**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, Object Detection, YOLO, 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. Recent free AI-powered tools, such as Watermark Remover.io, can seamlessly remove watermarks from images within seconds. These free services make it more convenient for thieves to strip the copyright protections of images and repost them on the internet without credit. 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 our own watermarks of various styles on our dataset of 12779 non-watermarked 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), 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 [13]. 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 a common approach for 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. Each image is 512 x 512 pixels and RGB-colored, thus yielding 786432 features. 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. Thus, we decided to create our own dataset by generating our own watermarks and annotations on the 12779 non-watermarked images. In creating our own watermarked dataset, we wanted to make the watermarks as realistic as possible. Many previous papers used watermarks that were large and easily detectable since they were tailored for a certain task, and hence were not realistic. We drew inspiration from multiple watermark styles on the internet in order to create a diverse, realistic watermarked dataset. Our watermarks contained random English words, random color, random thickness and font, random opacity level, and random location. 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 styles. For testing, we created six different test sets of watermarks: Cursive, No Rotation, Rotated, Small Text, Text High Opacity, and Text Low Opacity. Then, we used the results to compare which of these styles helped watermarks be most effective against detection and removal.

# 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, a random rotation, 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. Using this watermark embedment method, we created six different test datasets, with each containing a unique type of watermark with certain characteristics, so we can compare which characteristics are strong against detection and removal. The unique watermark types that we’ve tested are Cursive, No Rotation, Rotated, Small Text, Text High Opacity, and Text Low Opacity.



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 [12] is a popular deep learning architecture 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.

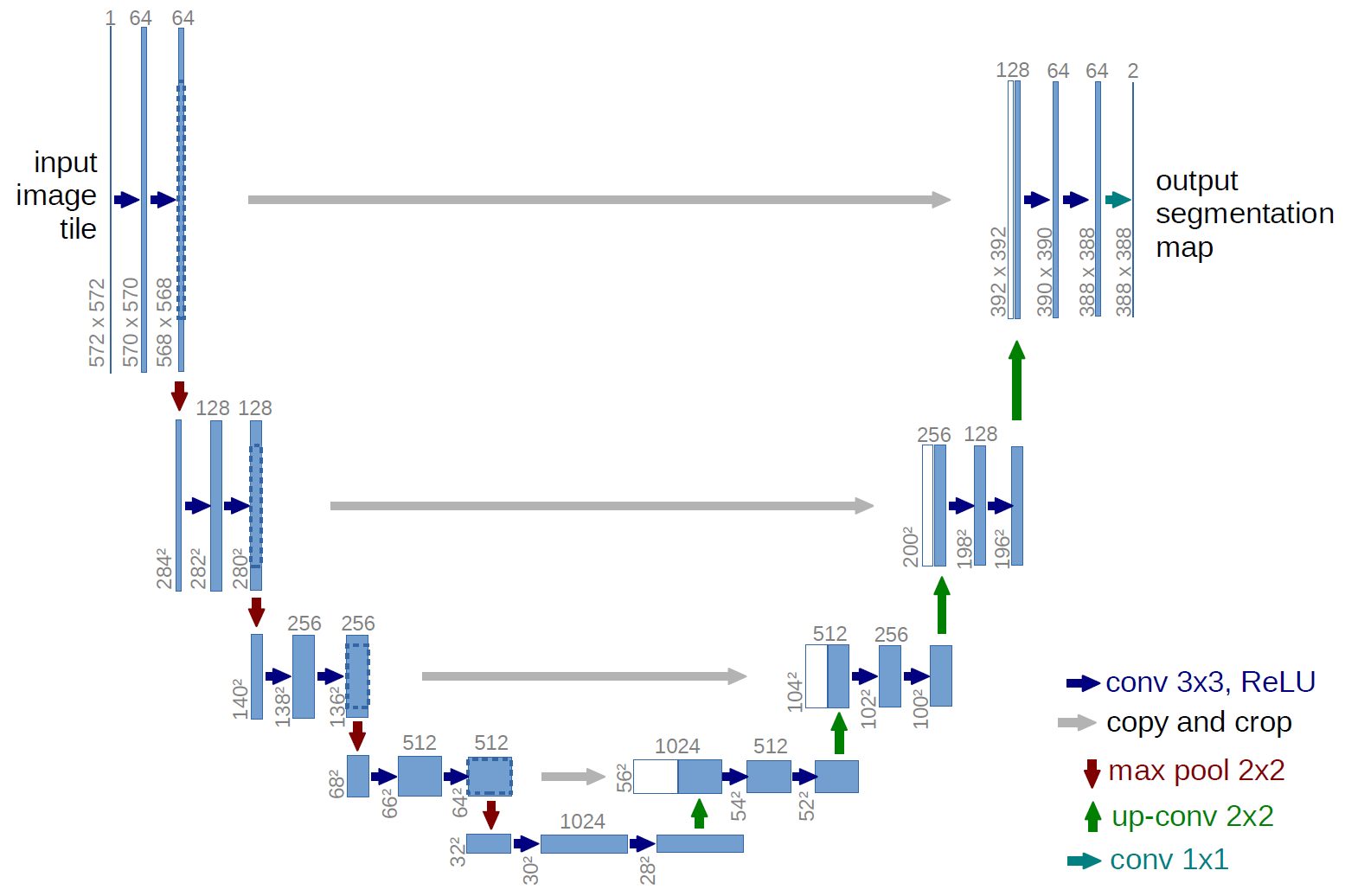


Figure 3: **UNet architecture [11]**



Figure 4: **Watermark-embedded image**



Figure 5: **Mask created for watermark**

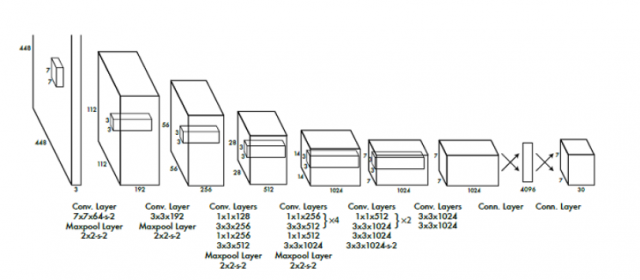


Figure 6: **CNN Architecture for YOLO [7]**

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 Mask Detection**

As mentioned previously, YOLO marks the region of interest by creating a box around the detected matermarks. However, the size of the box detected varies significantly on the size of the watermark and the rotation of the watermarks. In the case when the region of interest(bounding box) is relatively big in comparison to the image, blacking out the whole box leads to a significant loss in information and hinders the reconstruction of the images. For this reason, after the bounding box is predicted, the next step is to detect the exact masks of the watermarks in the bounding box as shown in Figure 5.

This detection of the exact mask of the watermarks has been first done using UNet. Here, the architecture of UNet leads to a problem. As can be seen in the architecture (Figure 3), the size of the output image significantly reduces in size compared to the original image (in Figure 3, the size goes from input - 572x572 to 392x392). This loss is due to the upsampling method which is used in UNet. Given the already small size of the bounding box, this leads to pixelated and incorrect predictions. Hence, a better model is used for predicting the masks in the bounding box.

The next method which is used in the watermark mask detection is the U2Net[11] architecture. This architecture is a nested UNet architecture which is able to deal with the problem of uneven output and input image size based on an improved upsampling function which uses interpolation. This model was trained on the predictions of YOLO on a dataset of watermarks which contained watermarks of all types. The loss function used here was Binary Cross Entropy Loss function. This was used since the final prediction of the pixels is binary: 1 for watermark, 0 for no watermark.

## 4.4**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.

The first method uses Generative Adversarial Networks that takes inspiration from the model used in the work Deep Image Prior [14], 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 8: **Masked Image: watermark is blacked out using the box predictions from YOLO V8**

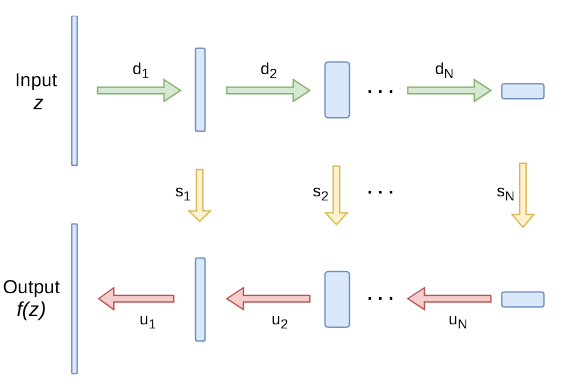


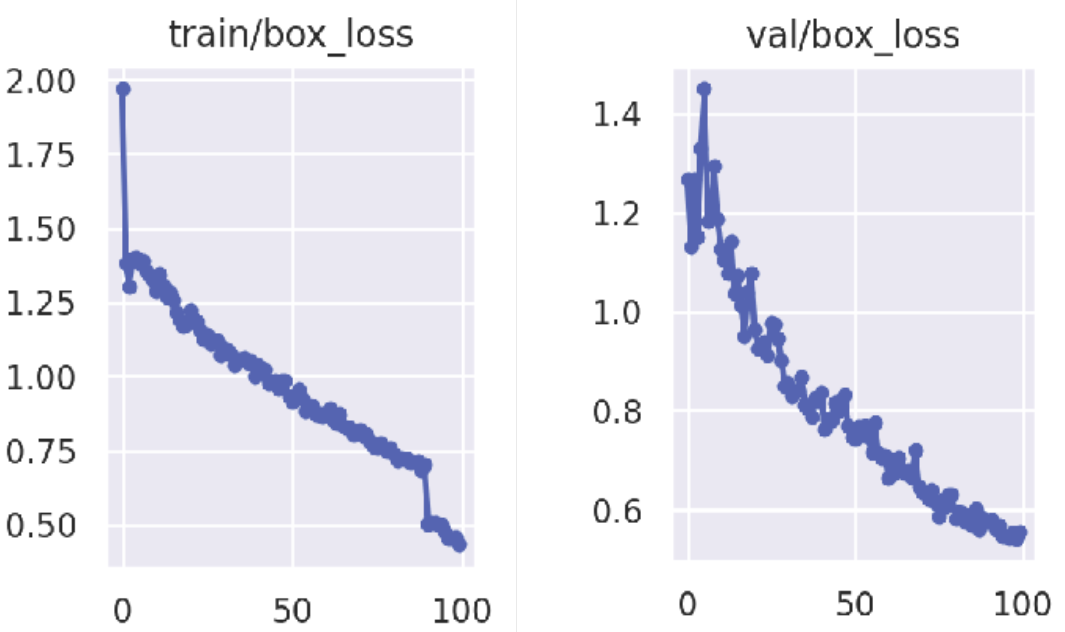
Figure 9: **Architecture Used [13]**

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. However, due to the simple structure, this has still not yielded significant results and hence not detailed upon.

# 5**RESULTS**

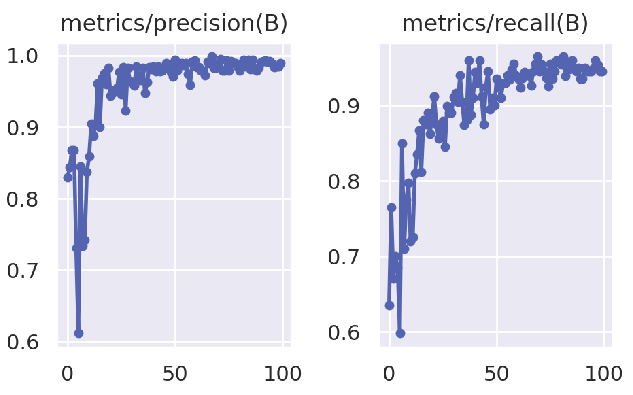
## 5.1**Watermark Detection**

For evaluating our YOLO Watermark Detection model, we decided to display combined results for all the unique watermark test sets, as there was a negligible difference in the performance of YOLO for the different test sets. We used a variety of metrics to evaluate our YOLO model. The first plots are the box loss of the model on the training and validation data, which shows how accurate and tight the bounding box is around the object. With low values for both train and validation sets, our model shows no signs of overfitting.



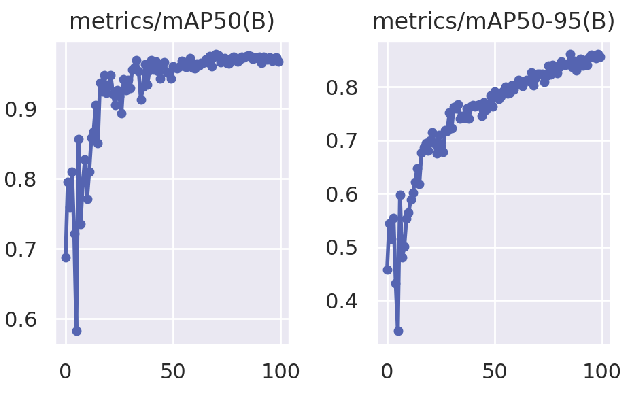
Plot 1: **Box loss on train and validation sets**

The next two plots show the precision and recall of the model on the test data.



Plot 2: **Precision and recall on test set**

Finally, we show the mean Average Precision of the model on the test data, with both mAP50 and mAP50-95. mAP50-95 is the most widely recognised metric for object detection. It is the average mAP for IOU thresholds which have been varied from 0.5 - 0.95, with 0.05 step intervals. Essentially, it gives us an idea of how much the predicted bounding box is overlapping with the ground truth bounding box.



Plot 3: **Mean Average Precision of the model on test data**

The following table shows our final values for the different metrics. With an mAP50-95 score of 0.881, our model is able to detect all the different styles of watermarks that we’ve generated very well. Therefore they must be compared against each other in the next step, which would be where we find out which watermark styles are better.

| **Metric** | **Final Value** |
| --- | --- |
| Box Loss (Train) | 0.488 |
| Box Loss (Valid) | 0.574 |
| Precision (Test) | 0.994 |
| Recall (Test) | 0.942 |
| mAP50 (Test) | 0.976 |
| mAP50-95 (Test) | 0.881 |

Table 1: **YOLO result metrics**

Upon the training and testing of the UNet model for Watermark detection, some of the results can be seen in Figure 10. It is evident here that UNet is unable to perform well as the predicted images are almost completely black. This is mostly because the whole image 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**



Figure 11: **Test Image with Predicted Watermark**

## 5.2**Watermark Mask Detection**

The training and testing of the U2Net[11] architecture on the boxed region of interest for the watermarks predicted by YOLO gives results as shown in Figure 12. From the figures, it is evident that this time the mask detection is much more successful in comparison to the previous attempt(Figure 10) which had been done using UNet on the whole image.

The loss metrics obtained from testing the U2Net model on test datasets with the different types of watermarks give the results shown in Table 2.

| **Watermark Type** | **Test BCE Loss Avg** | **Test MSE Loss Avg** |
| --- | --- | --- |
| Cursive | 0.7185474360585212 | 0.10710281009599566 |
| Random | 0.7340560410022735 | 0.12132237925380468 |
| No Rotation | 0.7148387194180516 | 0.10372630227269151 |
| Rotated | 0.7495674912393446 | 0.24372630227269151 |
| Small Text | 0.7267879092910953 | 0.11312015873300178 |
| Text High Opacity | 0.7208426183432734 | 0.10930173625470989 |
| Text Low Opacity | 0.7382167496798951 | 0.13100712957531818 |

Table 2: **Test BCE Loss Avg and Test MSE Loss Avg on different watermark types**

From the results of the Testing, at a glance it may seem that all the watermarks have very similar Loss Values. Even though all the watermarks have very similar Loss values, there are some insights that can be determined.

It is apparent that Rotated style has the highest MSE Loss with a value of around 0.243 which is almost double of the other values. A higher MSE Loss value shows that the model is unable to detect these images with high accuracy, meaning that Rotation style watermarks with the highest loss are the hardest to detect. Following this, Text Low Opacity and Random style watermarks are harder to detect. Finally, Text High Opacity, Cursive and Small Text and No Rotation have the least Loss values indicating that they are the easiest to predict.

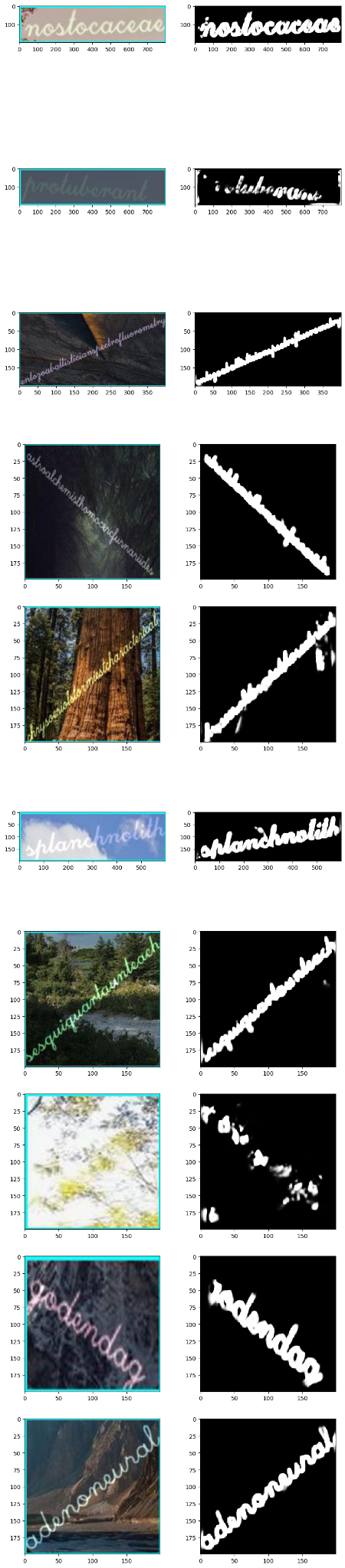
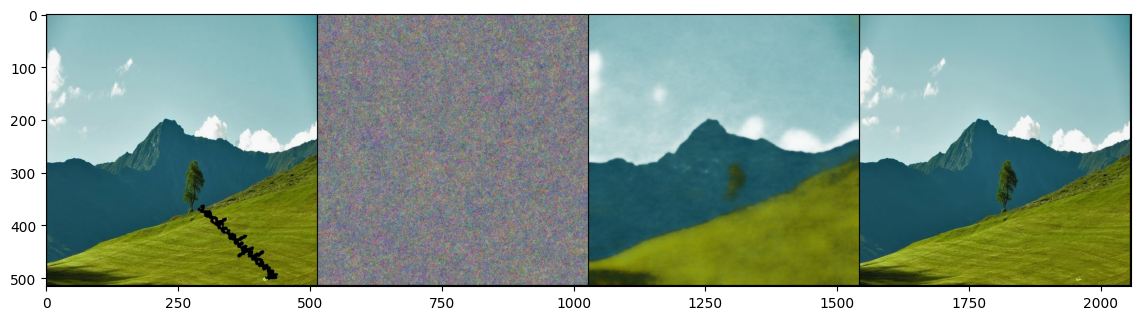


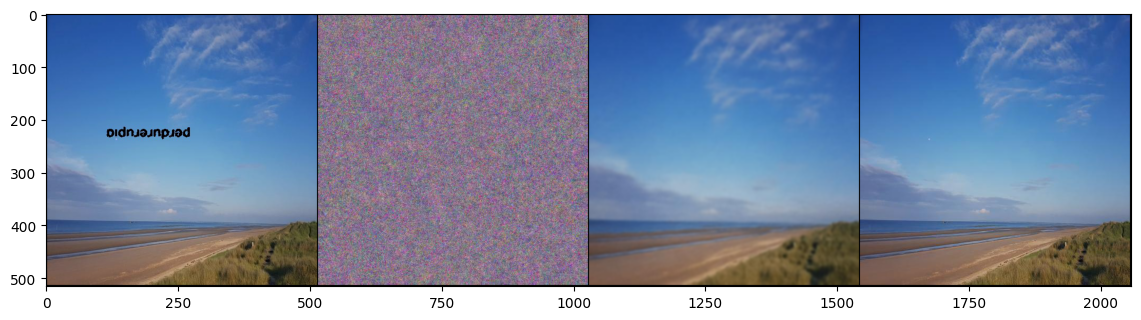
Figure 12: **U2Net Mask Prediction: Left: Masked watermark, Right: Predicted Mask**

These detected masks are added on top of the original watermarked images and subsequently used in the Watermark Removal step.

## 5.3**Watermark Removal**

The process of image reconstruction using GANs performed considerably well, as seen in Figure 13. If the watermark is detected well, then the watermark is completely removed. However in the cases where a part of the watermark is not detected in the mask, the GAN is unable to remove it. This can be seen from the reconstruction shown in Figure 14.





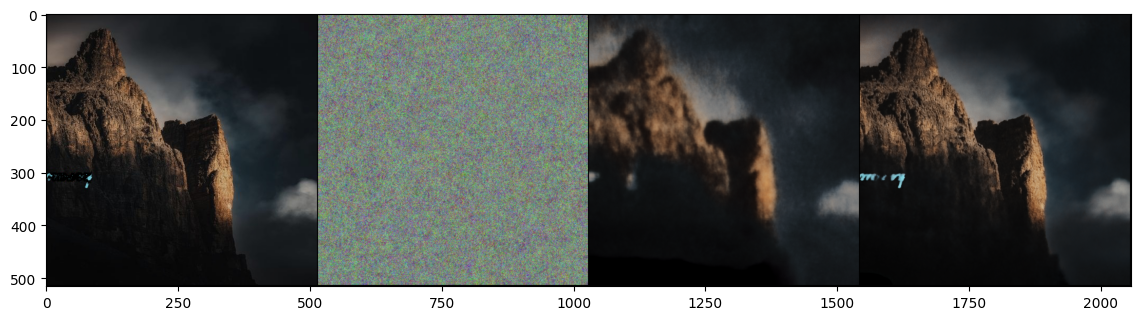


Figure 13: **GAN image reconstruction: From left to right- (i) Original masked image (ii) step 0 (iii) step 150 (iv) final step**

This model is able to reconstruct the images well but requires excessive amounts of computational power. With this constraint, the number of steps for which the GAN had to be run was limited to 7000 steps.



Figure 14: **GAN Prediction: Left: masked image, Right: reconstructed image, green artifact of watermark left in reconstructed image at the left**

The results of the watermark removal using the GAN has been quantified by comparing the generated images to the original images. There are multiple ways to achieve this comparison. Using Root Mean Squared Error can be a good metric as most of the image is unchanged except for the watermarked parts. However, for images, RMSE is not a perfect metric as it specifically focuses on the differences in the pixels and not on the form/structure of the images. Hence, a more domain specific metric, SSIM, has been used for comparing the performance.

Structured Similarity Index Measure(SSIM) is a metric used to measure the similarity between images, drawing parallels to the visual system used by humans.

This comparison of metrics using has been done for all different watermark types, and the mean of the values have been presented in Table 3.

| **Watermark Type** | **RMSE(Avg)** | **SSIM(Avg)** |
| --- | --- | --- |
| Cursive | 0.00384134512 | 0.98233172743057 |
| Random | 0.0147325602 | 0.92314108660594 |
| No Rotation | 0.00482902312 | 0.9851024759347 |
| **Rotated** | **0.02233051425** | **0.82436059233368** |
| **Small Text** | **0.0195237546** | **0.84075460474794** |
| Text High Opacity | 0.00828638712 | 0.95415887702955 |
| **Text Low Opacity** | **0.0145116769** | **0.89428445993468** |

Table 3: **Comparison of metrics for all watermark types**

From the results shown in Table 3 many clear trends are observable. First of it is seen that the order of SSIM of the watermark types are:

No Rotation>Cursive>Text High Opacity>Random>Text Low Opacity>Small Text>Rotated

As lower SSIM values represent that the predicted images are different from the original images, the watermark styles with the lowest SSIM scores should be used for watermark creation.

Out of all the watermark types, Cursive and No Rotation styles lead to extremely high values of SSIM (close to 0.98), meaning that in these cases, the watermarks are almost definitely detected and removed. This was also evident from the fact that both No Rotation and Cursive styles were easily detected with relatively high accuracy in the mask detection section(Table 2). Following this, Text High Opacity also had a high score, but lower than the previous two.

The best styles to use for increasing the watermark robustness include Rotated, Small Text and Text Low Opacity as these styles obtain a SSIM score of below 0.9. By sorting the average SSIM values of these 3 styles it is seen that Rotated is the most robust, followed by Small Text, and finally Text Low Opacity.

# 6**DISCUSSION**

Going into this task, we hardly imagined that the object detection step would be so effective against watermarks. Our YOLO v8 model was able to accurately detect almost every single watermark we fed it, regardless of the features we tweaked to make the watermark unique and hard to detect. Therefore, seeing that the difference in detection was negligible across different kinds of watermarks, we realized that the next step of watermark removal would be where we can compare different styles of watermarks against each other.

The initial results of UNet were good, but were heavy on resources and did not give us the output image size we needed. Upgrading to the U2Net Model, we saw a steep increase in performance, and the model did relatively well in predicting a segmentation mask for the given watermark. Again, our goal with fine tuning the models was to emulate the real world, where the most cutting edge deep learning solutions are currently being leveraged against watermarks, and therefore we needed to test our watermarks against models of similar performance.

As mentioned earlier, watermark detection was all too easy, and the main strength of watermark styles lay in their ability to pose a challenge for the GAN model in removing them without a trace. We found that the GAN model is very computation heavy, and if the entire watermark is segmented properly, performs very effectively. However, even if small artifacts are left over from the previous step of generating a segmentation mask, then the GAN ends up using those artifacts in its reconstruction, and the results are not very clean. As seen in Table 3, some watermarks withstood the U2Net model well, and thus the GAN step that followed did not create clean reconstructions.

# 7 FUTURE WORK

In the future, we would like to train and test more watermark styles to help our model detect and remove new types of watermarks. We would like to experiment with images and logos as watermarks since those kinds of watermarks are common in the real world in addition to text watermarks. We would also like to implement a stronger UNet model to better segment the watermark, as well as improve our Autoencoder model.

# 8**CONCLUSION**

In conclusion, multiple different methods for the detection and removal of various styles of watermarks have been tested and analyzed. For the task of watermark detection, YOLO V8 was used as it produced the best results and was able to perfectly detect all the different styles of watermarks. Furthermore, for the detection of the masks within the box detected by YOLO V8, the U2Net Architecture was used as it performed significantly well obtaining an average BCE Loss of 0.73 on testing over all the styles of watermarks. Here it is observed that adding rotation to the watermarks make them significantly more difficult to detect. For the subsequent task of Watermark removal, the GAN model was very effective. By comparing the similarity of the results obtained from the GAN architecture with the original images it is seen that an average SSIM of 0.91 is obtained. Examining the SSIM scores of each type of watermark shows that the most robust watermarking styles are that of Rotated, Small Text and Low Opacity.

Therefore, our findings suggest that watermarks which have a degree of rotation, are smaller in size, and have lower opacity, are stronger than watermarks that do not have these features. It was also observed that a cursive font is easily removed, and so are watermarks that are not rotated at all.

# **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] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R. Zaiane, and Martin Jagersand. 2020. U2-Net: Going deeper with nested U-structure for salient object detection. Pattern Recognition 106 (2020), 107404. <https://doi.org/10.1016/j.patcog.2020.107404>.

[12] 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>

[13] 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>.

[14] 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>