NCSR DEMOKRITOS
UNIVERSITY OF PIRAEUS

MSc in Artificial Intelligence

Classification & Localization of Inpainted Regions

Author :
Andreas Sideras

Supervisor:
Theodoros Giannakopoulos

June 23, 2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
UNIVERSITY OF PIRAEUS

Abstract

Generative models are AI models that leverage deep learning techniques to generate unique and lifelike content based on provided patterns and examples, often making it challenging to distinguish between real and artificial content. This has given rise to the urgent need for identifying artificial content in the age of generative models and deep fakes, as it can be exploited for malicious purposes such as the dissemination of fake news. Extensive research has been conducted in recent years to develop methods for recognizing whether specific content is generated by an AI system. Thus, the objective of this exercise is to delve into the detection of artificial parts within an image. We trained a model able to decide whether an image contains an artificial part. If so, it tries to localize it using a bounding box.Throughout our experimentation, we explored various renowned pretrained models, while leveraging a dataset that encompasses a broad range of content.

Contents

1 Introduction

Generative models have gained increasing popularity due to their wide-ranging applications, from generating art for entertainment purposes to the potential risk of spreading fake news. Extensive research has been conducted in recent years to address the detection of artificial content, as evidenced by Mirsky's and Lee's work on defense solutions in this domain [1]. In this exercise, our objective is to train a deep learning model capable of determining the presence of artificial part within an image and accurately detecting it if it exists. This task encompasses two prominent machine learning scenarios in computer vision: classification and localization. Localization determines the precise spatial coordinates of each object, and classification assigns a corresponding class label to each detected object. Models designed for such tasks have experienced significant popularity, especially since the advent of neural networks' application in computer vision tasks around 2012.

Let's attempt to provide a definition for what constitutes an artificial image. An artificial image can be defined as a visual representation that has been created or manipulated using digital tools or techniques, rather than being captured directly from the physical world. It may involve the use of generative models, image editing software, or computer-generated imagery (CGI). In our case, our focus will be on images that may or may not include a certain artificial part, rather than being entirely artificial. We are interested in examining and detecting these artificial components within the images, which could be manipulated regions to subtle alterations that may be challenging to distinguish from real content. These manipulations have been performed from a *Generative Adversarial Network*. One application of GANs is *image inpainting*, where GANs can be trained to fill in missing or damaged parts of an image. By training the generator to generate realistic and coherent pixels that match the surrounding context, GANs can effectively restore missing regions in images. We use image inpainting in order to create our dataset.

An inpainted image is an image that contains an artificial component. Our goal is for our model to determine whether an image has been inpainted (binary classification), and if so, to detect the artificially filled part using a bounding box (localization). Inpainted images are often challenging to distinguish from unedited images with the naked eye, especially when the inpainted region is relatively small or consists of a simple background, such as the sea or a wall, like figure [1a]. The most difficult scenarios occur when the inpainted region covers a significant portion of an object or when it is situated between two overlapping or touching objects like the people playing a sport in figure [1b]. In such cases, the GAN model must combine the semantics of the remaining image and prior knowledge to realistically recreate these objects. This task is inherently difficult, even for a human. While there are cases where the algorithm may struggle and produce blurry outputs, it often yields highly satisfactory results. Check how the inpainted part of figure [1c] maintains the original texture, structure and colors. Thus, our model will try to leverage the learned feature patterns to identify subtle or pronounced blurriness, inconsistencies in objects, and other artifacts that often generated by GANs and are typically imperceptible to the naked eye.

Although there are numerous datasets available for artificially generated images, we were unable to find one that specifically emphasizes inpainted regions. Consequently, we employed a pretrained GAN model and constructed our own dataset. Subsequently, we finetuned several widely recognized computer vision models that have consistently demonstrated impressive performance across various tasks. Through an experimental analysis, we compared the results of these models in both classification and localization performances, enabling us to identify the most optimal model. The most of experiments were run on the Google Colab platform, using the A100 Nvidia GPU.

(a) (b) (c)

Fig. 1: Inpainted Images with a bounding box denoting the artificial part

2 Image Inpainting

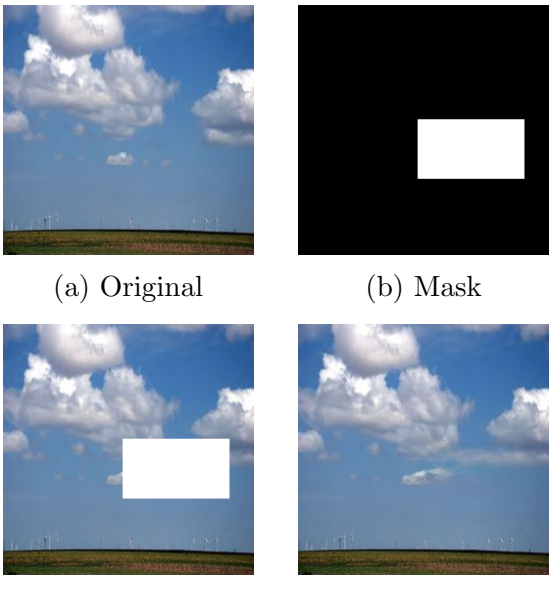
In this section, we will discuss image inpainting and the creation of our dataset. Regarding the dataset, we aimed to utilize one that exhibits a diverse range of content. The MIT Places365 dataset [2] serves this purpose perfectly, as it is a large-scale image dataset comprising millions of labeled 256x256 images representing various scenes and locations worldwide (the exact number may vary depending on the version). Widely employed in computer vision research for tasks like scene understanding, scene recognition, and image classification, this dataset offers a valuable resource for training and evaluating models in the realm of visual recognition and understanding. Due to restrictions in time and computational resources, we focused solely on the validation set of the version we choose that sums up to 36500 images. We further split it into training, validation, and test sets, with sizes of 25000, 5000, and 6500 images, respectively.

Let's now illustrate how we created the inpainted images. As we mentioned in the introduction, image inpainting is the process of filling in missing or corrupted parts of an image with plausible content, preserving the visual consistency and structure of the original image. GANs (Generative Adversarial Networks) can be used for image inpainting by training a generator network to generate visually coherent and realistic content to complete the missing regions, while a discriminator network provides feedback to ensure the generated content aligns well with the surrounding image and maintains a high level of visual quality. GANs have shown promising results in generating accurate and visually appealing inpainted images, making them a popular approach for tackling the image inpainting task.

Free-Form Image Inpainting with Gated Convolution [3] is a research paper that introduces a novel approach to the task of image inpainting, which involves filling in missing or corrupted regions in an image. The proposed method utilizes gated convolutions, a type of convolutional operation that selectively updates the feature maps based on the presence of missing regions. The paper begins by addressing the limitations of existing inpainting methods that struggle with generating realistic and coherent content for complex inpainting scenarios. It highlights the challenge of preserving semantic structures and texture details while filling in large and irregularly shaped holes.To tackle these issues, the authors propose a gated convolutional network architecture for free-form image inpainting. This architecture consists of an encoder-decoder framework with skip connections, where the gated convolutions play a crucial role in selectively updating the feature maps. The gating mechanism helps the model focus on relevant information while ignoring the missing regions,

thus improving the inpainting quality. The authors also introduce a new loss function, which combines both contextual and perceptual losses. The contextual loss encourages the completion to be consistent with the surrounding image content, while the perceptual loss ensures that the completed regions align with high-level visual features. To evaluate the proposed method, extensive experiments are conducted on benchmark datasets, and the results are compared with state-of-the-art inpainting techniques. The experimental results demonstrate that the approach achieves superior performance in generating visually pleasing and semantically coherent inpainted images. Furthermore, the paper provides a comprehensive ablation study, analyzing the effectiveness of different components of the proposed architecture.

We utilized a PyTorch re-implementation of the original code, provided by Qiang Wen in the repository located at `https://github.com/csqiangwen/DeepFillv2_Pytorch`. In order the the GAN inpainted model to run, we required masks that identify the missing regions to be inpainted. These masks are individual images of size 1x256x256, where a white rectangle denotes the region to be removed. As we had images without any missing parts, we developed a Python script to generate such images. These generated images consist of black backgrounds with a randomly placed white rectangle, indicating the area to be inpainted. The random variables determining the size and position of the rectangle follow a uniform distribution. We ran the script for all available half images (18250) and stored the corresponding ground truth values in a CSV file. The ground truths represent the coordinates of the inpainted regions, specifically the top left and bottom right x and y coordinates of the mask box. Additionally, we included a binary value in the CSV file to indicate the presence or absence of an inpainted region. Notably, we only inpainted half of the images, considering the other half as images without artificial parts. For these images, the 5-dimensional target vector consists of zeros. Finally, we shuffled the CSV file, which includes columns for the image ID, the binary value indicating the presence of an artificial region, and the xmin, ymin, xmax, and ymax coordinates. Subsequently, we cropped the corresponding part from the original image [2a] using the mask [2b], and then we fed the inpainting model with the masked image [2c] and the mask, which generated a new image with the cropped region inpainted [2d].



(a) Original (b) Mask

(c) Masked (d) Inpainted image

Fig. 2: Dataset creation

So, now we have dataset comprising of both inpainted and orginal images along with their corresponding ground truth labels. For this dataset we would like to train a model that is able to perform a binary classification on the first dimension value of the ground truth vector. If this is value is one, the model should perform regression on the 4-dimensional vector that contains the coordinates of the artificial inpainted region.

3 Inpainted Region Detection

In this section, we will describe the steps we followed to train a model that performs well on our task. Our task consists of two components: classification and localization. Our goal is to develop a model that outputs a 5-dimensional vector, representing our ground truth labels. The loss function should align with this defined representation. The loss function comprises two main components: the localization loss and the classification loss. The localization loss measures the difference between the predicted bounding box coordinates and the ground truth bounding box coordinates. The classification loss assesses the accuracy of the model's predictions regarding the presence of an artificial part in the image. To calculate the overall loss, the localization loss and the classification loss are typically combined using a weighted sum.

$$\text{Total Loss} = \alpha \cdot \text{Classification Loss} + \beta \cdot \text{Localization Loss}$$

In the equation for the total loss in object detection, α and β are weights that determine the relative importance of the classification loss and localization loss, respectively. These coefficients play a crucial role in balancing the contributions of the two components of the loss function, given that the two losses could have significant difference on their magnitudes. The classification loss component measures the accuracy of the predicted object labels. The coefficient α determines the relative importance of the classification loss compared to the localization loss. By adjusting the value of α, you can control the emphasis placed on accurately classifying objects. For example, if classification accuracy is considered more important than precise localization, a higher value of α can be used. Conversely, if localization accuracy is prioritized, a lower value of α can be set. The same apply to the localization loss and the β value. Both these weights are hyperparameters and tuning them allows you to customize the loss function to better align with the objectives and challenges of your particular object detection scenario. Hence, the output layer of our network consists of one unit with a sigmoid activation function, which aims to optimize the first component of the total loss. Additionally, the remaining four units are performing linear operations, striving to optimize the second component of the total loss.

By optimizing the above total loss, we hope to come up with a model that accurately detects an inpainted region. The evaluation of this system on the binary classification task is pretty standard. The classical Precision, Recall, F1 score and ROC curves can be used. But we would like also to quantify how well our model locates also the artificial region. This is simply a comparison of the ground truth rectangle coordinates and the predicted model's coordinates. IoU, short for Intersection over Union [3], is a metric commonly used in object detection and segmentation tasks. It measures the overlap between a predicted bounding box or region and the ground truth bounding box or region. IoU is calculated by dividing the area of intersection between the predicted and ground truth regions by the area of their union. The resulting value ranges from 0 to 1, where 1 indicates a perfect match and 0 indicates no overlap. In general, in computer vision community an IoU greater than 0.55 for a particular example, is considered as a successful prediction. While

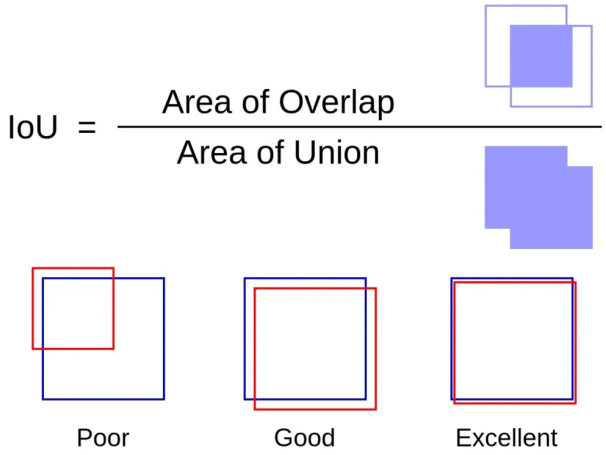


Fig. 3: Intersection Over Union metric

IoU is an effective metric for evaluating the accuracy of object detection and segmentation, it is not typically used directly as a localization loss for training object detection models. When two bounding boxes are non-overlapping, their IoU will be 0. In such cases, the IoU metric fails to provide meaningful gradients for training a model. This is because the IoU metric does not capture the spatial information or the localization accuracy of the predicted bounding boxes.

For the classification loss we will use a standard Binary Cross Entropy Loss. It measures the dissimilarity between the predicted probabilities and the ground truth labels.

$$\text{Binary Cross-Entropy Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right)$$

We have quite a few options for the localization loss. The most obvious option for such regression tasks is the Mean Squared Error. Mean Squared Error (MSE) is a commonly used metric to measure the average squared difference between the predicted and actual values in regression analysis. It provides a way to quantify the accuracy or quality of a regression model. To calculate the MSE, you take the predicted values from the model and the corresponding actual values, then compute the squared difference for each pair. Finally, you take the average of all the squared differences.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

In our case, y is 4-dimensional so the above equation is computed for these 4 dimensions and then averaged over the batch. While MSE loss is a very popular option for regression tasks, isn't the best option in the context of localization. MSE loss treats each pixel or coordinate independently and does not explicitly consider the spatial relationships between different parts of an object. Localization and object detection tasks require understanding the spatial structure of objects and accurately predicting their positions, which MSE loss fails to capture. MSE loss treats errors uniformly across all dimensions, which can lead to scale variance issues. Scale variance refers to the situation where the scale (size) of objects in an image varies significantly. In the context of localization, scale variance means that objects of interest can have different sizes, ranging from

small to large. For example, in an object detection task, you may have to detect objects of various scales, such as small pedestrians and large vehicles. When using a loss function that does not account for scale variance, such as MSE loss, the model's performance may suffer (note that IoU has that property.). MSE loss treats errors uniformly across all dimensions, including the width and height of bounding boxes. This means that errors in larger objects, which have larger bounding box dimensions, will have a proportionally larger impact on the loss compared to errors in smaller objects. As a result, the model may prioritize larger objects over smaller ones, leading to imbalanced predictions and reduced accuracy for objects of different scales. Check in figure [4] how a model trained with MSE loss correctly finds the artificial part, but assigns a relative bigger bounding box, resulting to a smaller IoU. After carefully examining numerous test results, we have verified the occurrence of this phenomenon.

Fig. 4: MSE Loss side effect

Smooth L1 loss is a loss function that addresses the limitations of MSE loss and is commonly used in object detection and localization tasks. It provides a balance between L1 loss (absolute difference) and L2 loss (squared difference) by using a piecewise function.

$$\text{smooth L1}_\beta(\hat{y}, y) = \begin{cases} 0.5 \cdot (\hat{y} - y)^2 & \text{if } |\hat{y} - y| < \beta \\ |\hat{y} - y| - 0.5 \cdot \beta^2 & \text{otherwise} \end{cases}$$

Smooth L1 loss includes an optional hyperparameter called "beta" that controls the point of transition between the quadratic and linear behavior of the loss function. Smooth L1 loss is less sensitive to outliers compared to MSE loss. It reduces the impact of large errors by using an L1-like loss

for small errors, which helps to handle noisy or inaccurate bounding box annotations. Smooth L1 loss handles errors in a scale-invariant manner. It avoids the scale variance issues associated with MSE loss by providing a more balanced update to the model parameters for errors in different scales. This enables the model to handle objects of different sizes equally well during training and prediction.

The general problem with these l1 and l2 based losses is that both are not scale invariant. Therefore, bounding boxes with the same level of overlap, but different scales will give different values.

In the paper *Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression* [4] a new option is proposed than encapsulates the benefits of IoU. As we said earlier IoU cannot be used directly as the objective function, so we use other approximate functions as we demonstrated earlier. The problem is that there is not a strong correlation between minimizing the commonly used losses (e.g. L2 norm) and improving IoU. Consider the following figure that we took from the paper.

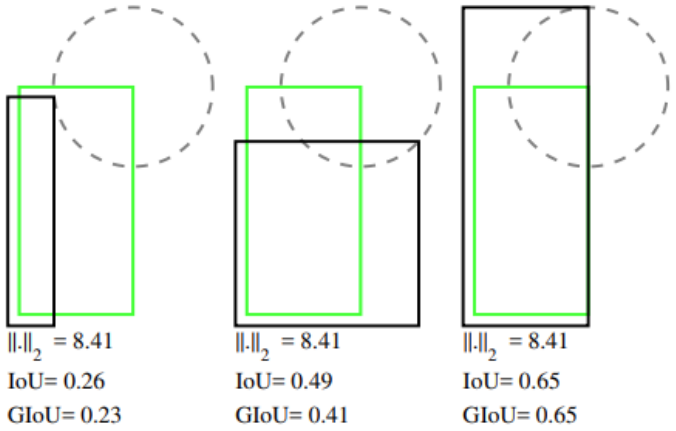


Fig. 5: L2 norm and IoU

The green bounding box is the ground truth and the black is the model's prediction. Let's assume that we keep the bottom left point fixed and mark with the grey circle all the L2 norms between the top right corners (of green and black boxes) of size 8.41. Now, if we sample just 3 of them, we can see that for the same L2 norms, we get bounding boxes with significant differences in the overlapping area (IoU values). The authors proposed a generalized version of IoU, as a new mertic for comparing two arbitary shapes.

$$\text{GIoU} = \text{IoU} - \frac{|\text{C}/(A \bigcup B)|}{|C|}$$

where C is the smallest enclosing convex object, the smallest object that contains both A and B. In the figure below we can see an illustration of the second term of GIoU.

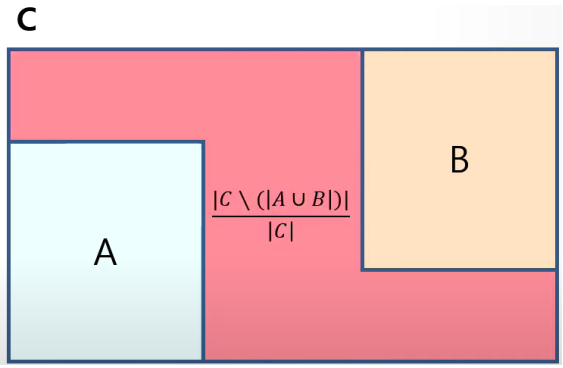


Fig. 6: GIoU second term

GIoU can be used directly as a loss. In non-overlapping cases, IoU value is always zero regardless of boxes' distance and IoU has zero gradient. In contrast, GIoU will have a negative value in non-overlapping cases. So, maximizing GIoU implies maximizing IoU when boxes are overlapped and minimizing distance between boxes in non-overlapping cases. Due to these properties, GIoU has a gradient in all possible cases and we can use it as an objective function.

Now, let's discuss the model architecture. In the computer vision community, a wide range of models has emerged over the past decade, demonstrating exceptional performance on various tasks. These models have been trained on extensive datasets, allowing them to learn complex representations and patterns. Frequently, these pre-trained models are utilized to tackle new challenges, aiming to benefit from the knowledge they have acquired. This approach is known as transfer learning, where we leverage the pre-existing knowledge to address our specific tasks. The pre-trained model used for transfer learning consists of two main parts: the convolutional base and the classifier. The convolutional base consists of multiple layers that extract hierarchical features from the input images, while the classifier is responsible for mapping these features to specific classes. We then change the last layer in order to be aligned with our case. For example we need the last output layer to be a layer with 5 units. Fine-tuning refers to the process of adapting the pre-trained model to a new task or dataset. On this process, we have two options. The first involves the freezing of the convolutional base and the training of only the layer(s) that we have modified. By doing so, we treat the pretrained model as just a feature extraction mechanism. The second option is to let the gradients flow back to the whole network and fully train it. In this case we use the pretrained weights as a very good starting point. The most important thing we should keep in our minds is the data and tasks that these models were trained on. They should have high similarity to our case, otherwise this prior knowledge shouldn't be useful. that's why in the pretrained models we used, we selected the versions that have been trained on ImageNet dataset, which demonstrates a big correlation with ours. For all the inputs we scale the image if needed in order to fit the networks input dimensions and then perfrom a normalization using the mean and std of the dataset they were trained on (ImageNet). Normalization is applied to ensure that the input data has zero mean and unit variance, which helps in improving the training process and convergence of the neural network.

The first network we finetuned was the VGG11. VGG11 specifically refers to the configuration of the VGG network with 11 layers, including 8 convolutional layers and 3 fully connected layers. VGG11 is a relatively deep neural network that has been widely adopted and proven successful in various tasks. It played a crucial role in the early days of computer vision with deep learning, delivering state-of-the-art results and driving breakthroughs in the field. We experimented also with

ResNet18 and ResNet50 which are the two versions of original ResNet architecture with 18 and 50 layers respectively. ResNet is composed of several residual blocks, which are the key components of ResNet. The architecture includes convolutional layers, batch normalization, ReLU activation functions, and a global average pooling layer followed by a fully connected layer for classification. ResNet introduced the concept of skip connections or shortcut connections. These connections allow the network to skip one or more layers and directly connect earlier layers to later layers. Skip connections help address the problem of vanishing gradients and enable the network to learn more effectively, especially when dealing with deeper architectures. Also, we tried AlexNet. This network consists of five convolutional layers followed by three fully connected layers. It introduced the concept of using rectified linear units (ReLU) as activation functions, which helped mitigate the vanishing gradient problem and speed up training.

In terms of hyperparameters and model selection, we optimized the validation set in order to specify them. We finally tested the models on a test set and reported the results in the next section. We used the Adam optimizer with a learning rate of 0.001. In general, all the models did really well on the classification part. We also performed very well on the localization part, although there's still some room for improvement there. Overall, all the models have shown that they have the capacity to tackle our problem. Their convergence rates were relatively similar, with the choice of loss function being the most crucial factor. As expected, the mean IoU of the test set (the accuracy metric for the localization part) was highly influenced by this specific choice. The losses with the best performances were the GIoU and the L1 smooth loss.

4 Results

In this section we demonstrate the results of our expreriments. In the table below we can see some metrics regarding the best fitted model for each case.

Table 1: Models' Performance on Test set

Model	Precision	Recall	F1	Mean IoU
AlexNet	0.878	0.852	0.8657	0.531
VGG11	0.882	0.897	0.8955	0.545
ResNet18	0.902	0.913	0.908	0.688
ResNet50	**0.928**	**0.99**	**0.958**	**0.713**

The hyperparameters of the best ResNet50 models were :

Table 2: Experimental Configuration

Hyperparameter	**Value**
NUM_EPOCHS	7
BATCH_SIZE	512
LEARNING_RATE	0.001
ALPHA	100
BETA	0.8

ResNet50 is clearly the best model we have managed to create. Below, you can see the corresponding confusion matrix and ROC curve. We can observe that it achieved exceptional performance in the classification task and performed very well in the localization as well. It is worth noting that a detection with an IoU greater than 0.55 is considered correct. Overall, it appears that the depth of the models played a crucial role in their performance. ResNet models share the same architecture, including residual connections, dropout, batch normalization, activation functions, and so on. The only distinction is that ResNet50 performs these operations more times, allowing the model to learn more low-level features and detect more complex patterns. In general, even though we employed some state-of-the-art models, the task has proven to be quite easy to handle. All the models demonstrate excellent performance in both tasks, and their tuning did not require any particular special considerations.

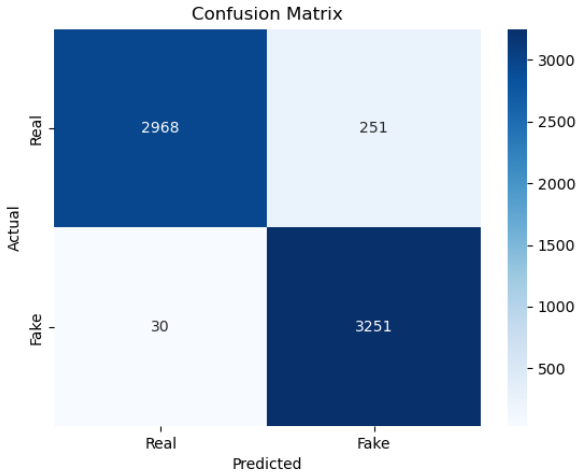


Fig. 7: Confusion matrix of best model

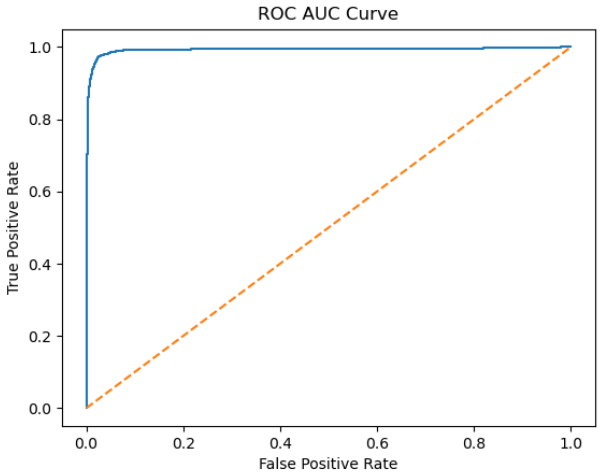


Fig. 8: ROC Curve

Intuitively, after conducting error analysis, we have been able to draw some conclusions. Firstly, all the models have exhibited high accuracy in cases where the GAN network generated blurry

outputs. A considerable portion of the test examples displayed such characteristics. Even in instances of minimal blurriness, imperceptible to the human eye, a neural network can effortlessly detect it. As we mentioned earlier, all these models have been trained on a similar dataset. Transfer learning seemed to play an important role as the models were able to capture discontinuities or artifacts created by GANs on objects where they had previously learned their structure (We ran also the models without pretrained weights and we say an important drop on their performance). Another aspect of the problem that may have made it easier for the model is the dataset creation method. By artificially generating regions with a fixed rectangular shape, we introduce a strong bias that the models are capable of capturing. Within these rectangles, the pixels generated by the GAN, may exhibit similar value distributions. These steep horizontal and vertical transitions may be challenging for the human eye to distinguish, but neural networks find them relatively easy to detect. In order to prevent this case, we could transform the problem to a classification and segmentation problem. The GAN algorithm we used to generate the data, is able to handle all possible shapes of masks. One could create masks (and corresponding annotations) with a more unorthodox/unpredictable way and make the model to predict a tensor of same size as the original input, where each pixel is a binary classification problem about whether it is part of the segmented artificial region or not.

References

[1] Yisroel Mirsky and Wenke Lee. The creation and detection of deepfakes: A survey. *arXiv*, 2020. 2

[2] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 3

[3] Jimei Yang Xiaohui Shen Xin Lu Thomas Huang Jiahui Yu, Zhe Lin. Free-form image inpainting with gated convolution. *arXiv preprint arXiv:1806.03589v2*, 2019. 3

[4] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union. June 2019. 8