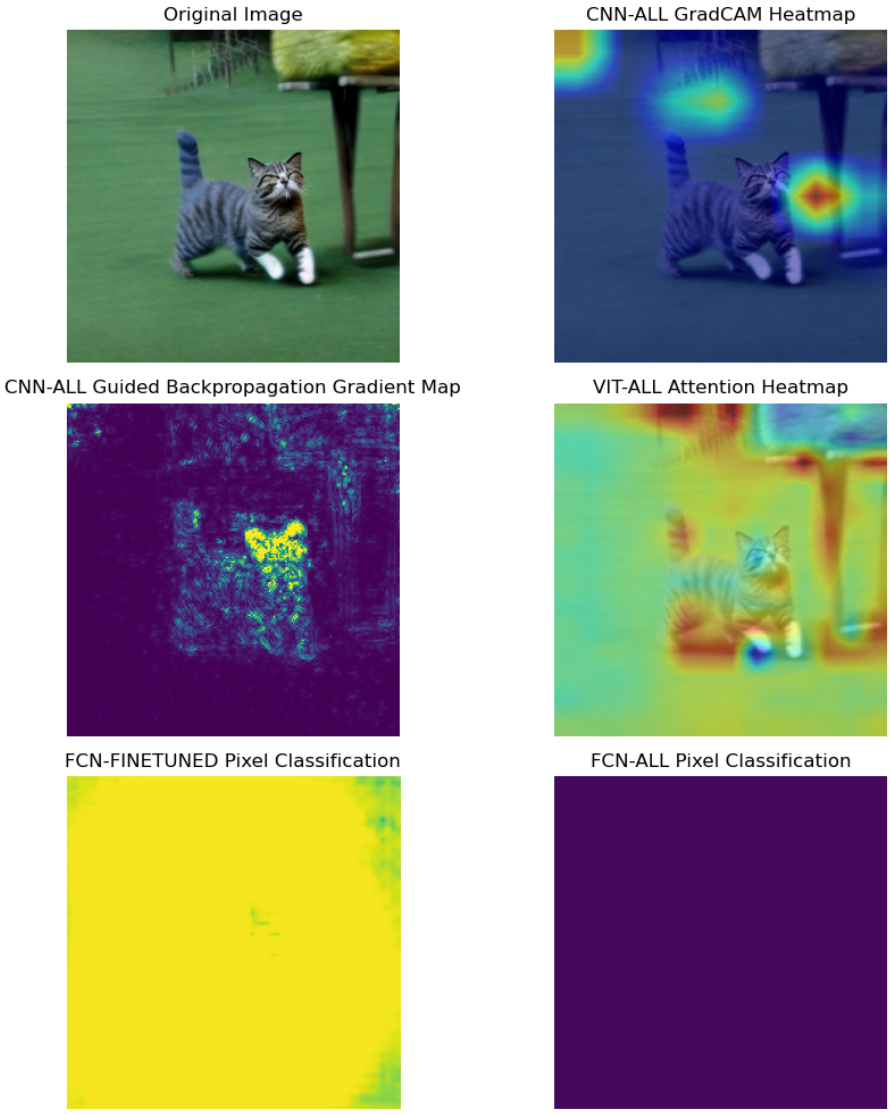
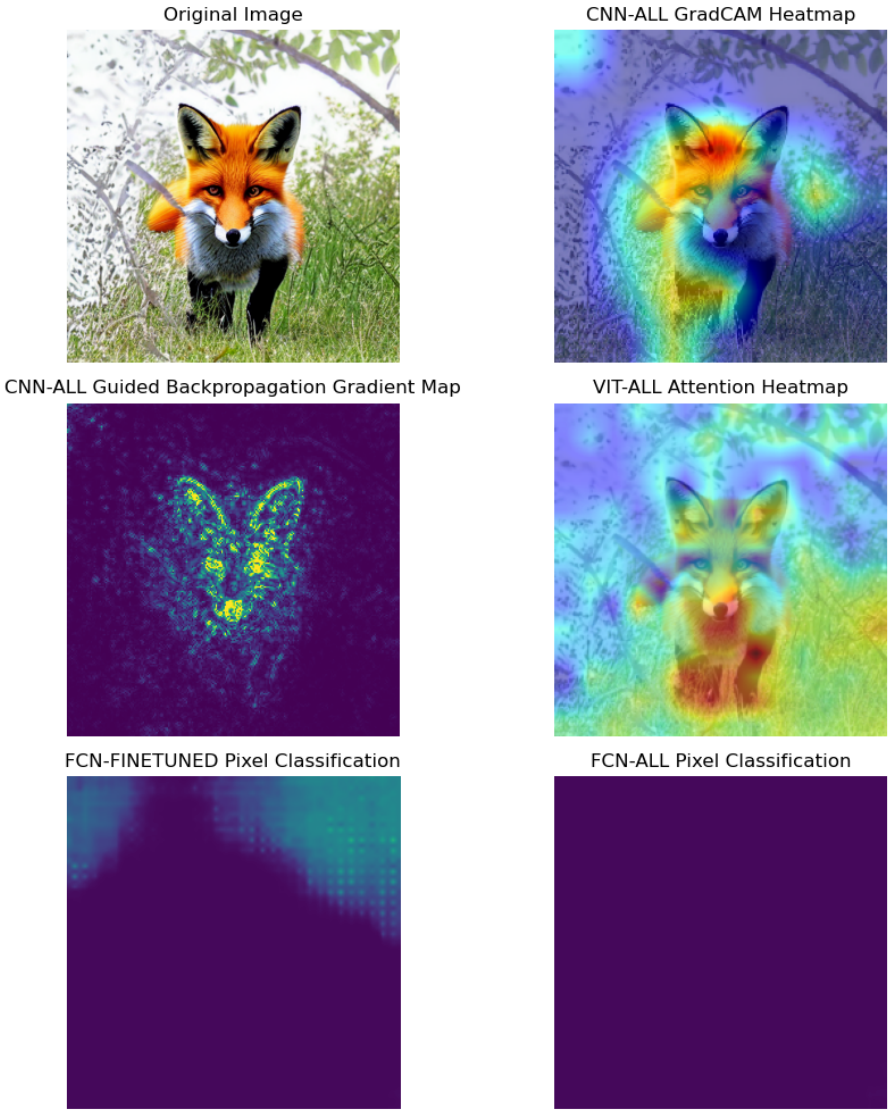


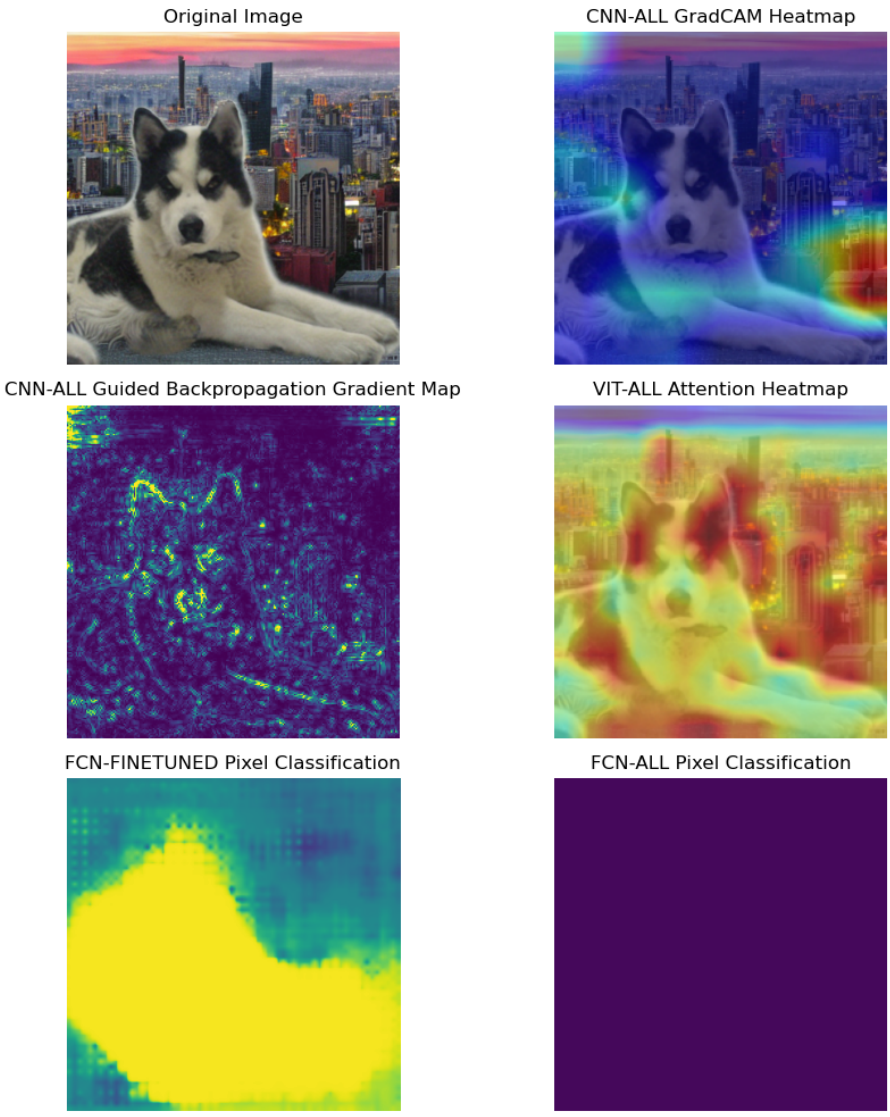

image generated with GLIDE



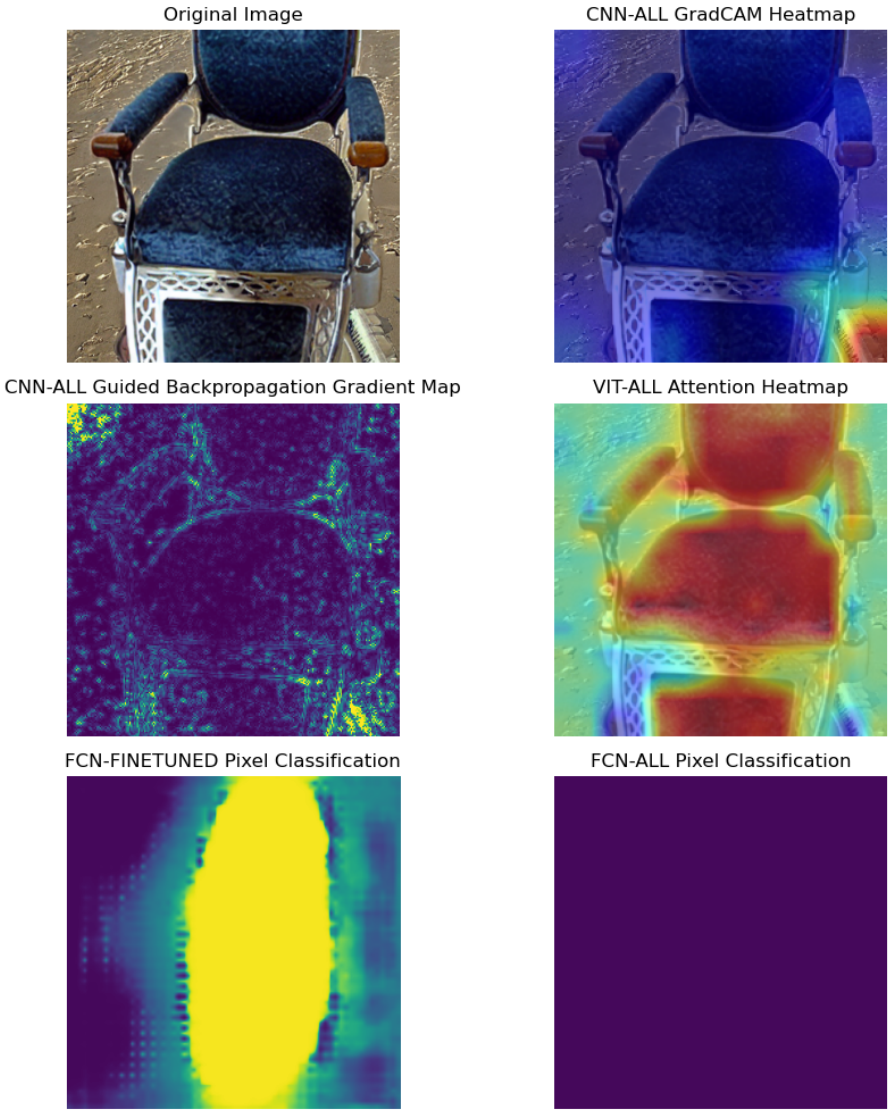

Inpainted image with AI-generated object




Inpainted image with AI-generated object




Inpainted image with AI-generated background




Inpainted image with AI-generated background

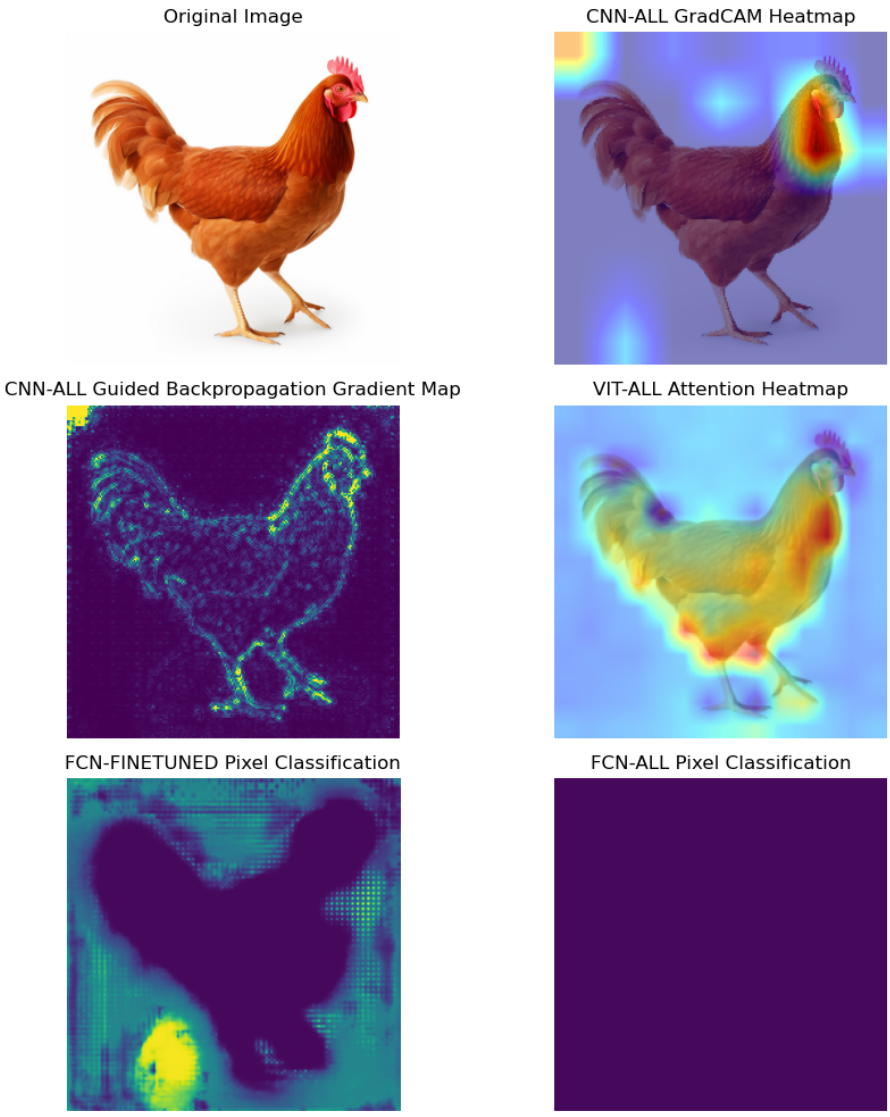
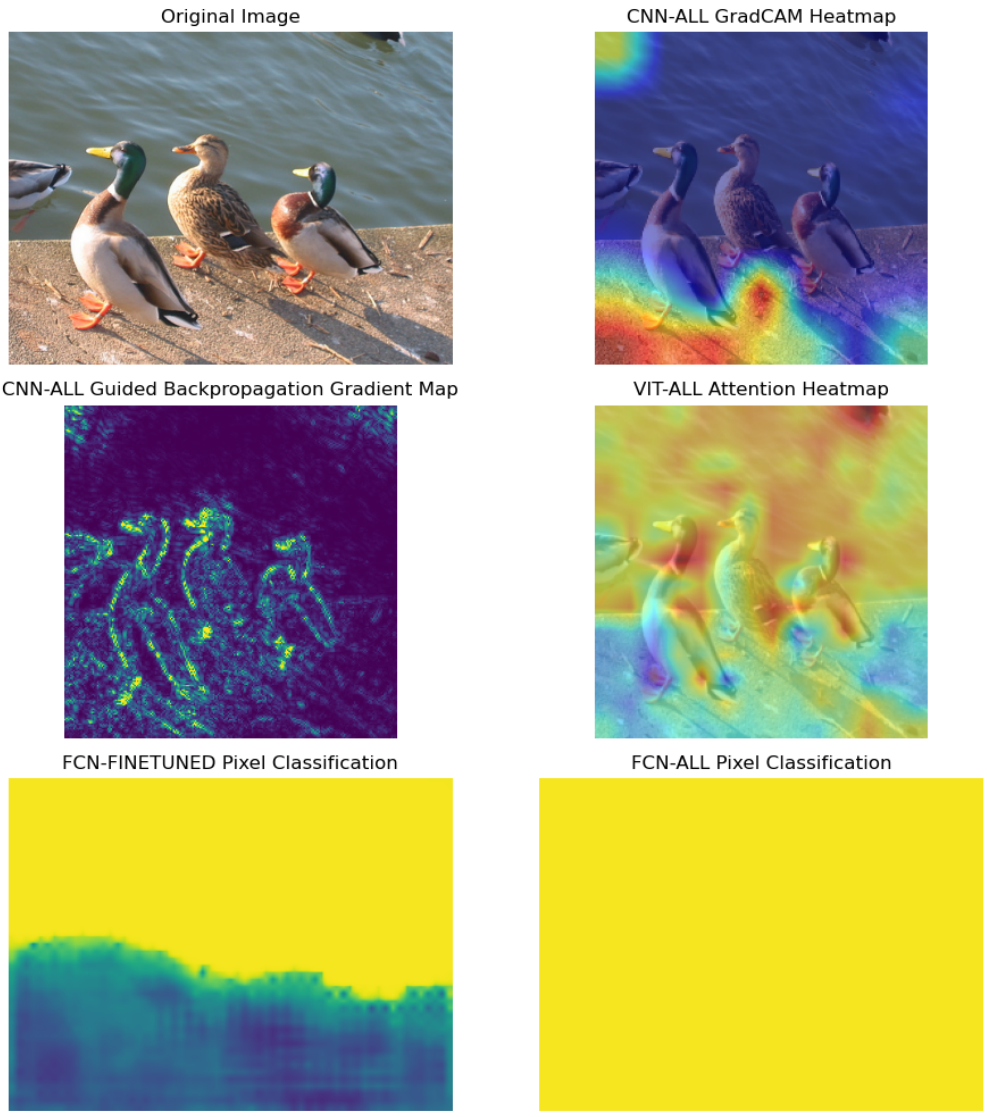
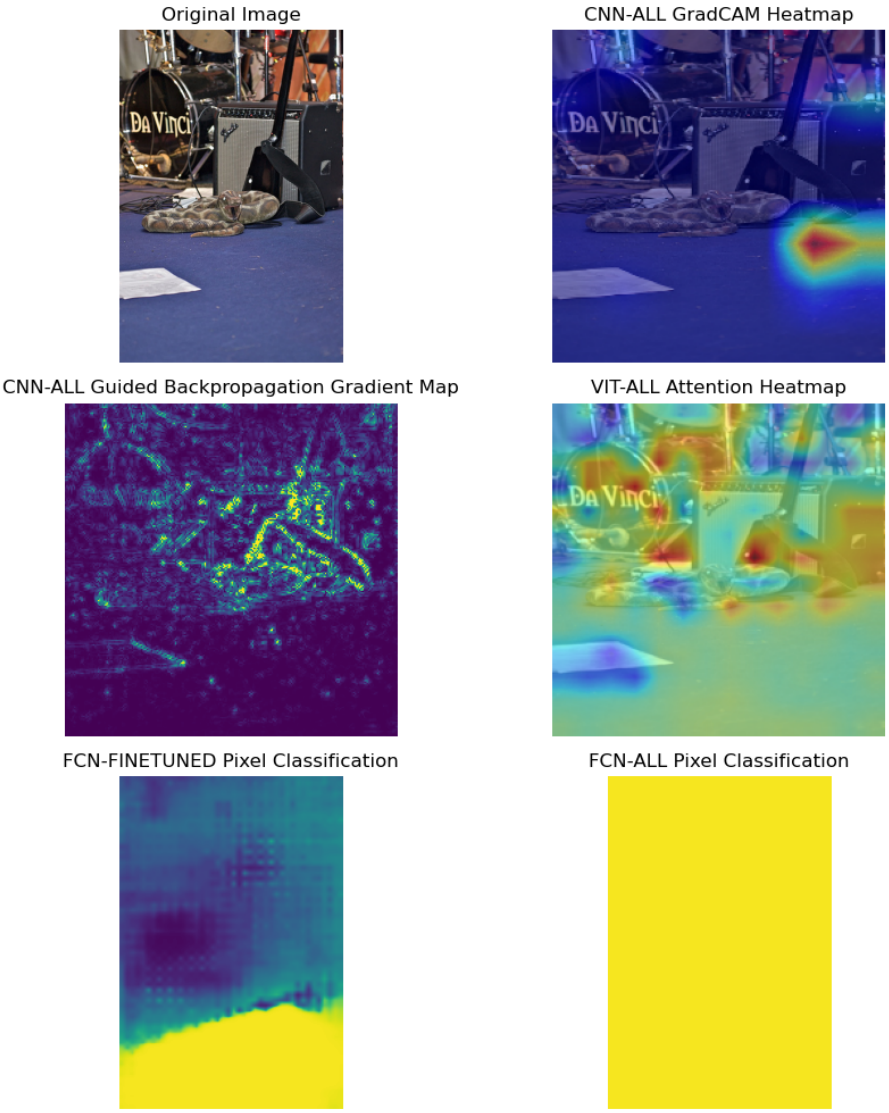


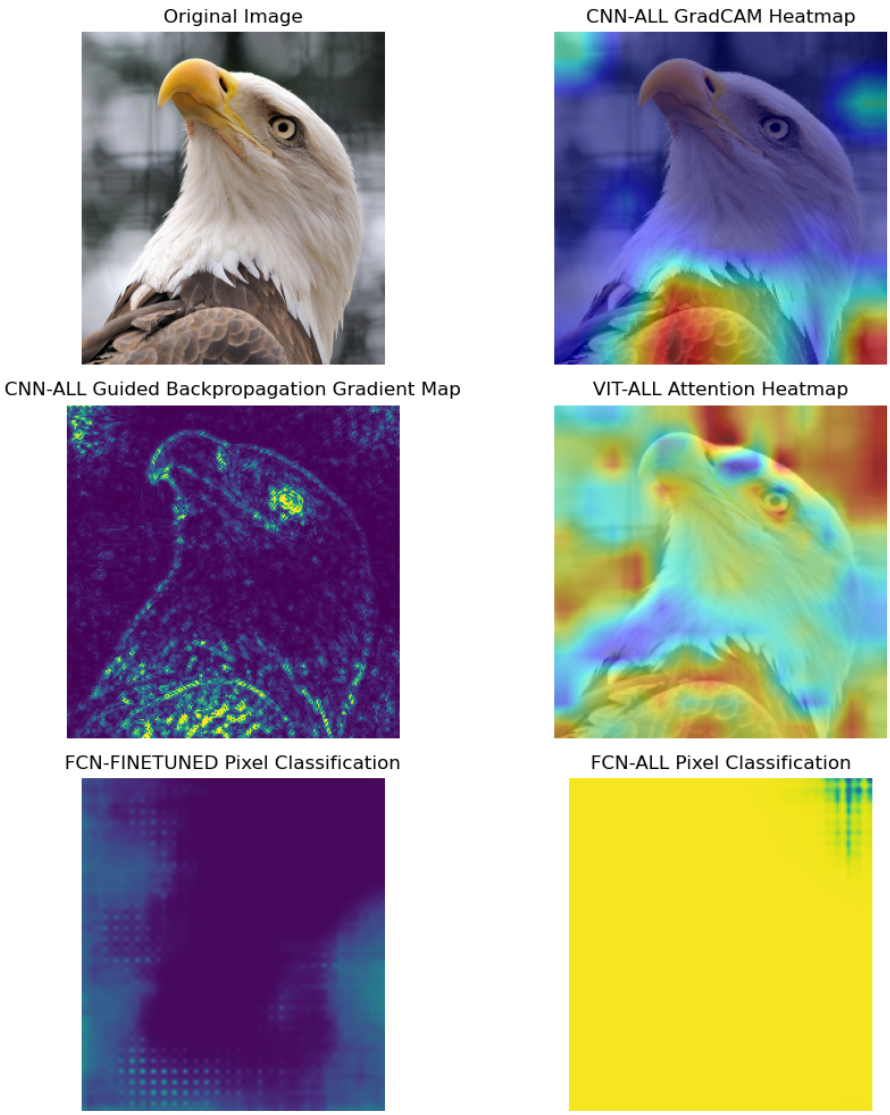

image generated with MIDJOURNEY


Original Image
CNN-ALL GradCAM Heatmap
CNN-ALL Guided Backpropagation Gradient Map
VIT-ALL Attention Heatmap
FCN-FINETUNED Pixel Classification
FCN-ALL Pixel Classification

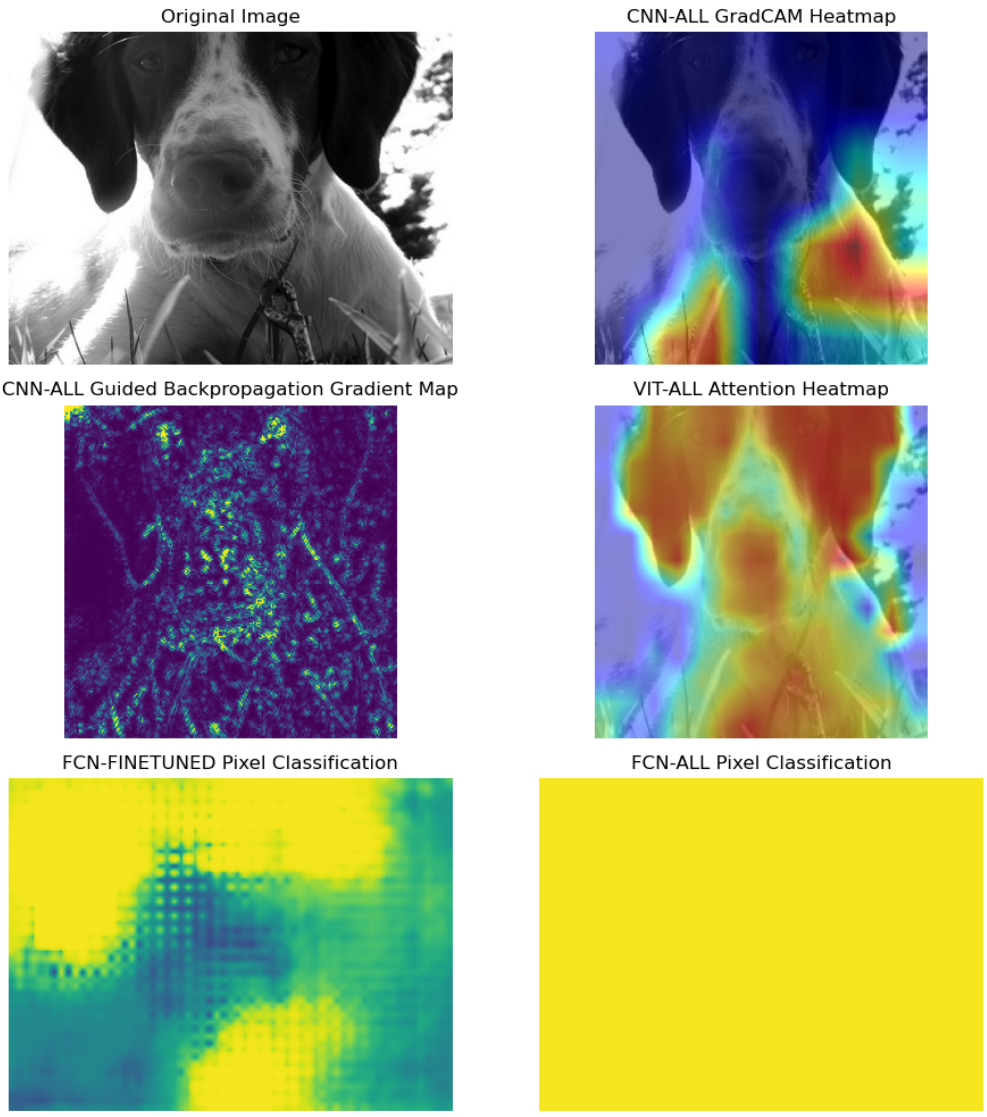
real image


Original Image
CNN-ALL GradCAM Heatmap
CNN-ALL Guided Backpropagation Gradient Map
VIT-ALL Attention Heatmap
FCN-FINETUNED Pixel Classification
FCN-ALL Pixel Classification

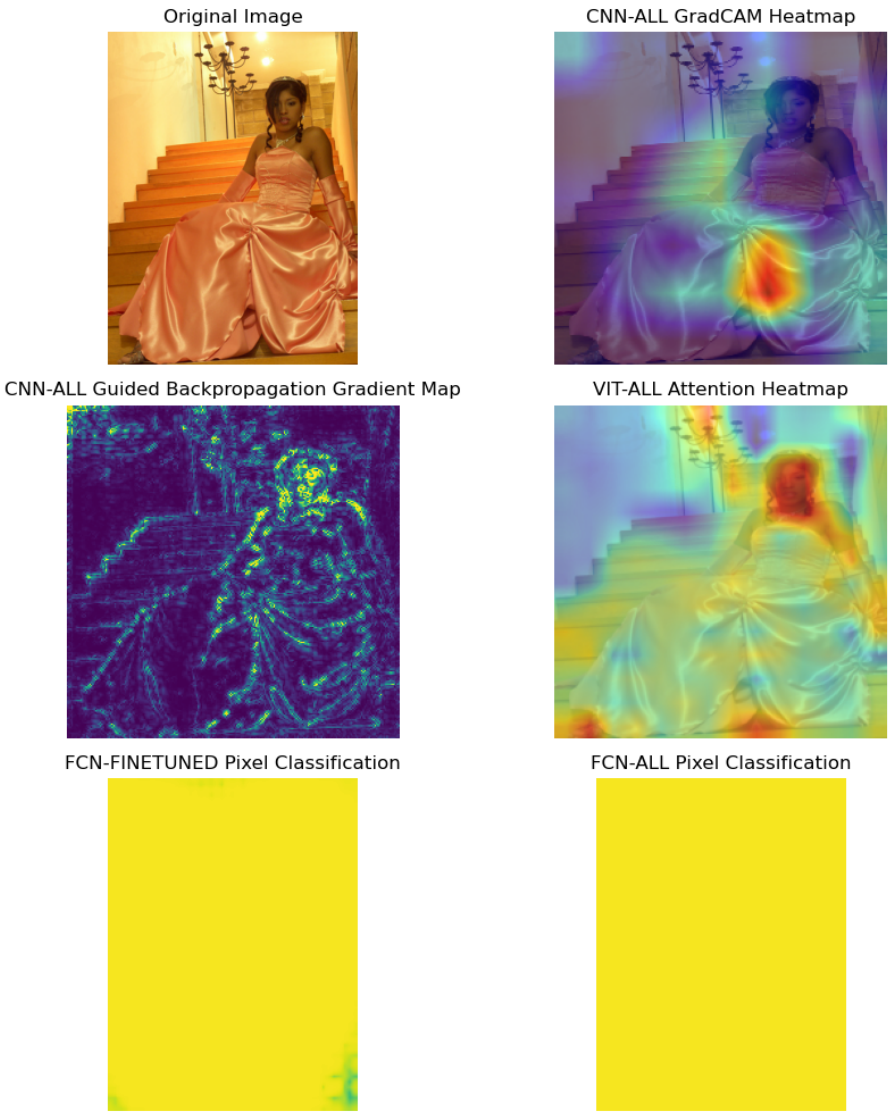
real image


Original Image
CNN-ALL GradCAM Heatmap
CNN-ALL Guided Backpropagation Gradient Map
VIT-ALL Attention Heatmap
FCN-FINETUNED Pixel Classification
FCN-ALL Pixel Classification

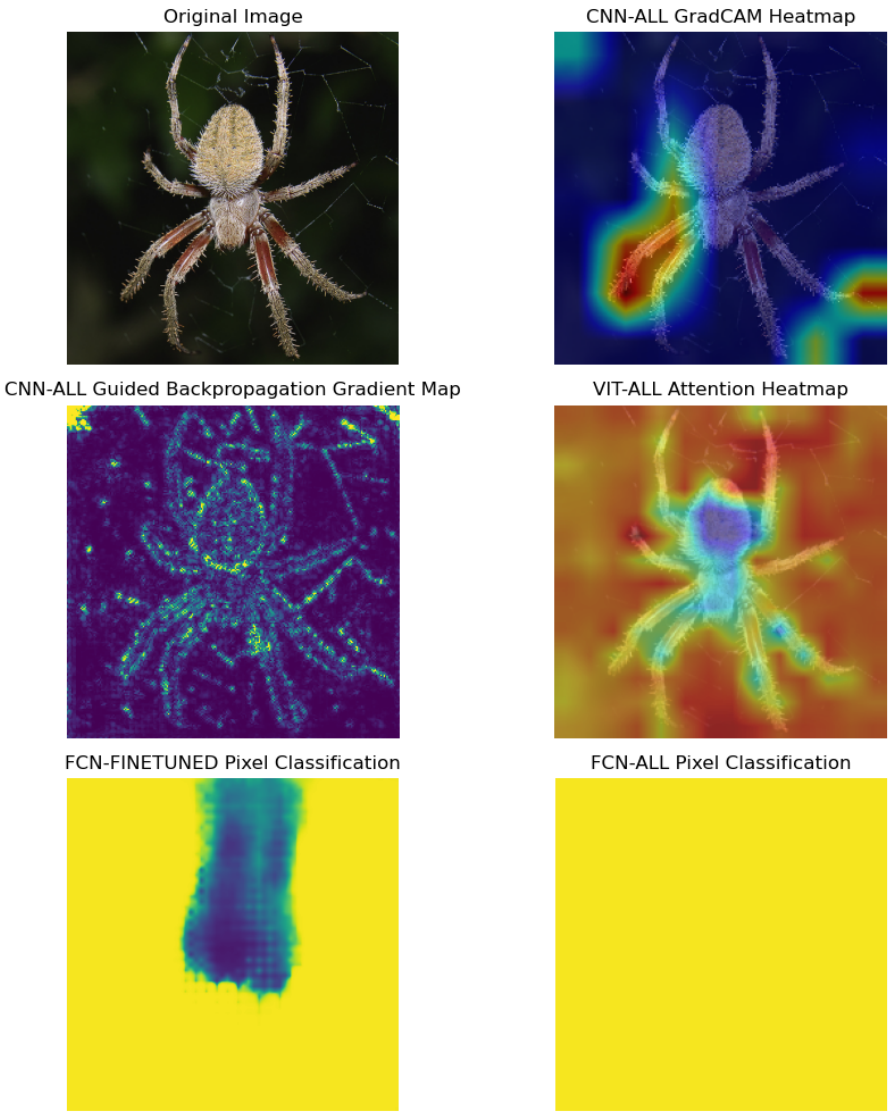
real image


Original Image
CNN-ALL GradCAM Heatmap
CNN-ALL Guided Backpropagation Gradient Map
VIT-ALL Attention Heatmap
FCN-FINETUNED Pixel Classification
FCN-ALL Pixel Classification

real image


Original Image
CNN-ALL GradCAM Heatmap
CNN-ALL Guided Backpropagation Gradient Map
VIT-ALL Attention Heatmap
FCN-FINETUNED Pixel Classification
FCN-ALL Pixel Classification

real image


Original Image
CNN-ALL GradCAM Heatmap
CNN-ALL Guided Backpropagation Gradient Map
VIT-ALL Attention Heatmap
FCN-FINETUNED Pixel Classification
FCN-ALL Pixel Classification

real image

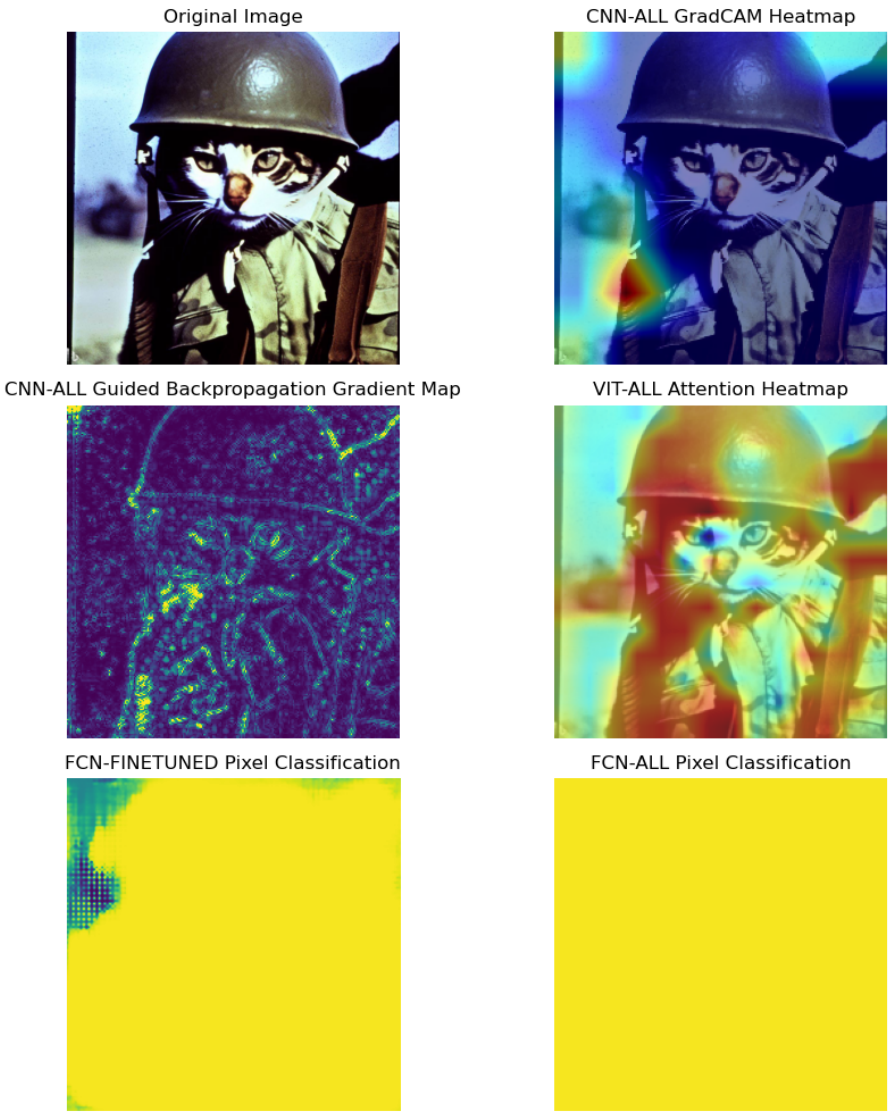



image generated with DALLE-3

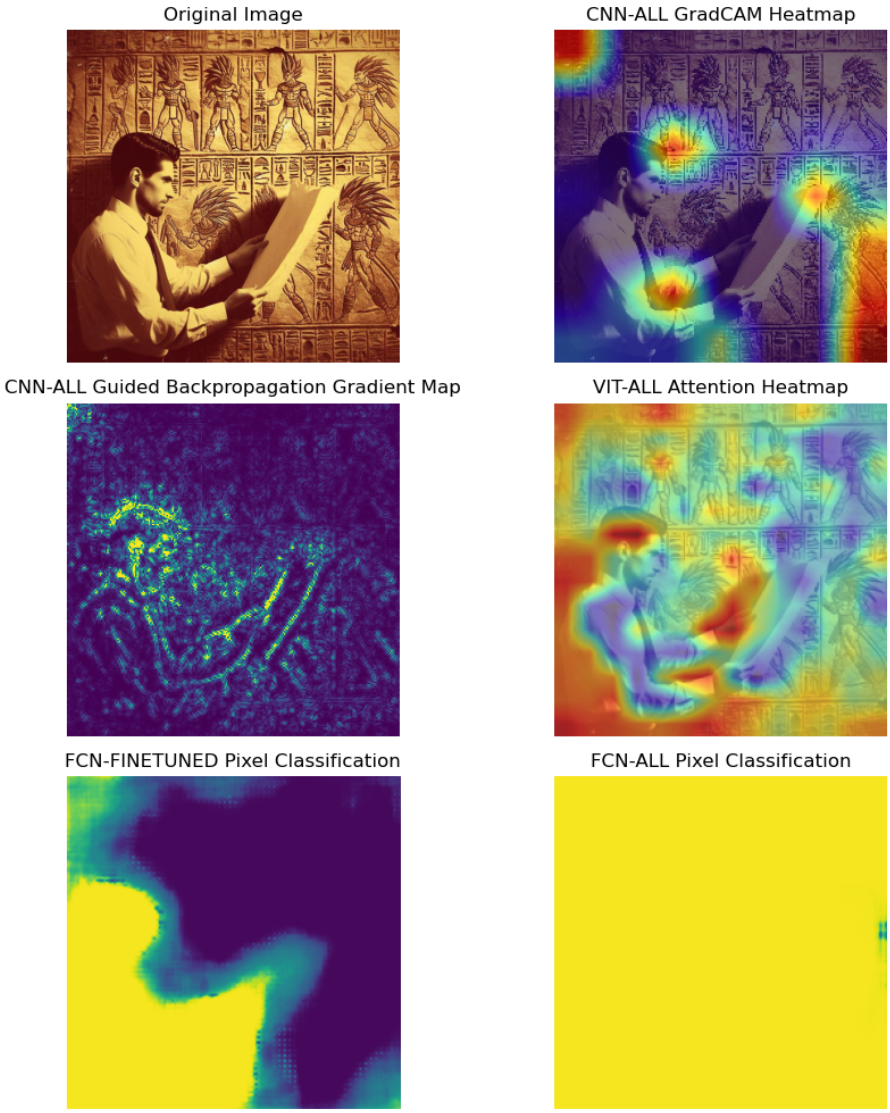



image generated with DALLE-3

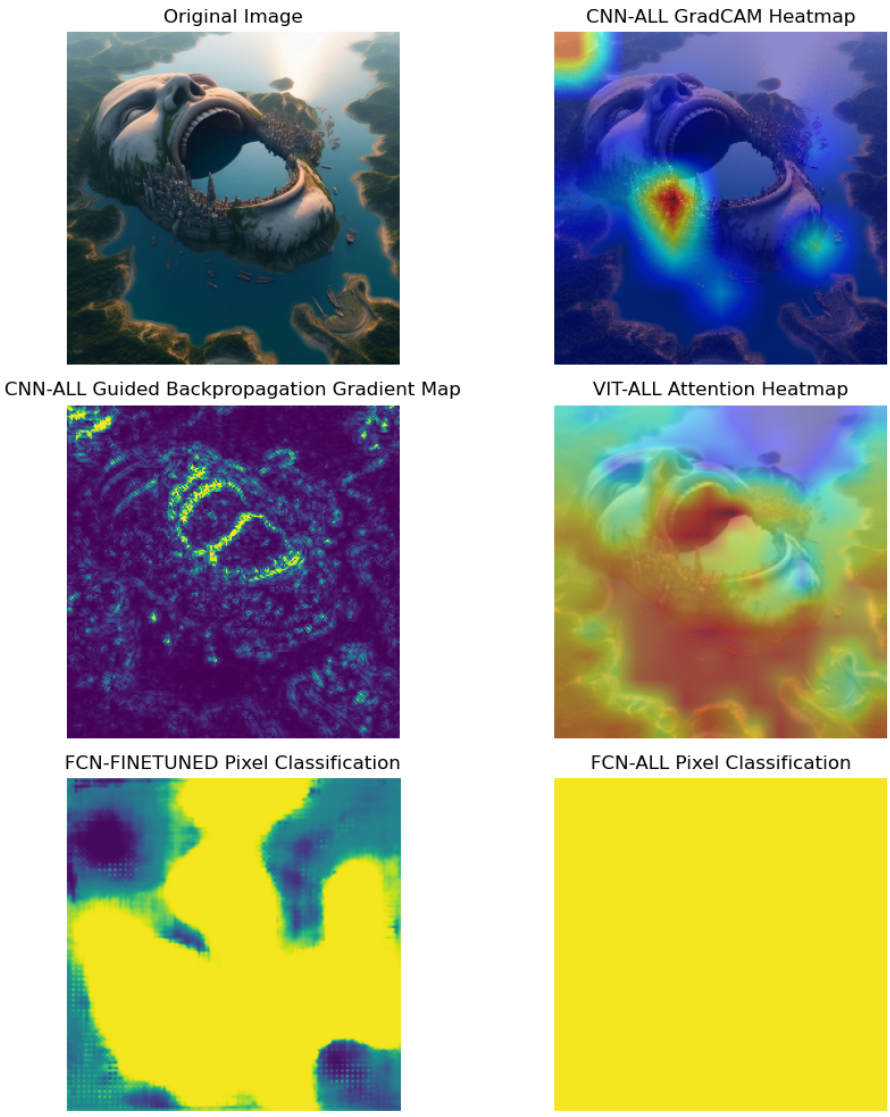



image generated with DALLE-3

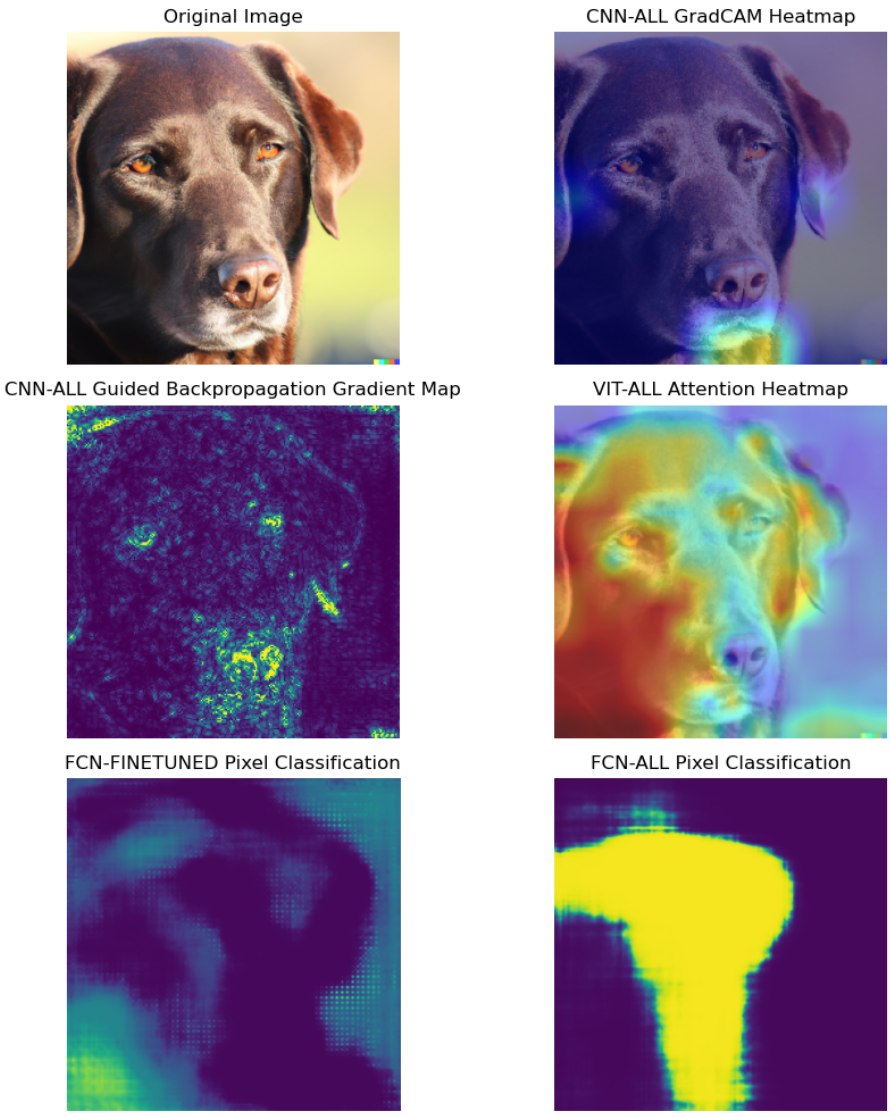



image generated with DALLE-3

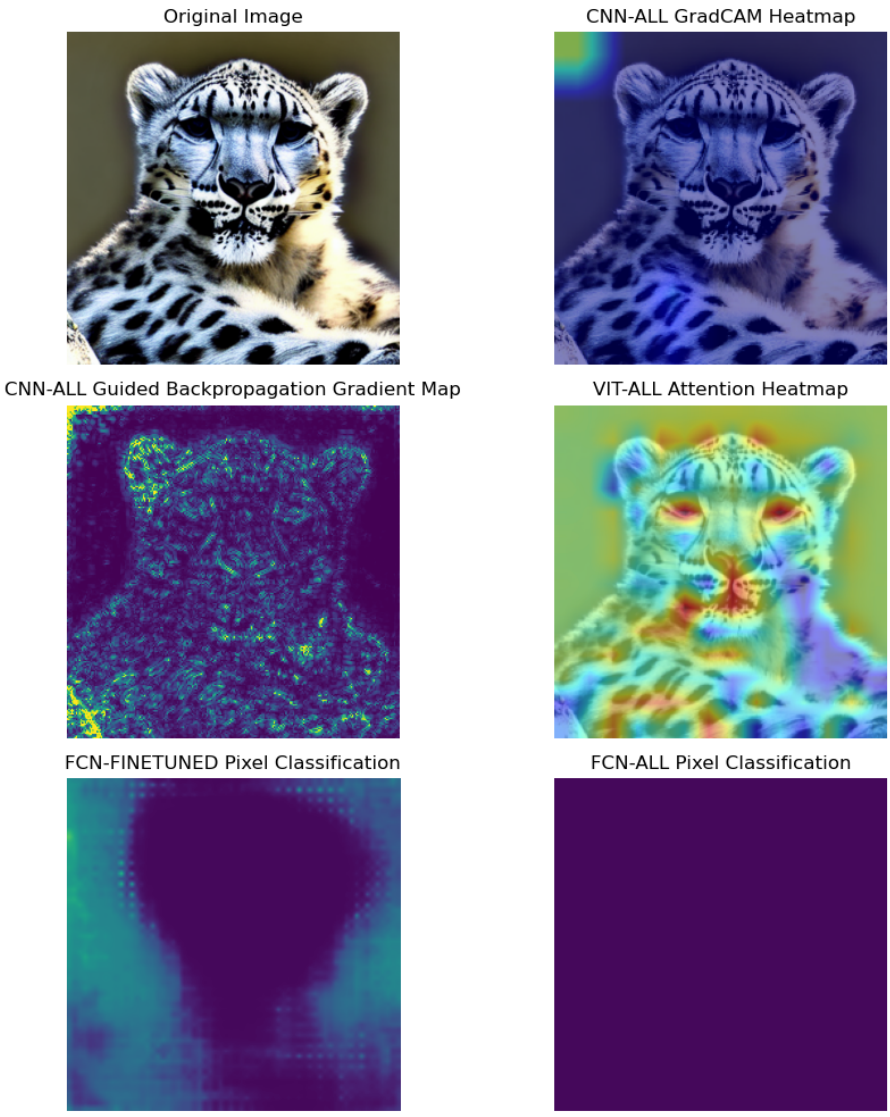



image generated with SDV 1.4

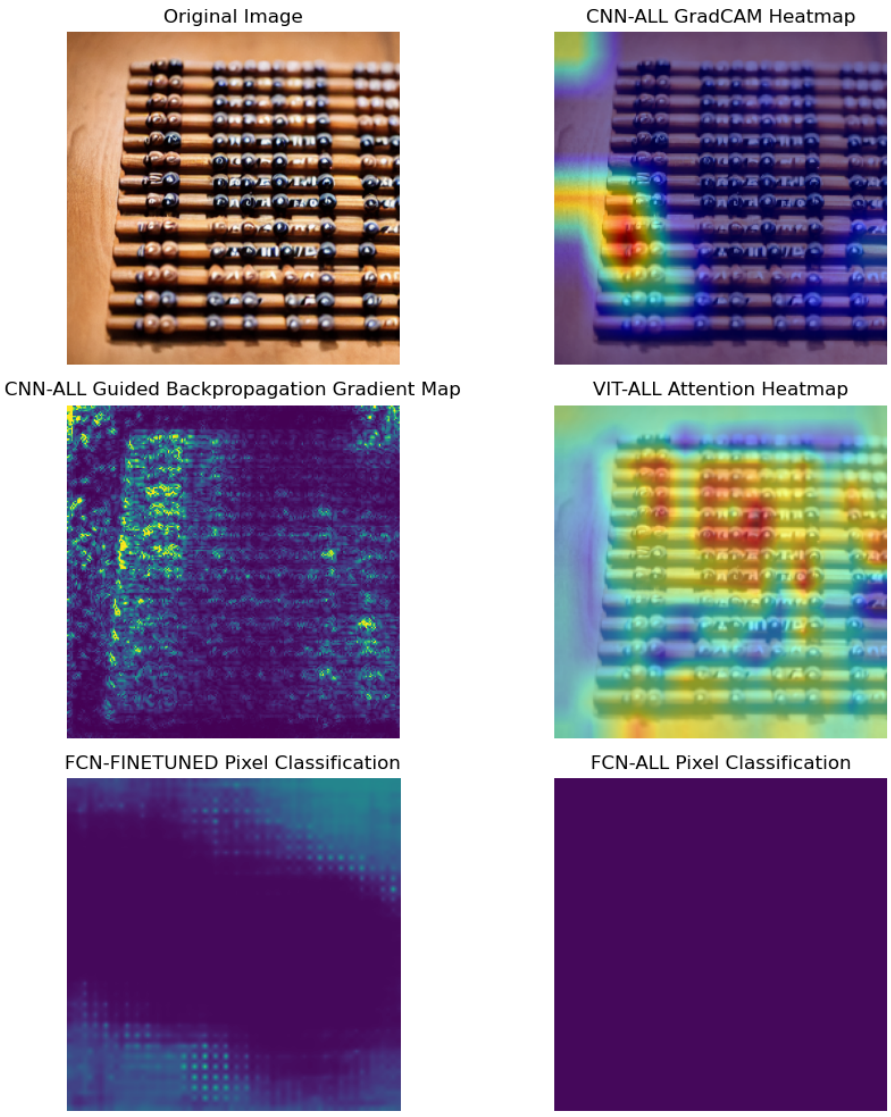



image generated with SDV 1.4

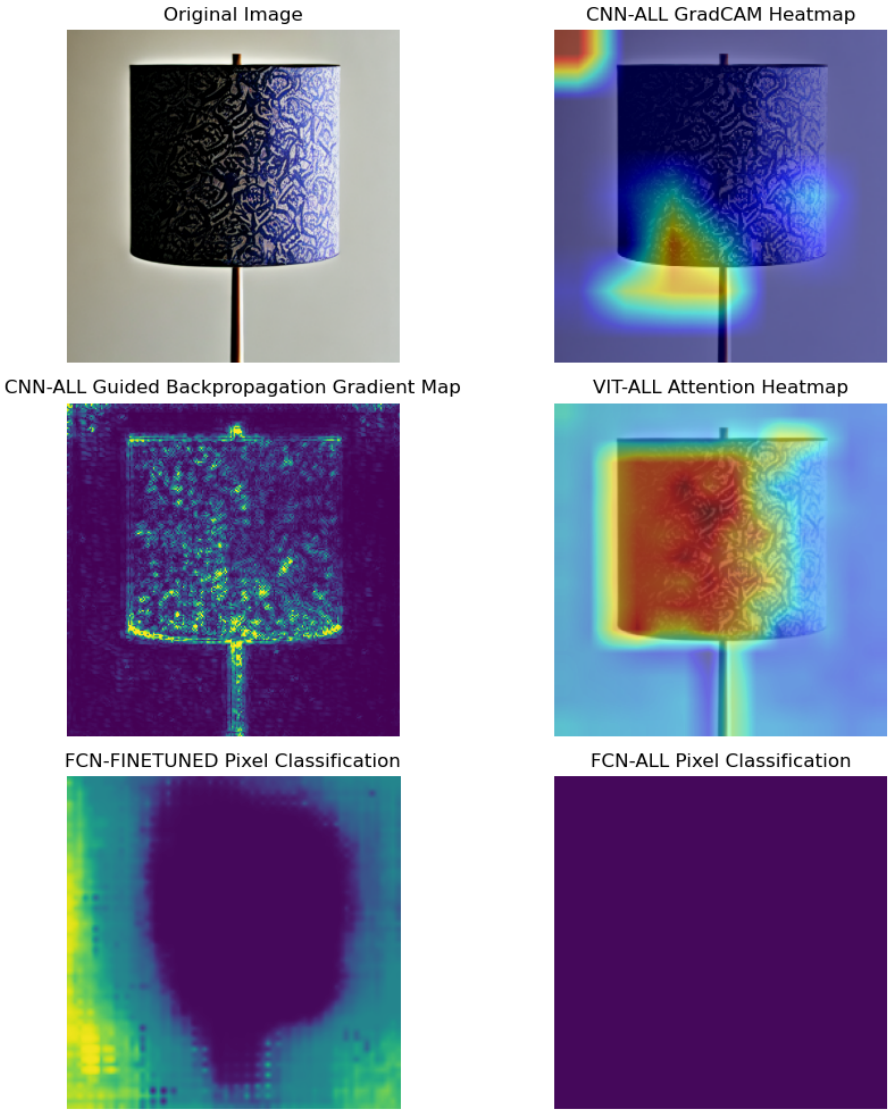



image generated with SDV 1.4

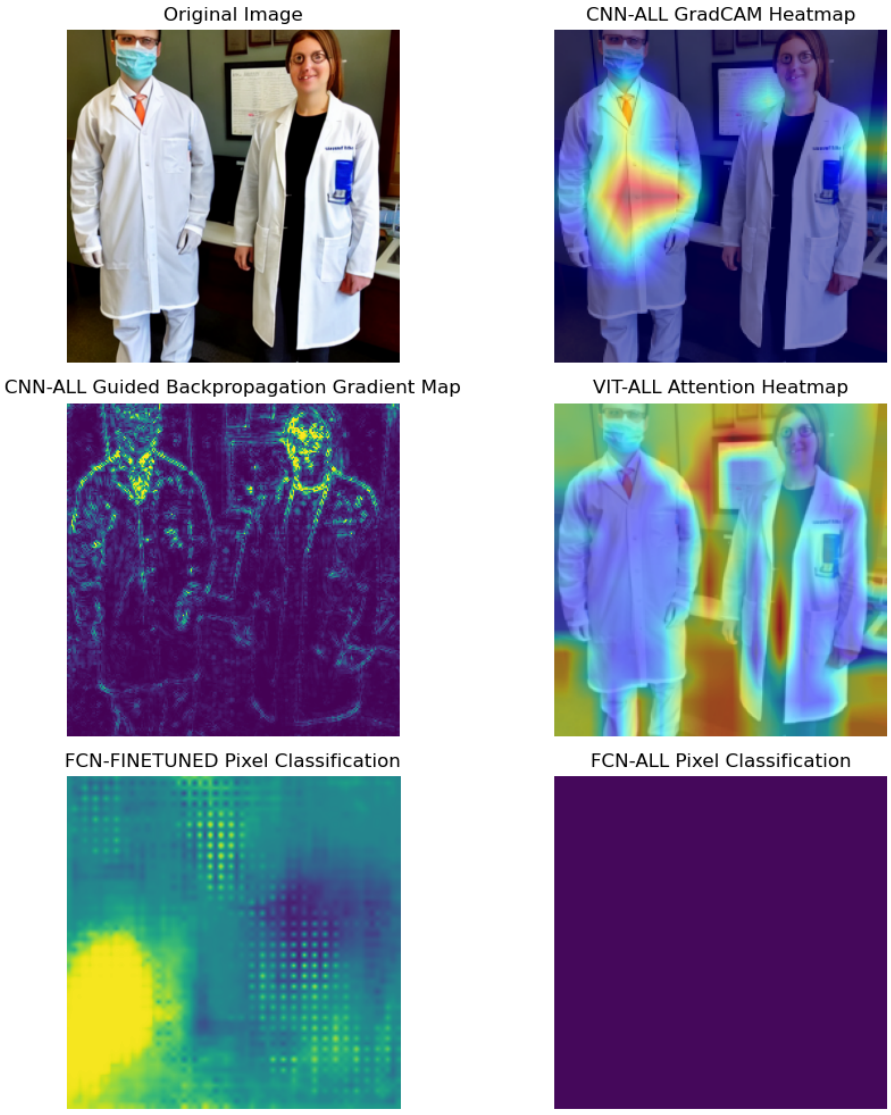



image generated with SDV 1.4

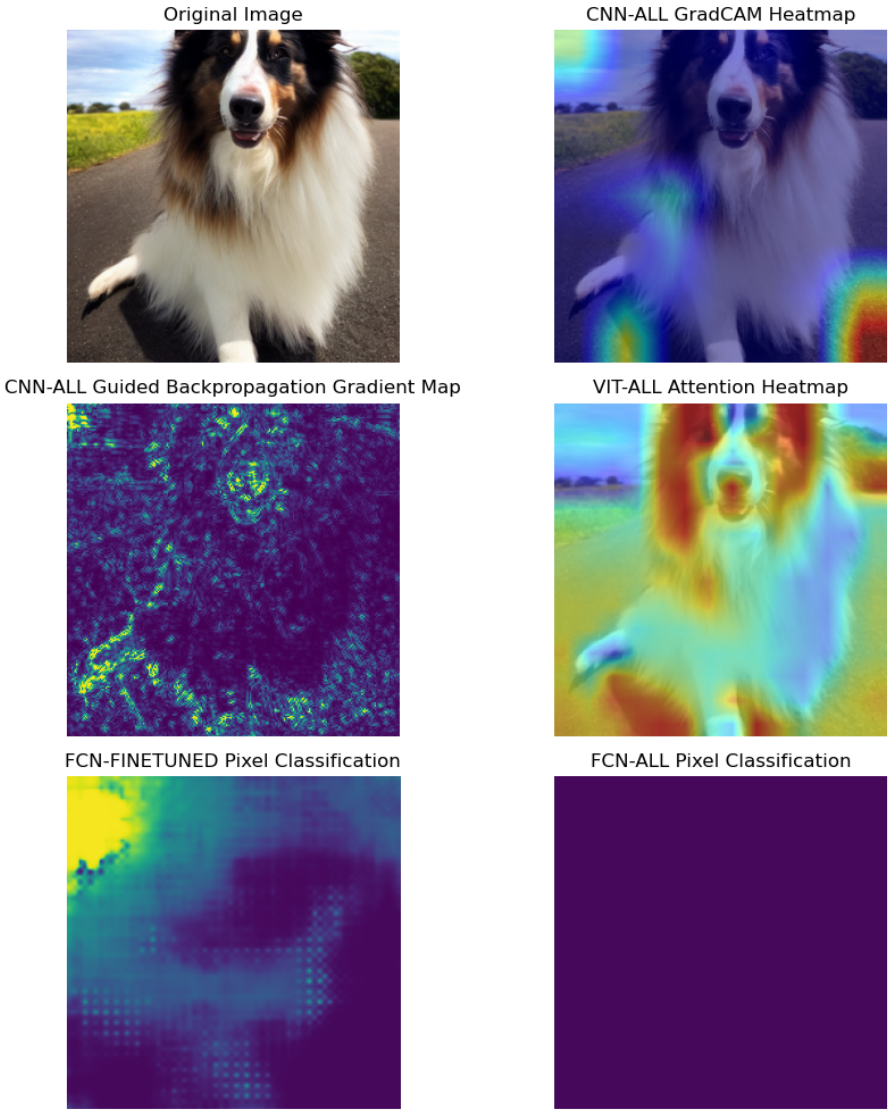



image generated with SDV 1.5

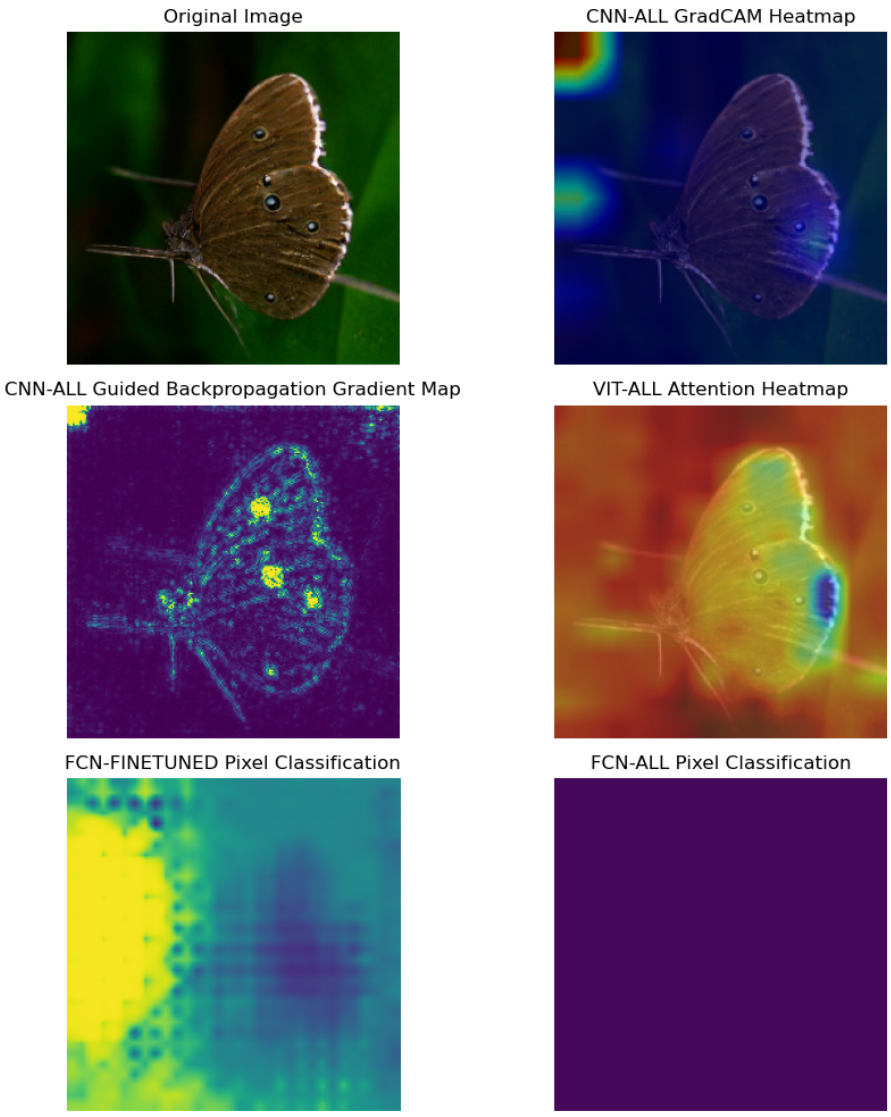



image generated with VQDM

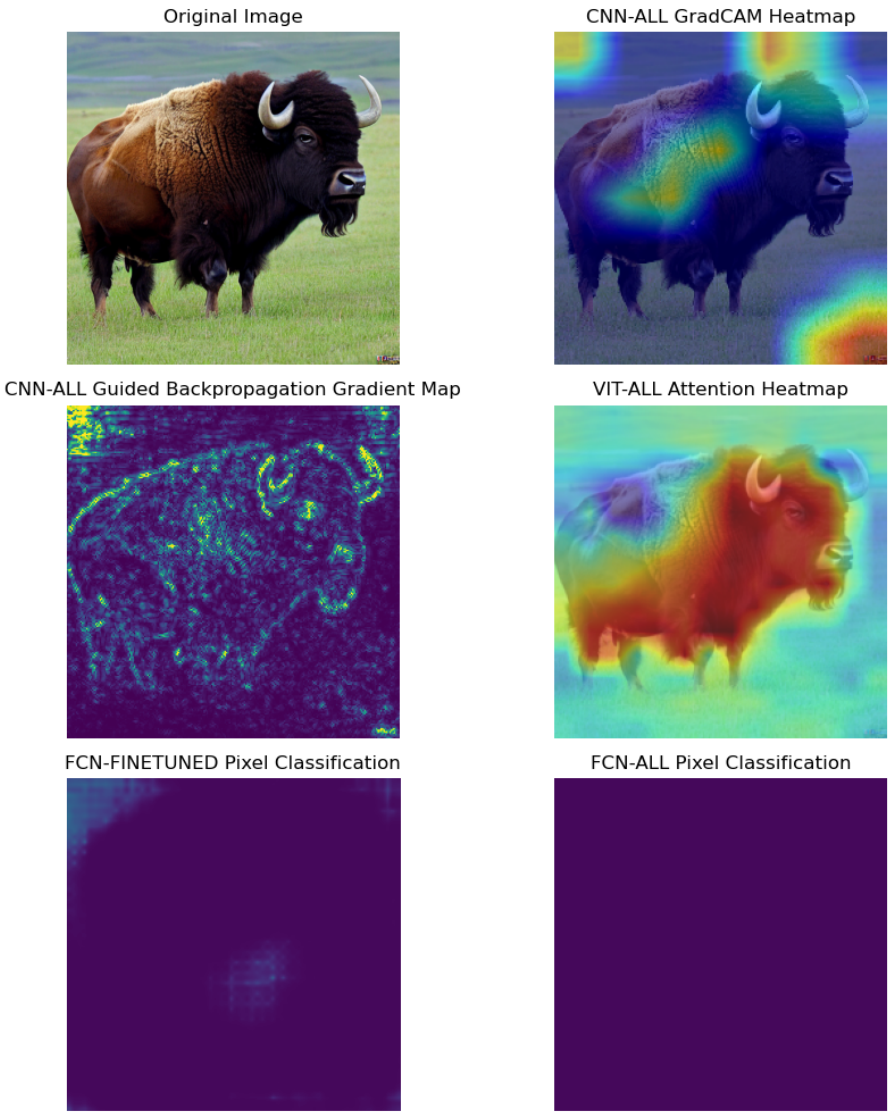



image generated with WUKONG

# B

# List of Files Removed from GenImage

imagenet_ai_0419_biggan\train\ai\116_biggan_00081.png
imagenet_ai_0419_biggan\train\ai\116_biggan_00094.png
imagenet_ai_0419_biggan\train\ai\116_biggan_00098.png
imagenet_ai_0419_biggan\train\ai\116_biggan_00107.png
imagenet_ai_0419_sdv4\train\ai\033_sdv4_00134.png
imagenet_ai_0419_sdv4\train\ai\033_sdv4_00137.png
imagenet_ai_0419_sdv4\train\ai\033_sdv4_00152.png
imagenet_ai_0508_adm\train\ai\115_adm_135.PNG
imagenet_ai_0508_adm\train\ai\115_adm_141.PNG
imagenet_ai_0508_adm\train\ai\115_adm_142.PNG
imagenet_ai_0508_adm\train\ai\115_adm_154.PNG
imagenet_ai_0508_adm\train\ai\115_adm_156.PNG
imagenet_ai_0508_adm\train\ai\115_adm_164.PNG
imagenet_glide\train\ai\GLIDE_1000_200_08_808_glide_00033.png
imagenet_midjourney\train\ai\208_midjourney_76.png
imagenet_midjourney\train\ai\208_midjourney_91.png
imagenet_midjourney\train\ai\208_midjourney_92.png

**DEPARTMENT OF ELECTRICAL ENGINEERING**
**CHALMERS UNIVERSITY OF TECHNOLOGY**
Gothenburg, Sweden
www.chalmers.se


CHALMERS