Mahmut Alp Erkent
Oğuzcan Nas

16/02/2021

Project Report

Introduction

The article we have chosen for our project was titled "Hiding Images within Images" (Baluja, 2020). The problem this paper deals with is embedding a large amount of information within an image with imperceptible distortions. Specifically, a full color image was embedded within another image with the same size. This problem is relevant because it creates the means for embedding authorship and copyright information to the image, which in turn enables detection of altered or fake images by hiding invisible markers throughout the original image. Another benefit is the ability to embed meta-information directly to the image, such as motion vectors that reveal hidden motion, bounding boxes for object detection, or extended color and depth information for each pixel. It can also be used to send secret messages without a secure channel.

Related Work

This problem is referred to in the literature as digital image steganography, which is the practice of concealing information in digital images. There are mainly two different approaches to hide the information: spatial and frequency data hiding. In spatial data hiding, raw image pixels are used to hide the information bits. Most commonly, least significant bits are used to hide the information. This is achieved by changing the last couple bits of a color channel of some pixels to store the information. Since any change in these bits causes a very slight change in the color of the pixel, the overall change is imperceptible to naked eye. This method enables high capacity to store information, but this hidden information is easier to detect and more fragile to image processing and compression compared to the other main approach, frequency data hiding. In frequency data hiding, instead of raw image pixels, coefficients of certain frequency transformations are manipulated. The most common method is using discrete cosine transform, which is also used for JPEG compression. This is achieved by splitting the image in a channel into eight-by-eight-pixel groups and encoding them as coefficients of different cosine waves, which in turn are obtained by adding different frequencies of cosine waves together. When the information is stored in the coefficients of these cosine waves, while there is less capacity, it is harder to detect the hidden information, and the information is more robust to image processing and compression (Cheddad et. al., 2010).

Models in this field are mostly non-ML based ones that incorporate these explicit embedding approaches with additional techniques like denoising, image dividing, factorization, etc. to increase the capacity and make the information resistant to steganalysis, which is the practice of discovering hidden information. There were many attempts to use deep neural networks in the steganalysis domain with high rates of success (Yedroudj et. al., 2018); however, such attempts are limited when it comes to information hiding. The state-of-the-art system in this domain uses deep neural networks that are simultaneously trained with generative adversarial networks that feeds an error signal to the hiding network that indicates how easy or hard it is to identify the hidden information (Hu et. al., 2018). Although it is indeed very hard to detect the hidden information in this system, its capacity to hide information is very limited.

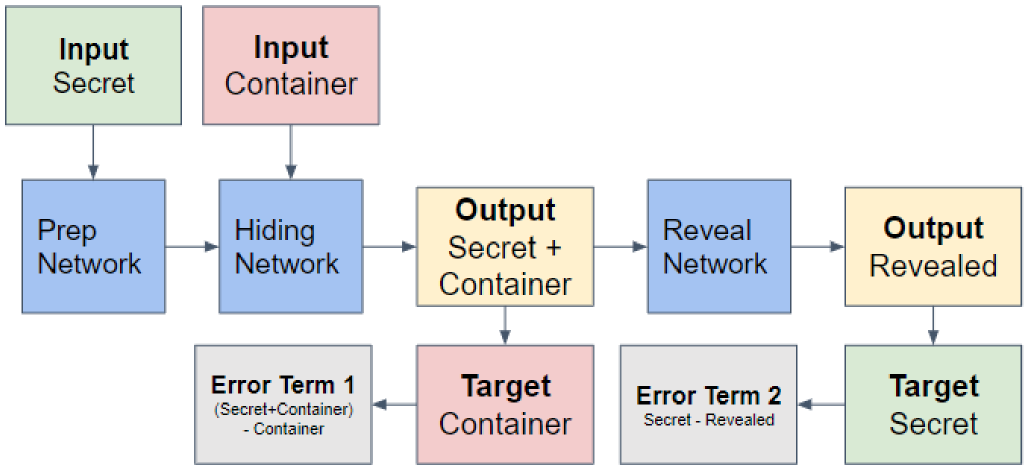

Figure 1: Our implementation of Baluja, S. (2020).

There are several differences between previously described steganography literature and this work. Firstly, this work does not necessitate errorless encoding and decoding of the hidden information. This error, which is inevitable due to the sheer size of the hidden information, is balanced between the container and recovered images. Secondly, in most steganographic models, hidden information is fragile, that any small changes made in the container image causes large errors in the revealed information. However, in this system, local disturbance to the container image only causes local errors in the recovered one. Thirdly, this system has very high capacity. It stores a full color image in another, so the ratio is 1:1. It can even store two full color images in one host image with minimal, imperceptible increase in error (2:1). This is the highest encoding density (48 bits per pixel) that we could find in the literature. Fourthly, rather than creating explicit models like most other systems in the literature, this system utilizes deep neural networks to determine where to allocate and how to spread the hidden information in the host image. Finally, due to the high capacity of the system, it is not very feasible to withhold the existence of the hidden information from statistical analyzers, but some methods are described in the article to obscure the hidden image.

Method

The method in the paper uses only implicit methods and consists of three convolutional neural networks which are trained simultaneously and can only work with each other. The first network is called "Prep Network" and picks which features were to be selected from the image to be hidden (secret image) into the host image. The second network, called “Hiding Network” decides how and where these features were to be stored in the host image, and outputs the container image which contains the host and secret images. The last network, “Reveal Network” is used to reveal the secret image from the host image. There were two error terms. Error Term 1 is the pixel-wise difference between the host image and the container image (host and secret image combined). Error Term 2 is the pixel-wise difference between the revealed image and the secret image. Reveal Network is only trained by the error term 2 and does not care about the fidelity of the container image, and the hiding and preparation networks are trained by both of the error terms. Among the improvements the authors suggest, one of the methods they use is shuffling the pixels of the secret image with a key and then hiding them in host images. Thus, when the container image is gone through the reveal network, the output is a shuffled secret image, which looks like noise. However, if the user has the shuffling key, the user can decode the secret image. This is intended to increase the security but drastically increases the training time. Another method they use is hiding two full sized images into one. This results in a bit of accuracy decrease but almost none visually perceivable. These two improvements also decrease the method’s susceptibility to an attack, in which the attacker can blur the container image and subtract it from the original (not blurred) version of container image, which results in the parts of the hidden image being visible.

Experiments/Results

We first looked on GitHub for the official code of the paper but could not find one. However, we could find implementations of an earlier version of this system (Baluja, 2017). But, since there were no details about the architecture of the system provided in this earlier paper, these implementations did not match the current system described in our paper. Therefore, we had to build the architecture ourselves by changing the kernel sizes, depths, activation functions and the hyperparameters. Due to computational limitations, we used Google Colab to run the codes. This proved to be an additional limitation due to the unreliability of the connection and retarded our progress frequently. Therefore, we did not have the resources to train the network long enough or make the resolution high enough to make the errors of the reveal network almost imperceptible (similar to the original paper). Due to these limitations, we could not implement the network with large training sets or using large image resolutions. We used Berkeley Segmentation Dataset (P. Arbelaez et. al., 2011) with 300 images for the training set and 200 images for the test set. We had to shrink the image resolutions (200-by-200-pixel images used in the article) to 80 by 80 pixels. In the final system, after many trial-and-errors, we decided to do the following: We preprocessed the images by normalizing them as a whole within each color channel. We used the Adam optimizer, with a mini batch size of 10 and trained for 3000 epochs. A representative sample from both sets can be seen in Figure 2. As a result, in the training set, we achieved an average error of 8.99 in R, G, B channels (out of 256) for the cover images and 10.42 for the secret images (errors and convergence curve over epochs can be seen in Figure 3 and 4). In contrast, these errors were 9.23 for cover and 13.04

for secret images. Compared to the results of the article (2.4 for cover and 3.4 secret) we had much higher error. This was expected due to the limitations we encountered. However, the visual quality of the output images were decent enough for some of them to the degree that the changes were imperceptible to the naked eye.

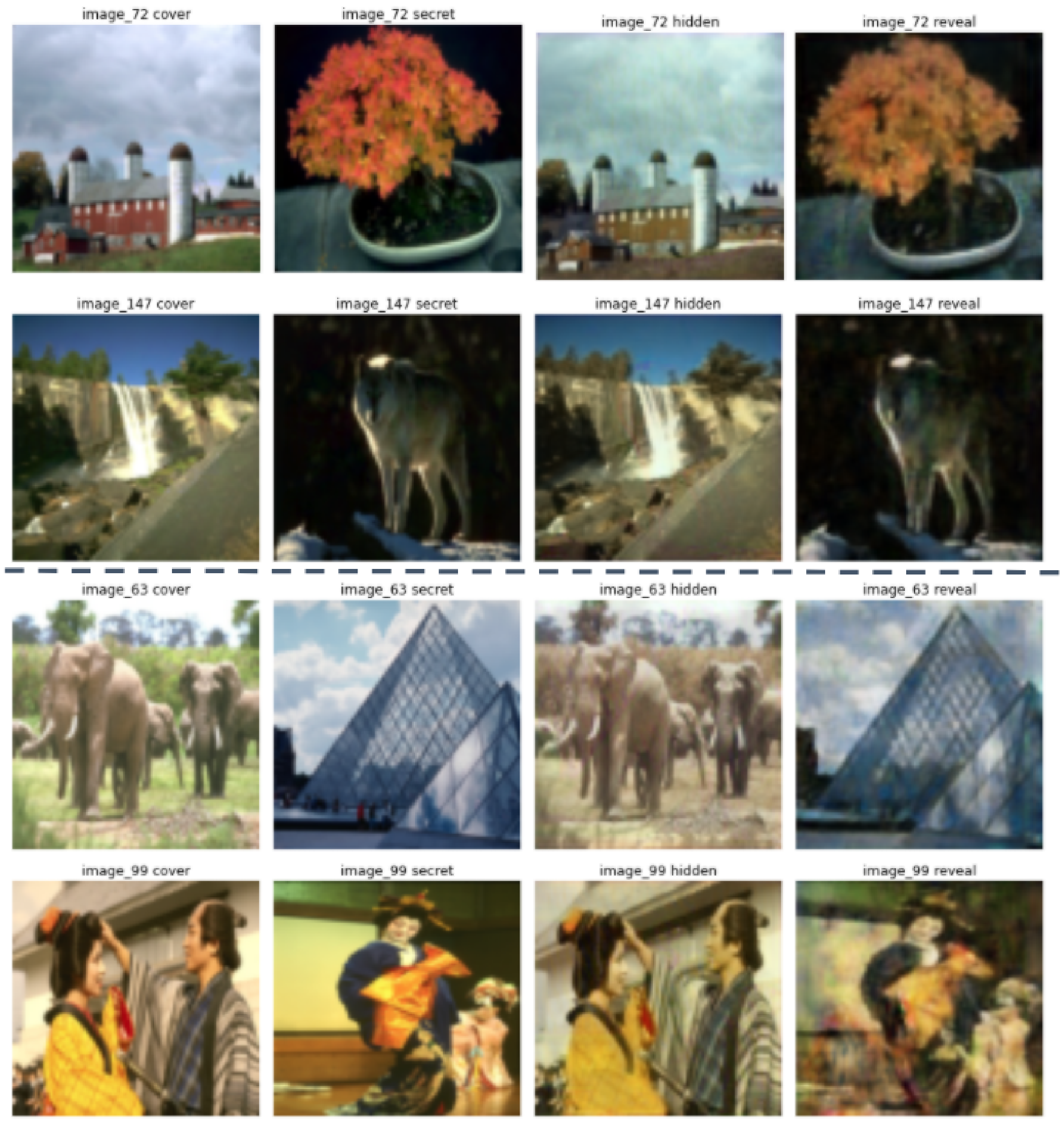

Figure 2: Sample inputs and outputs from our implementation (top two rows: training, bottom two rows: test).

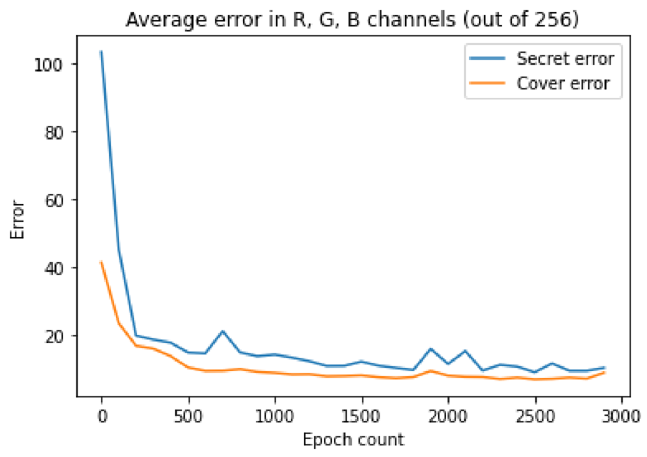

Figure 3: Average error in R, G, B channels across epochs.

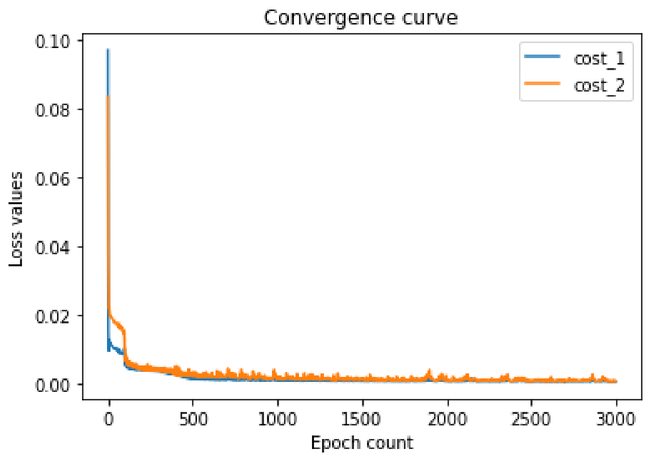

Figure 4: Loss values of each cost function across epochs.

The method described in the paper has several weaknesses and disadvantages. One disadvantage is the fact that the container image has a hidden image is discoverable. Not only this is the case when the original version of the container image is publicly available, however, discovering there is a secret image in the container and even discovering the parts of the secret image is possible even when there is no publicly available non-altered version of container image. It would most likely not be possible to completely hide the secret image due to the high density of information to be hidden. However, we can alleviate this limitation in a few ways. Firstly, we propose a discriminator network, which we will be trained simultaneously with the other three networks (see Figure 5). At each iteration, the discriminator network will take an input, either a container image (which contains a secret image) or the normal version of the same image without any hidden information. The discriminator network will try to guess if the input image has any hidden information or not. Thus, the output of the discriminator network can be fed as an error term to the hiding network, and would encourage it to embed the secret image into the host image in a less detectable way. The discriminator network's effect on the overall loss function would be proportional to its accuracy, and we hypothesise this would decrease its potential slowing down the training process. We also hypothesize that it would be easier to detect an alteration if the shapes in the hidden image were represented in the nearby pixels of the corresponding location in the container image. Therefore, the discriminator network would induce the hiding network to disperse the representation of the shapes to pixels further away. We would be able to test this by manipulating a nearby group of pixels in the input image and tracking their location in the output. Unfortunately, we were not able to implement this hypothesis due to its computational costs, and given that we would need to build this system from the ground up with no prior experience in building deep networks, we would need a lot of trial-and-error to find the best configuration, which would only increase the computational and time costs. Nonetheless, our general hypothesis for this technique is that it would make our network more robust to these types of discriminator networks as it would be trained to be undetectable to them.

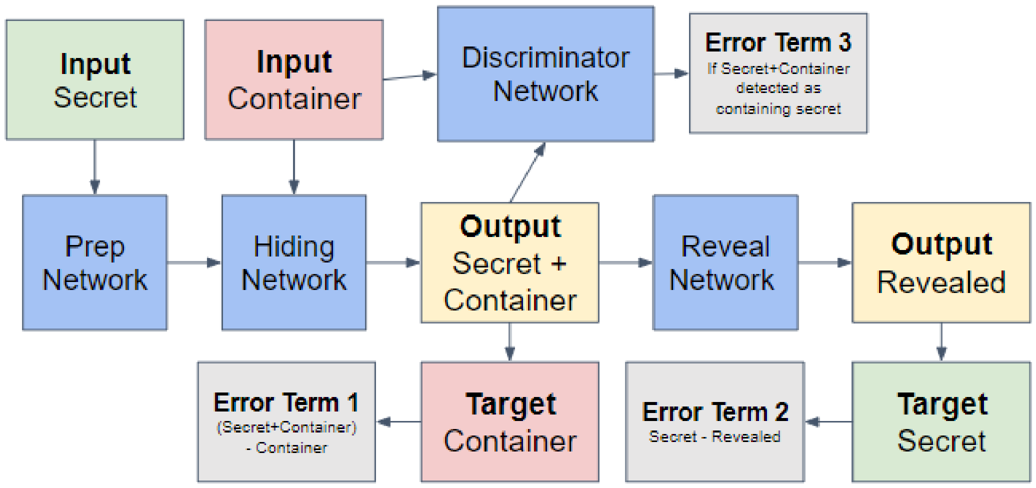

Figure 5: Our proposed network with a discriminator network. Reveal network is only affected by Error Term 2. Prep and Hiding networks are affected by all of the error terms.

Another weakness of the method proposed in the paper is that, after hiding the secret image into the host image (this results in container image), the attacker can blur the container image and subtract it from non-blurred version of the container image pixel-wise and the resulting image would reveal parts of the secret image. This means even when there is no publicly available unaltered version of the host image, it is possible to extract parts of hidden information by a simple technique. To overcome this issue, we hypothesized a blur-subtract operation (or a network) which would generate a metric of how susceptible our network is to this particular attack so that our network becomes more robust to it (see Figure 6). Our investigation of some computationally inexpensive methods like passing several kernels through container image and comparing it with the resultant image from the blur attack did not yield any useful results. Either the loss function did not represent the reality at all or it disrupted the recoverability of the secret images too much. We also propose a neural network solution to this problem. Just like the discriminator network, the blur-subtract network will get the output of the hiding network and the blur-subtract attack performed on that image (positive example) or blur-subtract attack performed on another image (negative example). This way, the network can estimate the susceptibility of the hiding network to a blur-subtract attack and provide this as a loss function. Because of computational constraints such a method brings, we could not implement this solution, but we hypothesize such an additional network would make the network robust against this particular attack.

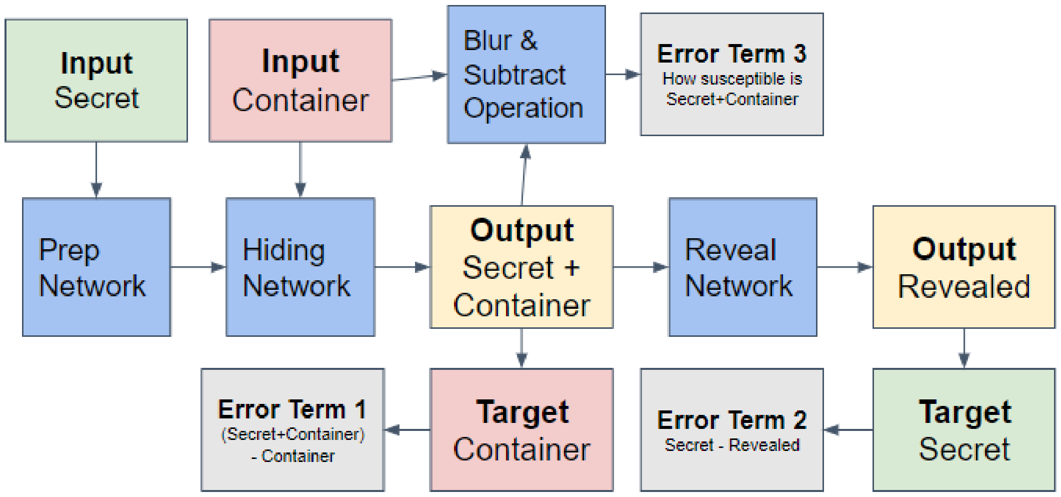
Input Secret
Input Container
Blur & Subtract Operation
Error Term 3 How susceptible is Secret+Container
Prep Network
Hiding Network
Output Secret + Container
Reveal Network
Output Revealed
Error Term 1 (Secret+Container) - Container
Target Container
Error Term 2 Secret - Revealed
Target Secret

Figure 6: Our proposed network with a blur & subtract operation. Reveal network is only affected by Error Term 2. Prep and Hiding networks are affected by all of the error terms

Lastly, we have extended the method from still images domain to the moving images domain with an image compression application. We have created a pipeline in which a GIF or a soundless video is separated into its frames, the last half of the frames are hidden into the first half, effectively reducing to data size around 42%. We used the same network we implemented for still images for this, and our results were satisfactory and the content of the GIFs were represented fully, while the color information was distorted to some degree. We think this can be a real application to increase data efficiency of large GIF hosting websites like Tenor or Giphy, as the computational cost of running the reveal network is low, meaning the users can decrypt the second half of the GIFs with ease.

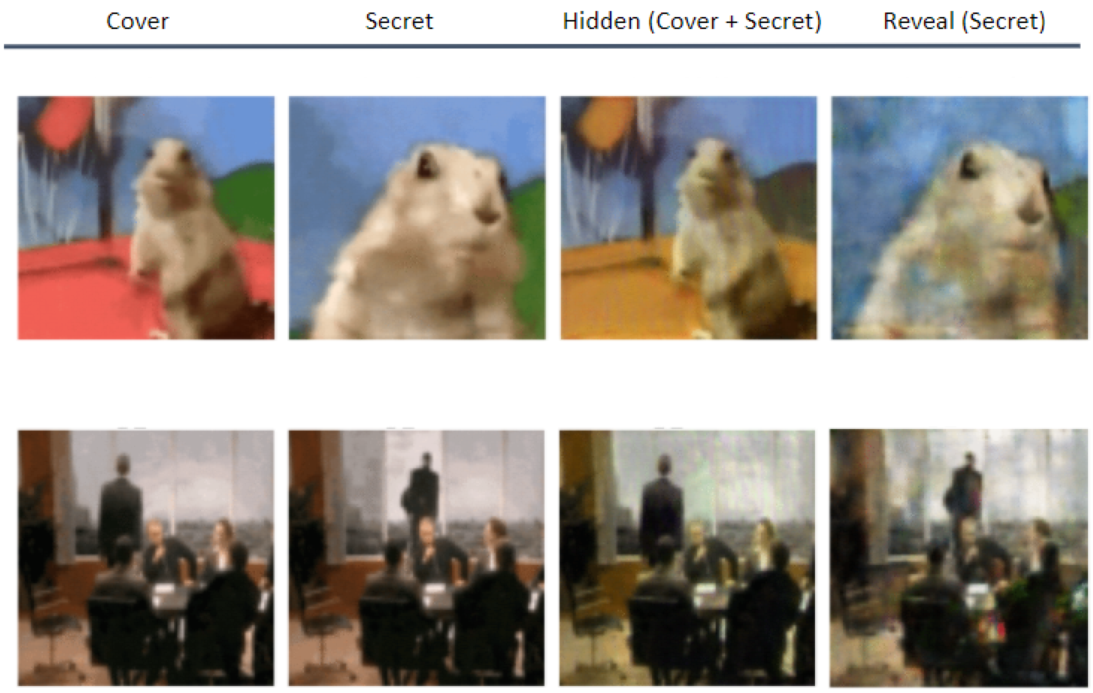

Figure 5: Our implementation of the GIF compression. Animated versions of GIFs can be obtained from the Appendix 1.

To do this project pushed us to learn more about neural networks and libraries like Tensorflow. We learned how to implement a model from a paper, how to create convolutional neural networks and how to use online image databases. We also practiced critically analysing a paper, understanding its drawbacks and limitations and trying to find solutions to the weaknesses of it. Choosing this paper led us to learn a lot about steganography in general and allowed us to appreciate how overreaching applications it can have in various different areas like secure communication, metadata embedding, content delivery and copyright protection.

References

1. Baluja, S. (2020). Hiding Images Within Images. *IEEE transactions on pattern analysis and machine intelligence*, 42(7), 1685-1697. https://doi.org/10.1109/TPAMI.2019.2901877
2. Cheddad, A., Condell, J., Curran, K., & Mc Kevitt, P. (2010). Digital image steganography: Survey and analysis of current methods. *Signal processing*, 90(3), 727-752. https://doi.org/10.1016/j.sigpro.2009.08.010
3. Yedroudj, M., Comby, F., & Chaumont, M. (2018). Yedroudj-net: An efficient CNN for spatial steganalysis. In *IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 2092-2096). https://doi.org/10.1109/ICASSP.2018.8461438
4. Hu, D., Wang, L., Jiang, W., Zheng, S., & Li, B. (2018). A novel image steganography method via deep convolutional generative adversarial networks. *IEEE Access*, 6, 38303-38314. https://doi.org/10.1109/ACCESS.2018.2852771
5. Baluja, S. (2017). Hiding images in plain sight: Deep steganography. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 2066-2076).
6. P. Arbelaez, M. Maire, C. Fowlkes and J. Malik. (2011), Contour Detection and Hierarchical Image Segmentation, IEEE TPAMI, Vol. 33, No. 5, pp. 898-916.

Appendix

1. Link to our presentation: <https://docs.google.com/presentation/d/1NkAtMFvDbTC0eUP1okJyvidrLxfhnmu3MEttyF_3Fyg/edit?usp=sharing>