**Analysis of Effective Watermarking Techniques in Images**

Analysis of effective watermark techniques against detection and removal by Deep Learning Attacks

Meesum Khan  
 Computer Science  
 Arizona State University  
 Tempe, Arizona, USA  
 makhan35@asu.edu

Abichal Ghosh  
 Computer Science  
 Arizona State University  
 Tempe, Arizona, USA  
 aghosh55@asu.edu

Zuy Pham  
 Computer Science  
Arizona State University  
Tempe, Arizona, USA  
 zpham@asu.edu

Nikhil Bindem  
 Computer Science  
 Arizona State University  
 Tempe, Arizona, USA  
 nbindem@asu.edu

**ABSTRACT**

Watermarks are a standard way of protecting a person’s rights to the photos and images they have created. However, with advancements in Deep Learning and other Machine Learning methods, it is now possible to attack these watermarks, by detecting and removing them seamlessly from images. This project explores the effectiveness of Deep Learning methods against watermarks, by trying to do just that. We work with different styles of watermarks, and try to find the ones that are least susceptible to detection and removal.

**KEYWORDS**

Watermark, Embedment, Detection, Removal, Convolutional Neural Networks, Generative Adversarial Networks.

1**INTRODUCTION**

According to studies by Copytrack [5], around 85% of the three billion images shared online daily are unlicensed. Moreover, around 64% of professional photographers reported instances of image thefts over 200 times. Every time a professional image is stolen, photographers and agencies lose an average of $446. Many of these photographers do not take legal action or cannot afford to take legal action. The rate of image theft continues to rise due to the increase of media content on the internet. With the rise in popularity of social media, it is easier than ever to distribute unlicensed images across the internet. A popular method of mitigating image theft is the embedment of watermarks on images. However, even with the addition of watermarks, around 68% of images containing watermarks have their watermarks removed since the watermarks are often not strong enough. Thus, this project analyzes the effectiveness of different styles of watermarks and finds which ones are most susceptible to detection and removal.

This project contains three main steps. First, we embed watermarks of various styles on our dataset of 22762 scenery images. Then, we use Convolutional Neural Networks to train an object detection model to detect the region of interest of watermarks in these images. Lastly, we use Adversarial Learning-based approaches for pixel value prediction to reconstruct the original image after the watermark is removed. After we complete these steps, we compare the strengths of the different styles of watermarks using the accuracy of watermark detection, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and structural similarity index measure (SSIM). Using these metrics, we find out which watermark styles are most resistant to detection and removal.

2**RELATED WORK**

2.1**Model Free Methods**

The most primitive methods of embedding, detecting, and removing watermarks from images use various model-free methods without any deep learning techniques. There has been a plethora of research done over the last few decades to make watermarks harder to detect and remove in order to protect the images’ copyrights. One method to make watermarks harder to detect is by embedding watermarks in a way that adapts to the texture, edge, and luminance of the pixels in the underlying image [8]. This causes variations between different watermarks, making it difficult for detection algorithms to learn the patterns. Another method for embedding watermarks is an algorithm based on a user-key structure that adapts to the host image’s features [6]. With the correct key, the watermark can be removed without affecting the original image. If the watermark is removed illegally, then the image will have a lower quality.

For automatic detection of watermarks, one algorithm uses total variation based on L1 norm [12]. The watermarked image is decomposed based on its structure and texture, and the watermark’s edges are distinguished from the host image’s edges. Another model free method [3] uses a simple approach for the detection and removal of watermarks simply using mathematical equations and image processing techniques. This method is based on the assumption of consistency and uniformity of the locations, size, and types of watermarks.

These methods can be effectively used once we have a localized region for the watermark, which can be achieved by Convolutional Neural Networks.

2.2**Convolutional Neural Networks**

Convolutional Neural Networks (CNNs) are the traditional approach for any object detection tasks. This is due to the fact that they are much better suited to detect features, and simplify the processing of features in high dimensional data such as images. There is only one paper in this given grouping as this is the only approach we found that was using object detection using CNNs as an initial step. The method proposed in [2] is a two-step method, consisting of two specialized CNN models for two different tasks. The first task is to locate the watermark in the given image, and thus the first CNN model is an object detection model. The second model is trained as an Image Translation model, which takes in the region of pixels affected by the watermark and attempts to convert them back to their original form. We believe that by taking the learnings of this paper from task one, we can build a strong watermark detection model, and then try different approaches to convert the pixels in the watermarked region, using both Model Free and Model Based approaches. This is how we arrived at our initial technical approach in the proposal.

The CNN-based model’s method is adept at detecting/identifying the watermarks in the images, but the step for the reconstruction of the images leads to the creation of some artifacts. Adversarial methods are able to deal with this problem by using a generator-discriminator model.

2.3**Adversarial Learning**

Adversarial Learning-based approaches have become more prevalent for watermark removal techniques. An approach using a Conditional GAN model that reconstructs images with a patch-based discriminator trained on reconstructed and original images [9] is able to deal with the problem of reconstruction of artifacts produced. This model incorporates a patch-based discriminator which uses the reconstructed image and original image to improve the reconstruction by reducing traces of watermarks (artifacts) that were left by the original model. This patch-based discriminator can thus be used on top of the CNN-based model previously mentioned to improve the results by removing the artifacts. However, it is difficult to train the discriminator to differentiate between the reconstructed and original images, as it could lead to overfitting.

Other recent approaches using Adversarial Learning are able to perform significantly well on the watermark removal task, but these methods utilize substantially greater computational power. One such approach is a general-purpose watermark removal framework called REFIT [1]. This technique uses fine-tuning and two different methods, EWC algorithm and unlabeled data augmentation, to remove watermarks without affecting the model's functionality under a weak threat model. It can remove watermarks against a wide range of watermarking schemes. Although it only evaluated image classification models under a weak threat model, this technique shows promising results in removing watermarks without affecting the model's performance or accuracy. An alternative approach used in [10] involves a deep learning framework based on an encoder-decoder structure for the task and is able to perform well even in the case when the images used have lower quality due to image compression or image resizing.

3**DATA**

We have used the Scenery Watermark Detection dataset [4] from Kaggle for this project. It contains 22762 scenery images. Of these images, 9983 images (43.9%) contained watermarks and 12779 images (56.1%) did not contain watermarks. While we can use the watermarked images for testing, it is not possible to use them for training because they have not been annotated. We decided to generate our own watermarks on the non-watermarked images, and thus create our own dataset. Each image is 512 x 512 pixels and RGB-colored, thus yielding 786432 features. Having a large dataset with a variety of different watermarks helps reduce potential biases in the watermark detection and removal models since they will be trained with a wide variety of features. We splitted the dataset into 60% Train, 20% Validation, 20% Test classes to train, validate, and test our watermark embedment, detection, and removal models.

4**METHODS**

Our solution to the given problem is to split it into three major steps. First, we need to embed different styles of watermarks on our scenery images. Then, we need to detect the region of interest of a watermark in a given image. Once we are able to accurately locate the watermark, we need to fill in the watermark pixels with values that will blend in with the surrounding pixels.

4.1**Watermark Embedment**

For watermark embedment, we used Image Processing techniques with OpenCV and Python, and came up with a watermark text generator. This watermark generator chooses a small set of random English words, a random color, a random thickness and font, a random opacity level, and a random location on the original image. Combining all these parameters, we created our watermark and added it onto the image. We also saved the bounding box of the watermark, and converted it into PASCAL VOC annotation format for object detection. We are currently working on randomizing the parameters in even more ways, to get as much variety in our watermarks as possible. A sample image before and after watermark embedment is as follows:



Figure 1: **Image before watermark embedment**



Figure 2: **Image after watermark embedment**

Once the watermark is embedded, we can move to the next step of detecting the watermark.

4.2**Watermark Detection**

We used two different methods for the task of watermark detection: UNet and YOLO. We will compare the watermark detection accuracies between the two methods to determine which method is better at detecting watermarks.

UNet [11] is a very popular deep learning architecture which is used in the tasks of image segmentation, Salient Object Detection and many more. Here, masks were created for all the watermarks as shown in Figure 5. The masks divide the image such that the parts of the image that are 0 (black) correspond to no watermark and the parts that are 1 (white) correspond to watermarks. The UNet architecture consists of an encoder network (downsampling) and decoder network (upsampling) in the form of an U-shaped structure as shown in Figure 3.

The downsampling consists of a contracting network which extracts the features present in the image through a series of convolution networks, ReLU activations and max poolings. The decoder (upsampling) is a symmetric copy of the downsampling step, except that the Convolutions are replaced with Deconvolutions, and the max pooling that leads to downsampling is replaced with an upsampling step that increases the image size using “bilinear” interpolation.

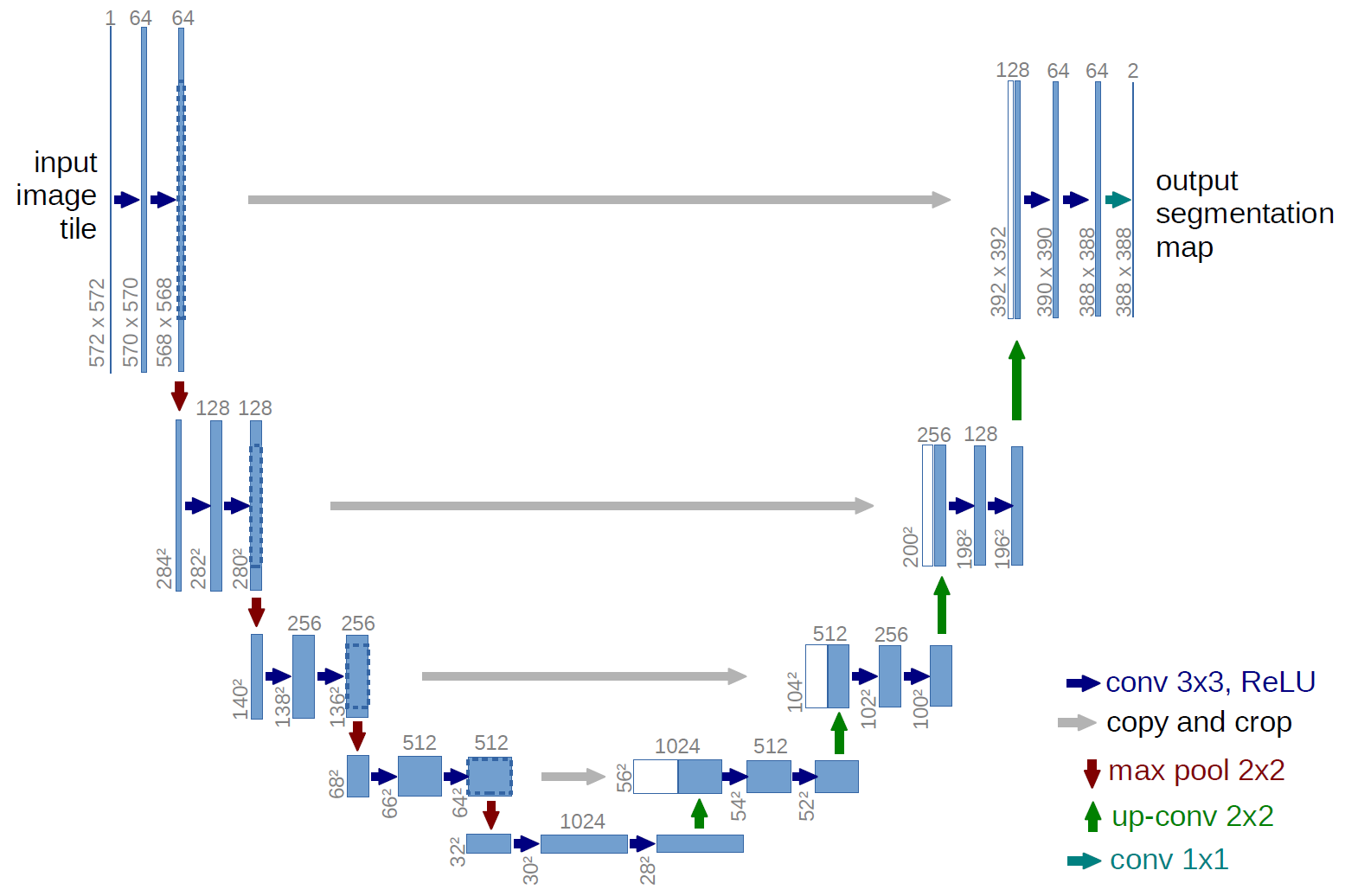


Figure 3: **UNet architecture [11]**



Figure 4: **Watermark-embedded image**



Figure 5: **Mask created for watermark**

The second method used is You Only Look Once (YOLO) Object Detection [7]. YOLO is a very strong Object Detection algorithm, based on a Convolutional Neural Network and a unique approach to searching for objects in a given image. Other algorithms like R-CNN follow a two step approach to object detection, which involves initially detecting possible areas of interest, and then passing over those areas again to confirm whether there is an object present, as discussed in [2]. With YOLO, we only pass through the image once, dividing it into blocks that the algorithm goes over once and gives a prediction score based on how likely there is an object in that block. It is much faster, and more accurate as well. Therefore, we have selected the open source implementation of YOLO V8 for the Watermark Detection step. The CNN architecture for our YOLO V8 implementation is as follows:

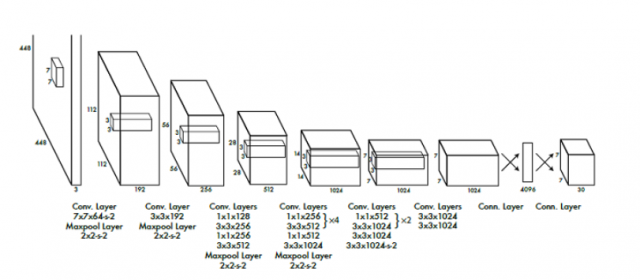


Figure 6: **CNN Architecture for YOLO [7]**

We trained the YOLO V8 Model on our generated dataset. The inputs to our model are the watermark-embedded scenic images and the bounding box annotations of the watermark.

After training the model on the watermarked images and the watermark annotations, the model will predict the region of interest in unseen test images by drawing a bounding box around the watermark as follows:



Figure 7: **Sample Image with Bounding Box**

Once the watermark’s region of interest has been detected, we can move to the next step of removing the watermark and filling in the vacated pixels.

4.3**Watermark Removal**

For the step of watermark removal, two different strategies have been implemented. These methods aim at predicting the pixels correctly (image reconstruction) given the watermarks have been detected and the part of the image containing the watermark is masked as shown in Figure 8.



Figure 8: **Masked Image: watermark is blocked out using the box predictions from YOLO V8**

The first method uses Generative Adversarial Networks that takes inspiration from the model used in the work Deep Image Prior [13], which helps in image restoration. This method uses an hourglass (encoder-decoder) architecture as shown in Figure 9. The model uses skip connections that are added from one part of the encoder layer to the decoder layer where the depth is the same. Similar to how autoencoders are used for dimensionality reduction, this model attempts to forget the black pixels in the process of going to the bottleneck. Following this, it attempts to relearn (predict the pixel values) of the marked pixels in the decoder part.

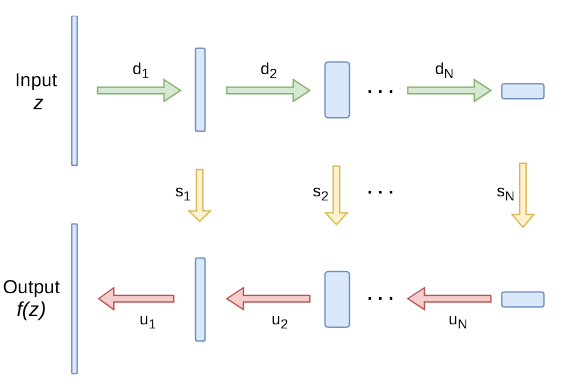


Figure 9: **Architecture Used [13]**

The depth of the model used in this method is extremely large and requires considerable computational power. The second method we are implementing is a simple autoencoder structure similar to the above implementation but considerably less complex. This model simply uses Convolutions and ReLU in the encoder, and Deconvolutions and ReLU in the decoder. We are currently working on this model.

5**RESULTS**

Upon the training and testing of the UNet model for Watermark detection, some of the results can be seen in Figure 10. From this figure, it is evident that UNet is unable to perform well as the predicted images are almost completely black. This is mostly because the watermark masks (Figure 5) are mostly black and only have small portions that are white (watermark). Due to this, the loss function is relatively low even when the prediction of the whole image is black.

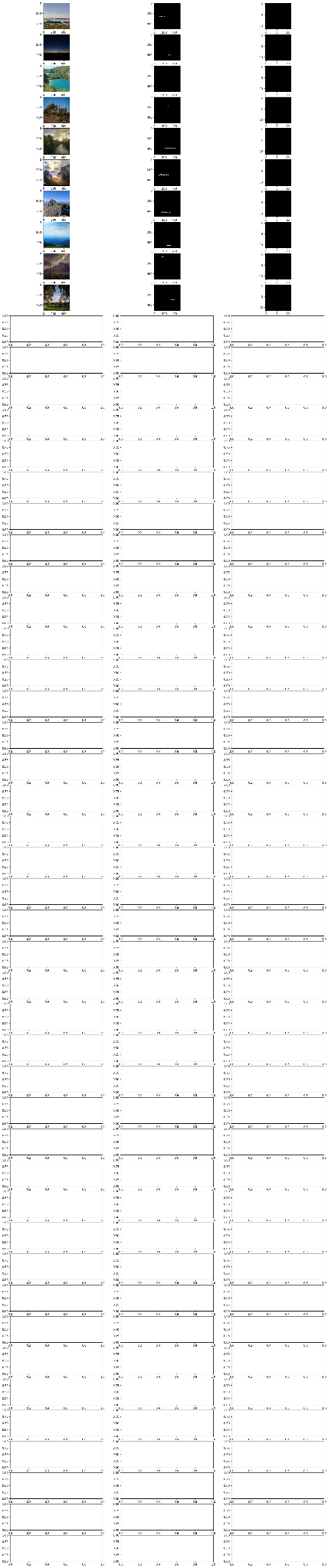


Figure 10: **UNet Predictions: Leftmost column: Watermarked images, Middle column: Watermark masks, Rightmost column: Predictions**

The YOLO V8 algorithm performed very strongly on our generated watermark dataset, and gave very high accuracy in predicting the bounding boxes for the watermarks. Visualizations for the training metrics and confusion matrix are as follows:

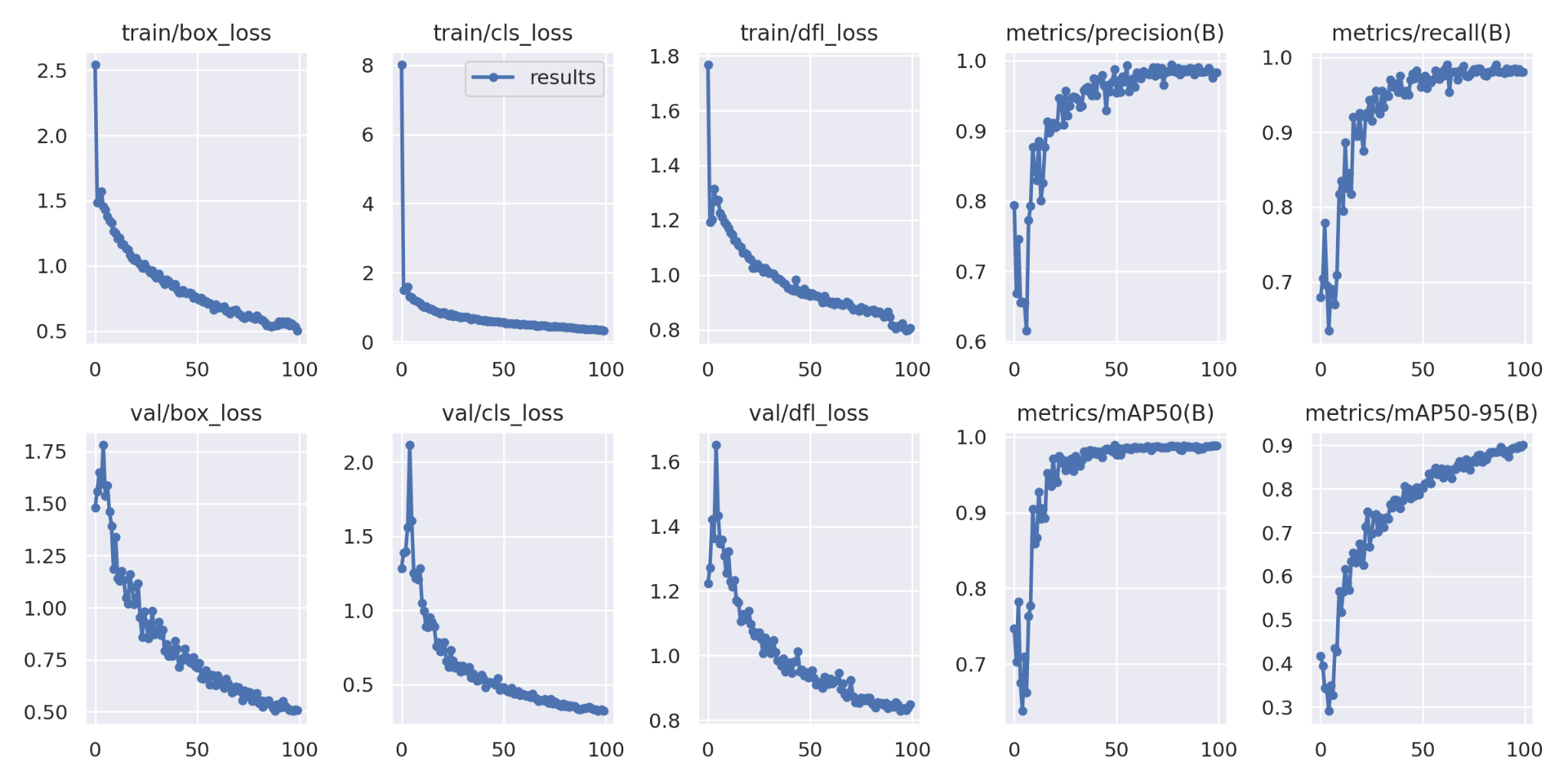


Figure 11: **YOLO V8 Training Metrics Graph**

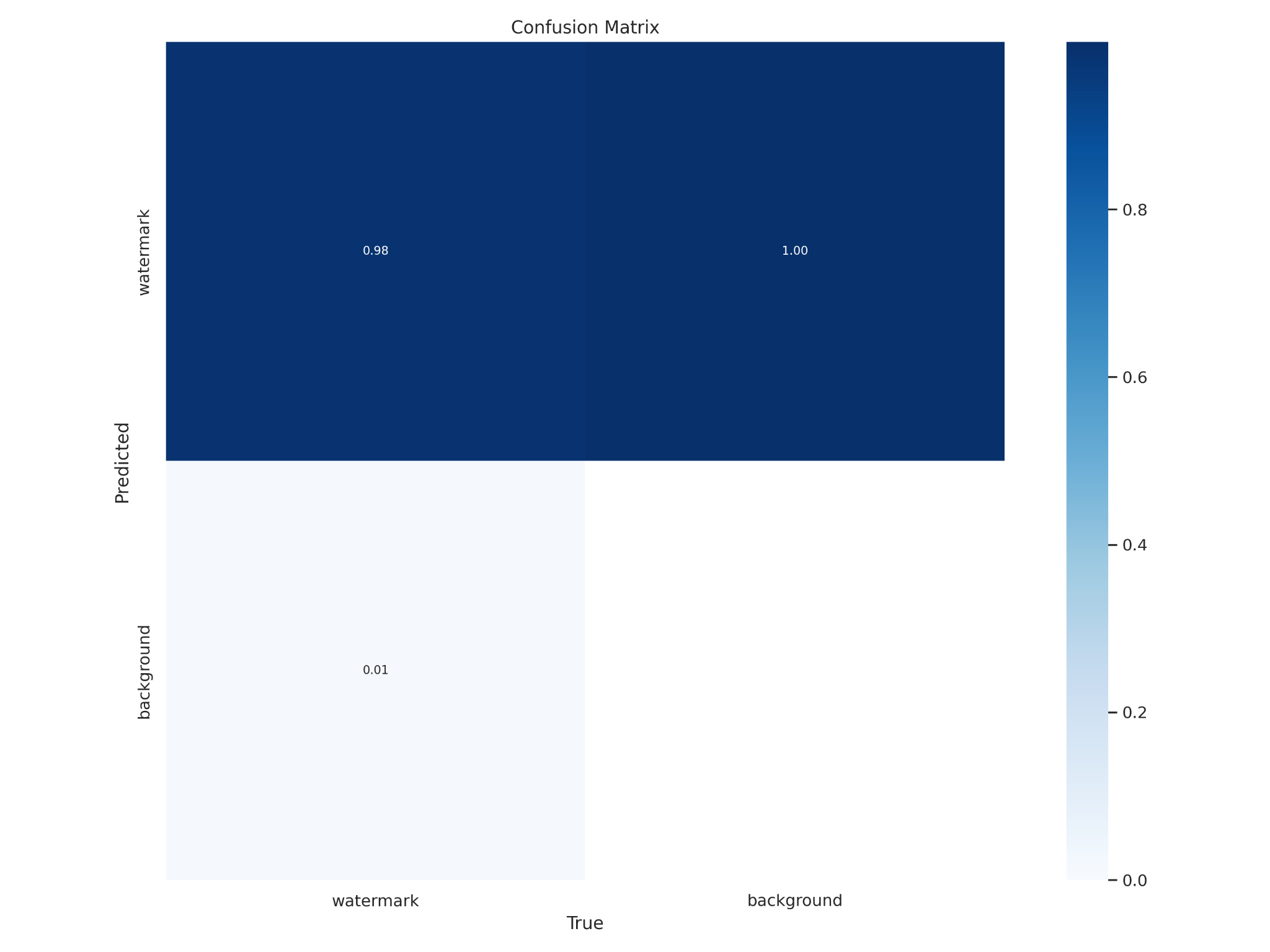


Figure 12: **Validation Data Confusion Matrix**

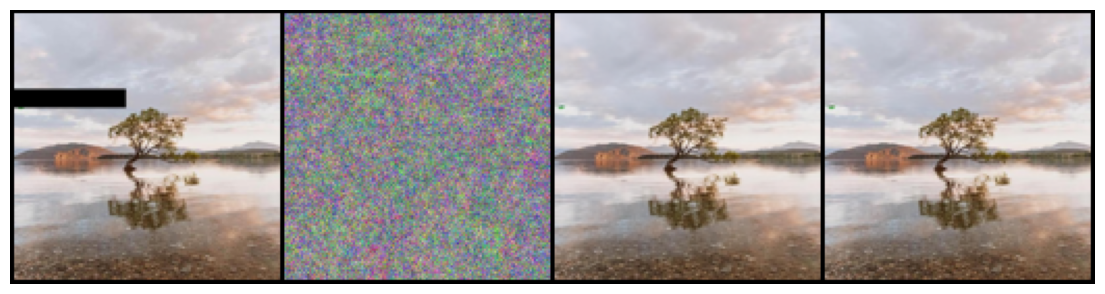
From the metrics, we see that the model is not overfitting on the training data and has similarly good performance on the validation set. Also, in the confusion matrix, we see how well the model is doing, with a 99% prediction accuracy for our watermark objects.

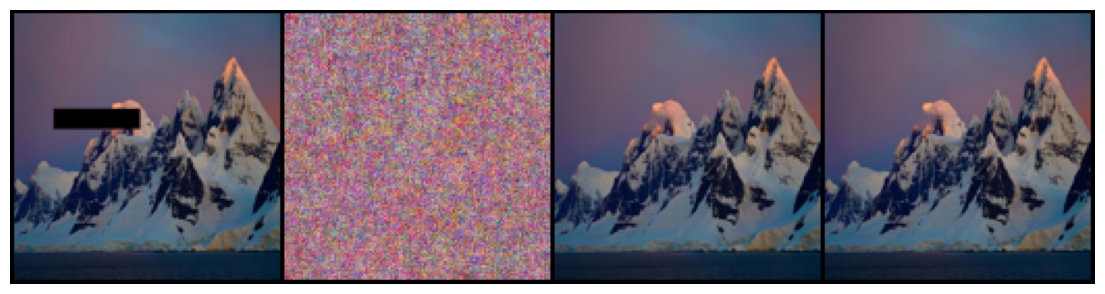
Our goal here is not just to make a good object detection model for watermarks, but to find the styles of watermarks that are strongest at being undetected by such a model. Therefore, having set our baseline detection model for this project, we will generate datasets of different kinds of watermarks, and individually evaluate how well this model performs on them.



Figure 13: **Test Image with Predicted Watermark**

The process of image reconstruction using GANs performs considerably well, as seen in Figure 14. However, it cannot remove the watermark completely in the cases where some parts of the watermarks are not enclosed within the boxes detected using YOLO V8 (Figure 15). Furthermore, since the image is covered by a box rather than an outline, it does not give the model an exact representation of what is close to the watermarks, which leads to some incorrect reconstructions, as seen in Figure 16.





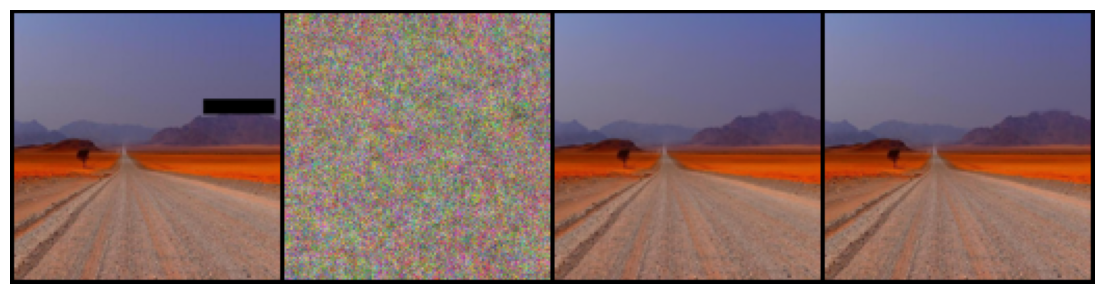


Figure 14: **GAN image reconstruction: From left to right- (i) Original masked image (ii)step 0 (iii)step 4000 (iv)final step**

This model is able to reconstruct the images well but requires excessive amounts of computational power. With this constraint, the images had to be resized down considerably in order for efficient training. The images were resized from 640x640 to 128x128 which is 5 times smaller which can be seen from the low qualities of both the masked and the reconstructed images.



Figure 15: **GAN Prediction: Left: masked image, Right: reconstructed image, green artifact of watermark left in reconstructed image at the left**

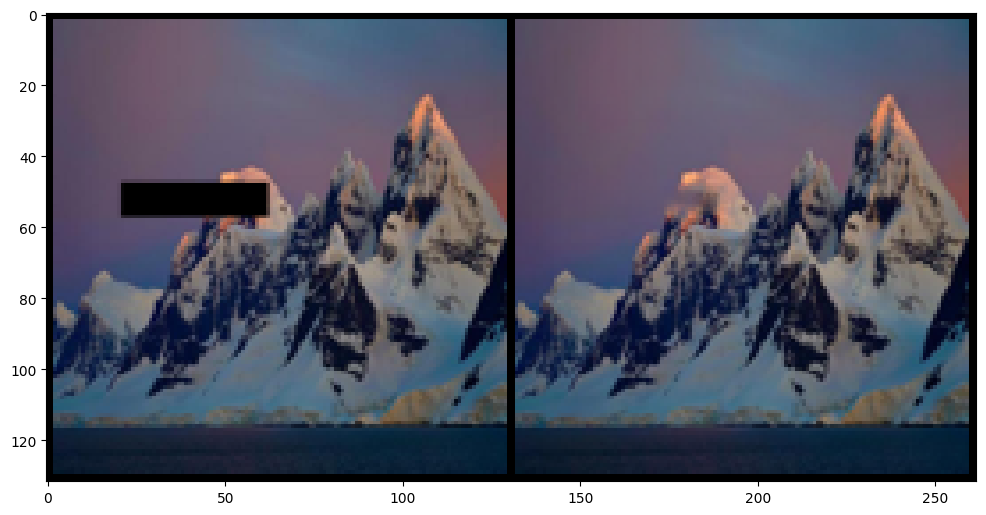


Figure 16: **GAN Prediction: Left: masked image, Right: reconstructed image, part of the mountain covered by mask gets incorrectly reconstructed as sky**

6**DISCUSSION**

With our current watermark embedding method, it is turning out to be too easy for the object detection model to find the watermarks. We need to add more variety, in terms of rotations, size, and other transformations that we observe in real-world watermarks. This will also enable us to analyze which styles of watermarks are best to use.

The bounding box annotations in the generated dataset are not fully capturing the watermark, and this is leading to incorrect bounding box predictions by the object detection model. Because of this, the next step of image reconstruction to remove the watermarks is affected. We need to improve the annotation code so that we are able to capture the trailing lines of letters such as ‘g’ and ‘y’, and this will improve our predictions. We also need to figure out how to give rotated bounding box annotations, to represent rotated watermarks.

Currently, we are completely masking out the bounding box for reconstruction in the second step. However, this is not optimal and will give subpar results. Instead, we need to use Image Processing techniques to mask out only the letters in the given bounding box, and this will give us a much better result during reconstruction.

The UNet model wasn’t able to perform well and predicted almost black images for the masks. In order to improve this performance, we could crop the images close to the watermarks to increase the size of the watermarks relative to the effective size of the images. As mentioned above we could use UNet to better detect the watermarks near the bounding boxes detected by YOLO V8 instead of using Image Processing methods. We could also improve the detection process by using a more powerful architecture such as U2Net which builds upon UNet as a nested structure.

As seen in the previous sections, the GAN model is very computationally expensive and due to limited resources, the images had to be resized down by almost five times. However, we want to retain the details present in the image originally. One way to approach this problem is to crop out a 128x128 image that contains the mask, and then run this image through the model and crop the reconstructed image back.

Another way to increase the efficiency of the reconstruction is to create an autoencoder model that is relatively simple in comparison to the GAN, which we are currently working on. This model should be able to perform comparatively to the GAN while using less resources.

7**CONCLUSION**

We analyzed different methods for detection and removal of watermarks, and found that YOLO V8 is best for watermark detection, and reconstruction using GANs is the most effective for removing watermarks. Some drawbacks we faced are that GAN reconstruction can be computationally expensive, which can be difficult for larger images. Now that we have successfully attacked watermarks, the next step of the project is to analyze different watermarking techniques and compare their effectiveness against the attack models we implemented. After completing this, we will be able to fully answer our problem statement in finding the strongest watermarking techniques against detection and removal.

**REFERENCES**

[1] Xinyun Chen, Wenxiao Wang, Chris Bender, Yiming Ding, Ruoxi Jia, Bo Li, and Dawn Song. 2021. REFIT: A Unified Watermark Removal Framework For Deep Learning Systems With Limited Data. In Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (Virtual Event, Hong Kong) (ASIA CCS ’21). Association for Computing Machinery, New York, NY, USA, 321–335. DOI:<https://doi.org/10.1145/3433210.3453079>.

[2] Danni Cheng, Xiang Li, Wei-Hong Li, Chan Lu, Fake Li, Hua Zhao, and Wei-Shi Zheng. 2018. Large-Scale Visible Watermark Detection and Removal with Deep Convolutional Networks. In Pattern Recognition and Computer Vision, Jian-Huang Lai, Cheng-Lin Liu, Xilin Chen, Jie Zhou, Tieniu Tan, Nanning Zheng, and Hongbin Zha (Eds.). Springer International Publishing, Cham, 27–40. DOI:<https://doi.org/10.1007/978-3-030-03338-5_3>

[3] Tali Dekel, Michael Rubinstein, Ce Liu, and William T. Freeman. 2017. On the Effectiveness of Visible Watermarks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 6864–6872. DOI:<https://doi.org/10.1109/CVPR.2017.726>.

[4] Vesnin Dmitry. 2023. Scenery Watermark Detection. <https://doi.org/10.34740/KAGGLE/DSV/4926870>

[5] Mary Eising. 2019. Global infringement report 2019 - Copytrack. (2019). Retrieved April 13, 2023 from <https://www.copytrack.com/wp-content/uploads/2019/03/Global_Infringement_Report_2019_EN.pdf>.

[6] Yongjian Hu, S. Kwong, and Jiwu Huang. 2005. An algorithm for removable visible watermarking. IEEE Transactions on Circuits and Systems for Video Technology 16, 1 (December 2005), 129–133. DOI:<http://dx.doi.org/10.1109/tcsvt.2005.858742>.

[7] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, et al. 2022, November 22. ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation. Zenodo. <https://doi.org/10.5281/zenodo.7347926>.

[8] M.S. Kankanhalli, Rajmohan, and K.R. Ramakrishnan. 1999. Adaptive visible watermarking of images. In Proceedings IEEE International Conference on Multimedia Computing and Systems, Vol. 1. 568–573 vol.1. DOI:<https://doi.org/10.1109/MMCS.1999.779263>.

[9] Xiang Li, Chan Lu, Danni Cheng, Wei-Hong Li, Mei Cao, Bo Liu, Jiechao Ma, and Wei-Shi Zheng. 2019. Towards Photo-Realistic Visible Watermark Removal with Conditional Generative Adversarial Networks. In Image and Graphics, Yao Zhao, Nick Barnes, Baoquan Chen, Rüdiger Westermann, Xiangwei Kong, and Chunyu Lin (Eds.). Springer International Publishing, Cham, 345–356. DOI:<https://doi.org/10.48550/arXiv.1905.12845>.

[10] Jing Liang, Li Niu, Fengjun Guo, Teng Long, and Liqing Zhang. 2021. Visible Watermark Removal via Self-calibrated Localization and Background Refinement. CoRR abs/2108.03581 (2021). DOI:<https://doi.org/10.48550/arXiv.2108.03581>.

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. CoRR abs/1505.04597 (2015). arXiv:1505.04597 <http://arxiv.org/abs/1505.04597>

[12] Hector Santoyo-Garcia, Eduardo Fragoso-Navarro, Rogelio Reyes-Reyes, Gabriel Sanchez-Perez, Mariko Nakano-Miyatake, and Hector Perez-Meana. 2017. An automatic visible watermark detection method using total variation. 2017 5th International Workshop on Biometrics and Forensics (IWBF) (April 2017), 1–5. DOI:<http://dx.doi.org/10.1109/iwbf.2017.7935109>.

[13] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. 2017. Deep Image Prior. CoRR abs/1711.10925 (2017). arXiv:1711.10925 <http://arxiv.org/abs/1711.10925>