NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Science
Bachelor's Programme "Data Science and Business Analytics"

**Software Project Report on the Topic:**

# Module for DeepFake Image Detection and Analysis

**Submitted by the Student:**

| | |
|---|---|
| group #БПАД221, 2rd year of study | Belikov Daniil Maksimovich |
| group #БПАД221, 2rd year of study | Zarembo Mikhail Mikhailovich |
| group #БПАД221, 2rd year of study | Shakhmin Pavel Romanovich |

**Approved by the Project Supervisor:**

Andrey Borevskiy Olegovich
Research Fellow
Faculty of Computer Science, HSE University

Moscow 2024

# Contents

# Annotation

Our project covers all key stages in working with generated images, from their creation to the interpretation of the model's decisions, which is a non-trivial approach for future model improvement. Our team was divided into two parts, where the first was engaged in segmentation and interpretation of the model, and the second was in generating new images. Next we will talk about the whole work, but we will dive into details only on our part, namely classification, segmentation and interpretation.

Firstly, the generation of images is carried out using the generative model StyleGAN (Style Generative Adversarial Network), which is similar to a standard GAN but allows for improved quality of the generated images by generating details more accurately, which is important for tasks such as face generation. Thus, a sufficient number of images were generated for the subsequent retraining of the classifier, which is detailed below.

For the classifying model, we used ResNet34, which was pretrained on the ImageNet dataset. Retraining occurred in two stages: training on a dataset of 24,000 images, and during sequential and simultaneous training together with the second version of StyleGAN as a discriminator. Considering that the generated images from the original dataset were created using another type of generative models (namely a diffusion model), we were able to diversify the quality of our sample and successively improve the model's accuracy to 99% in validation.

It is also important to mention the interpretation and segmentation module, which is distinctive from all other works in the field of generated image detection. The interpretation includes LRP (Layer-wise Relevance Propagation) and Grad-CAM (Gradient-weighted Class Activation Mapping) modules, which are necessary to understand which areas the model focuses more attention on, thereby indicating which areas are poorly generated by the model. For a deeper analysis, facial segmentation was also created focusing on key areas (nose, ears, skin, hair, lips, eyes, eyebrows, beard, neck). Thus, after combining segmentation and the LRP module, it was possible to understand which areas are worst generated by the models and which areas are key for the model's decision-making.

# Аннотация

Наш проект охватывает все ключевые стадии в области работы со сгенерированными изображениями начиная от их генерации и заканчивая интерпретацией решения модели, что является нетривиальным подходом для будущего улучшения модели. Наша команда бы-

ла разбита на две части, где первая занималась сегментацией и интерпретацией модели, а вторая генерацией новых изображений. Далее будет рассказано про всю работу, однако в детали будем погружаться только на нашей части, а именно на классификации, сегментации и интерпретации.

Первым делом идет генерация изображения с помощью генеративной модели StyleGAN (Style Generative Adversarial Network), которая похожа на простой GAN, однако позволяющая улучшить качество сгенерированных изображений, генерируя детали более точно, что важно для такой задачи как генерация лиц. Таким образом было сгенерировано достаточное количество изображений для последующего дообучения классификатора, о котором пойдет речь далее.

В качестве классифицирующей модели был взят предобученый на датасете ImageNet ResNet34. Дообучение проходило в 2х этапа: обучение на датасете из 24 тысяч картинок, затем одновременное обучения вместе со второй версией StyleGan в качестве генератора и ResNet34 в качестве дискриминатора. С учетом того, что сгенерированные изображение из изначального датасета были созданы с помощью другого вида генеративных моделей (а именно, с помощью диффузионной модели), мы смогли разнообразить качество нашей выборки и последовательно улучшить точность модели то 99% на валидационной выборке. Также, важно упомянуть модуль интерпретации и сегментации, которая является отличием от остальных работ в области детекции сгенерированных изображений. Интерпретация включает в себе модули LRP (Layer-wise Relevance Propagation) и Grad-CAM (Gradient-weighted Class Activation Mapping), которые необходимы для получения сведений о том, на какие зоны модель больше обращает внимание, давая информацию о том, какие участки плохо генерируются с помощью моделей. Для более глубокого анализа также была создана сегментация лица по важным участкам (нос, уши, кожа, волосы, губы, глаза, брови, борода, шея). Таким образом после совмещения сегментации и модуля LRP удалось понять, что хуже всего генерируют модели и какой участок является ключевым для принятия решения модели.


# Keywords

Machine learning, deep learning, classification , segmentation, interpretation, generation, LRP, Grad-CAM, Deepfake, GAN

# 1 Introduction

In the modern reality with a high rate of spreading information it became very important to be sure at distinguishing real information and materials from the fake ones. By the fake ones, authors mean any kind of photographic, videographic or audio material that was not captured in reality but was produced using any kind of software, especially generative AI, which will be discussed a lot in this paper as it is its main concept.

Among all the possible options of using generative AI, it can be used to cause harm as any other kind of instrument, and people nowadays start to witness the cases of misuse of such AI in many spheres of social, political and economic life of a society. From the latest (year 2024) news, it is seen that generative AI is used to produce fake voice recordings to defame and cause reputational damage to famous politicians and opinion leaders. Fake photographic and video content is produced to deceive workers in the online banking area and get unauthorized access to private data and operations. It can be clearly seen that more scenarios of cyber fraud will be created in future, so it is highly important to develop countermeasures before the massive spread of generative AI.

Although now it is still possible to detect fake content with the naked eye, generative techniques evolve and start to produce content for which it is hard to tell whether it is generated or real. So, modern computer vision and classification techniques should be utilized to develop a solution for AI detection of fake material, as it can be more accurate in it by accumulating larger knowledge about visible and hidden features or artifacts that make it easier to detect fake content.

The main requirements for countermeasures are its accuracy, ease of use and capabilities of being integrated into existing products and solutions, such as online-call services, photostocks and video hostings, social networks and job search websites. Moreover, legal authorities should be able to use the tool to provide expertise for courts and prevent crime.

With all being said, the massive spread of misuse of generative AI should be considered as a threat to companies, individuals and public figures, so it is highly important to have a tool that is able to distinguish the real content from the generated one. The main requirements are specified and are technically possible to implement.

All files, models, weights can be seen on the GitHub of the project.

## 2 Literature overview

The main tensions about deep fakes started several years ago because of the rising power of generative models. That is why all works considering deep fakes use approximately the same methods for classification. The most interesting work as our team found was research [13] that combined Error level analysis, CNN and SVM for classification. Our investigation showed that such error level analysis is not as efficient and even redundant as augmentation (because of this, photos also can not be applied filters to make the data set diverse). The authors in Ciftci [5] extracted medical signal features and performed classification via CNN with 97% accuracy. However, the system is computationally complex due to a very large feature vector. McCloskey [11] developed a deep fake detector by using the dissimilarity of colors between real camera and synthesized and real image samples. The SVM classifier was trained on color based features from the input samples. However, the system may struggle on non-preprocessed and blurry images.

There are several good works connected with face segmentation such as article from 2020 [3] , in which a model was created that could segment part of face in different poses even if head is turned very far. The main problem is connected with bad quality of the model on a new face images. This were fixed with publishing a new huge data set and related work in 2023 [10], however in this data set there is no class for nose; ears, neck, body and hair is the same class; sometimes it is badly marked which makes work with it a little difficult. In our case we found a method of how we can train segmentation for every class we want even without markup using zero-shot image segmentation with combination of models CLIP, Grounding DINO and Grounding SAM [9]

As for LRP, the base information was taken from the research about brain activity segmentation with LRP [4] which is quite interesting approach due to existence of small but visuable places that need to be detected. For visual representation some articles also use Grad-CAM. For example this work [8] about medical text classification used activation maps for underlying word of each class. This method also can be used for brief retelling of such texts. Although there is a lot of information and works considering different stages of our project, there was no such one that combined generation, classification, segmentation and LRP.

# 3 Data augmentation + Error Level Analysis

*Implemented by Daniil Belikov*

One can say that size and quality of the sample is as important as a good model in every ML task. That is why the first stage was understanding how to make our data set larger, diverse and containing maximum useful information. For first generative models is a good idea and this will be investigated later in this paper. To make our data set diverse we can implement augmentation techniques which include data manipulation such as flip of the image, random crop, changing brightness and so on. All of this helps slightly change data representation from the usual view. Thus the model will need to 'think better and more abstract' as known taks for it is slightly harder and it can not always rely on some patterns that can be destroyed with light data manipulation. Also we need to make all images the same size which is 256 by 256 as on this size were trained almost all large CNN models.

So, after resizing the image we decided to use color jittering which slightly changes contrast, saturation, and brightness allowing the model not to focus as much on color but more on structure. Next goes a slight injection of random noise that can stimulate the imperfections on the image. After that we decided to use Gaussian Blur that can mimic the loss of detail that might occur during the generation process, making a model to identify subtle cues in blurred and smoothed facial features. The last one was normalization of the image, mean and variance were taken as for training large CNN such as ResNet34, that we will use later.

Now goes the third idea of extracting important features on the stage of preprocessing. For that we decided to use Error Level Analysis (ELA). It is an image analysis technique that is used to detect image manipulation. This method is based on JPEG compression analysis, which allows you to identify areas of the image that have been changed or edited. ELA can be especially useful in determining whether changes have been made to an image since it was originally created. Error Level Analysis works by comparing the compression levels in different parts of the same image. When you save the entire image again, the error level may increase. However, if a part of the image has been modified or added from another source and saved at different stages of compression, then those areas will have different error levels. ELA identifies these differences and displays them in the form of visually distinct colors.

Figure 3.1: Example of using ELA as image preprocessing

Changing the compression rate can make such indicating more visible. But after some tests our team understands that this type of image preprocessing does not fit to our deep fake classification as this method is good for noticing changes in photos rather than understanding that the whole photo was generated. Perhaps this technique can be used while developing a project even more so the model could detect not only generated images but places on the photo that were changed or which went through a filter.

# 4 Classification

*Implemented by Pavel Shakhmin*

This is the core part of the project, as it provides the main functionality of distinguishing real images from the generated ones. It utilizes many machine learning and computer vision techniques, such as Convolutional Neural Networks (CNN), Knowledge Distillation, Automated Hyperparameter Tuning, Skip Connection, Transfer Learning. The final result of this part of the project is a tuned Computer Vision model that is capable of detecting fake images. All models were trained on a dataset that consists of 140k images of humans, having 70k of them real and 70k generated using diffusion models.

## 4.1 Simple CNN Approach

The first effort was tuning a simple CNN that consists of two Convolutional layers and two MaxPooling layers as part that is able to extract features, and two Dense (Linear) layers as a part of a classifier. This architecture is considered small, compared to popular Computer Vision Classifiers, such as ResNet or GoogLeNet, providing fast training and inference. The model was trained using a Back Propagation approach with its convolutional layers being trained from

scratch. This led to a more agile but long training process, as both feature extraction and classification mechanisms were trained simultaneously. During the training process, hyperparameters, including Optimizer, learning rates and number of neurons in Dense layers were tuned using Optuna framework [1] with parallelism and pruning of unsuccessful trials. The best accuracy that was achieved with this approach was 68

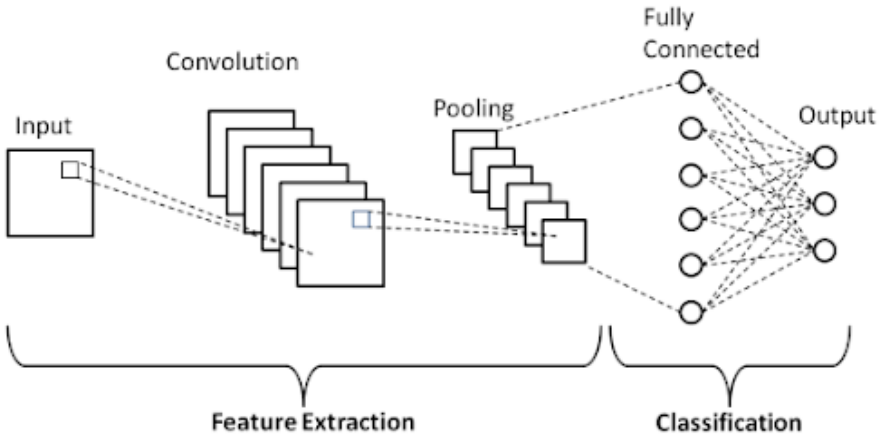



Figure 4.1: Architecture of a simple CNN model (source)

## 4.2 Transfer Learning Approach

As an attempt to increase the quality of the classification, a Transfer Learning technique was adopted. It consists of taking a pre-trained feature extraction layer from an existing CNN, such as ResNet and adding a classification layer to it, utilizing both high feature extraction capabilities of pre-trained layers and agility of training new classification layers from scratch. As a pre-trained model, three options were tried: EfficientNet, ResNet34 and GoogleNet. Classification layers were added to feature extraction layers of each model, which then were trained and evaluated. Number of classification Dense layers and the number of neurons in each of them were tuned via hyperparameter search, which also included optimizer and learning rate tuning. The best result was achieved by a ResNet34 model with classification accuracy of 99.64%. Although the result is high, ResNet34 is a large model that requires many computational resources to train and inference. So, the goal for next iterations was to achieve comparable accuracy with a smaller model that requires less computations.

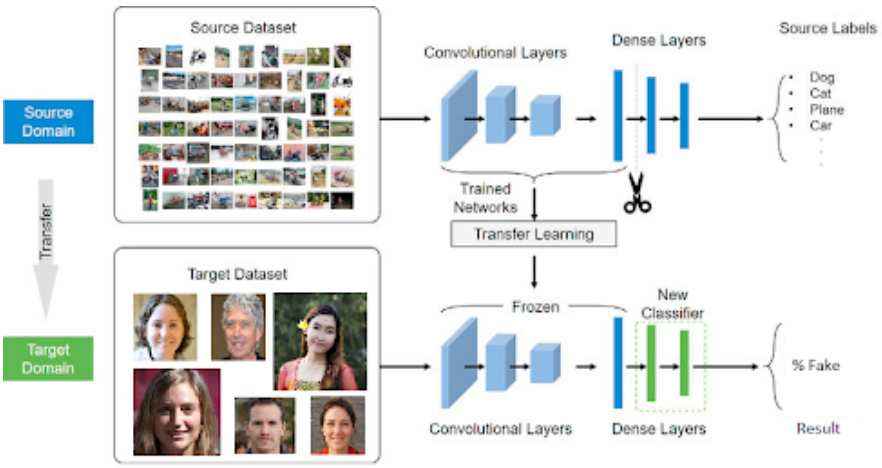



Figure 4.2: Illustration of Transfer Learning approach (source)

## 4.3 Knowledge Distillation Approach

As the ResNet34-based model achieved high results, it made it possible to extract its classification knowledge and transfer it to a smaller model. This can be done using a Knowledge Distillation approach that consists of using a trained model, which achieves great results and using it to train another model, which can be smaller and computationally easier. During the training process, input data is fed to both trainer and trainee models, while trainee loss is computed as a weighted sum of both models. This allows to pass the knowledge from the large, computationally expensive model to a smaller one without losing validity. Together with hyperparameter tuning, this approach resulted in a simple model with accuracy of 81.87%, which is obviously lower than of the ResNet34 model, but comes with low-resource computing. This result should then be improved by introducing more complex approaches to the small model.

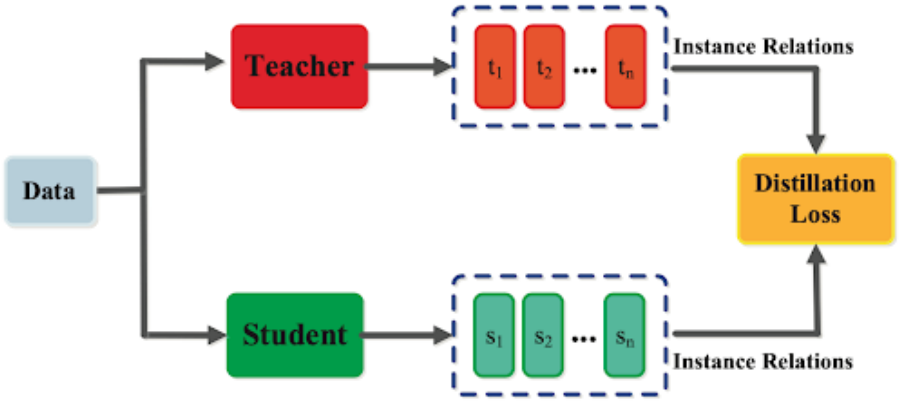



Figure 4.3: Illustration of Knowledge distillation approach (source)

## 4.4 Adding Skip Connection Layers

Another attempt of increasing accuracy was utilizing Skip Connection technique, which skips some of the layers in the neural network and feeds the output of one layer as the input to the next layers. Mainly this approach allows gradients to pass freely to improve convergence and deal with accuracy degrading. Combined with hyperparameter tuning, this technique was applied to the previous model and resulted in 89.3% accuracy.

# 5 Visualization of model decision

This part is not usually and not mandatory for all projects that are connected with Computer Vision, but one can do useful things with information from modules that can visualize model decisions. In our project this part is interesting because of the combination of it with segmentation which gives clear statistics of how the model makes decisions and what parts of the face are important for that. First, let's consider the interpretation modules.

## 5.1 Grad-CAM method

*Implemented by Daniil Belikov*

As was mentioned earlier, interpretability of convolutional neural networks is a tricky stage allowing to understand which parts of the image are important. Grad-Cam, or Gradient-weighed Class Activation Mapping is one of the techniques for that purpose. Grad-CAM method as it is stated in its name is connected with visualizing activation maps which can be described as features or patterns that tries to find each layer. Speaking for CNN, on the first layers activation maps represent simple lines, on the next layers there are circles or curves, on the next there are more complex patterns. When taking some part of the image and then applying it through activation maps we get some sum that represents the number, which can not only be used for the future calculations, but also for understanding how well our window fits the pattern. This is one of the main principles that use Grad-CAM as it checks the activation of each map in the model. Thus we can get a clear and understandable picture of how the model proceeded.

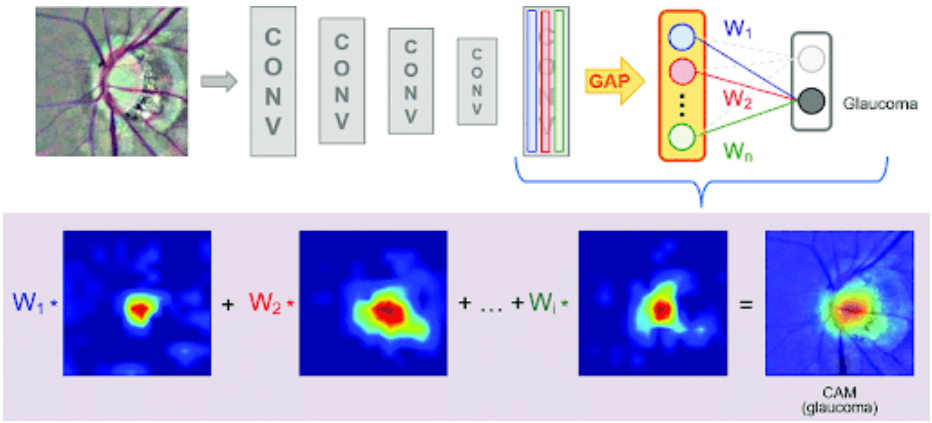



Figure 5.1: Grad-CAM explained

This visualization can be useful in applications where understanding model reasoning is crucial, such as in medical image diagnostics. Also with this tool developers can understand potential biases and errors in trained models (for example focusing on irrelevant features to make predictions). Grad-CAM is a useful tool for understanding a model's behavior, but as was tested in our case it does not fit well with our problem. If a person clearly could say that the image is fake, then there are some fake parts that can be found by model and so Grad-CAM will highlight them. In hard cases when humans can not make the right decision there are no potential places that can indicate the fake photo. In case of deep fakes, models need to look by pixels and not by whole parts of the face. That is why the next method fits better to our task of visualizing.

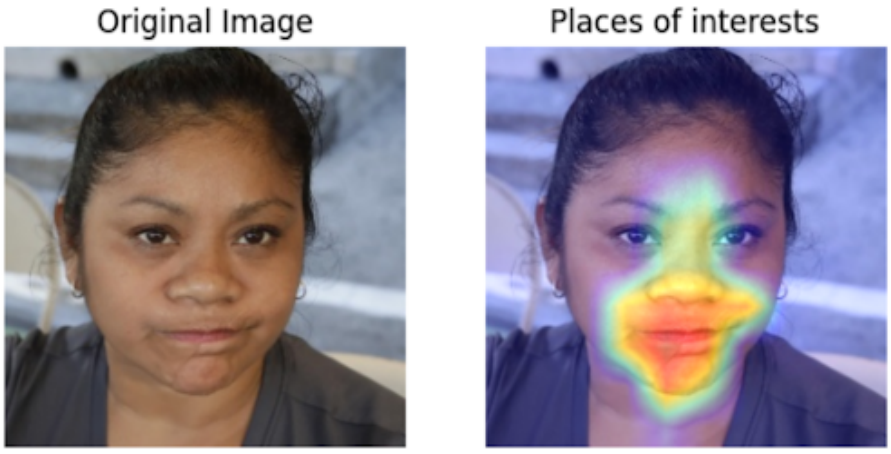



Figure 5.2: Visualization with Grad-CAM

## 5.2 LRP method

*Implemented by Daniil Belikov*

# Layer-wise Relevance Propagation (LRP)

Now we need more deep analysis of the model's decision and by that we also need to understand the impact of every neuron and every pixel. LRP states for Layer-wise Relevance Propagarion and was created exactly for this purpose. This approach uses back propagation from output layers to input layers to evaluate the impact of each pixel into classification. Relevance of each neuron is calculated based on its impact in activations that lead to the final decision. The propagation procedure implemented by LRP is subject to a conservation property, where what has been received by a neuron must be redistributed to the lower layer in equal amounts.

We can get propagation relevance scores $R_k$ at each layer onto neurons of the lower layer by the following rule:

$$R_j = \sum_k \frac{z_{jk}}{\sum_j z_{jk}} R_k$$

where $z_{jk}$ is the extent to which neuron $j$ has contributed to make neuron $k$ relevant.

LRP has several methods that could give different results considering the purpose and the number of current layers.

## Basic Rule (LRP-0)

This rule redistributes relevance to each input in proportion to their contributions to neuron activations as follows:

$$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k$$

This rule ensures that basic properties are satisfied. Thus deactivating (when $a_j = 0$ leading to $w_{ij} = 0$) as well as the absence of a connection is maintained. Although this rule can be applied to the whole network and layers, the gradient can be too noisy which would result in a not clear image.

## Epsilon Rule (LRP-$\varepsilon$)

This rule looks similar except adding a small positive term in the denominator, the role of which is to absorb some relevance when contributions to the activation of neuron $k$ are weak or contradictory

$$R_j = \sum_k \frac{a_j w_{jk}}{\varepsilon + \sum_{0,j} a_j w_{jk}} R_k$$

Increase in $\varepsilon$ leads to survival the absorption of only the most salient explanation. This rule makes explanations sparser in terms of input and makes them less noisy.

## Gamma Rule (LRP-$\gamma$)

Another change in rule consider favoring effect of positive contributions over negative:

$$R_j = \sum_k \frac{a_j \cdot (w_{jk} + \gamma w_{jk}^+)}{\sum_{o,j} a_j \cdot (w_{jk} + \gamma w_{jk}^+)} R_k$$

New parameter $\gamma$ indicates how much positive contributions are favored and increasing $\gamma$ leads to disappearing negative contributions.

Sounds like the Gamma Rule is better in terms of visualization, but as stated in one of the research from Berlin university [7] it is better to combine these three rules due to the amount of neurons in each layer. Thus, LRP-0 picks many local artifacts of the function and it is better to use it on the upper layers as the number of neurons is less than on the lower or middle. Middle layers have a more disentangled representation but also there are many spurious variations which can be filtered out if LRP-. This can remove noise elements in the explanation to keep only a limited number of features that matches the needed object to classify. On the lower layers LRP- fits the best due to spreading relevance uniformly to the whole feature rather than capturing the contribution of every individual pixel.

In our project we tested several methods. First one is connected with mixing Gamma rule and Epsilon Rule with an already implemented library for RLP Captum.

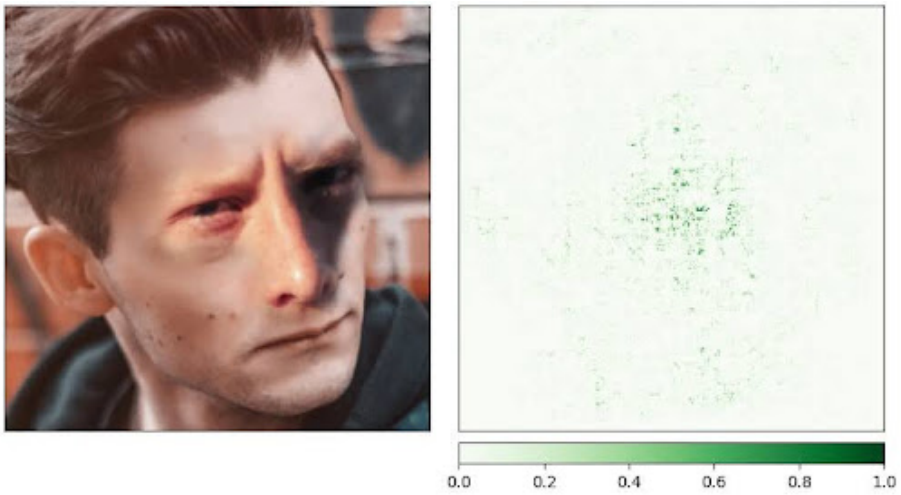



Figure 5.3: Image processed by Captum

The next version consists of combination of different  in Epsilon Rule on each layer, thus we could get more illustable picture

Figure 5.4: Image processed Epsilon Rule implementation

By looking at visual modules we can understand not only how our model thinks, but also what are the disadvantages of the sample. For example it was established that models are bad at generating moles which our classifier noticed and often used as a positive sign for fake images. Also when person have glasses it works bad as model does not seen much glasses and thinks that it is also fake sign (this was fixed by extending our data sets)

# 6 Image segmentation

*Implemented by Mikhail Zarembo*

## 6.1 Idea

Image segmentation is a process of partitioning images into classes or clusters based on specific criteria. As every model, segmentation models also need datasets for training, validation and tests with annotations or masks, but for our task (face segmentation) there is no such dataset that has enough classes. So the idea was to use an ensemble of models CLIP[12] (Contrastive Language-Image Pre-training), Grounding DINO[14] (DIstillation with NO labels) and SAM[6] (Segment Anything Model). The combination of CLIP-DINO-SAM[9] models allows for highly accurate segmentation of desired areas based on a text prompt only. This method is highly convenient because there is no need to create a dataset for segmentation which is a very long and expensive process.

## 6.2 Structure of each model

The structure of the CLIP-DINO-SAM[9] combined module is complicated so first let's see every model architecture and detailed explanation separately.

### 6.2.1 CLIP

CLIP is not a neural network but a framework combining an image and text encoders. Generally it processes images and texts through the encoders and gets the embeddings which are then processed and the model creates a matrix where each cell (i, j) represents the confidence of the model that the j text description belongs to the i image. The whole process is made without directly optimizing for the task which means that the model can predict images and texts that it had never met during the process of training (zero-shot transfer learning)[2] .

**CLIP structure:** Figure 6.1

- **Visual Encoder:** This component is designed for image processing which converts input images into feature vectors (embeddings). It can use different architectures for this task such as Vision Transformers (ViT) or CNNs (Convolutional Neural Networks). Both architectures are designed to process and understand images but they do this in different approaches. In ViTs the feature vectors are pooled to create a single embedding that represents the entire

image. In CNNs the feature maps are averaged globally to create a single feature vector that represents the entire image. The output feature vector from either ViT or CNN is passed through a normalization layer to ensure it has unit length. This final normalized vector is the image embedding which can be compared with text embeddings to find similarities.

- **Text Encoder:** This component processes texts. It is built on a Transformer architecture similar to those used in NLP (Natural Language Processing) tasks. The text encoder converts text descriptions into feature vectors (embeddings). Transformer arranges tokens in vector space in a way that the model understands the relationships between tokens. The final layer's output of the Transformer is a sequence of feature vectors, one for each token, token - is a smaller unit of input text (word or subword). These vectors are then aggregated to create a single feature vector that represents the entire text input.

- **Combination of the embeddings:** The two embeddings from Text Encoder and Visual Encoder go through linear transformations to match their dimensions and to align the features from both modalities in the same space and then they are normalized. Then the model creates a matrix of element-wise cosine similarity between pairs of image and text vector representations multiplied by the temperature parameter. In this matrix the highest value in the row represents the most suitable text for the picture.

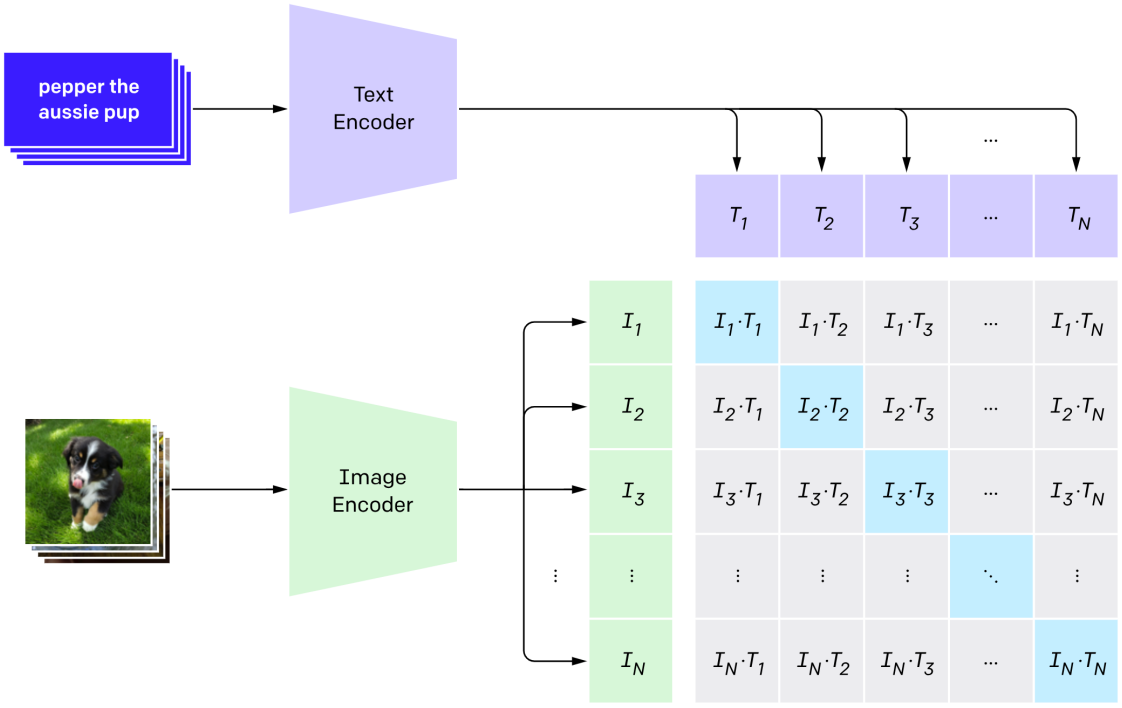



Figure 6.1: CLIP architecture (source)

### CLIP pre-training 6.2

CLIP was trained on a dataset consisting of 400M pairs of various images and their text descriptions. The model used contrastive learning where it learned to match images with corresponding text descriptions and distinguish non-matching pairs. This way it developed generalized representations that are not limited by training classes. During the training process N-many images and N-many texts were passed through corresponding Image and Text encoders and after normalization a paired matrix was made. The model tried to maximize the elements (i, i) on the main diagonal, being the cosine similarities between embeddings, as there are correct text descriptions for corresponding images. With this logic Encoders were trained by minimizing the cross entropy on each vertical and horizontal of the matrix.

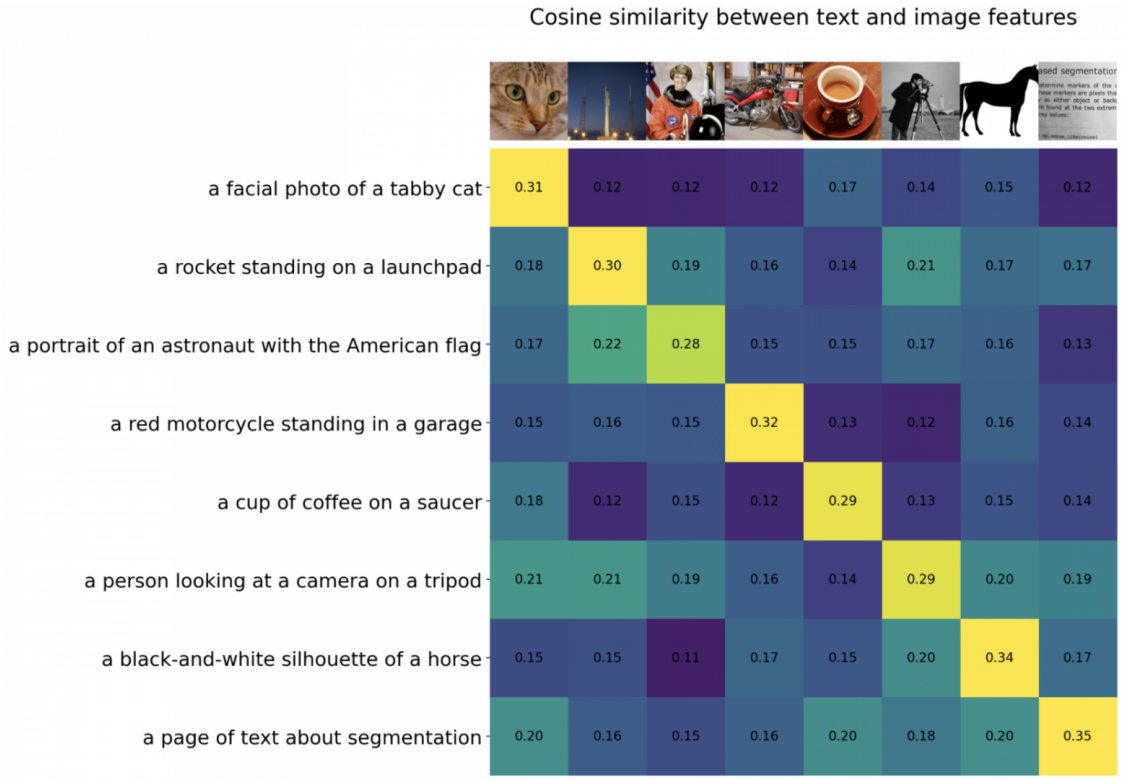
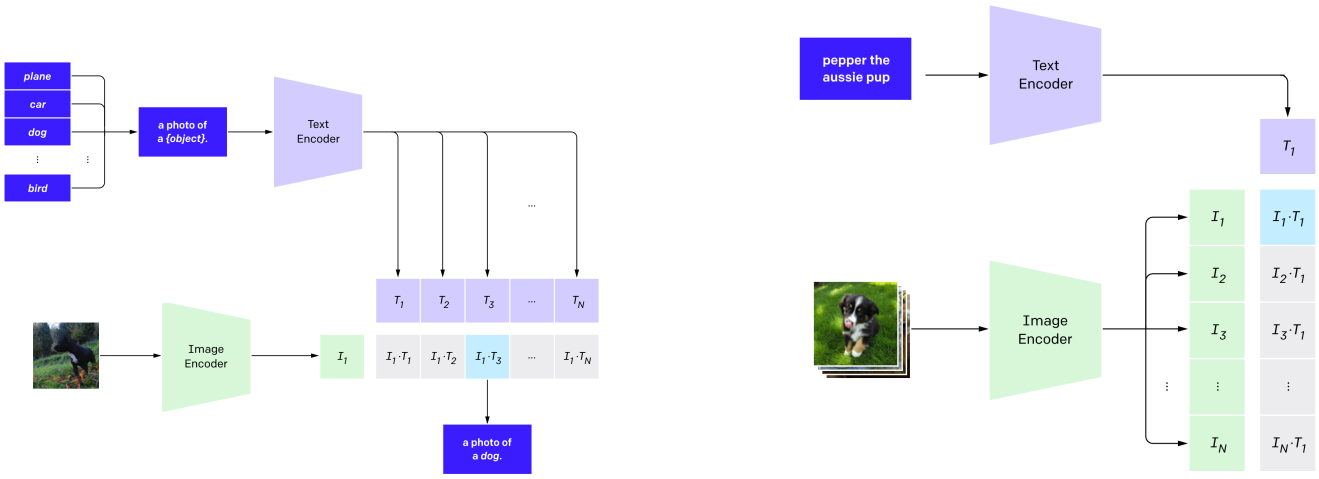



Figure 6.2: CLIP output matrix (source)

### Useful implementations of CLIP 6.3

If CLIP receives one picture and several text descriptions it will compare texts to the image. This way the output of the model will show text descriptions that is the most suitable for the image which is similar to what common image classification models do. On the other hand, if CLIP receives one text and several images it will perform as a search engine comparing images to text.




(a) CLIP as classificator (b) CLIP as search engine

Figure 6.3: Comparison of CLIP functionalities (source)

### 6.2.2 Grounding DINO

Grounding DINO is capable of creating bounding boxes around the target areas provided by text description. It is capable to detect objects that are new to the model - were not in its training dataset. This is because of constructive learning and text-based prompts along with large and diverse training data similarly to CLIP

**Grounding DINO architecture** Figure 6.9

Grounding DINO consists of 5 main parts: Text Backbone and Image Backbone,

- **Text Backbone and Image Backbone:** Figure 6.4 Figure At this stage image is passed through image backbone like Swin Transformer and it extracts text features, on the other hand text is sent to BERT-like Transformer Backbone which also extracts text features.

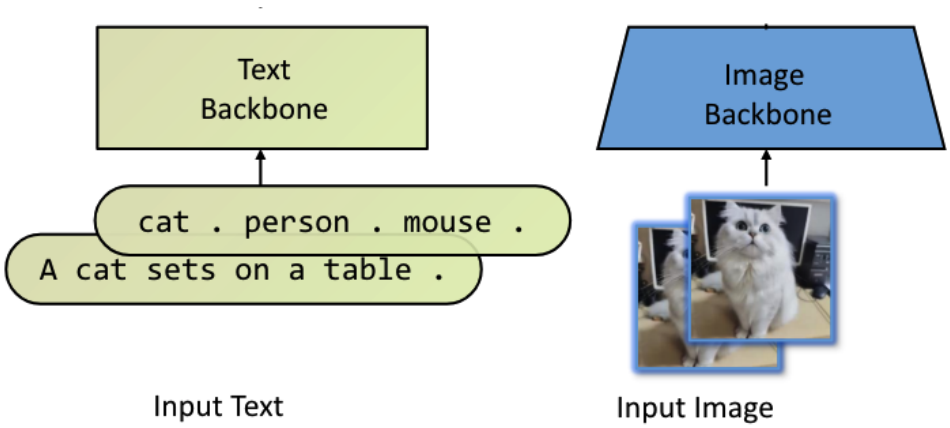



Figure 6.4: DINO text and image backbones (source)

- **Feature enhancer:** Figure 6.5 Features from the Backbones are passed through the Feature enhancer for cross-modality feature fusion. Feature enhancer includes four layers: Self-

attention, Image-to-Text Cross-Attention, Text-to-Image Cross-Attention and FFN(Feed-Forward Network).

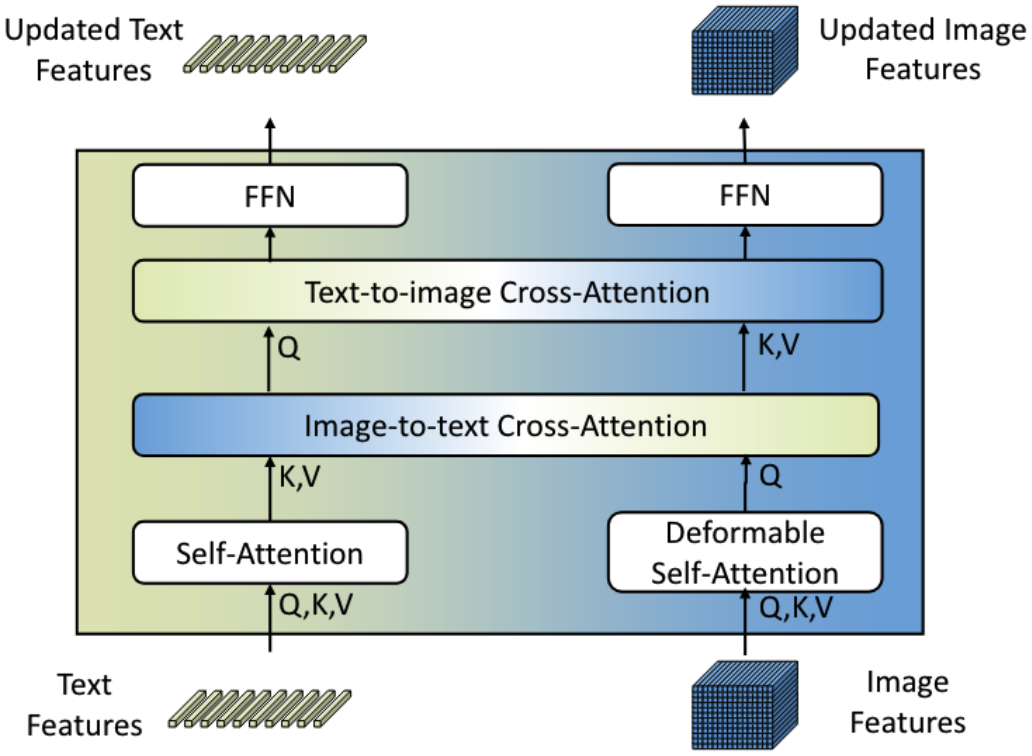



Figure 6.5: DINO Feature enhancer (source)

Self-attention for image features processes pixel embeddings by determining the relevance of each pixel to others so this shows the model contextual relationships within the image. For text features self-attention processes token embeddings with the same logic showing the modal contextual relationships within text.

Image-to-Text Cross-Attention aligns text tokens with relevant image patches which allows the model to understand the contextual relationships within image and text parts in other words which parts of the image correspond to which parts of the text.

Text-to-Image Cross-Attention enables the model to align image tokens with relevant text tokens. This process helps the model understand which parts of the text correspond to which parts of the image allowing for more precise localization and interpretation of textual descriptions for image areas.

FFN processes and integrates visual and textual information further preparing it for the next part of the model.

- **Language-guide Query Selection:** Figure 6.6 After new image and text features leave the Feature enhancer, they are passed through Language-guide Query Selection and there

textual information is utilized to guide the attention mechanism when processing image patches. This helps in effectively grounding textual descriptions in corresponding image regions. Then from the Language-guide Query Selection model gets Cross-Modality Queries (queries derived from one modality (text) used to attend to and retrieve relevant information from another modality (images))

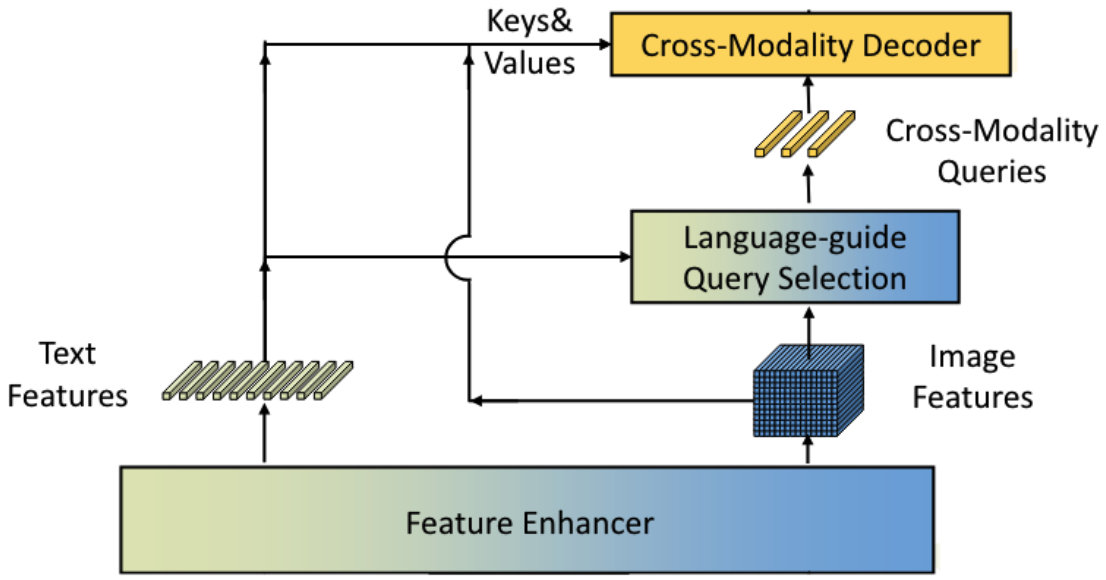



Figure 6.6: DINO Language-guide Query Selection (source)

- **Cross-Modality Decoder:** Figure 6.7 Cross-Modality Queries from the Language-guide Query Selection, image features and text features all together are passed through the Cross-Modality Decoder producing a unified and contextually rich representation decoding input features from each modality by leveraging the inter-modal relationships. Cross-Modality Decoder includes four layers: Self-attention, Image Cross-Attention, Text Cross-Attention and FFN.

  Self-attention for Cross-Modality query helps to connect textual and visual data enabling the model to create more integrated representations that capture the relationships between different types of information

  Cross-Attention combines two embedding sequences, first is a query and the second is a key and value input. Then they are processed and the output sequence having the same dimension and length as the first query represents is a set of feature vectors that integrate information from both modalities.

  The FFN processes and refines the integrated feature vectors from the cross-attention layers producing the final output.

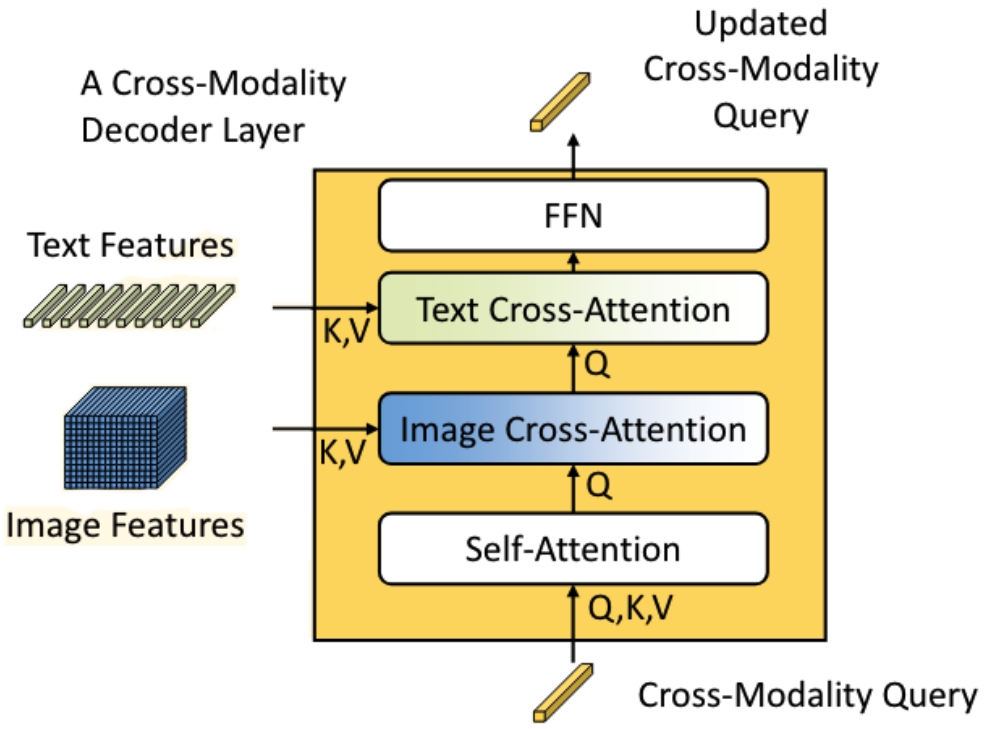



Figure 6.7: DINO Cross-Modality Decoder (source)

- **Output of the DINO model:** Figure 6.8 After the text features and the new Cross-Modality Query are combined, model gets the final understanding of which part of image corresponds to which part of text creating bounding boxes for object detection for corresponding classes from text

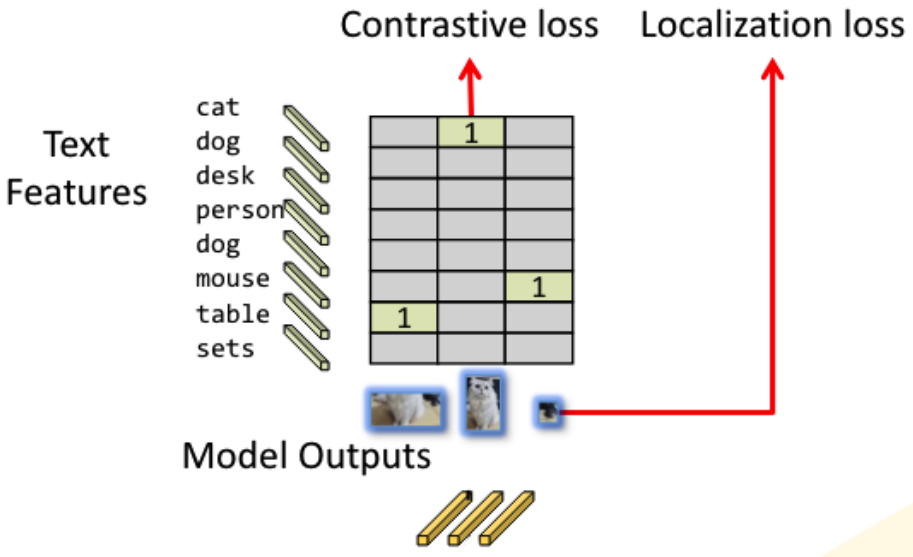



Figure 6.8: DINO output (source)

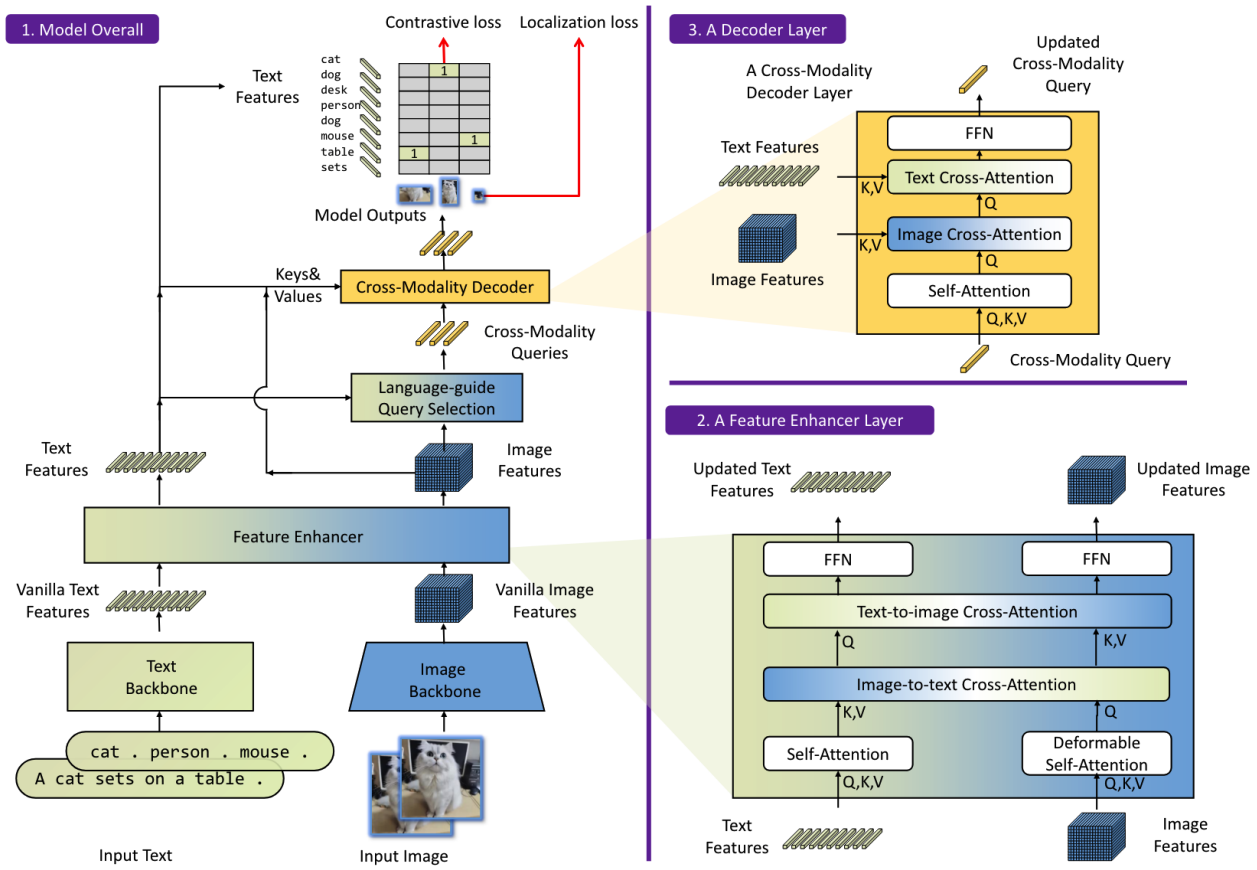



Figure 6.9: whole DINO architecture (source)

### 6.2.3 Grounding SAM

SAM (Segment Anything Model) is capable of zero-shot segmentation of anything if it gets a proper prompt. Segmentation prompts supported by SAM are points, bounding boxes, masks and text descriptions. SAM was trained on a large and diverse dataset and it's prompt based structure allows it to segment areas that were not present in it's training data

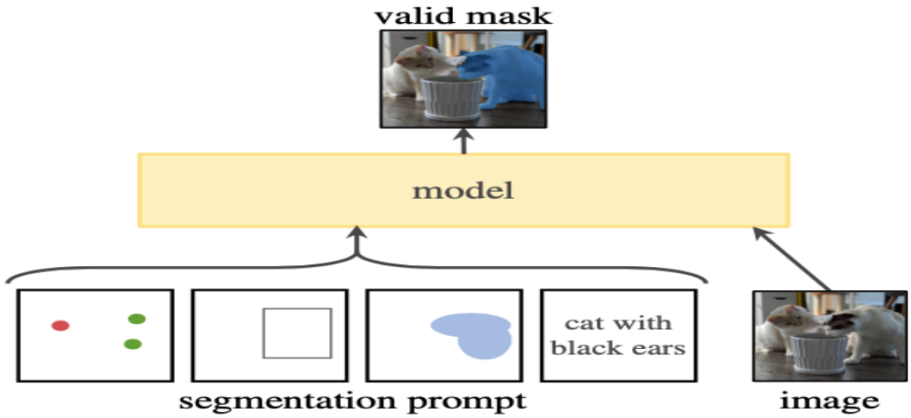



Figure 6.10: SAM promptable architecture visualization (source)

**SAM architecture** Figure 6.11 consists of five major parts: Input Image Processing,

Prompt Encoder, Cross-Attention Module, Transformer Decoder and Segmentation Head.

- **Input Image Processing:** The input image is fed into the backbone network to extract feature maps. Here is used MAE pretrained Vision Transformer adapted to process high resolution inputs effectively capturing rich visual representations. The output of the image backbone are feature maps (embeddings).

- **Prompt Encoder:** It encodes the prompts (points, boxes, masks, texts) that help the model to understand it's objective. Point Encoder encodes points provided as prompts converting them into a format that can be used by the model. Box Encoder encodes bounding boxes providing spatial context to guide the model. Mask Encoder encodes initial masks that roughly outline the area of interest refining the segmentation process.

- **Cross-Attention Module:** Integrates the encoded prompts with the image features extracted by the backbone using cross-attention to combine the information from the prompts and the image features. Query vectors are derived from the prompt encodings and key and value vectors are derived from the image feature maps. Cross-attention scores are computed to determine the relevance of different image regions with respect to the prompts. These scores are used to produce attention weights, which are then applied to the value vectors to obtain a contextually integrated feature representation.

- **Transformer Decoder:** It refines the integrated features and produces the final segmentation output. It consists of three components: Multi-Head Self-Attention, FFN, Layer Normalization and Residual Connections. Multi-Head Self-Attention captures relationships within the integrated feature representation. FFN processes the features further to refine the segmentation boundaries. Layer Normalization and Residual Connections stabilizes training and enhances model performance.

- **Segmentation Head:** The refined features are passed through the segmentation head to generate the final segmentation masks segmenting the target objects in the image. The segmentation head consists of convolutional layers followed by upsampling operations to produce masks that matches the resolution of the input image.

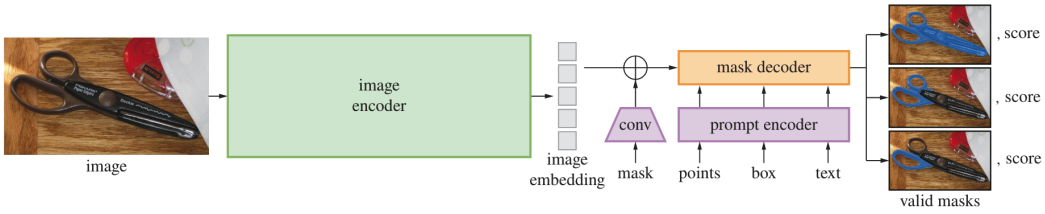



Figure 6.11: SAM simplifyed architecture (source)

## 6.3 CLIP-DINO-SAM

The combination of the above mentioned models consists of 4 major stages: Setup, Creation of bounding boxes, Image segmentation and Final classification of masks.

- **Setup:** Image, text descriptions and text prompts (classes) are provided to the system
- **Creation of bounding boxes:** The text prompts and the original image are sent to Grounded DINO which uses them to generate bounding boxes around the regions that match the text descriptions.
- **Image segmentation:** The bounding boxes generated by DINO are used as prompts for SAM. Then it produces segmentation masks for the regions corresponding to text descriptions
- **Final classification of masks:** CLIP here is used for classification of text prompts. Text prompts and segmented regions from SAM are provided to CLIP. Then based on the similarity between the segmented regions embeddings and the classification prompt embeddings CLIP assigns a label to each segmented region along with a confidence score.

Then the final image with segmentation masks is saved and the zero-shot segmentation based on text description is finished.

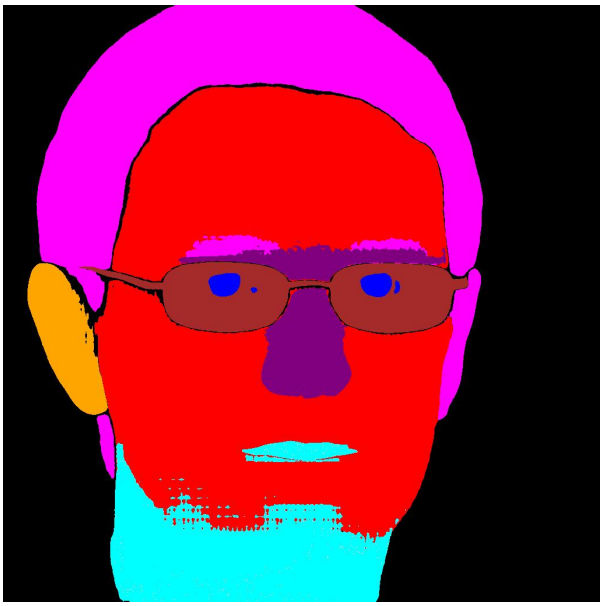
## 6.4 Model tuning

The final version of the text prompts (classes) needed for segmentation are nine classes: Hair, Ears, Eyes, Eyebrows, Glasses, Nose, Mouth, Neck and Face (every other face region all together without those already mentioned) with corresponding text descriptions of each class - these are all necessary classes covering the whole face.


Figure 6.12: Classes and corresponding mask colors

The raw combines module[9] showed miserable results.

(a) original image



(b) segmented image

Figure 6.13: Comparison of CLIP functionalities

Here the regions of hair and eyebrows are labeled as neck, neck is labeled as mouth and labels for hair and eyebrows are missing. This is an inferior situation for our task as classes are labeled wrong and the further analysis will go wrong as well.

**Explanation:**

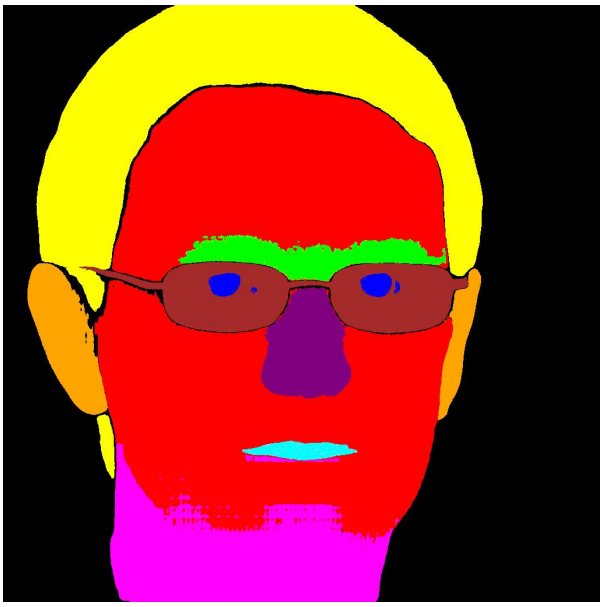
When CLIP-DINO-SAM model gets many text descriptions (classes) CLIP performance of classification of the segmented areas reduces. In our situation DINO processes multiple text prompts and produces multiple bounding boxes for SAM, then the latter creates segmentation masks and CLIP receives multiple text descriptions and multiple images (segmented areas) and has to find the corresponding ones. CLIP does not get the initial image so it does not know the context of each segmented area and gets many distracting images (for each text prompt it receives all the areas but most of them do not belong to this particular class) so CLIP can easily get confused and label the masks wrong.

**Problem fix:**

The segmentation process was changed: now the whole model receives each text prompt sequentially getting only one text prompt at iteration. Hence DINO makes less bounding boxes and almost all of them correspond to the prompt, SAM creates masks and then CLIP only has to choose correct segmented areas for the prompt from a smaller batch with far less distractors.

This way the performance of the model was significantly increased.

(a) original image



(b) segmented image

Figure 6.14: Comparison of CLIP functionalities

The Figure 6.14 represents the sample result of the final version of segmentation and it looks much better than with raw model.

# 7 Combination of Segmentation and LRP

*Implemented by Mikhail Zarembo*

## 7.1 Idea

After an image is classified it can be further analyzed using Layer-wise Relevance Propagation (LRP) methods (see Section 5.2). Generally it highlights the regions of the image that the model focuses on this information alone is often not sufficient, especially when dealing with large datasets used in deep fake classification. Manually inspecting each image is not practical hence the process needs to be automated. This is where Segmentation (see Section 6) comes into play. The combination of LRP with Segmentation helps to identify and label the specific regions of interest that the classification model considers. In other words the combination module can automatically classify a what region of face does the classification model looks the most at a particular image. This way the analysis of generated images will become automated and simple process.

## 7.2 Implementation

The module combining LRP and Segmentation aims to map the LRP heat-map to each segmented mask. So the output of the combined module will show the percent of pixels in the LRP heat-map that correspond to each facial region (class). The segmentation module generates not only the final segmentation mask but also individual masks for each class. This allows the combination of LRP and Segmentation to be executed as a two-step recursive algorithm.

- **Dimension match:** The LRP heat-map and segmentation mask are resized to match the dimension of the original image.

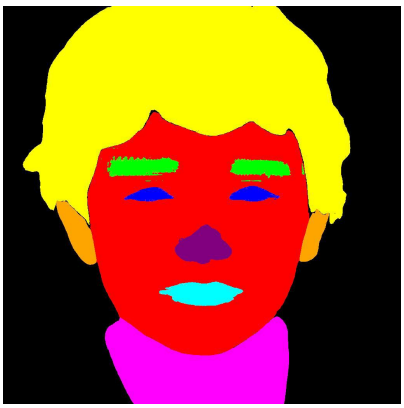
- **Combining process:** New segmentation mask (with matched dimensions) is converted to binary mask, then it is multiplied bitwise by the LRP image and then non-zero pixels are counted, this way every pixel that belongs both to LRP and the segmentation mask (a particular class) is counted and the number is saved.

Example of implementation

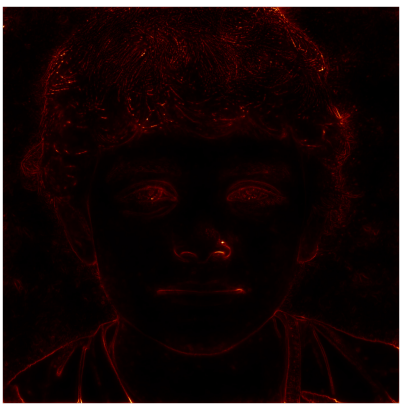
First image is processed by the LRP algorithms (Epsilon Rule implementation see Section 5.2) and then segmentation masks are created Figure **??**

Then the combination process begins and the results are represented in a class-wise bar chart with percentages Figure 7.2

(a) Original image



(b) Segmentation masks



(c) LRP image

Figure 7.1: Sample implementation

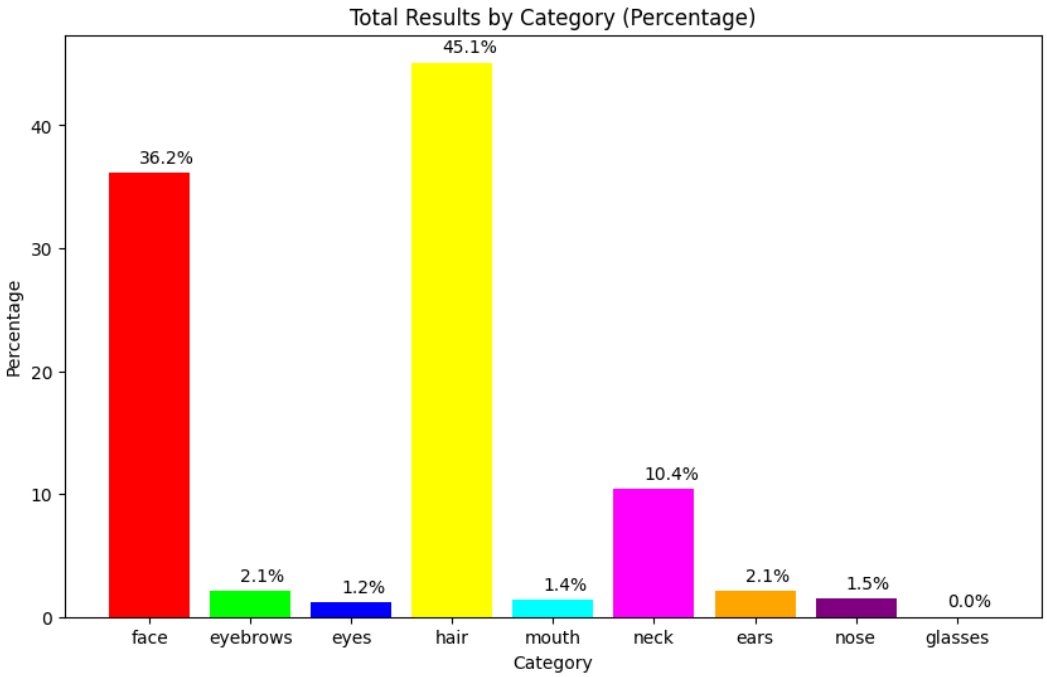



Figure 7.2: Result of combination of sample image

The sample result shows that the classification model pays more than 90% of attention to the face and hair of the generated person in the image. A person inspecting image will never get such percise results so the advantage and usefullness of the combination module is undeniable

The module combining LRP and Segmentation processed a thousand images from the generated dataset made by our colleagues (see Section 8). The results (Figure 7.3) showed that almost 80% of attention of the classification model for all pictures is located at two classes: face and hair.

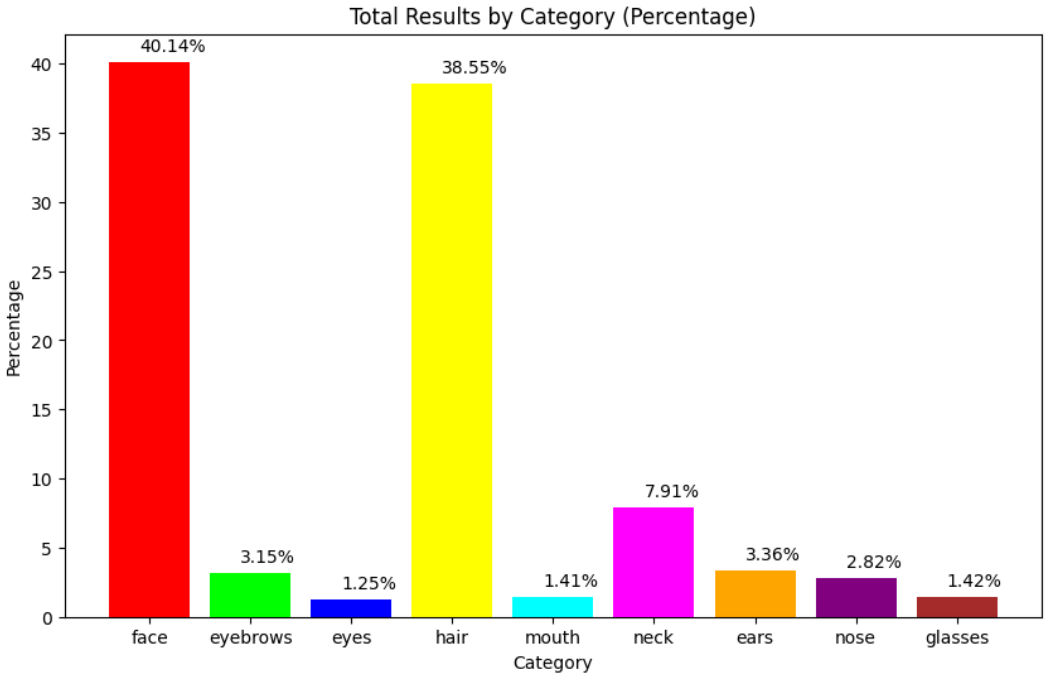



Figure 7.3: Result of combination of LRP and Semgentation for generated dataset

# 8 Image Generation

Generating new images is another important part of our research as it helps not only to make samples larger but also to make them more diverse as different generative models create images in unique ways which can be important for the model. To make this we have chosen GAN models and we have tried several versions of them.

## 8.1 GAN Model

Generative adversarial network is a class of generative models that consists of two parts: generator and discriminator. It can be said that these two parts try to compete with each other as the generator tries to create a new image that will not look like the image from the sample and the discriminator tries to make the new image the same as from the sample. This is why the model needs to learn how to create new images that will save the logic from the sample. Discriminator in simple GAN can be described as a classifier as it evaluates the quality of the generated image. The same generator tries to understand how to map noise into images. For that it is trained to convert image to noise and conversely. Important thing to note is that such generative models have two losses for both parts and that is why training can be harder.

## 8.2 Deep Convolutional GAN (DCGAN)

More complex form of the GAN that was implemented. The difference between GAN and DCGAN is in the structure of the discriminator as in that case it could be CNN model that can better work with unstructured data as image. Thus the quality of the classifier is increased

## 8.3 Wasserstein GAN (WGAN) with gradient penalty

WGAN is another architecture of GAN that was also implemented. Instead of using the binary cross-entropy loss used in standard GANs, WGAN uses the Earth Mover's distance (also known as the Wasserstein distance) to measure the difference between the model distribution and the target distribution. This distance provides a more effective gradient which helps in stabilizing the training. Also, it subtracts gradient penalty to the loss function making some sort of regularization.

Due to lack of computational resources, it was decided to look at other alternatives such as StyleGAN that is more popular than the previous two and give much better quality.

Figure 8.1: DCGAN Generation

## 8.4 Style GAN

Style GAN is more sophisticated model from GAN family and is highlighted for its ability to generate high-quality synthetic images of human faces. Style GAN uses a technique called progressive training of generative adversarial networks. The model starts out training on low-res images and then gradually increases the image resolution as it learns. This helps to make the learning process more stable from the start and improves the quality of the images it generates. Also Style GAN uses innovative normalization techniques, including adaptive instance normalization at each level of the generator. These methods help to stabilize the color and style of the generated images throughout the learning process which eventually helped to create a proper model for face generating task. Although our team faced the same problem with lack of computation resources, our work use pretrained Style GAN for deep fake generation that helped to enlarge training sample.

Figure 8.2: WGAN Generation

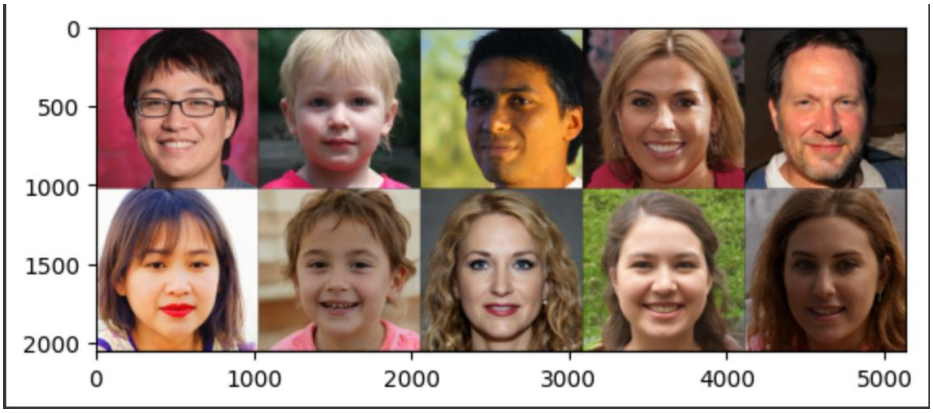



Figure 8.3: Style GAN Generation

## 8.5 Combining ResNet and Style GAN

At this stage we have a good classification model and a good model for generation. That is why we decided to make them even better by combining and training them simultaneously by putting pretrained ResNet34 on our large dataset as a discriminator. Thus we managed to make a model that takes real images, generates fake and then produces binary classification. This helps us to make a more stable classifier (as after training on a large data set model managed worse on the new generated images and only after 20 epochs with the new method model managed to reach accuracy of 98% which is less then it was initially but on a wider range of images.

# 9 Conclusion

The result of the project consists of three parts: classification, interpretation and segmentation modules. Classification module's main purpose is utilizing Computer Vision methods to distinguish real material from the fake one. The final version of this method provides classification with accuracy of 98%, which can be considered as a high result. After the classification model was developed, it became useful to interpret the findings to look for the improvement options. Applying different interpretation techniques resulted in many ways to visualize the work of the classification model, which gave additional knowledge about its attention zones. This information is then used to run segmentation algorithms to gather human-readable statistics on a whole dataset.

The main goals, which were stated in the Introduction, such as classification accuracy, ease of use and capabilities of being integrated into existing products and solutions are achieved in the resulting solution. The research and the work can be classified as done, achieving all its goals and providing its findings that can be useful in social, economic, media and political areas of life.

# References


[1] URL: https://optuna.org/ (visited on May 27, 2024).

[2] Mikhail Konstantinov @Dirac. "OpenAI's CLIP Neural Network: A classifier that doesn't need to be trained. Long live Learning without Learning". In: *habr:539312* (2021). URL: https://habr.com/ru/articles/539312/.

[3] Khalil Khanm Rehanullah Khan Kashif Ahmad Farman Ali. "Face Segmentation: A Journey From Classical to Deep Learning Paradigm, Approaches, Trends, and Directions". In: (2020).

[4] S. Bach. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: (2015).

[5] Ciftci U. A. Demir I. Yin L. Fakecatcher. "Detection of Synthetic Portrait Videos Using Biological Signals". In: (2021).

[6] Alexander Kirillov Eric Mintun Nikhila Ravi Hanzi Mao Chloe Rolland Laura Gustafson Tete Xiao Spencer Whitehead Alexander C. Berg Wan-Yen Lo Piotr Dollár Ross Girshick. "Segment Anything". In: *arXiv:2304.02643* (2023). URL: https://arxiv.org/abs/2304.02643.

[7] Sebastian Lapuschkin Gregoire Montavon Alexander Binder. "Layer-Wise Relevance Propagation: An Overview". In: (2019).

[8] Katsuhiko Ogasawara Hongjian Zhang. "Grad-CAM-Based Explainable Artificial Intelligence Related to Medical Text Processing". In: (2023).

[9] James. "Licensed MIT License Copyright (c)". In: (2023). URL: https://github.com/capjamesg/sam-clip/blob/main/LICENSE.

[10] Soffia Kirillova. *EasyPortrait: Face Parsing and Portrait Segmentation Dataset*. URL: https://betterprogramming.pub/easyportrait-face-parsing-and-portrait-segmentation-dataset-bb38eaf1b082 (visited on May 27, 2024).

[11] M McCloskey S. Albright. "Detecting GAN-generated imagery using saturation cues". In: (2019).

[12] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. "Learning Transferable Visual Models From Natural Language Supervision". In: *arXiv:2103.00020* (2021). URL: https://arxiv.org/abs/2103.00020.

[13] Rashid Amin Rimsha Rafique Rahma Gantassi. “Deep fake detection and classification using error-level analysis and deep learning”. In: (2023).

[14] Shilong Liu Zhaoyang Zeng Tianhe Ren Feng Li Hao Zhang Jie Yang Chunyuan Li Jianwei Yang Hang Su Jun Zhu Lei Zhang. “Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection”. In: *arXiv:2303.05499* (2023). URL: arXiv:2303.05499.