University of Padova

Department of Mathematics

Master Thesis in Data Science

Machine Learning-based Image Forgery Detection

Supervisor
Wolfgang Erb
University of Padova

Master Candidate
Jan Elfes

Academic Year
2022-2023

This work is dedicated to all the people that filled this time with inspiration and joy.

Abstract

Image manipulation tools are constantly improving. Most recently, the productization of generative models in popular software like Adobe Photoshop provided a whole new range of possibilities. Though many applications might be harmless, image forgery is not. Tampered images can spread false information, manipulate opinions, and erode trust in media. Therefore, being able to detect fake images is of the utmost importance. The majority of Image Forgery Detection models consist of specialized architectures, often trained with limited data and computational resources. In contrast, image segmentation has found substantial interest and investment. In this work, I explore the capabilities of state-of-the-art general image segmentation models to adapt to the task of Image Forgery Detection to leverage the extensive resources and advancements in this field. I assess their performance on the detection of classical Photoshop manipulation like splicing. Further, I extend the scope to the detection of AI-inpainted images, i.e. images that were manipulated using deep generative models. I show that image segmentation models can keep up with state-of-the-art forgery detection tools. Moreover, the models can detect AI-inpainted regions by identifying the characteristic frequency signature of the generative models.

Contents

Listing of figures

Listing of tables

Listing of acronyms

APS . . . . . Active Pixel Sensor

CFA . . . . . Color Filter Array

CNN . . . . . Convolutional Neural Network

CMOS . . . . . Complementary Metal-Oxide Semiconductor

DCT . . . . . Discrete Cosine Transform

DETR . . . . . Detection Transformer

DFT . . . . . Discrete Fourier Transform

FPN . . . . . Feature Pyramid Network

GAN . . . . . Generative Adversarial Network

IoU . . . . . Intersection over Union

LBP . . . . . Local Binary Pattern

LSTM . . . . . Long Short-Term Memory

MAT . . . . . Mask-Aware Transformer for Larger Hole Inpainting

SIFT . . . . . Scale Invariant Feature Transform

SMM . . . . . Style Manipulation Module

SRM . . . . . Spatial Rich Model

VAE . . . . . Variational Autoencoder

ViT . . . . . Vision Transformer

1 Introduction

In today's digital age, images have become ubiquitous and are accessed and shared through numerous channels. They play a crucial role in conveying information, shaping opinions, and influencing our understanding of the world. However, with the widespread availability of image manipulation tools, creating fake images has become increasingly easier. This has serious implications for the credibility of images and raises concerns about the reliability of visual information. Manipulated or fake images can mislead viewers, spread false information, and erode trust in media.

Image editing software like Adobe Photoshop or GIMP is not new by any means, and so are fake images. But, with technology constantly improving and especially through the introduction of generative models, the skill and effort it takes to produce a photorealistic fake image has been drastically reduced. What once took years of practice can now be done in seconds. All it takes is selecting the area of an image, describing a change in natural language and passing it on to a model like Midjourney [14]. Only a couple of years ago, AI-generated images only made for a nice cover of a book about deep learning [15]. Nowadays, the results are truly stunning and can already cause confusion. Just this year, an image of the pope wearing a stylish puffer jacket caused outrage about the perceived waste of money by the Catholic Church (figure 1.1). The "success" of the image is likely based on two main features. First, the fake was very good. The image did not give any clear indications of being fake except for potentially portraying something slightly unrealistic. Second, it triggered a confirmation bias [16]. The church wasting money was not a new sentiment. The image just had to be realistic enough to enforce this

Figure 1.1: Viral Tweet showing an AI-generated picture of Pope Francis in a puffer jacket [1].

idea. But, rather than diving deeper into the intricacy of the human mind, this work focuses on solving the first part of the problem. If humans cannot detect images generated by models anymore, it might be time for models to do the detection themselves.

Image Forgery Detection means detecting if an image was manipulated and identifying which areas of the image are fake. It differs from other image classification tasks, as models are supposed to detect something hidden between the pixels. Therefore, the architectures used in this field are highly specialized to emphasize image features that go beyond the three color channels and can indicate forgery. In this work I will address Image Forgery Detection as an image segmentation task. Instead of dividing the image into semantic categories a human can see, like dog, sky or car, I train a model to separate the image into categories only the model can see, i.e. real or fake. With my experiments, I explore if general image segmentation models with millions of parameters are able to outperform highly specialized but much smaller Image Forgery Detection models. Moreover, I will address a new family of "AI-powered" forgery tools. For this I will combine the advances in detecting AI generated images and Image Forgery Detection and train a model for AI-inpainting detection. AI-inpainting is a technique that uses deep generative models to alter or recreate parts of an image. The output is an image that is partly

AI generated and partially authentic. The goal of the detector is to distinguish the generated area from the rest of the image.

The document is structured in the following way: chapter 2 reviews relevant literature on Image Forgery Detection as well as AI image generation, chapter 3 describes the research design as well as the data and models used, chapter 4 describes the experiments and discusses the results and chapter 5 provides final conclusions as well as an outlook for further research.

2

Literature Review

This chapter is divided into six sections: section 2.1 reviews common Image Forgery techniques, section 2.2 focuses on the contributions of generative AI, section 2.3 summarizes classical Image Forgery Detection methods, section 2.4 gives an overview of Deep Learning-based Image Forgery Detection approaches, section 2.5 describes frameworks for the detection of AI-generated images, and section 2.6 summarizes recent approaches to AI-inpainting detection.

2.1 Classical Image Forgery

Image Forgery is the act of altering the content of an image with the intent to modify or misrepresent its information. The term "classical" is used to describe forgery methods predating the introduction of AI tools to this field. In this section, I will review the most common classical Image Forgery techniques.

First are techniques that do not change the actual content of the image. This includes changes of contrast, colorization, application of different filters, etc. Second are traditional techniques for inserting or removing content from an image which are, splicing, copy-move and inpainting (figure 2.1). Splicing describes the act of taking an object from one image and inserting it into another. Copy-move works similarly but copies an object inside of an image and inserts it again within the same image at another position. Lastly, inpainting is mainly used when removing objects. The empty space left is filled by extrapolating and copying other image parts.

Figure 2.1: Examples of classical Image Forgery, with the original images on the top and on the bottom from left to right: inpainting, copy-move and splicing [2].

Third are complementary techniques used to conceal the impact of the ones mentioned above. These include reshaping and resizing of objects, blurring or smoothing of edges, etc.

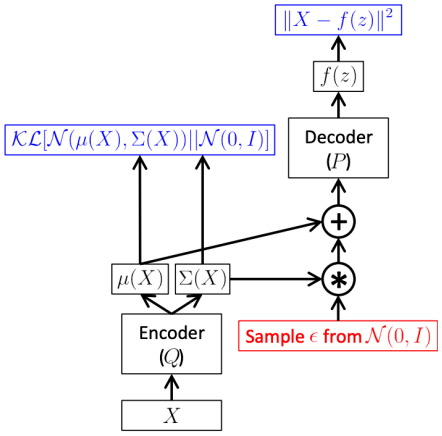
2.2 AI Generated Images

The two most common architectures for generative models for images are Variational Auto-Encoders (VAE) [17] and Generative Adversarial Networks (GAN) [18]. Both models learn to generate images from a random distribution by approximating the training data distribution.

The VAE is built like a traditional Autoencoder consisting of an encoder and a decoder. During training, the encoder maps input images to a lower dimensional latent space. The decoder reverses this process, taking as input a feature vector from the latent space and producing an image as output (figure 2.2). The difference to an Autoencoder is that, during generation, the input of the decoder is sampled from a random distribution which is parametrized using the latent variables. The system is trained by maximizing the variational lower bound

$$\mathrm{E}_{z\sim Q}[\log P(X|z)] - D_{KL}[Q(z|X)\|P(z)].$$

This equation is maximized by increasing the likelihood for the decoder P to reproduce the data X with the input z (first term). Here, z is sampled from the distribution Q of the encoder. The second term is the Kullback-Leibler divergence of $Q(z|X)$ and the prior distribution for



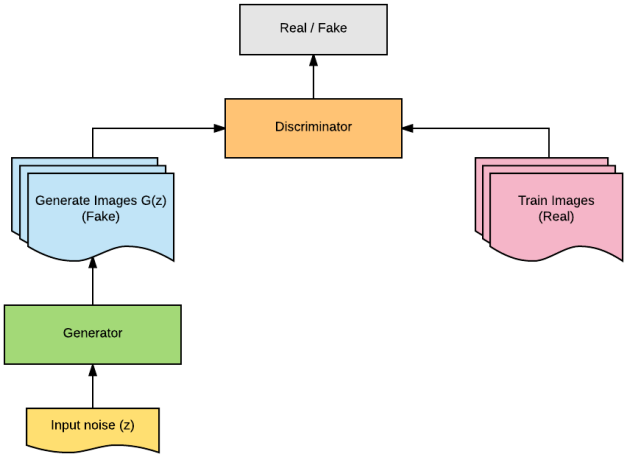


Figure 2.2: High-level architecture of a VAE [3]. **Figure 2.3:** High-level architecture of a GAN [4].

the latent variables $P(z)$, which is generally assumed to be Gaussian. Maximizing both terms together increases the likelihood of producing data similar to the input while keeping the latent distribution close to a Gaussian.

A GAN consists of a generator which generates images from a random distribution and a discriminator which attempts to distinguish between real and fake images (figure 2.3). During the training, the two compete until the generator learns to mislead the discriminator by generating images that are not distinguishable from the training data. The learning can be formulated as a zero-sum game

$$\min_G \max_D \mathrm{E}_{x \sim p_{\text{data}}} \log D(x) + \mathrm{E}_{z \sim p_{\text{model}}} \log (1 - D(G(z)).$$

with D the Discriminator, G the Generator, p_{data} the distribution of real data, p_{model} the distribution of noise going into G, x a sample from the real data, and $G(z)$ a sample from the Generator. D assigns a probability for its input to come from p_{data}. Therefore, the goal for the Discriminator is to assign high values to the real samples x (left term) and low values to the fake samples $G(z)$ (right term). The Generator, on the other hand, tries to mislead the Discriminator to assign high probabilities to generated samples.

After generating images from a random distribution, the next step was generating images from a natural language prompt. One of the first models that got much public attention for text-to-image generation was DALL-E [19]. The researchers combined the generative abilities of a VAE with a transformer [20]. First, the VAE is trained in the same way as before. Then,

given a text and image pair, the transformer is used to learn to map the text input to the latent space of the VAE for the respective image. Once both parts are trained, text can go as input into the transformer, which produces the latent embedding that then goes into the decoder of the VAE, which then creates an image.

Another milestone in the development of image generation was latent diffusion [21], which combines the idea of a latent space representation of images from VAEs and GANs and diffusion models. On a high level, a diffusion model learns to reverse a process of gradual addition of random noise. Although showing remarkable results for images, these models have been too costly to use directly on the pixel domain. This limitation was overcome by combining them with an Autoencoder and applying the diffusion process to the lower dimension latent space.

Besides the mentioned techniques, there have been many further improvements and tweaks to these models over the past years. With many models easily accessible and companies like Photoshop starting to offer "AI-powered" solutions to their customers, AI-generated or AI-manipulated images are everywhere. Whether it is an image of the pope wearing his favourite jacket or Donald Trump getting arrested [22], these images already manage to spread false information.

2.3 Image Forensics

In this section, I will summarize the main characteristics of forged images and how to use them for detection. A great source to learn about image forensics is Prof. Hany Farid's tutorial on Digital Image Forensics [5]. In this extensive work, Farid explains many different approaches to digital image forensics ranging from format-based methods like double JPEG compression to physics-based forensics, analyzing if the content of an image obeys the laws of physics.

Format

A popular format for digital images is JPEG. To be precise, JPEG is less a format but a lossy compression algorithm. Following different steps of applying Discrete Cosine Transform and a quantization step allows storing images with little loss of information. Characteristics of this process can be used to identify tampered images.

Besides the pixel values, JPEG files contain metadata. That includes information like the compression factor or the JPEG version, but also focal length, exposure time, etc. which differ from camera to camera and smartphone to smartphone. While this combination may not

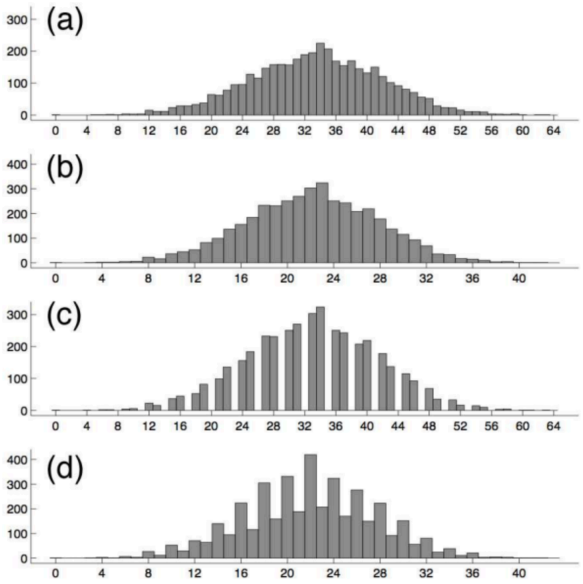


Figure 2.4: Distribution of pixel values in an image for single (a)-(b) and double (c)-(d) JPEG compression [5].

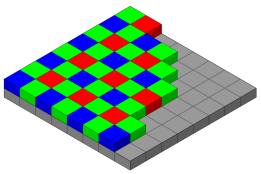


Figure 2.5: Example of a Bayer Filter [6].

always be a unique identifier for the camera model, if an image header does not align with any common combinations, it is likely that the image did not come directly from a camera.

Before loading a JPEG image, for example, into Photoshop, it might have already gone through a first stage of compression. When being altered and saved again, it undergoes a second compression. This Double JPEG Compression can leave artefacts in the image histogram. The main reason for this phenomenon is the quantization step of the process. Figure 2.4 shows an image histogram for four quantization steps. With compression factor 2 (a), compression factor 3 (b), double compression first with factor 3 and then with a factor 2 (c), and double compression the other way around (d). The histogram in (c) shows periodically empty bins. This artefact arises because the first compression is stronger than the second. The first quantization step, with a factor of 3, redistributes the values from 128 to 42 bins. In the second step these 42 bins are redistributed in 64 bins, leaving several new bins empty. Similarly, when the quantization increases, as shown in (d), values concentrate in certain bins.

Camera

Taking a photo with a camera differs from generating an image with AI or alternating it with classic Photoshop tools. A major difference is that taking a photo involves a camera sensor

that underlies certain physical constraints. Most digital cameras (including smartphones) use CMOS image sensors, a specific type of Active Pixel Sensor (APS). A property of these sensors used for Image Forgery Detection is the Color Filter Array (CFA). Here, each pixel is assigned a value of red, green or blue that repeats over the area of the sensor in periodic order, similar to a checkerboard. One of the most common designs is the Bayer Filter (see figure 2.5). To obtain an image where each pixel is not monochromatic but rather a superposition of all three colours, a step called Demosaicing is required. This process employs an algorithm to interpolate colours between the different pixels*. This process leaves periodic artefacts in the image. When an image is tampered with, these patterns break. For example, when copy-pasting a part of another image, the frequency of the colours will be different between the old and the new image. This can be detected with CFA analysis. Further methods that utilize the characteristics of the camera sensor are Chromatic Aberration and Sensor Noise.

Image Features

When adding an object from another image, this object likely has to be rescaled to a new size, which introduces correlations between the pixels. For a one-dimensional signal x of length n which is resampled to a new signal y of length m, when only considering linear transformations, this process takes the form of

$$y = Ax$$

with $A \in \mathbb{R}^{m \times n}$. In this case each element of y can be expressed as a linear combination of elements in x

$$y_i = \sum_{j=0}^{n} a_{i,j} x_j.$$

This correlation can be detected.

Detecting cloned parts within an image involves comparing various regions of the image with one another. Utilizing a brute force approach for this task would become unwieldy quickly. Therefore, approximate methods like Scale Invariant Feature Transform (SIFT) are employed, which transform the image into a series of local feature vectors.

Beyond the pixels

Previous approaches have in common that they analyse the image file. What was not taken into account yet is the semantic image content. Farid mentions several possible indicators beyond

*For the curious reader, I recommend Capturing Digital Images (The Bayer Filter) - Computerphile

the pixel values or the file header. These include reflections and shadows, lighting, perspective, and the general plausibility of the scene.

The two main takeaways from this brief review of Image Forensics techniques are there is No Free Lunch, but we are getting some appetizers. Some methods can be clear indicators in some cases but not in others. If the header of a JPEG file contains the information that the file was saved from Adobe Photoshop, this is good news, but if someone goes through the work to remove this data, this feature becomes useless. There is not one solution that fits them all. On the other hand, there are several features a Deep Learning framework could make use of. Statistical differences in the pixel values, repeated areas in the images or periodic colour distributions could all be detected by a model. Considering the capabilities of current Deep Learning models in "understanding" the input content, it would even seem possible for a model to pick up on misaligned shadows or non-physical behaviour. The following section reviews the advances made in Deep Learning Image Forensics and how models use the indicators discussed in this section.

2.4 Deep Learning Approaches to Image Forensics

Deep Learning models led to improvements in many different disciplines, including Image Forensics. Two surveys from 2020 review the first approaches making use of deep neural networks [23], [24]. Both studies find that Deep Learning models perform much better than traditional methods.

Although there are different approaches, all models aim to detect a change in the distribution of pixel values within the forged area compared to the original image. To achieve this, most of the reviewed models do not use raw RGB values of an image as input but rely on image preprocessing from classic image forensics like Discrete Cosine Transform or Laplacian Kernels.

To overcome the constraint of predefined features, Bayar and Stamm [25] proposed a new convolutional layer. Instead of relying on predefined filters, the model learns the dependencies between pixels itself. To detect the structural changes between different areas, the model first needs to focus on local structures rather than global ones. The proposed layer uses prediction error filters which predict the value of the central pixel of the window using the remaining pixels in the filter window. This process emphasizes local dependencies, allowing to detect changes throughout the image. This concept is closely related to the noise residuals from the Spatial Rich Model (SRM) [26] known from steganalysis.

Another interesting example is the model by Cozzolino and Verdoliva [27]. The authors

treat the problem of splicing detection as anomaly detection using an Autoencoder. First, the model is trained to represent the images. Then each pixel is classified based on the error level. As spliced regions are assumed to be different from the rest of the image, the model is expected to make more errors in that region. The output is a pixel-level classification mask.

Most of these works operate on a specific type of forgery with a specialized model architecture. In a more recent survey, the authors reviewed models with multipurpose capabilities [2]. For example, to improve the detection of inter-pixel dependencies and enable the model to detect different types of forgeries, Bappy et al. [28] introduced a hybrid architecture including an Autoencoder and an LSTM. The LSTM detects dependencies between pixels that might occur due to resampling of the spliced object. These features are then added to the encoder embedding of the Autoencoder and fed into the decoder.

Zhou et al. [29] proposed a multi-branch model. One branch uses the RGB channels, while the second branch uses the SRM noise features as input. The overall architecture is that of a Faster R-CNN [30] known from object detection. The model outputs bounding boxes that mark the doctored regions.

A more recent contribution that was not part of the reviews is TransForensics [31]. It is the first image forensics model to use self-attention [20]. The attention mechanism improves the model's flexibility in learning the inter-pixel dependencies leading to state-of-the-art results on the most common Image Forgery datasets.

To summarize, Deep Learning has drastically improved the capabilities of Image Forgery Detection. But, most models still rely on predefined features from classical approaches. This is implemented either by preprocessing the data and using the features as input into the model or by implementing the functionality into an end-to-end system. Whilst some models use features related to the file format or camera model, most approaches focus on inter-pixel dependencies as the main feature for detecting Image Forgery. Over time models have improved from image-level classification to either pixel-level classification (similar to image segmentation) or bounding box prediction (similar to object detection).

2.5 Detecting AI Generated Images

As described in the previous section, the detection of classic Image Forgery techniques with Deep Learning is mainly based on statistical dependencies between pixels. These dependencies are altered during forgery operations, for example, through resampling pixels or combining different images. This section investigates the characteristics of AI-generated images.

Similar to the "Beyond the Pixel" methods, some techniques can detect logical inconsistencies in the generated material. For example, detecting unusual blinking patterns in AI-generated videos [32], or mouth shapes that do not fit the spoken words [33]. Bau et al. [34] made an interesting analysis of what a GAN cannot generate. For instance, in a comparison between real and fake images using image segmentation, they found that GANs produced significantly fewer pixels classified as humans.

Making use of inter-pixel dependencies also proved to be an effective approach to detecting AI-generated images. Mo et al. [35] were able to detect AI-generated images of faces with over $95\,\%$ accuracy, picking up on statistical artefacts by the generating model. Their detector was a simple CNN-based architecture taking as input the images and applying a high pass filter.

More research into GAN-specific artefacts showed that the up-sampling and deconvolution process in the decoder of the network produces characteristic structures in the images. When transforming the data from a lower dimensional latent space to the high dimensional image space, the model has to create high-resolution features from low-resolution data. The introduction of artefacts during this process was first shown by Odena et al. [36] detecting a checkerboard structure in GAN-generated images. Zhang et al. [7] continued this analysis using Discrete Fourier Transform (DFT). Figure 2.6 shows a comparison of a real and a fake image and their respective spectrum in the frequency domain after applying DFT[†]. The spectrum shows distinct high-frequency artefacts for the GAN image that are not present in the real image. Although the detector could detect images generated by a specific generative model, it failed to generalize to data from other models not seen during training. Later it was shown that this can be improved by making use of more diverse training data as well as data augmentation [37]. Goebel et al. [38] managed to detect which out of six different GANs had generated which images.

The aforementioned research was done before text-to-image models became widely accessible. In a recent paper Sha et al. [39] explored the capabilities of text-to-image models and their detection. Two of these models are DALL-E 2 and Stable Diffusion. Though the authors were able to detect images from Stable Diffusion with about $90\,\%$ accuracy, the classification accuracy for images from DALL-E 2 was only slightly above $50\,\%$ (on a balanced dataset) and therefore close to chance. To overcome this issue, the authors considered a new detection paradigm. For each image, they created an image description with the image captioning model BLIP [40]. Then, they embedded the image and the caption using CLIP [41] and used the concatenated embeddings for classification. This approach partly improved the performance of the detec-

[†]The frequencies are shifted, with low frequencies in the centre of the spectrum.

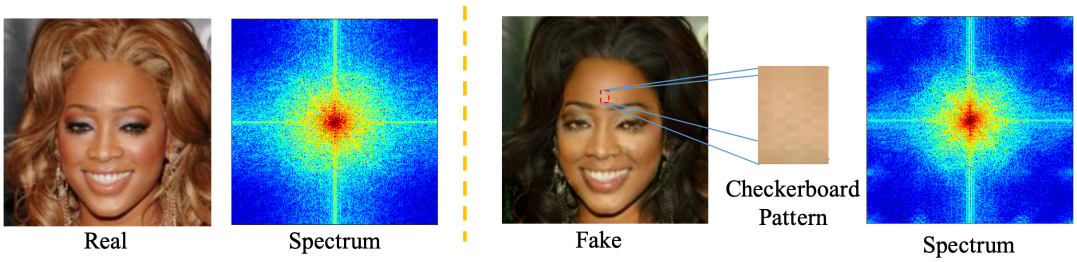


Figure 2.6: Real image (left) and AI generated fake image (right) with their respective spectral image showing frequency artefacts for the generated image [7].

tor[‡].

Detecting AI-generated images is highly model dependent as detectors pick up on specific characteristics of the generative models. These can be categorized into two main types: high-frequency image characteristics, which emerge during the upsampling process, and semantic characteristics, which contribute to the overall content and meaning of the image.

2.6 AI-Inpainting Detection

As described in section 2.1, image inpainting is used to recover lost or removed parts of an image. There exist different deterministic algorithms like Navier-Stokes-based inpainting [44]. The method uses the Navier-Stokes differential equations from fluid dynamics to fill in missing image information from its surroundings. Newer approaches use deep generative models, i.e. AI-inpainting, which can fill in missing image information or alter image parts based on natural language prompts. Given the relatively recent emergence of these models, there hasn't been a well-established state-of-the-art in terms of their detection. The main bottleneck for this research is the lack of established datasets. Therefore, each researcher had to create their own data, making it difficult to compare the results. So far, approaches rely on the use of noise features, as already seen in section 2.4. Two contributions are outlined below in more detail.

Wang et al. [45] trained a model to detect different inpainting methods, including traditional algorithms and Deep Learning. They approached the problem as an object detection task using a modified R-CNN to predict a bounding box around the tampered area. Similar to previous approaches in Image Forgery Detection, they used noise features as an additional input into

[‡]The authors considered two datasets, MSCOCO [42] and Flickr30k [43]. On the first, the performance increased up to 80 % accuracy, on the second, there was no improvement at all.

the network. It was done using the Local Binary Pattern (LBP) [46] similar to the SRM. The data source were images from MS COCO [42] that the authors manipulated themselves.

Li et al. [47] introduced the Noise-Image Cross-fusion Network (NIX-Net). The network uses both RGB and SRM features as input and predicts a binary segmentation mask. Moreover, the authors used special training data to improve the generalizability of the detector between different inpainting techniques. First, they trained an Autoencoder to fully reconstruct authentic base images from the Places dataset [48]. Then, they replaced parts of the real images with the respective area of the reconstructed images to emulate differences in the noise patterns between real and fake image regions. A detector trained on this model-agnostic dataset was able to detect inpainted regions from different generative models.

3

Research Design & Data

This chapter is divided into three sections: section 3.1 states the main research questions and describes the experiments, section 3.2 gives an overview of the data used, and section 3.3 describes the architecture of the used models in depth.

3.1 Research Design

I approach the problem of Image Forgery Detection as an image segmentation task. Unlike previous approaches that relied on highly specialized models, I make use of the advances in universal image segmentation. Recent models, like MaskFormer [9], successfully combined different segmentation paradigms showing a high level of flexibility to adapt to the different tasks. Thus, I explore the applicability of these advanced models in the context of Image Forgery Detection. To address this, I focus on two main research questions:

1. How do large image segmentation models perform on the task of classical Image Forgery Detection?

2. How do modern large image segmentation models perform on the task of AI-inpainting detection?

To answer these questions, I carry out two sets of experiments.

3.1.1 Classical Image Forgery Detection

The first set of experiments focuses on identifying classical image forgery techniques as described in section 2.1. Existing detection models have been specifically designed for this purpose. In addition to RGB image features, these models used specialized modules like the Bayer filter [6], which allowed them to pick up on inter-pixel dependencies that change during the image tampering. A downside of this approach is the required level of feature engineering, limiting the model's flexibility. For example, this includes selecting suitable kernels for surfacing noise features.

In recent years, image segmentation models like the MaskFormer [9] made significant advances, demonstrating strong versatility. I will test how these models adapt to the Image Forgery Detection task. Therefore, I will compare the performance of large image segmentation models in detecting forged image regions to state-of-the-art forgery detection models.

I will use the CASIA Image Tampering Detection Evaluation Database v2 [49] for the experiments. This dataset is a cornerstone of Image Forgery Detection experiments, providing an excellent basis to compare the performance of the models trained during this work to other forgery detection models. Section 3.2.1 describes the dataset in more detail.

The models used during the course of this work are MaskFormer [9] and Mask2Former [10]. The models were selected due to their availability via the huggingface library and good performance on the benchmark *Semantic Segmentation on ADE20K* [50]. The architecture for each model is described in detail in section 3.3.

3.1.2 AI-Inpainting Detection

The second set of experiments extends the work beyond the detection of classical image forgery tools. Over the past years, even months, many AI image manipulation tools have become widely accessible. In May this year (2023) Adobe released it's AI-inpainting tool Generative Fill for beta testers [51]. I will test the models from the previous section in their capabilities to detect image sections created with one of these tools.

Due to the novelty of the tools, there does not exist a common AI-inpainting dataset. Therefore, I create a new dataset combining two different sources, i.e. the Places205 dataset as well as test data from the MAT model. This dataset is described in more detail in section 3.2.2.

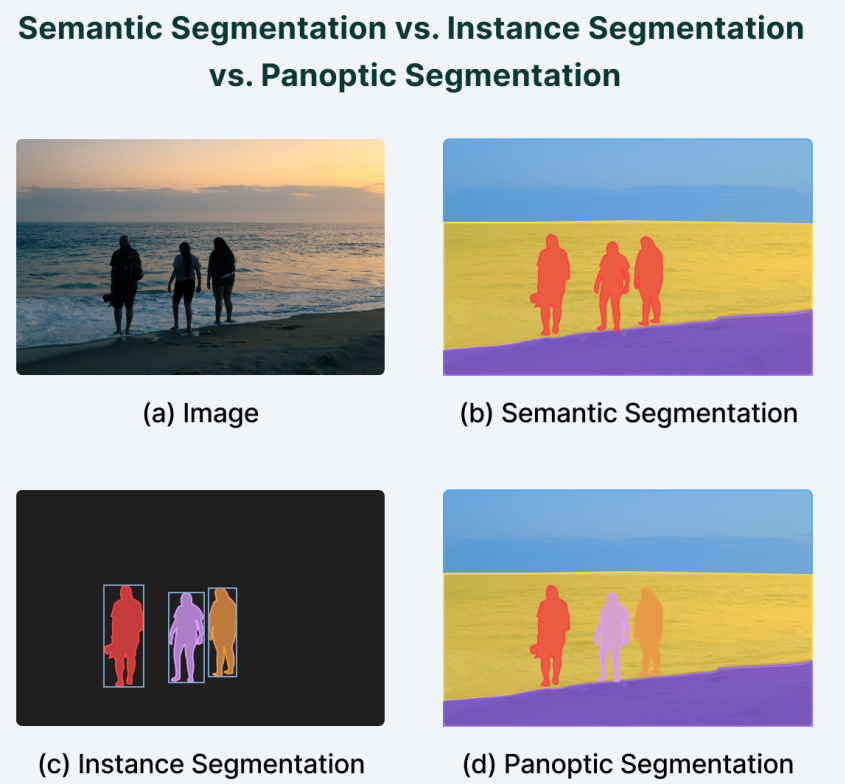


Figure 3.1: Examples of the three image segmentation paradigms [8].

3.1.3 Image Segmentation

There are three main image segmentation paradigms, i.e. instance, semantic, and panoptic segmentation (figure 3.1). Instance segmentation means to identify different objects or *things* in the image. The model outputs different object masks indicating which pixel belongs to an object and which not. Semantic segmentation does not focus on individual objects but rather on semantic categories. As shown in figure 3.1, semantic segmentation does not differentiate between different people in the image as all of them belong to the same *person* category. The model outputs one semantic map indicating the category of each individual pixel. Lastly, panoptic segmentation combines the previous two approaches by separating the image into semantic categories and distinguishing between different instances within these categories.

Initially, the three tasks were addressed separately by different models. Carion et al. [13] successfully combined them with their Detection Transformer (DETR). The model first predicts object masks for instance segmentation and, in a second step, merges the different masks into one panoptic segmentation map. This approach has been refined with the models MaskFormer

[9] and Mask2Former [10]. These models are described in more detail in section 3.3.

In the experiments, the models are trained using the semantic segmentation paradigm. The goal is to not focus on objects but rather on regions. This enables the model to detect a different manipulations, for example, inpainting which usually removes and object and replaces it with background. This would not fall under the scope of an object detection model. Similar to previous approaches that detected discrepancies between the forged region and the rest of the image, the scope of this work is to train the models to distinguish between real and fake regions independent of whether they are showing an object or not. Therefore, the models are trained on a binary image segmentation task classifying each pixel as either *real* or *fake*.

3.2 Data

3.2.1 CASIA v2

The CASIA Image Tampering Detection Evaluation Database v2 [49] was first introduced in 2013 and is a cornerstone of Image Forgery Detection research. Most papers that were reviewed in section 2.4 evaluated their results on this dataset, making it a great source for comparison. Moreover, with 12.614 examples it is the largest dataset available. 5123 are tampered, and 7491 are authentic images.

The images have different sizes and formats, i.e. compressed (JPEG) and uncompressed (TIFF). The authentic images are classified in 9 categories: scene, animal, architecture, character, plant, article, nature, indoor and texture.

The tampered images were created with either copy-move, copying one part of the image and inserting it again at another position, or splicing, taking an object from one image and inserting it in another image. In addition the creators used blurring within the tampered regions to make the images look as realistic as possible to the human eye. Moreover, the dataset contains binary masks, indicating the tampered region for each image. Some examples are shown in figure 3.2. Figure 3.5 shows the distribution of mask sizes.

The size and widespread use of the dataset make it a good candidate for machine learning experiments. Other common datasets are COVERAGE [52] and Columbia [53], which only contain around 100 and 1000 tampered images and are therefore too small to fine-tune a large model like MaskFormer. The dataset is available on Kaggle [54].

A problem with the dataset is its entanglement. All 5123 tampered images were created from either one (copy-move) or two (splicing) of the authentic images. Under closer investigation,

Figure 3.2: Examples from the CASIA v2 test data with forged images on top and the respective ground truth masks below.

it turns out that only a subset of 1956 authentic images were used to create the tampered ones. This can lead to the situation that parts of an image that was used in training also appear in testing. This data leakage could inflate the results. One way to prevent this would be to remove all the base images that were already used during training from the test set. Unfortunately, for an 80/20 split, this would cause 88 % of the smaller test set to drop from the experiments. None of the reviewed works addressed this problem. The main scope of my first set of experiments is to evaluate if modern large image segmentation models can keep up with the performance of specialized architectures that use image forgery-specific features like noise. For this reason, I will test the models in the same way as previous approaches to be comparable. In the second step, I will assess how far data leakage affects the performance of the models during testing.

The data is split into training (80 %) and testing (20 %) in the same way for all experiments. Further, a fixed 20 % share of the training data is held out as a validation set during training. The exact numbers are reported in table 3.1.

Type	Train	Validation	Test
Tampered	3206	814	961
Authentic	4775	1182	1534

Table 3.1: Size of the train, validation and test set for the experiments with the CASIA v2 dataset.

3.2.2 Places205+MAT

The Places205+MAT dataset was created during the course of this work from two different datasets. The first one is Places205 [48], a common dataset for scene recognition. The second is test data from the Mask-Aware Transformer for Large Hole Image Inpainting (MAT) [11]. This model takes as input an image and a mask and tries to recreate the masked area. The model is described in more detail in section 3.3.3. The test data contains images that were reconstructed by the model and the respective groundtruth masks.

Places205. Places205 is a scene recognition dataset with 205 different categories of environments like *teenage bedroom*, *winter forest path* or *sunny coast*. Each category has more than 5000 images, and the dataset contains 2.5 million images in total. The dataset is a subset of the larger Places356-Standard. For this work I used a subset of 36.320 images, sampled randomly across all categories. The data is available on Kaggle [55].

MAT. The dataset contains images from Places365-Standard partially reconstructed by the MAT model. Figure 3.3 shows examples of reconstructed images and their masks. The data

and the model were selected due to both performance and accessibility. The model is currently on second place of the Places2 Image Inpainting benchmark [56]. Moreover, the model is available on github [57], and some of the test data is openly available on OneDrive [58]. The dataset contains 36.000 reconstructed images. The masks were created randomly sampling [0, 3] rectangles of different sizes and [0, 4] brush strokes of different widths. The average mask density for the training set used in this work is 0.217. The distribution of mask density is shown in figure 3.4.

The combined Places205+MAT contains 72.320 examples, half inpainted by MAT, the other half original. Unfortunately, the authors of MAT did not provide the original images or file names making it impossible to match the reconstructed images with the respective originals. But, due to the size of the dataset and the random sampling of the authentic images, data leakage should not be a problem. The data is split in $80\,\%$ training and $20\,\%$ testing data. Furthermore, $20\,\%$ of the training data is held out during training as a validation set. The exact sizes of the different splits are reported in table 3.2.

Type	Train	Validation	Test
Inpainted	23 360	5840	7300
Authentic	23 360	5760	7200

Table 3.2: Size of the train, validation and test set for the experiments with the Places205+MAT dataset.

3.2.3 Stable Diffusion

The MAT data is comparable to prompt base image inpainting data like the one generated through Photoshop Generative Fill in the way that the final product in both cases is an image that is partially authentic and partially AI-generated. But, there are two main differences between these two types of data.

First, inpainting tools like Photoshop Generative Fill take as input not only an image and a mask, but also a text prompt telling the model what to fill the masked area with. For MAT, this additional constraint does not exist. The model only has to match the style of the residual image, but it does not have to follow an additional semantic constraint. This changes the way the model is trained and, therefore, impacts the kind of data it generates.

Second, for MAT, the masked information is lost. Therefore each pixel within the masked area is generated by the model. For prompt-based inpainting models this information is still accessible. In fact, the model tries to extend this part of the image rather than recreating it.

Figure 3.3: Examples from the MAT test data with manipulated images on top and the respective groundtruth masks below.

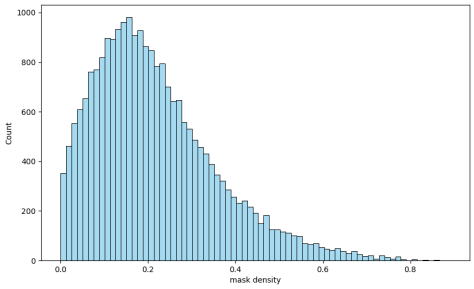


Figure 3.4: Distribution of mask density for the MAT training data.

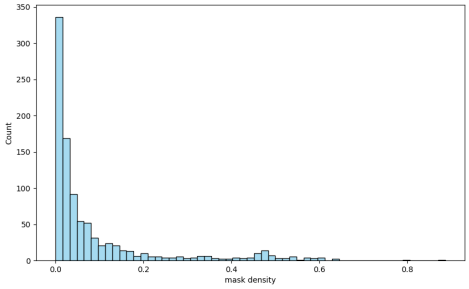


Figure 3.5: Distribution of the mask density for tampered images in the CASIA v2 training set.

This also changes the model's behavior and the value that can be provided by groundtruth masks. For the prompt-based inpainting model, not every pixel within that mask will be model generated and, therefore, share certain model-related properties. This means, that the mask also contains falsely labeled pixels, making it harder to train a model.

To better understand the performance of the MaskFormer on detecting AI inpainting, I have created a second dataset. This dataset contains 100 images created using stable diffusion. Though too small to train a model, it will allow me to test the generalizability of the models trained on Places205+MAT.

I use the implementation of stable diffusion inpainting by runwayml on huggingface [59]. The inputs are 100 images from Places205 that were not part of the sample used in Places205+MAT to prevent data leakage. The masks are randomly sampled squares implemented in the same way as for the MAT model. Lastly, the prompts are randomly sampled from a selection of 5 prompts with increasing levels of detail.

- a person
- a small box
- a basketball, high resolution
- a cute sea otter high resolution
- a person on a skateboard doing a kickflip in high resolution

Examples for each prompt are shown in figure 3.6. The images are very different in quality. For some of the images, the queried results are visible, while in other images, the results do not resemble the query, or the image did not change noticeably. As described above, the inpainted elements often do not fill the complete mask area. The average masked area is 24 %.

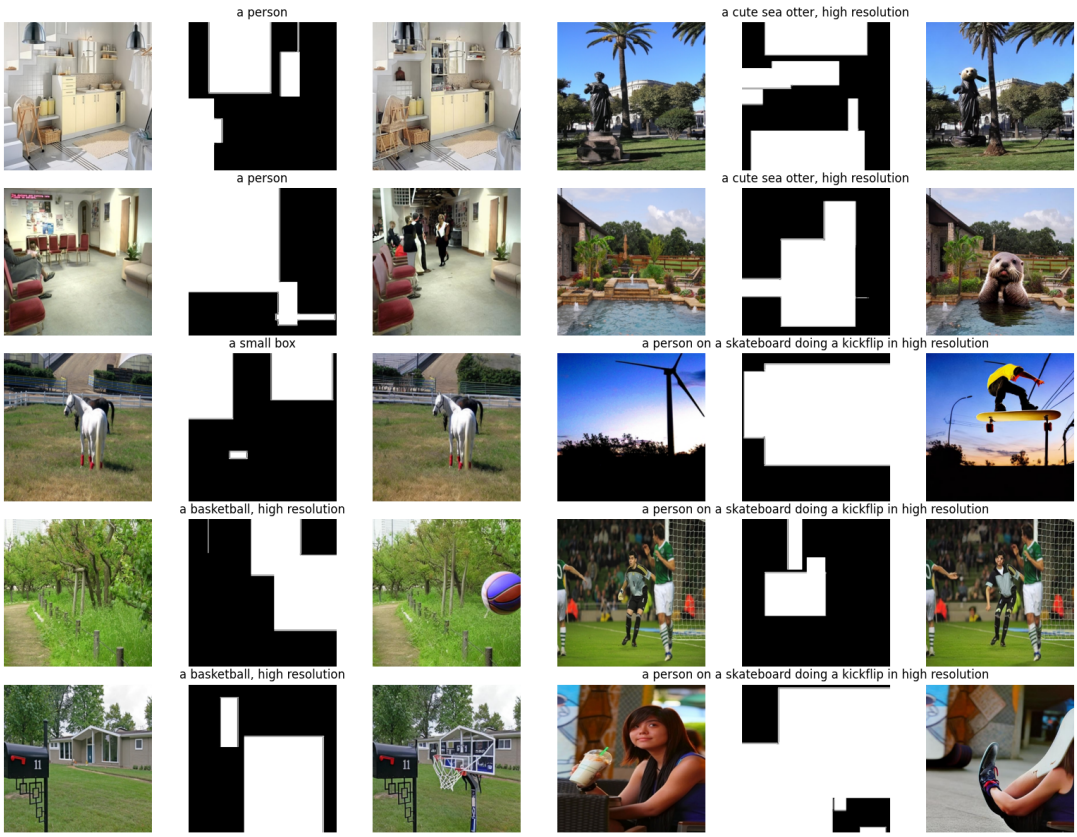


Figure 3.6: Examples from the data created with stable diffusion with original (left), mask (middle), stable diffusion image (right) and prompt (top).

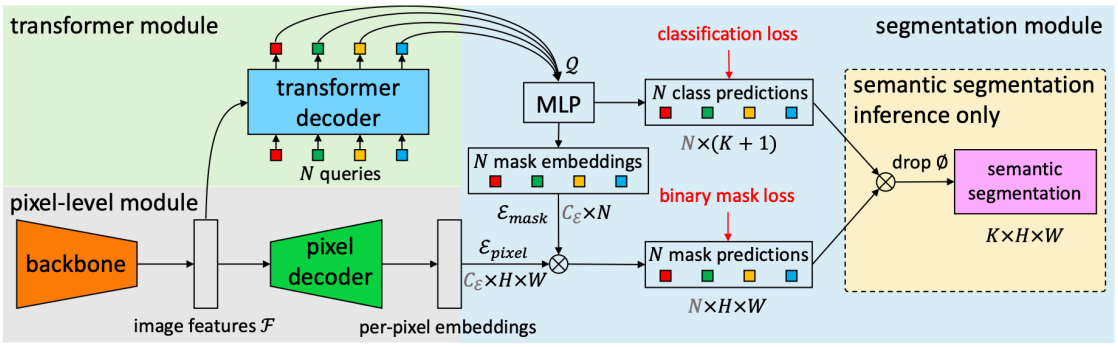


Figure 3.7: High-level architecture of the MaskFormer model [9].

3.3 Models

This section describes the different models that were used during the experiments.

3.3.1 MaskFormer

The MaskFormer [9] extends the previous DETR model [13] from combining instance and panoptic segmentation to including semantic segmentation. It achieved state-of-the-art results when it first came out and is currently on place 36 of the benchmark for *Semantic Segmentation on ADE20K* with a mIoU of 55.6 [50].

Figure 3.7 shows the overall architecture of the model. First, a backbone extracts image features to a lower dimensional feature map $\mathcal{F}$. In one part of the workstream (bottom), a pixel decoder upsamples the feature map to produce per-pixel embeddings $\mathcal{E}_{pixel}$. The second path applies a transformer decoder to $\mathcal{F}$, creating N per-segment embeddings $\mathcal{Q}$. Here, N is a hyperparameter and does not correspond to the number of final predicted classes K. Then a multi-layer perceptron takes $\mathcal{Q}$ to produce both, N class predictions and N respective mask embeddings $\mathcal{E}_{mask}$. The final mask predictions are obtained via a dot product between mask and pixel embeddings. In the last step, the N predicted masks and corresponding class predictions are combined into one final semantic map prediction.

Pixel-level module. The model uses either a ResNet [60] or a Swin Transformer [61] as a backbone. The latter is an improved version of the Vision Transformer (ViT) [62]. While the ViT computes attention globally on the whole image, the Swin Transformer computes attention locally in smaller windows. In the second step, these windows are shifted to allow for inter-window dependencies. This **S**hifted **Win**dow (Swin) approach reduces the computa-

tional complexity from quadratic to linear with respect to image size. During the initial testing of the MaskFormer, the Swin Transformer performed better than the ResNet. The feature map generated by the backbone has lower dimension than the input, scaled down by a factor of S with $\mathcal{F} \in \mathbb{R}^{C_{\mathcal{F}} \times \frac{H}{S} \times \frac{W}{S}}$ where $C_{\mathcal{F}}$ is the number of channels, and H, W are height and width of the input image.

The pixel decoder is based on the architecture of a Feature Pyramid Network (FPN) [12]. The network upsamples the low-dimensional and semantically dense feature map from the backbone in a step-wise manor. Here, the network follows the same steps as the backbone in reversed order*. It allows the network to not only take the output from the previous step as input, but also the respective feature map with the same dimensions from the backbone. This procedure is shown in figure A.1. At the end, a 1×1 convolution projects each pixel to and embedding of dimension $C_{\mathcal{E}}$.

Transformer module. The model uses a standard transformer decoder [20]. It takes as inputs the backbone feature map $\mathcal{F}$ and N queries. These queries are learnable positional embeddings. The exact architecture of the decoder is shown in figure A.2. The modules' outputs are N embeddings of size $C_{\mathcal{Q}}$, one for each segment.

Segmentation module. The segmentation module combines the outputs of the pixel-level module and the transformer module. The transformer output $\mathcal{Q} \in \mathbb{R}^{\mathcal{C}_{\mathcal{Q}} \times \mathcal{N}}$ passes through a linear classifier followed by a softmax to generate the N class predictions. For the mask embeddings, a Multi-Layer Perception transforms $\mathcal{Q}$ into N vectors with the same dimension as the pixel embeddings $C_{\mathcal{E}}$. Using a simple dot-product followed by a sigmoid activation function, these two embeddings are combined into N mask predictions assigning a probability to each pixel $[h, w]$ to belong to a certain mask. For semantic segmentation, the model combines the N mask predictions with the N class predictions. Each pixel is assigned to a class with the maximum combined probability of class and mask prediction

$$\hat{c}[h, w] = \underset{c \in \{1,\dots,K\}}{\arg\max} \sum_{i=1}^{N} p_i(c) \cdot \hat{M}_i[h, w]. \tag{3.1}$$

Here $p_i(c)$ is the probability of set i to belong to class c, $\hat{M}_i$ is the corresponding predicted mask and $\hat{M}_i[h, w]$ is the probability of pixel $[h, w]$ to be in that mask. $\hat{c}[h, w]$ is the class the model predicts for pixel $[h, w]$.

*For example, if the backbone reduces the dimension (H, W) by factors of 4, 8, 16 and 32 successively, the decoder increases the dimension again following the same steps in reversed order.

Loss function. The model is trained using a combination of three different losses, i.e. binary cross entropy for the class prediction and a combination of a dice loss [63], and a focal loss [64] for the mask prediction. Given a set of N predictions $\hat{z}$ and the corresponding ground truths z we get:

$$\mathcal{L}(\hat{z}, z) = \sum_{j=1}^{N} [-\log p_{\sigma(j)}(c_j) + \mathcal{L}_{mask}(\hat{M}_{\sigma(j)}, M_j)] \tag{3.2}$$

Here $\sigma(j)$ is a bipartite matching that assigns each of the N segments to a matching class mask.

The focal loss is a modified cross-entropy loss that puts less weight on good predictions and more on bad ones. This is achieved by the factor $(1 - \alpha_i)^\gamma$ with

$$\alpha_i = m_i \cdot \hat{m}_i + (1 - m_i)(1 - \hat{m}_i) = \begin{cases} \hat{m}_i & \text{if } m_i = 1 \\ (1 - \hat{m}_i) & \text{if } m_i = 0 \end{cases}. \tag{3.3}$$

m_i is the probability that pixel i belongs to this specific mask. If the prediction is good, i.e. close to either 0 if it is not or 1 if it is, the factor becomes very small (< 1) for values of $\gamma > 0$, reducing the weight of the loss. On the other hand, if the prediction is bad, the factor is close to one, and the loss stays close to its full value. For $\gamma = 0$, the loss is equivalent to the regular cross-entropy.

The dice loss is similar to the IoU, i.e. dividing the number of correct positive predictions by the number of total positive predictions. The only difference is that, here, each prediction is neither 1 nor 0 but the probability of the prediction being one. The combined mask loss is formalised as

$$\begin{aligned} \mathcal{L}_{\text{mask}}(\hat{m}, m) &= w_{\text{dice}} \cdot \mathcal{L}_{\text{dice}}(\hat{m}, m) + w_{\text{focal}} \cdot \mathcal{L}_{\text{focal}}(\hat{m}, m) \\ &= -w_{\text{dice}} \cdot (1 - \frac{2 \sum_{i=1}^{H \cdot W} \hat{m}_i \cdot m_i + 1}{\sum_{i=1}^{H \cdot W} \hat{m}_i + m_i + 1}) - w_{focal} \cdot \sum_{i=1}^{H \cdot W} (1 - \alpha_i)^\gamma \log \alpha_i \end{aligned} \tag{3.4}$$

where w_{dice} and w_{focal} are tunable weights.

Parameters. Different versions of the model are available via huggingface. Each version differs in the backbone used as well as the training data (table 3.3). Apart from the backbone the remaining hyperparameters are the same for each version (table A.1). The versions used in this work are either trained on ADE20K-full, ADE20K or COCO-Stuff-10k. ADE20K-full has 25k training examples and over 3000 semantic categories. Of these, only 847 were used during the training of the model. The authors also considered a further reduced version of

Model	Backbone	mIoU	#paramters
MaskFormer	R101c	47.2	60M
	Swin-S	51.0	63M
	Swin-B	52.3	102M
	Swin-L	55.6	212M
Mask2Former	Swin-S	52.4	69M
	Swin-B	53.7	107M
	Swin-L	57.3	216M

Table 3.3: Different versions of the MaskFormer and Mask2Former with either a ResNet backbone (R) or a Swin Transformer (Swin). The mIoU values are reported for the ADE20K validation set with 150 categories [10].

ADE20k with only 150 classes. COCO-Stuff-10k has 9k training examples and 171 different categories.

3.3.2 Mask2Former

The Masked-attention Mask Transformer (Mask2Former) has the same general architecture as the MaskFormer with two main improvements. These are the introduction of masked-attention and the use of multi-scale features [10].

One of the main efforts when training a transformer for object detection is learning which region one module should attend to. In the classical formulation of attention, each head initially can attend to the whole input. During training the learned attention weights narrow that field of attention down. For models like DETR and MaskFormer this leads to very slow convergence during training. Masked-attention alleviates this problem. The idea is the following: Each transformer module consists of several layers. Each layer takes as input the standard set of Query, Key and Value, as well as a mask. This mask is the binarized prediction mask of the previous layer. Therefore, all points outside of the mask predicted by the previous layer are discarded. This updated version of cross-attention is formalized in equation 3.5.

$$\mathbf{X}_l = \text{softmax}(\boldsymbol{\mathcal{M}}_{l-1} + \mathbf{Q}_l \mathbf{K}_l^{\mathrm{T}})\mathbf{V}_l + \mathbf{X}_{l-1} \tag{3.5}$$

Here, l is the layer index, $\mathbf{X}_l \in \mathbb{R}^{N \times C}$ are the N query features of that layer and $\mathbf{Q}_l = f_Q(\mathbf{X}_{l-1})$ is a linear transformation of the output of the previous layer. $\mathbf{K}_l, \mathbf{V}_l \in \mathbb{R}^{H_l W_l \times C}$ are the images features after linear transformations $f_K(.)$ and $f_V(.)$. H_l and W_l are the height

and width of the image features. Lastly, the attention mask is defined as

$$\mathcal{M}_{l-1}(x,y) = \begin{cases} 0 & \text{if } \mathbf{M}_{l-1}(x,y) = 1 \\ -\infty & \text{otherwise} \end{cases} \tag{3.6}$$

with $\mathbf{M}_{l-1} \in \{0,1\}^{N \times H_l W_l}$.

The second enhancement of the model is the use of multi-scale features in the transformer. Using high-resolution features is computationally costly, therefore, the MaskFormer uses the spatially low-dimensional output of the backbone as input into its transformer decoder. The Mask2Former, on the other hand, uses feature maps from all the upsampling stages of the pyramid-like pixel decoder (figure 3.8). Here, the first layer of the transformer decoder takes as input a low dimensional feature map (similar to the MaskFormer). The second layer takes as input a higher dimensional feature map as well as the mask prediction of the previous layer resized to the spatial dimension of the second layer. This allows for the use of high-dimensional features without having to compute attention on the whole feature map. The feature maps of the pixel decoder have resolution $1/32$, $1/16$ and $1/8$ of the original image.

With it's improvements, the Mask2Former achieves better results than its predecessor with only about 5M parameters more (table 3.3). Moreover, the training time was reduced drastically from 300 epochs for the MaskFormer to only 50 epochs. In this work I only consider versions of the model pretrained on ADE20K as described in the previous section.

3.3.3 MAT

The Mask-Aware Transformer for Large Hole Image Inpainting (MAT) [11] takes as input an image and a mask and reconstructs the masked area using the residual image information. The model is currently in the first place for *Image Inpainting on CelebA-HQ* [65] and second for *Image Inpainting on Places2* [56]. The key components of the model are its transformer modules using both shifted windows [61] and masked-attention as well as its Style Manipulation Module (SMM), which is similar to a StyleGAN [66]. The architecture of the model is shown in figure 3.9.

First, several convolutional layers downsample the input to aggregate information and reduce computational complexity by applying attention on a lower dimensional space. Next are five transformer modules. The size of the shifted windows increases first and then decreases again. Masked-attention, as described in the previous section, guides the model to focus on the

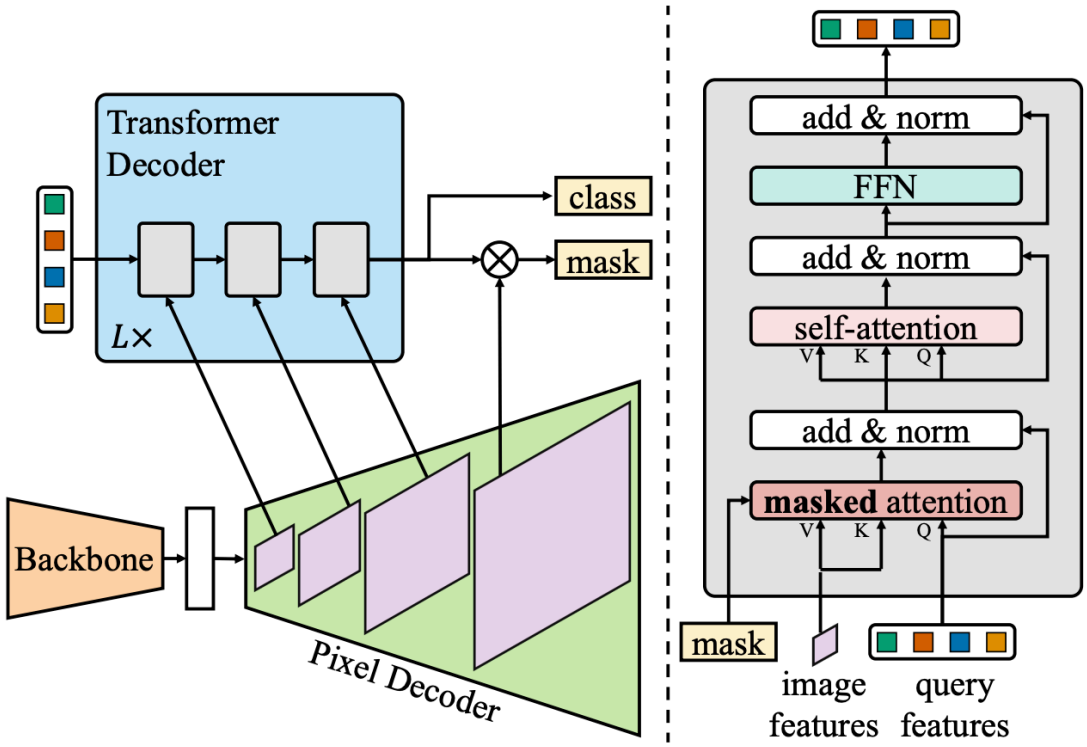


Figure 3.8: High-level architecture of the Mask2Former model (left) and design of the masked-attention transformer module (right) [10].

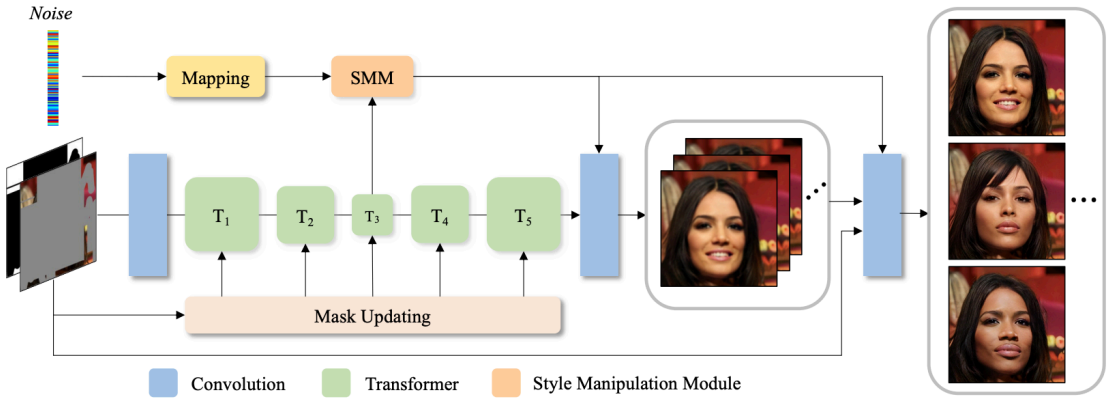


Figure 3.9: High-level architecture of the MAT model [11].

image parts that are not masked. The output of the last transformer module is passed again into a set of convolutional layers that upsample the features to the size of the original input and add the input from the SMM. The SMM replaces parts of the transformer output with random noise and produces scaling factors for the weights of the final convolutional layers, which upsample the data back to the input size. This allows for pluralistic and photorealistic image inpainting.

4 Experiments

This chapter is divided into two sections: section 4.1 describes the experiments carried out on the CASIA dataset for photoshop detection and section 4.2 reports the results for the experiments on AI-inpainting detection using both datasets Places+MAT and Stable Diffusion.

4.1 Image Forgery Detection

4.1.1 MaskFormer

Comparing architectures

The first experiment compares five different versions of the MaskFormer model. Here I consider three different backbones, namely Swin-S, Swin-B and ResNet101c. For the Swin-based models I test both versions trained on ADE20K with the reduced set of 150 labels and COCO with 171 labels. The ResNet101c is trained on ADE20K with the full label set which includes 847 categories. The model, parameters and training data are described in detail in section 3.3.1.

All five models are fine-tuned on the CASIA training dataset including both tampered and authentic images. The latter are included to improve the precision of the model to prevent the detection of fake regions in every image. The models are then trained for 20 epochs and an additional 20 if they did not overfit within the first round. The models are evaluated after each epoch on the validation set. Figure 4.1 shows learning curves for training and validation loss of

all models. Section A.2 describes the specifications during training in more detail.

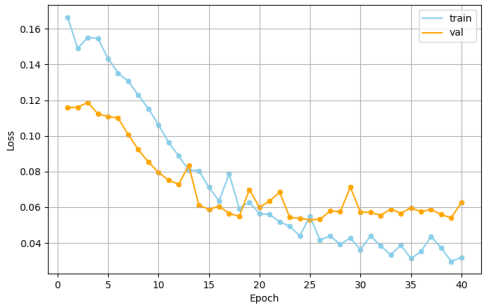
From a first evaluation of the learning curves, it is evident that all models overfit the data within a maximum of 40 epochs. All models reach about the same loss, around 0.06. But, the models pre-trained on ADE20K reached the same level earlier than models trained on COCO. ResNet101c starts overfitting first after 11 epochs. It is likely due to the reason that it is the model with the lowest number of parameters. Second are Swin-S-COCO, Swin-S-ADE20K and Swin-B-ADE20K. All three of them start overfitting at around 20 epochs. The slows model Swin-B-COCO where overfitting becomes visible at around 25 epochs.

All Swin models perform similarly on the validation set at the overfitting threshold. The ResNet101c performs slightly worse. Due to its faster convergence than the other Swin-based models, whilst having the same performance on the valdiation set, Swin-S-ADE20K was used for all further experiments.

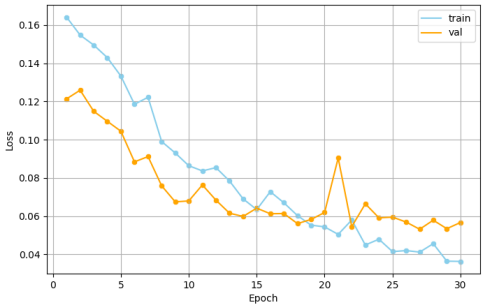
Swin-S-ADE20K was retrained with an early-stopping mechanism that stopped the training after 16 epochs when the validation loss did not improve for three epochs with respect to a threshold of 0.005. The f1 score of an image is calculated on pixel-level, where each pixel of the segmentation mask counts as one single prediction. Therefore, for example, the number of true positives is the number of pixels that were correctly classified as fake. As the f1 score for any authentic image is zero, I only report the score for the tampered images. Figure 4.2a shows the distribution of f1 scores for the tampered images from the test set. The plot shows that scores concentrate in the extremes, with most scores being close to 0 or 1. The median f1 score is 0.777. Moreover, the best 25 % of images all have an f1 score above or equal to 0.942. The mean value of predictions, on the other hand, is only 0.561. This is due to the 25 % of examples that have a score of 0 meaning the predicted mask did not overlap with the ground truth or the model did not predict any mask at all. Image-level precision and recall follow the same distribution as the f1 score (figure A.3) with a mean precision of 0.609 and mean recall of 0.578.

Table 4.1 compares the f1 scores of different models from the literature with the models trained during this work. The MaskFormer comes in second, two points ahead of the MFCN and 7 points behind TransForensics, the best model found.

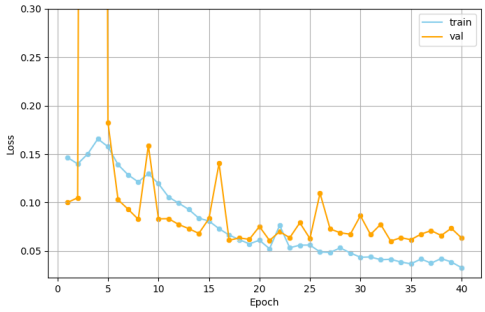
Figure 4.5 shows the ten examples from section 3.2.1 with their respective predictions from Swin-S-ADE20K. Six of the predictions match the groundtruth mask almost exactly. For the examples in rows four and seven the model failed to predict most of the mask. Both are examples where the image was mirrored. This could be an indication of a technique the model struggles to detect. Figures A.4 and A.5 show the ten best and ten of the zero predictions. The best



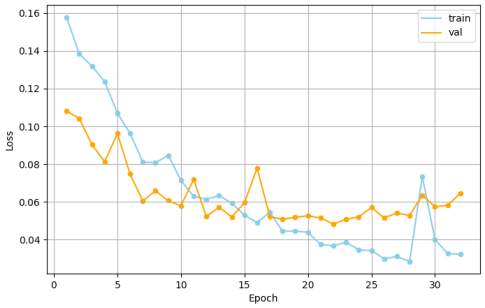
(a) MaskFormer Swin-S-COCO.



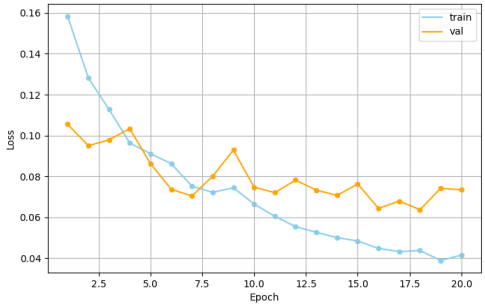
(b) MaskFormer Swin-S-ADE20K.



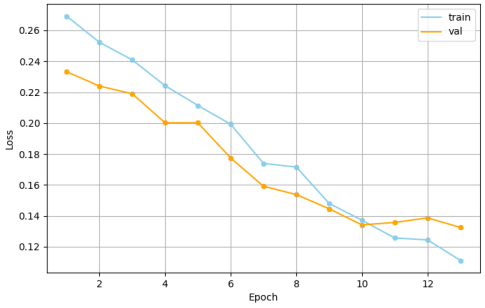
(c) MaskFormer Swin-B-COCO.



(d) MaskFormer Swin-B-ADE20K.

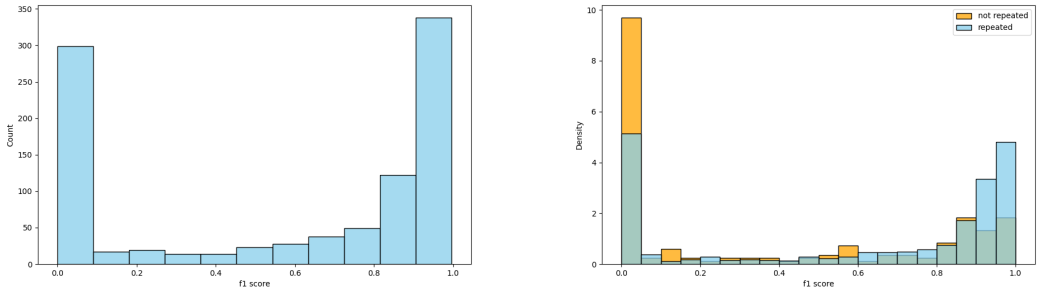


(e) MaskFormer ResNet-ADE20K.



(f) MaskFormer Swin-S-ADE20K.

Figure 4.1: Learning curves for different MaskFormer architectures. Reported are both training and validation loss while fine-tuning the models on the CASIA v2 dataset (a)-(e). The model in (f) is fine-tuned on tampered examples only.



(a) (b)

Figure 4.2: Distribution of f1 scores for the predictions of Swin-S-ADE20K on the tampered images from the CASIA v2 test set (a). On the right (b) these scores are separated in test images that were partly seen during training and completely new images. Here the scale is relative to be able to compare the two distributions.

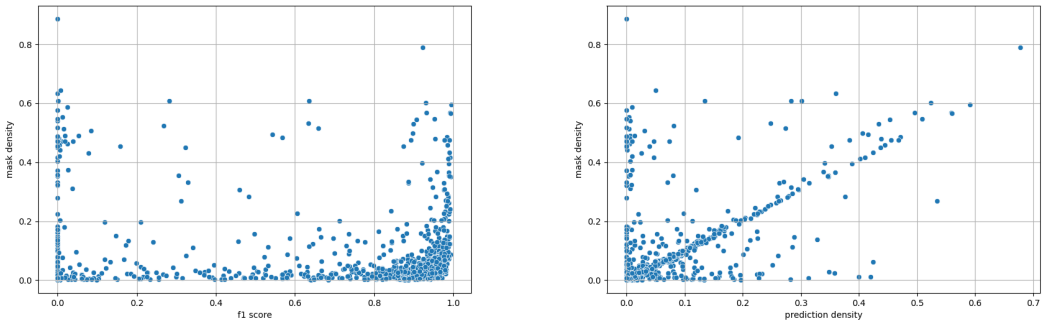
examples are mainly landscapes, where much of the image was replaced (often the sky). Reason for this could be the size of the fake region with respect to its border, inflating the f1 score. Further, the images often seem to have clear panes, like sky and mountains. Separating those two elements is similar to the original segmentation task of the model. The bad predictions are mainly images with small changes where a small object was either added, copied or removed.

Prediction trends

The distribution of the mask density gives further information on which trends influence the model performance. Figure 4.3a plots the mask density relative to the f1 score of the respective prediction. No correlation is visible, negating the assumption that large masks would inflate the f1 scores. On the contrary, the cluster in the lower right corner of the plot (high f1 score, small mask) indicates that the model predicts many small objects relatively well. With regard to prediction accuracy, figure 4.3b compares the size of the groundtruth mask with the predicted mask. The best predictions lay on a one to one line, with the predicted mask being the same size as the groundtruth. Besides that, points scatter above and below the line, not indicating any trend in predicting masks smaller or larger than the groundtruth.

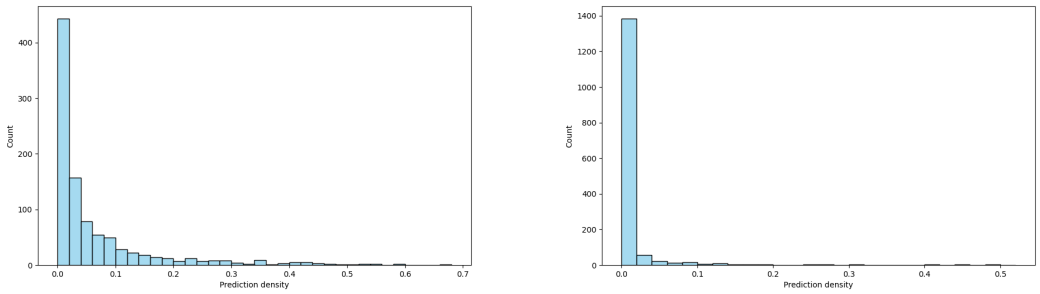
Model precision

In previous works, model performance was only evaluated on tampered images. What has not been addressed in the literature before is the performance on authentic images, i.e. how many false positives does the model produce. I measure the overall model precision as the share of au-



(a) (b)

Figure 4.3: Mask density as a function of the f1 score (a) and the density of the predicted mask (b) for the MaskFormer Swin-S-ADE20K on the CASIA v2 test set.



(a) (b)

Figure 4.4: Distribution of the prediction mask density for MaskFormer Swin-S-ADE20K on tampered (a) and authentic images (b) from the CASIA v2 test set.

thentic images that were correctly classified as fully authentic, i.e. the predicted mask does not contain any fake pixels. Figure 4.4 shows the distribution of predicted mask sizes for tampered and authentic images. For authentic images, the model predicted a mask in 30 % of cases. In comparison, for tampered images, the model predicted a mask for 89 % of the cases*.

Training without negative examples

In most examples from the literature, models were trained on tampered images only. To assess the effect of authentic images on the training, I fined-tuned another version of the MaskFormer Swin-S-ADE20K without authentic images. The learning curves are shown in figure 4.1f and

*In some cases the model predicted a mask that did not overlap with the groundtruth mask. This led to the f1 score being zero in 25 % of cases, but the prediction density only in 11 %.

the f1 score on the test set is reported in table 4.1. With a mean score of 0.607, the model performs better than the model trained on authentic and tampered images and is only 0.02 points behind the current state-of-the-art. This is caused by the lower share of predictions with 0 f1 score, which decreased by 4 percentage points to 21 % for tampered images. But, this performance improvement comes with a cost. Before, the share of authentic images where the model predicted a mask was 30 %. This value increased to 88 %. As expected, the model performs better in detecting tampered regions in tampered images but decreases strongly in precision.

Data leakage

Another important factor for the performance of the model could be data leakage. As described in section 3.2.1, many of the tampered examples in the test set are built from the same images as images in the training set. To get a better estimate of the true performance of the model, figure 4.2b compares the f1 score for both partly repeated and completely new images. The number of zero-predictions is much higher for the new images with 44 % compared to 21 % for the partially repeated ones. The mean f1 scores are 0.602 for the repeated images and only 0.362 for the new ones. Note that most of the images 83 % are partially repeated. This result clearly shows that data leakage influences the model performance.

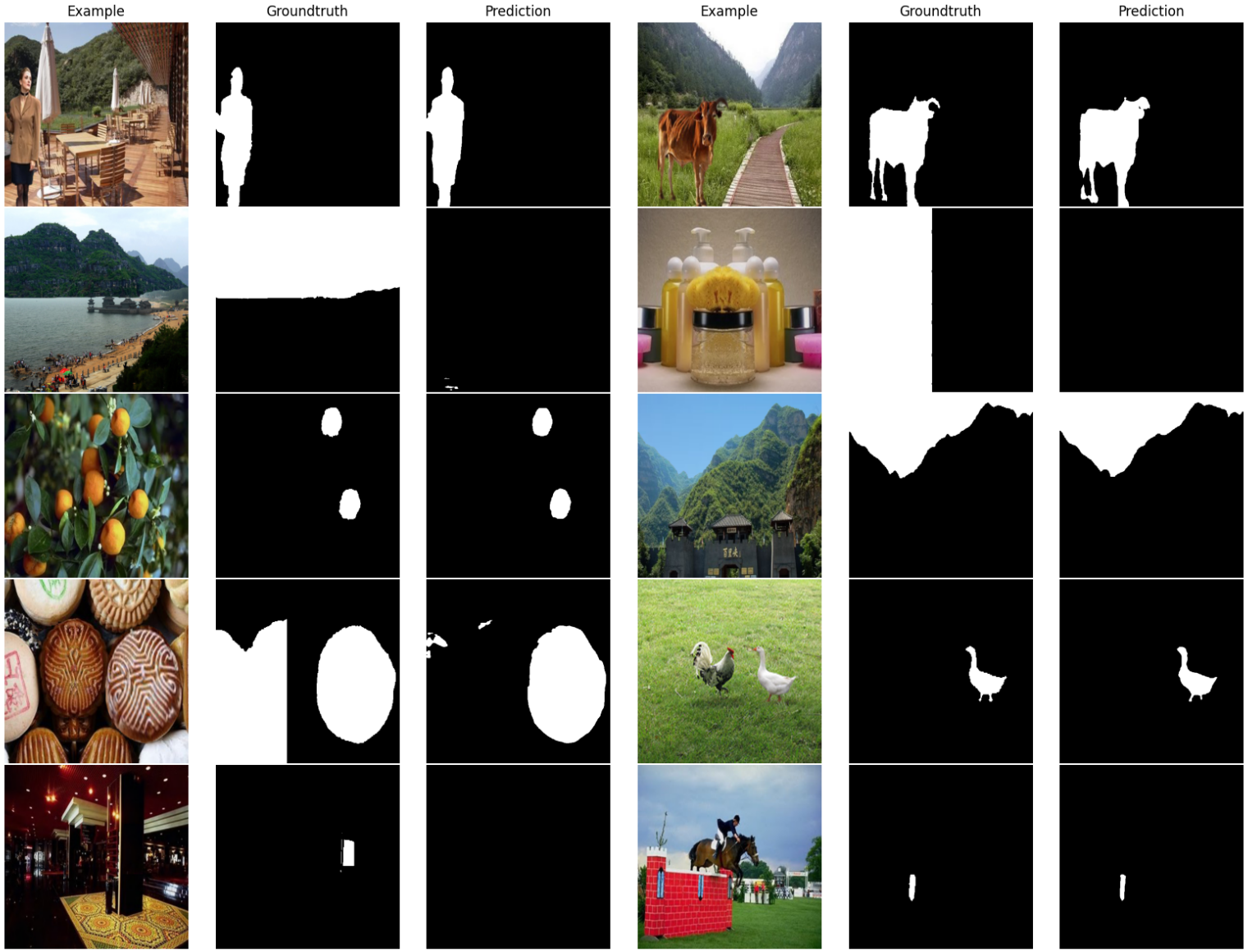
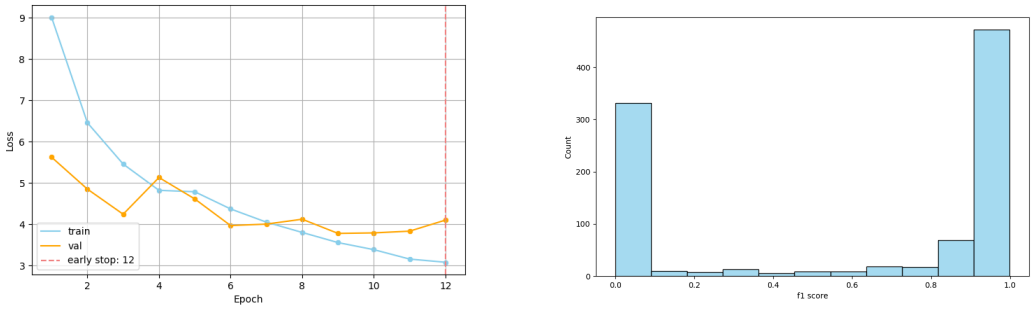


Figure 4.5: Example images from the CASIA v2 test set (left), groundtruth masks (middle), and the mask predictions from MaskFormer Swin-S-ADE20K (right).



(a) (b)

Figure 4.6: Learning curves on training and validation set for the fine-tuning of the Mask2Former Swin-S-ADE20K on CASIA v2 (a). Distribution of f1 scores for the predictions of the model on the tampered images from the test set (b).

4.1.2 Mask2Former

The Mask2Former, as described in section 3.3.2, is an optimized version of the MaskFormer. The experiments in this section were designed to test if it also performs better on the task of Image Forgery Detection. As the model is very similar to the MaskFormer I only considered the Mask2Former Swin-S-ADE20K. Moreover, the model has high GPU memory usage making it impossible to train models larger than size small on the machine used in this work.

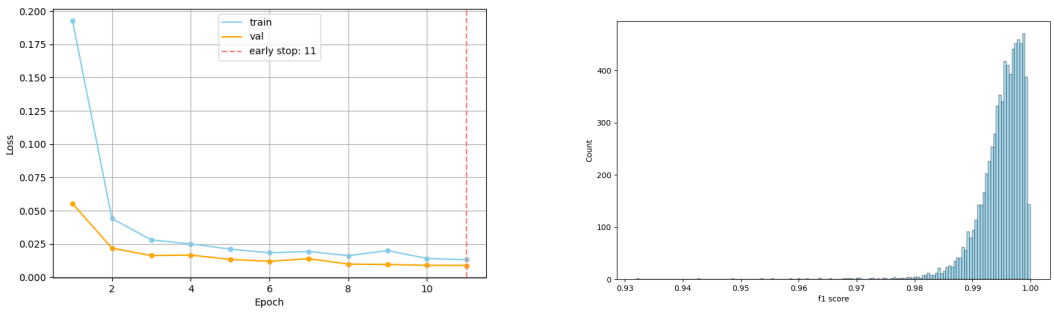
In the same way as before, I fined-tuned the model on the whole CASIA v2 dataset, including authentic images. The learning curves are shown in figure 4.6a and the mean f1 score is reported in table 4.1. The model performs better than the MaskFormer with a mean f1 score of 0.583. But, the share of 0 f1 score predicitons is 32 %, two percentage points higher than for the MaskFormer. The higher average f1 score comes from a more polarised distribution of scores. As shown in figure 4.6b, the scores are even more distributed to the edges, i.e. either 1 or 0 rather than in between. The median f1 score is 0.901. This is 0.112 points more than for the MaskFormer. In addition to its good performance on tampered images, the model also has high precision. The share of authentic images in the test set where the model predicts a mask is only 5 %.

4.2 AI-Inpainting Detection

The following experiments aim to assess the capabilities of modern large image segmentation models to discriminate between real and AI-generated pixels in an image. This section describes

Model	F_1	Details
SPAN [67]	0.382	CNN with Bayer and SRM layers
RGB-N [29]	0.408	R-CNN with RGB and Noise features
MFCN [68]	0.541	Multi-task CNN detecting surface and edge mask
TransForensics [31]	**0.627**	CNN with attention
MaskFormer	0.561	Swin-S-ADE20K
MaskFormer (Tp)	0.607	Swin-S-ADE20K trained on tampered images only
Mask2Former	0.583	Swin-S-ADE20K

Table 4.1: Performance of different models on the CASIA v2 dataset.



(a) (b)

Figure 4.7: Learning curves on training and validation set for the fine-tuning of the MaskFormer Swin-S-ADE20K on Places205+MAT (a). Distribution of f1 scores for the predictions of the model on the tampered images from the test set (b).

the fine-tuning of the MaskFormer model on the Places205+MAT dataset and the evaluation of the model performance on various data sources.

Places205+MAT

I trained the MaskFormer Swin-S-ADE20K on the Places205+MAT dataset, including both inpainted and authentic images. The learning curves are shown in figure 4.7a and the distribution of f1 scores on the test set in figure 4.7b. The results are extraordinary with a mean f1 score on the test set of 0.995 and a minimum f1 score of 0.932. Some examples are shown in figure 4.8. As indicated by the f1 score, the predicted masks match the groundtruth masks almost exactly. Moreover, the performance on the authentic test images reveals that the model is very precise. The mask prediction density on the authentic images is exactly zero. The model did not predict any fake pixels in any of the authentic images.

Figure 4.8: Example images (left), groundtruth masks (middle) and the mask predictions from MaskFormer Swin-S-ADE20K (right) for tampered images from the Places205+MAT test set.

I conducted a series of rigorous tests to ensure that the performance is measured accurately and not inflated by any side effects. These are described in the following.

MAT

As a first comparison, I trained the same model only on the images reconstructed by MAT. Learning curves and f1 distribution are shown in figure A.6. The performance on the reconstructed images is similar to before with a mean f1 score of again 0.995. The performance on the authentic images on the other hand now dropped significantly. The model predicted a non-zero mask in $55\,\%$ of examples. These masks seem to not pick up on any clearly distinguishable objects in the scene but rather form a porous cover of parts of the image (figure A.7).

Stable Diffusion

To test if the first model can generalize to other inpainting sources, I tested it's performance on the 100 images from the stable diffusion dataset (section 3.2.3). The model achieves a mean f1 score of 0.367. More importantly, the average prediction density of the model on the stable diffusion images is $99.9\,\%$. In other words, the model predicts nearly every pixel to be fake for every image. Examples are shown in figure A.8. I also tested the model on the original images from Places205 from which the stable diffusion data was created. These images were not part of neither train, validation or test set of the previous experiments. The result is the same as before, the model predicts every pixel as real on the authentic images without exception.

CASIA

Next, I tested the dependency of the model on the main data source Places205. The model was applied to both parts of the CASIA test set, i.e. authentic and tampered images. For the authentic images the model predicted each pixel in each image to be authentic without mistake. For the tampered images the results are nearly the same, except for one single image where the model predicts $99.6\,\%$ of the pixels to be fake. For every other image the result is the same, no fake pixels were detected.

Frequency Analysis

As a final test, I analyzed the spectrum of the MAT images. As shown in [7], GANs can leave frequency artefacts in the generated images. To analyze if the MaskFormer picks up on artefacts

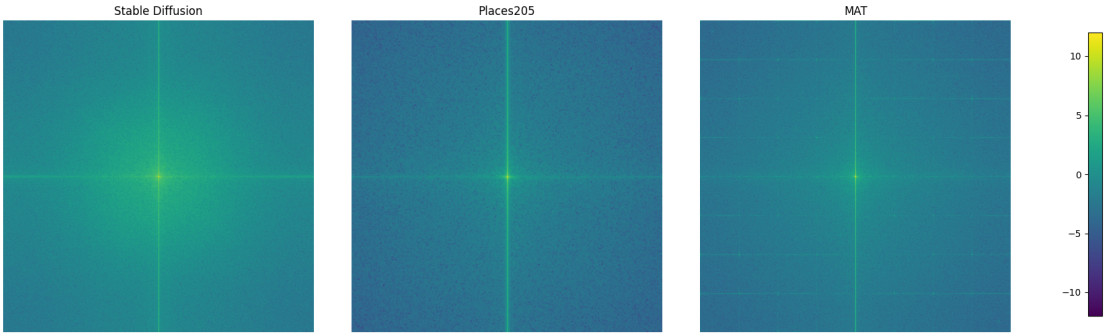


Figure 4.9: Mean shifted DCT for the training images in Stable Diffusion (left), Places205 (middle), and MAT (right). Displayed are the DCT coefficients for the different frequencies as described in section A.1. High values refer to a high abundance of the respective frequency and vice versa. The two datasets that were altered by a generative model (left and right) show characteristic frequency artefacts compared to the authentic images (center).

of the MAT model, I replicated this analysis here. I calculated the DCT for each image (see section A.1 for details). Then, I took the mean for each of the five sets of images, i.e. CASIA (tampered), CASIA (authentic), Places, MAT and Stable Diffusion. Further, I shift the spectrum so that low frequencies are in the centre of the spectral image. The spectra for MAT, Places205 and Stable Diffusion are shown in figure 4.9. Both models, Stable Diffusion and MAT, altered the original spectrum of the Places205 dataset. Within the spectrum of Places205, frequencies are distributed randomly. For Stable Diffusion, the increased intensity around the centre indicates a higher abundance of low frequencies. Following this, the model creates less detailed features which are associated with high frequencies. The MAT images, on the other hand, show another trend. Low frequencies are not overrepresented, but the spectrum contains discrete spectral lines. This phenomenon was discovered before by Zhang et al. [7] and is shown in figure 2.6. These three spectra provide a good explanation for the model performance. The MAT model leaves a clear frequency signature in the images which is not detectable by the human eye, but can be picked up by a model. The images by Stable Diffusion on the other hand do not match the distribution of the Places205 images and are therefore also not detected as authentic. But, as they have another signature as the MAT images, the model cannot detect the altered regions. Lastly, the spectra for both tampered and authentic images from the CASIA training set are shown in figure A.9. Both spectra look similar to the one of Places205. Thus, it is not surprising that they were also classified as authentic by the model.

Comparison with NIX-Net

There does not exist an exact reference for AI-inpainting detection on images created from Places205 using the MAT inpainting model. But, as described in section 2.6, Li et al. [47] tested their NIX-Net on images from Places205 that were inpainted using different models. The best mean IoU achieved on their test set was 0.921. The MaskFormer scored 0.069 points higher with a mean IoU of 0.990 without specialized noise features. But, in contrast to the NIX-Net, it was not able to generalize to other inpainting techniques.

5 Conclusion

In this work, I approached the detection of image manipulation with classical manipulation tools like Adobe Photoshop as an image segmentation problem. For this task, I fine-tuned the large image segmentation models MaskFormer and Mask2Former to classify the pixels of an image into two categories, real and fake. I evaluated the results, calculating the f1 score for the predicted masks on tampered images and the share of correct zero-mask predictions on authentic images. Furthermore, I compared the f1 scores of the different models with the current state of the art (table 4.1).

The best model, MaskFormer Swin-S-ADE20K trained on tampered images only, achieved an f1 score on the test set of 0.607 which is only 0.02 points behind the best model performance found in the literature and better than any other model. In contrast to previous research, I extended the analysis of forgery detection and considered the model performance on authentic images as an evaluation criterion. Therefore, the rate of true negative predictions, i.e. the model predicting all pixels as real within a real image, was used to asses the models image-level precision. The best performance, considering both metrics, was achieved by the Mask2Former Swin-S-ADE20K, which achieved an image-level f1 score of 0.583 on the test set and correctly classified $95\,\%$ of the authentic images.

In a final step, I assessed the problem of data leakage when training on the CASIA dataset. It was motivated by the fact, that many tampered images were created from the same authentic images. When simply splitting the data in train and test set, many of the test images were partially seen during training (section 3.2.1). This has not been addressed in any of the reviewed

works using this dataset. To assess the extent of this problem, I compared the model performance on images only made of components not seen during training with partially repeated images. The model achieved a mean f1 score of 0.421 on the completely new images and 0.590 on the partially repeated ones. This inflation of f1 scores through the repeated images has to be considered when making assumptions about the model performance beyond the CASIA dataset.

With my experiments, I could show that current state-of-the-art image segmentation models can keep up with specialized architectures for Image Forgery Detection. Further, I demonstrated that these models can achieve high precision when including authentic images in the training. Finally, I assessed the problem of data leakage for the CASIA dataset allowing future research to take this problem into account.

The second set of experiments was focused on the detection of AI-inpainted images, i.e. authentic images that were altered by a generative model. I created a new dataset, combining a set of authentic images with the test data of a model for image inpainting. Similar to the previous experiments, I trained the MaskFormer Swin-S-ADE20K on the data and evaluated its performance in terms of image-level f1 score and precision. The model achieved exceptional results with a mean f1 score of 0.995 and zero false predictions on the authentic test images. I further tested the model on a set of images created by Stable Diffusion as well as the CASIA dataset. The model detected the Stable Diffusion images as manipulated but could not generate an accurate segmentation mask. Instead, it classified nearly all pixels in the images as fake. The model could not detect any fake images created with Photoshop (CASIA). Previous research has found distinct frequency artefacts in images created by generative models. Based on these results, I analyzed the spectra of the different datasets using Discrete Cosine Transform. It revealed unique frequency artefacts for both MAT and Stable Diffusion that were not present in the authentic images or the images that were manipulated with Photoshop. The different characteristics of artefacts found in MAT and Stable Diffusion images could explain why the model did not generalize to the new data.

I have shown that it is possible to identify AI-inpainted regions in otherwise authentic images with high precision using an image segmentation model. Furthermore, the model also was able to identify authentic images but struggled with the detection of inpainted regions generated by other generative models. The next steps for this research will be comparing different generative models and analyzing if one can detect models with a less distinct signature as MAT. Furthermore, this research would benefit from creating a large, diverse and publicly available AI-inpainting dataset. Such a dataset would need to contain examples from different

models with a variety of prompts and input images. It would enable further investigation of AI-inpainting detection.

To conclude, the detection of Photoshop and AI-inpainting using an image segmentation model is possible with a competitive performance compared to specialized state-of-the-art forgery detection models. Similar to other detection tasks, Image Forgery Detection is constantly competing with improving manipulation tools. Furthermore, the problem of mis- and disinformation caused by fake images is unlikely to be solved solely through detection. As outlined in the introduction, there are two factors to the success of a fake image. First, the fake must be good enough to not be detected by either the human eye or a model. Second, it has to be believable. This cuts back to the pope wearing an expensive jacket, an image that had it both. But, additional images like the pope going scuba diving, finding the holy grail or riding a large motorcycle [69] did not cause outrage, despite having the same image quality. In the end, solving the problem of manipulated images requires both effective detection models and a critical approach to new visual information.

A Appendix

A.1 DCT

DCT is similar to the Discrete Fourier Transform, but is limited to the space of real numbers. It is used to transform an image from the spatial to the frequency domain. For an image of size $N \times M$, a 2D type II DCT without normalization is defined as:

$$F(u,v) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \cos\left[\frac{\pi u}{2N}(2i+1)\right] \cos\left[\frac{\pi j}{2M}(2j+1)\right] f(i,j). \tag{A.1}$$

Here $f(i,j)$ is the intensity of pixel $[i,j]$ and $F(U,v)$ the DCT coefficient at $[u,v]$. For most images, most of the signal lies in the upper left corner (low frequencies). In this work, the spectrum is shifted, with low frequencies in the center.

A.2 Technical Specifications

The models were trained using a Vertex AI notebook from the Google Cloud Platform. The machine was a n1-highmem-8 with 8 vCPUs, 52 GB RAM and an NVIDIA Tesla P4 GPU. Training The main model, MaskFormer Swin-S-ADE20K, for 20 epochs on the CASIA training set with 7981 images takes about 26 hours.

All training runs on both MaskFormer and Mask2Former used a learning rate of $5e-5$ and

Parameter	Value
S	32
N	100
$C_{\mathcal{E}}$	256
$C_{\mathcal{F}}$	256
w_{dice}	1
w_{focal}	20
$C_{\mathcal{F}}$ (R101c)	2048
$C_{\mathcal{F}}$ (Swin-S)	768
$C_{\mathcal{F}}$ (Swin-B)	1024

Table A.1: Hyperparameters of the MaskFormer.

the Adam optimizer. The MaskFormer was trained with a batch size of 4 and Mask2Former with a batch size of 2. This was mainly selected due to memory reasons selecting the maximum batch size possible before the machine ran out of GPU memory. The final models for predictions on the test set were selected using early stopping. The training stopped if the validation loss did not improve for 3 epochs. All the code is publicly available on Github*.

A.3 Tables

A.4 Figures

*https://github.com/jelfes/ImageForgeryDetection.git

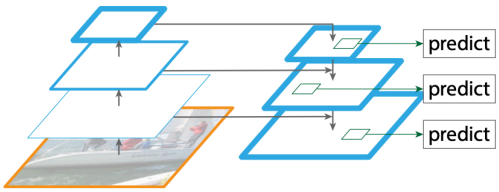


Figure A.1: Architecture of a Feature Pyramid Network [12] with backbone on the left and pixel-decoder on the right. The prediction on each level of the pyramid is omitted in the MaskFormer.

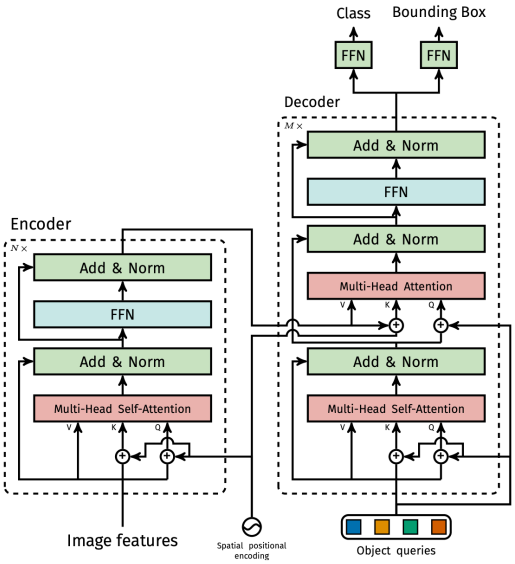
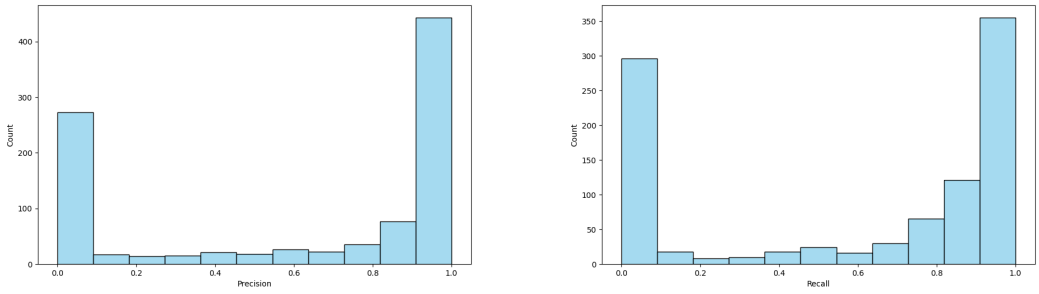


Figure A.2: Architecture of the transformer module used in DETR [13]. The MaskFormer only uses the Decoder (right side) taking as input the backbone feature map $\mathcal{F}$.



(a) (b)

Figure A.3: Distribution of precision (a) and recall (b) of the MaskFormer Swin-S-ADE20K on the CASIA test set.

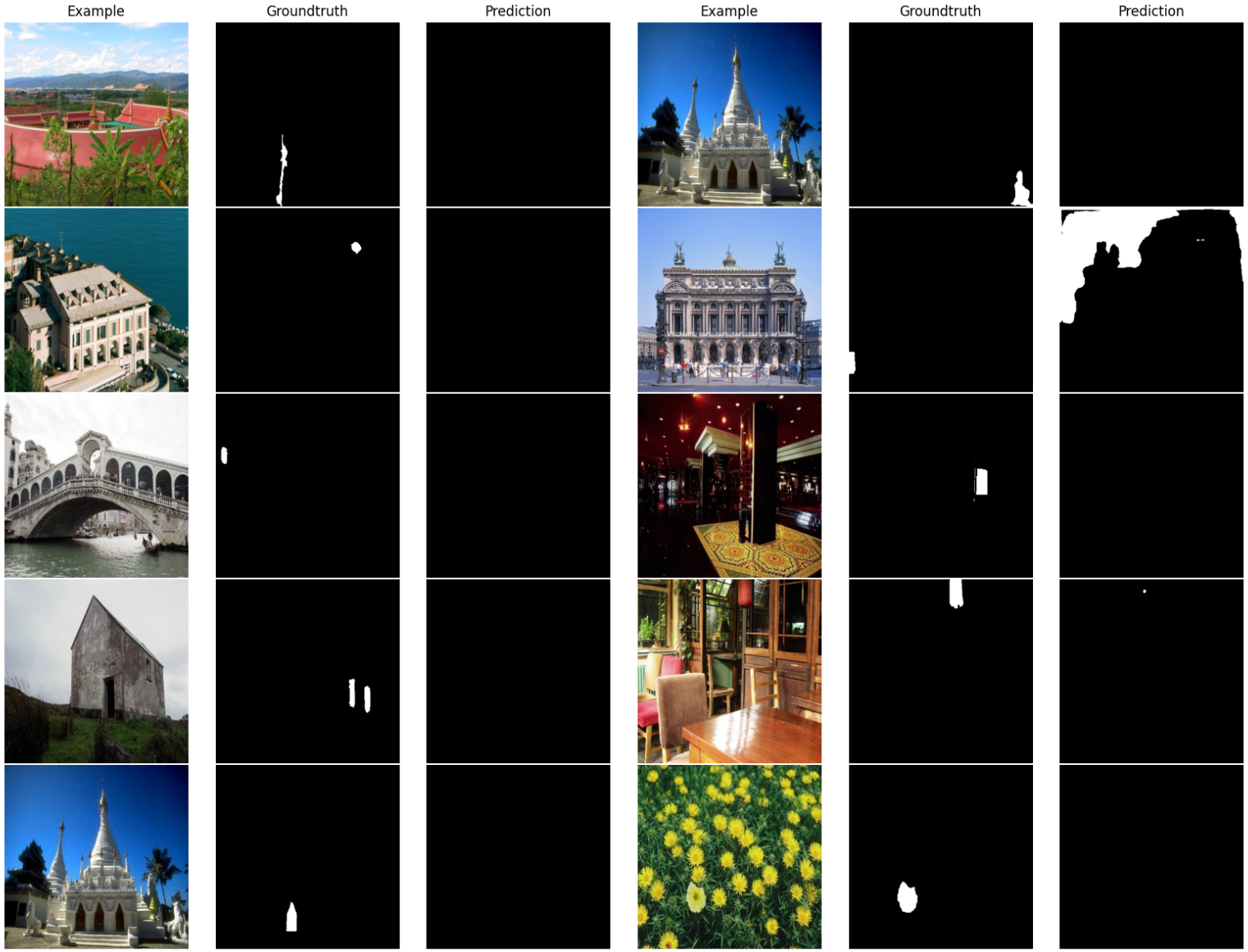


Figure A.4: Example images of bad predictions (f1 score of 0) with the tampered images (left), groundtruth mask (middle), and the mask predictions of MaskFormer Swin-S-ADE20K (right).

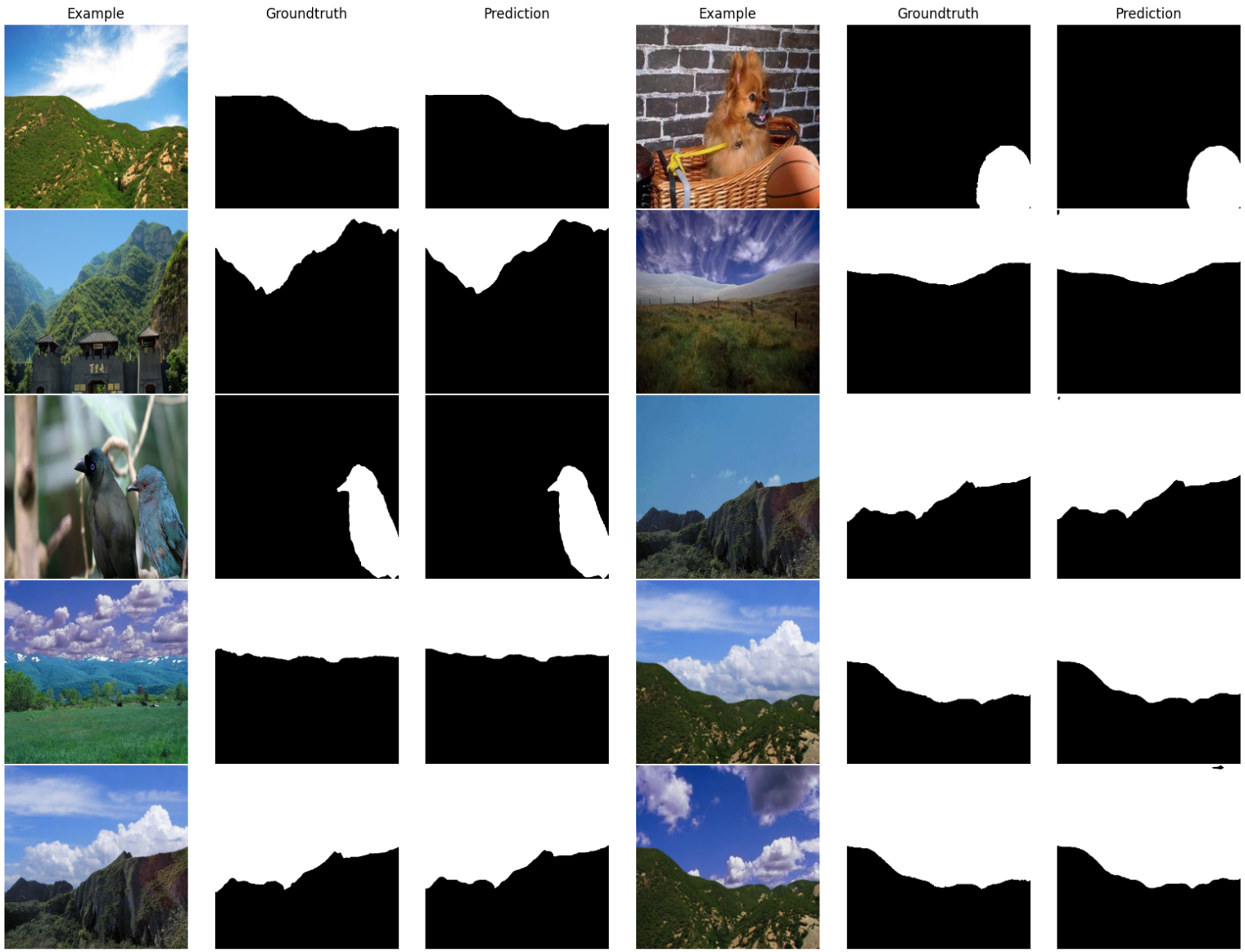
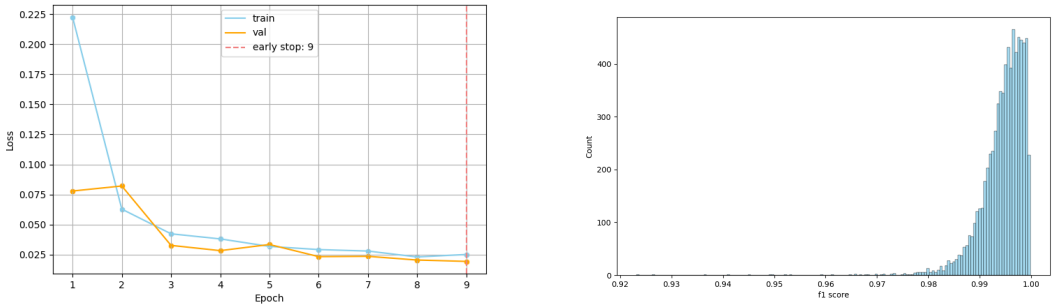


Figure A.5: Example images of good predictions (f1 score > 0.99) with the tampered images (left), groundtruth masks (middle), and the mask predictions of MaskFormer Swin-S-ADE20K (right).



(a) (b)

Figure A.6: Learning curves on training and validation set for the fine-tuning of the MaskFormer Swin-S-ADE20K on tampered images only from Places205+MAT (a). Distribution of f1 scores for the predictions of the model on the test set (b).

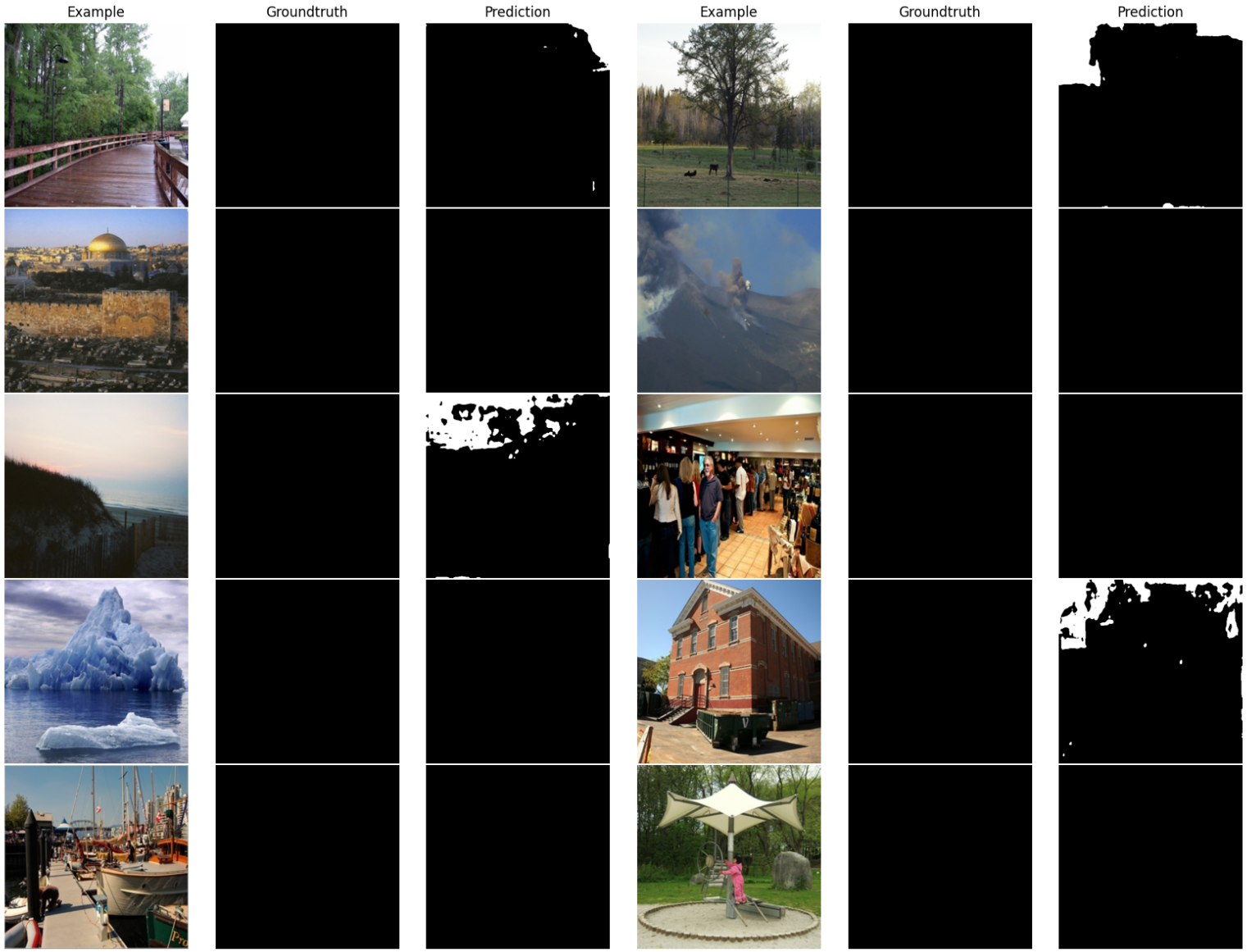


Figure A.7: Examples of authentic images from the Places205 test set (left), groundtruth masks (middle), and the mask predictions from the MaskFormer Swin-S-ADE20K that was trained on tampered images from MAT only (right).

Figure A.8: Example images (left), groundtruth masks (middle), and the mask predictions from MaskFormer Swin-S-ADE20K (right) for images created with stable diffusion.

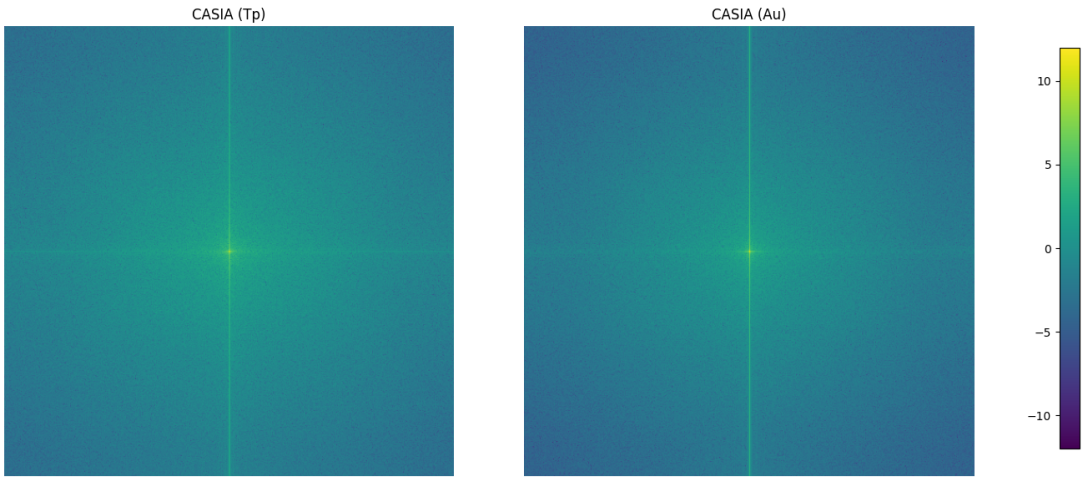


Figure A.9: Mean shifted DCT of tampered and authentic training images from the CASIA v2 dataset. Low frequency are shifted to the center. Displayed are the DCT coefficients for the different frequencies as described in section A.1. High values refer to a high abundance of the respective frequency and vice versa.

References

[1] @singareddynm. The boys in brooklyn could only hope for this level of drip. [Online]. Available: https://twitter.com/singareddynm/status/1639655045875507201

[2] I. Castillo Camacho and K. Wang, "A comprehensive review of deep-learning-based methods for image forensics," *Journal of Imaging*, vol. 7, no. 4, p. 69, 2021. [Online]. Available: http://dx.doi.org/10.3390/jimaging7040069

[3] C. Doersch, "Tutorial on variational autoencoders," 2016. [Online]. Available: https://arxiv.org/abs/1606.05908

[4] P. Pandey. Deep generative models. [Online]. Available: https://towardsdatascience.com/deep-generative-models-25ab2821afd3

[5] H. Farid. Digital image forensics. [Online]. Available: https://farid.berkeley.edu/downloads/tutorials/digitalimageforensics.pdf

[6] Cburnett. Bayer filter. [Online]. Available: https://en.wikipedia.org/wiki/Bayer_filter#/media/File:Bayer_pattern_on_sensor.svg

[7] X. Zhang, S. Karaman, and S.-F. Chang, "Detecting and simulating artifacts in gan fake images," in *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2019, pp. 1–6. [Online]. Available: http://dx.doi.org/10.1109/WIFS47025.2019.9035107

[8] N. Barla. The complete guide to panoptic segmentation. [Online]. Available: https://www.v7labs.com/blog/panoptic-segmentation-guide

[9] B. Cheng, A. G. Schwing, and A. Kirillov, "Per-pixel classification is not all you need for semantic segmentation," 2021. [Online]. Available: https://arxiv.org/abs/2107.06278

[10] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," 2021. [Online]. Available: https://arxiv.org/abs/2112.01527

[11] W. Li, Z. Lin, K. Zhou, L. Qi, Y. Wang, and J. Jia, "Mat: Mask-aware transformer for large hole image inpainting," 2022. [Online]. Available: https://arxiv.org/abs/2203.15270

[12] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. [Online]. Available: https://doi.org/10.1109/CVPR.2017.106

[13] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," 2020. [Online]. Available: https://arxiv.org/abs/2005.12872

[14] D. Holz. Midjourney. [Online]. Available: https://www.midjourney.com/home/?callbackUrl=FappF

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[16] B. J. Casad and J. E. Luebering. confirmation bias. [Online]. Available: https://www.britannica.com/science/confirmation-bias

[17] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013. [Online]. Available: https://arxiv.org/abs/1312.6114

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, p. 139–144, 2020. [Online]. Available: https://doi.org/10.1145/3422622

[19] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," 2021. [Online]. Available: https://arxiv.org/abs/2102.12092

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[21] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2021. [Online]. Available: https://arxiv.org/abs/2112.10752

[22] BBC News. Fake trump arrest photos: How to spot an ai-generated image. [Online]. Available: https://www.bbc.com/news/world-us-canada-65069316

[23] R. Agarwal, D. Khudaniya, A. Gupta, and K. Grover, "Image forgery detection and deep learning techniques: A review," in *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 2020, pp. 1096–1100. [Online]. Available: https://doi.org/10.1109/ICICCS48265.2020.9121083

[24] Z. J. Barad and M. M. Goswami, "Image forgery detection using deep learning: A survey," in *2020 6th international conference on advanced computing and communication systems (ICACCS)*. IEEE, 2020, pp. 571–576. [Online]. Available: https://doi.org/10.1109/ICACCS48705.2020.9074408

[25] B. Bayar and M. C. Stamm, "A deep learning approach to universal image manipulation detection using a new convolutional layer," in *Proceedings of the 4th ACM workshop on information hiding and multimedia security*, 2016, pp. 5–10. [Online]. Available: https://doi.org/10.1145/2909827.2930786

[26] J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images," *IEEE Transactions on information Forensics and Security*, vol. 7, no. 3, pp. 868–882, 2012. [Online]. Available: http://dx.doi.org/10.1109/TIFS.2012.2190402

[27] D. Cozzolino and L. Verdoliva, "Single-image splicing localization through autoencoder-based anomaly detection," in *2016 IEEE international workshop on information forensics and security (WIFS)*. IEEE, 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/WIFS.2016.7823921

[28] J. H. Bappy, C. Simons, L. Nataraj, B. S. Manjunath, and A. K. Roy-Chowdhury, "Hybrid LSTM and encoder-decoder architecture for detection of image forgeries," 2019. [Online]. Available: http://arxiv.org/abs/1903.02495

[29] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis, "Learning rich features for image manipulation detection," in *Proceedings of the IEEE conference on computer*

vision and pattern recognition, 2018, pp. 1053–1061. [Online]. Available: https://doi.org/10.1109/CVPR.2018.00116

[30] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[31] J. Hao, Z. Zhang, S. Yang, D. Xie, and S. Pu, "Transforensics: image forgery localization with dense self-attention," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 055–15 064. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.01478

[32] T. Jung, S. Kim, and K. Kim, "Deepvision: Deepfakes detection using human eye blinking pattern," *IEEE Access*, vol. 8, pp. 83 144–83 154, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.2988660

[33] S. Agarwal, H. Farid, O. Fried, and M. Agrawala, "Detecting deep-fake videos from phoneme-viseme mismatches," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020, pp. 2814–2822. [Online]. Available: https://doi.org/10.1109/CVPRW50498.2020.00338

[34] D. Bau, J.-Y. Zhu, J. Wulff, W. Peebles, H. Strobelt, B. Zhou, and A. Torralba, "Seeing what a gan cannot generate," 2019. [Online]. Available: https://arxiv.org/abs/1910.11626

[35] H. Mo, B. Chen, and W. Luo, "Fake faces identification via convolutional neural network," in *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*, ser. IHamp;MMSec '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 43–47. [Online]. Available: https://doi.org/10.1145/3206004.3206009

[36] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard

[37] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "Cnn-generated images are surprisingly easy to spot... for now," 2020. [Online]. Available: https://arxiv.org/abs/1912.11035

[38] M. Goebel, L. Nataraj, T. Nanjundaswamy, T. M. Mohammed, S. Chandrasekaran, and B. S. Manjunath, "Detection, attribution and localization of gan generated images," 2020. [Online]. Available: https://arxiv.org/abs/2007.10466

[39] Z. Sha, Z. Li, N. Yu, and Y. Zhang, "De-fake: Detection and attribution of fake images generated by text-to-image generation models," 2023. [Online]. Available: https://arxiv.org/abs/2210.06998

[40] J. Li, D. Li, C. Xiong, and S. Hoi, "BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *Proceedings of the 39th International Conference on Machine Learning*, vol. 162, 2022, pp. 12 888–12 900. [Online]. Available: https://proceedings.mlr.press/v162/li22n.html

[41] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning*, vol. 139, 2021, pp. 8748–8763. [Online]. Available: https://proceedings.mlr.press/v139/radford21a.html

[42] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," 2014. [Online]. Available: https://arxiv.org/abs/1405.0312

[43] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 67–78, 2014. [Online]. Available: https://doi.org/10.1162/tacl_a_00166

[44] M. Bertalmio, A. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*, vol. 1, 2001, pp. I–I. [Online]. Available: https://doi.org/10.1109/CVPR.2001.990497

[45] X. Wang, S. Niu, and H. Wang, "Image inpainting detection based on multi-task deep learning network," *IETE Technical Review*, vol. 38, no. 1, pp. 149–157, 2021. [Online]. Available: https://doi.org/10.1080/02564602.2020.1782274

[46] Z. Guo, L. Zhang, and D. Zhang, "A completed modeling of local binary pattern operator for texture classification," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1657–1663, 2010. [Online]. Available: https://doi.org/10.1109/TIP.2010.2044957

[47] A. Li, Q. Ke, X. Ma, H. Weng, Z. Zong, F. Xue, and R. Zhang, "Noise doesn't lie: Towards universal detection of deep inpainting," 2021. [Online]. Available: https://arxiv.org/abs/2106.01532

[48] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in Neural Information Processing Systems*, vol. 27, 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/3fe94a002317b5f9259f82690aeea4cd-Paper.pdf

[49] J. Dong, W. Wang, and T. Tan, "Casia image tampering detection evaluation database," in *2013 IEEE China summit and international conference on signal and information processing*. IEEE, 2013, pp. 422–426. [Online]. Available: http://dx.doi.org/10.1109/ChinaSIP.2013.6625374

[50] paperswithcode.com. Semantic segmentation on ade20k. [Online]. Available: https://paperswithcode.com/sota/semantic-segmentation-on-ade20k

[51] Adobe. Adobe unveils future of creative cloud with generative ai as a creative co-pilot in photoshop. [Online]. Available: https://news.adobe.com/news/news-details/2023/Adobe-Unveils-Future-of-Creative-Cloud-with-Generative-AI-as-a-Creative-Co-Pilot-in-Photoshop-default.aspx/default.aspx

[52] B. Wen, Y. Zhu, R. Subramanian, T.-T. Ng, X. Shen, and S. Winkler, "Coverage — a novel database for copy-move forgery detection," in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 161–165. [Online]. Available: https://doi.org/10.1109/ICIP.2016.7532339

[53] T.-T. Ng and S.-F. Chang, "A data set of authentic and spliced image blocks," 2004. [Online]. Available: https://api.semanticscholar.org/CorpusID:59823140

[54] D. Goe. Casia 2.0 image tampering detection dataset. [Online]. Available: https://www.kaggle.com/datasets/divg07/casia-20-image-tampering-detection-dataset

[55] S. Mittal. Places. [Online]. Available: https://www.kaggle.com/datasets/mittalshubham/images256

[56] paperswithcode.com. Image inpainting on places2. [Online]. Available: https://paperswithcode.com/sota/image-inpainting-on-places2-1

[57] W. Li, Z. Lin, K. Zhou, L. Qi, Y. Wang, and J. Jia. MAT: Mask-aware transformer for large hole image inpainting. [Online]. Available: https://github.com/fenglinglwb/MAT

[58] W. Li. MAT data. [Online]. Available: https://mycuhk-my.sharepoint.com/:f:/g/personal/1155137927_link_cuhk_edu_hk/EuY30ziF-G5BvwziuHNFzDkBVC6KBPRg69kCeHIu-BXORA?e=7OwJyE

[59] Runwayml. Stable diffusion inpainting. [Online]. Available: https://huggingface.co/runwayml/stable-diffusion-inpainting

[60] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[61] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021. [Online]. Available: https://arxiv.org/abs/2103.14030

[62] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[63] F. Milletari, N. Navab, and S. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," 2016. [Online]. Available: http://arxiv.org/abs/1606.04797

[64] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2017. [Online]. Available: https://arxiv.org/abs/1708.02002

[65] paperswithcode.com. Image inpainting on celeba-hq. [Online]. Available: https://paperswithcode.com/sota/image-inpainting-on-celeba-hq

[66] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," 2018. [Online]. Available: https://arxiv.org/abs/1812.04948

[67] X. Hu, Z. Zhang, Z. Jiang, S. Chaudhuri, Z. Yang, and R. Nevatia, "SPAN: spatial pyramid attention network forimage manipulation localization," 2020. [Online]. Available: https://arxiv.org/abs/2009.00726

[68] R. Salloum, Y. Ren, and C. J. Kuo, "Image splicing localization using A multi-task fully convolutional network (MFCN)," 2017. [Online]. Available: http://arxiv.org/abs/1709.02016

[69] roaming-buffalo. Pope francis's day out (14 ai-generated images collected from around reddit). [Online]. Available: https://www.reddit.com/r/CatholicMemes/comments/12doq7r/pope_franciss_day_out_14_aigenerated_images/

Acknowledgments

I want to thank everybody that supported me with this work. In particular, I want to thank my supervisor Dr. Wolfgang Erb and my team at Wayfair for their help and feedback. A special thank you also to my friend Jasper Anders for reviewing my work and always holding me up to high standards. Further, I want to thank my fellow Data Science students for making this an exceptional time. I learned so much from you all. Last but not least, a big thank you to all the people in my life that do not care about Data Science but still showed interest in my work and supported me on this journey. Sincerely, thank you all!