APS360 Watermark Remover Group 4 Progress Report

Xun Wei, 1001173507
Geling (Gloria) Li, 1001290417
Ta Jiun Ting, 1001188500

Data Gathering

The goal of our project is to train a machine learning model that can remove watermarks from images. In order to achieve this, we first need a large dataset of images to train on. Since watermarks are typically applied to photographs, the resolution of our images cannot be too small. ImageNet is an image database that has hundreds of thousands of images; therefore, it is a suitable source for our image data collection. Due to copyright reasons, ImageNet only provides the URLs of where the images can be found to the public. These images come in vary in dimension, and there are problems with the URL in some cases.

To prepare the image data for training, we first download the pictures from valid URLs on the ImageNet database, then resize them to the same size of 300 x 500 pixels. The images that cannot be resized to 300 x 500 pixels are discarded. In addition, we discovered in our research that watermarks applied to images are typically either text or an image of a logo. Therefore, we set up a script to automatically download images, then apply either a transparent text or image watermark on these images. To account for different scenarios, we created two sets of data. The first set of images have the watermarks applied at the same position in each image, while the second set has the watermarks applied randomly in each image. We believe that the set with watermarks applied in a random fashion will be more difficult to remove by a machine learning model.

Meeting with TA

On the morning of March 11th, our group met with one of the TAs of this course, Bibin Sebastian. We mentioned to him that we have finished the data gathering, and have started to train an autoencoder model. We chose this model because it is something we have learned about in the course and have the code available. He first said that we can possibly try to solve this problem first using libraries that do not use machine learning, such as OpenCV. This could be done to compare the performance of our model.

On the feedback given to us on the project proposal, Lisa suggested that we can use a residual network to generate images. We mentioned this to Bibin, and he suggested that we can perhaps train multiple models and compare performance. Lastly, he mentioned that since our model is training on large images, training will likely take a long time on our local machines. He suggests that we can try to move our set-up to the Google Colaboratory, and use the accelerated hardware available to train our model on the cloud.

Preliminary Models

Autoencoder

The first preliminary model we have built is a simple fully-convolutional auto-encoder. We picked fully-convolutional networks because we want the model to be able to take images of any sizes. During training, we pass in the watermarked image's tensor as the input to the model and compare the output with the original image (without watermark) by calculating the MSE loss. The results of this auto-encoder are not as expected. The reconstructed pictures are getting clearer and clearer; however, the watermark does not seem to be fully removed, although its intensity is indeed slightly lowered.

Below are the results after running 50 epochs:

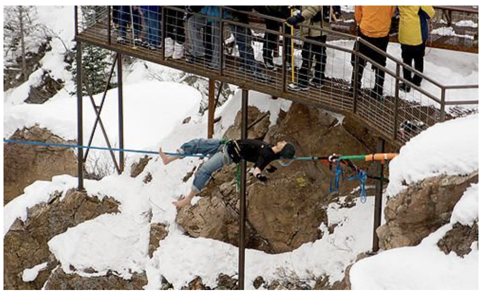

Figure 1. Original Image

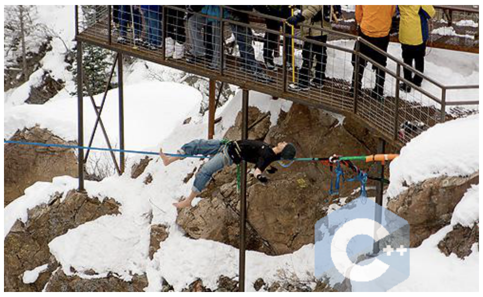

Figure 2. Watermarked Image

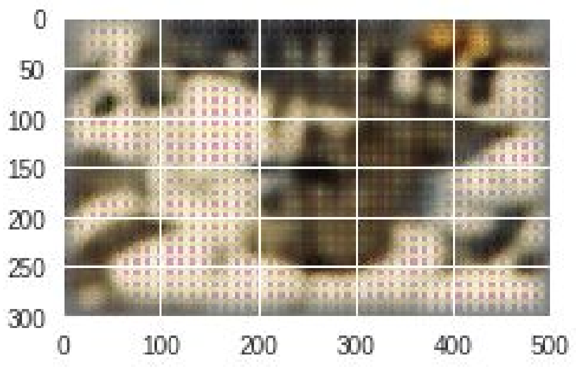

Figure 3.1 Reconstruction at Epoch 0

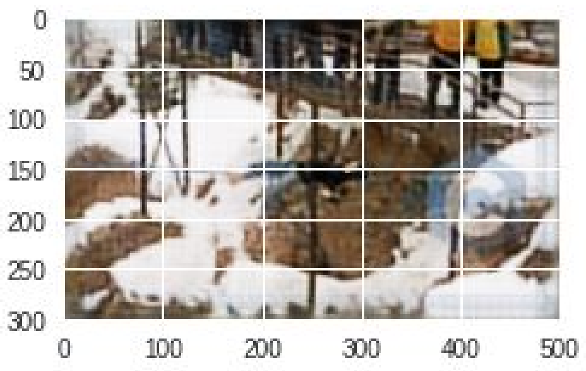

Figure 3.2 Reconstruction at Epoch 10

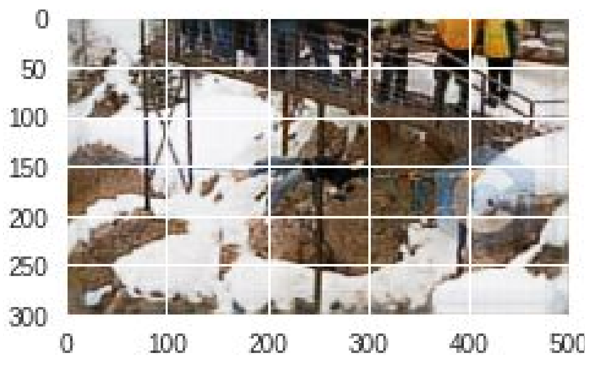

Figure 3.3 Reconstruction at Epoch 20

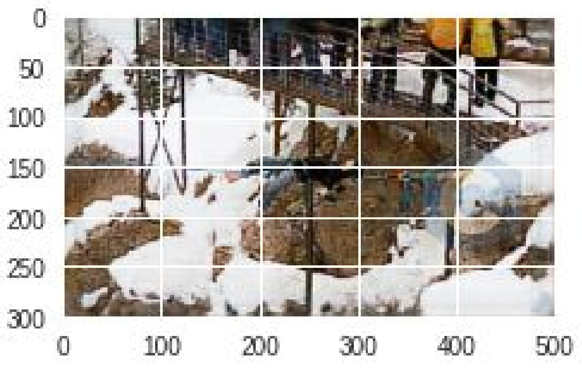

Figure 3.4 Reconstruction at Epoch 30

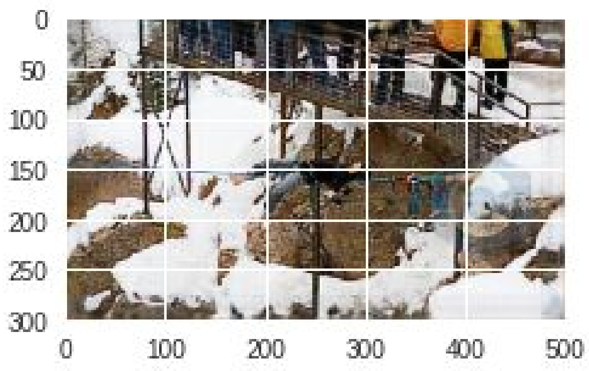
Figure 3.5 Reconstruction at Epoch 40

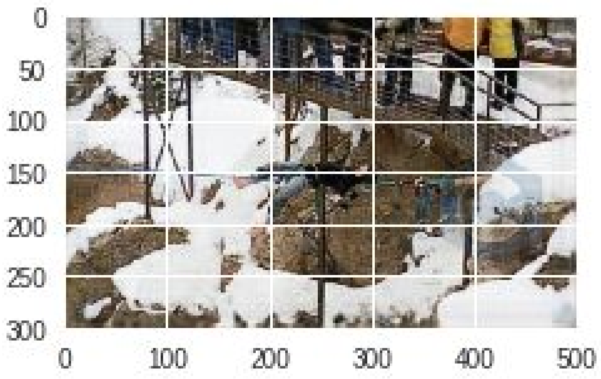
Figure 3.6 Reconstruction at Epoch 50

Residual Block + Dilated Transpose Convolution

In addition, we also trained a fully convolutional network with residual blocks based on the recommendations of Lisa and Bibin. The network is built with residual blocks as well as transpose convolution with dilation. We selected this model in an attempt to recreate the results from a study done by Ulyanov et al. [1]. This model utilizes two fully convolutional residual blocks with no downsampling, then four transposed convolutional with dilation are used to recreate the image. We initially started with a deeper model with more layers, but we had to reduce the model size given our limited computational resources. This model uses close to the 12GB GPU memory available on Google Colab.

To train this model, we use MSE to compare the loss between the original unwatermarked image with our reconstructed image. We started off using a learning rate of 0.02 for 100 epochs, then lower the learning rate to 0.01 for 50 epochs, and 0.001 for 100 epochs, and 0.001 for the last 100 epochs. The graduate reduction of the learning rate is to ensure that we will reach the lowest loss possible. After 350 epochs of training using various learning rates, the results we obtained from this network are unpromising as the watermark still visible after training. However, similar to our first auto-encoder model, we do observe a lower intensity of the watermark in the reconstructed image.

<u>Training set results:</u>

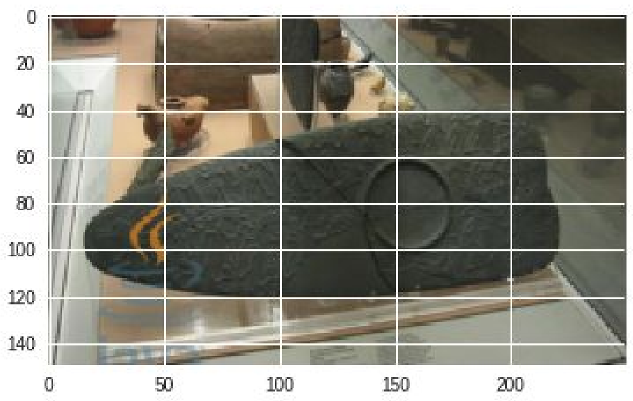
Figure 4.1 Original watermarked image

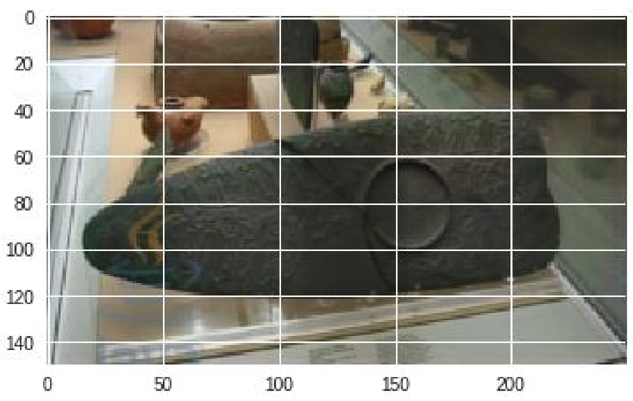
Figure 4.2 Reconstructed image

Testing set results:

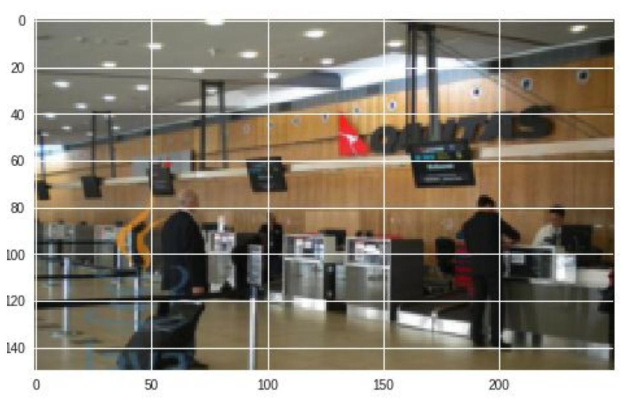
Figure 5.1 Original watermarked image

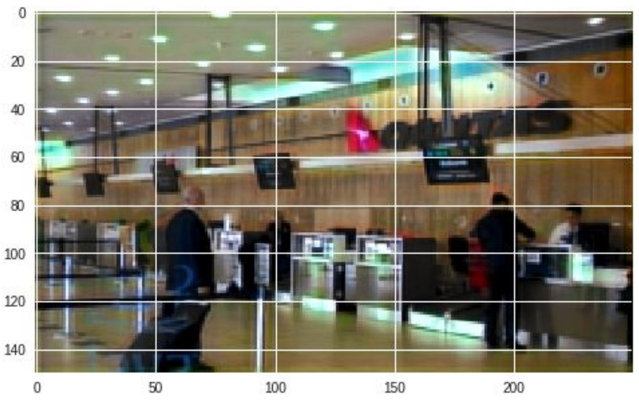
Figure 5.2 Reconstructed image

Next Steps

There is still about 2 weeks left before the presentation. We need to train our model and make adjustments accordingly. Training our model will likely take a long time due to the large image dataset. Furthermore, we are also exploring other ways to train our model since the original image without watermark is unlikely to be available to the users in real-world scenarios.

We are currently attempting to construct a model that applies a simple mask at a random location during training, and train the residual block with dilation network described above to detect the mask. This is an attempt to recreate the real-world situation where the original image is unavailable to the user.

We plan to finalize the model selection by Friday, March 22, and spend the week after tuning the hyperparameters before presenting the results on Monday, April 1. Afterwards, we will summarize our procedures and our findings in the final report submission.

Reference

[1] D. Ulyanov, V. Lempitsky, and A. Vedaldi, “Deep Image Prior,” Deep Image Prior. [Online]. Available: https://dmitryulyanov.github.io/deep_image_prior. [Accessed: 17-Mar-2019].