Detect secret data hidden within digital images

Saurabh Shrinivas Maydeo
Department of Computer Science
Georgia State University
Atlanta, GA 30302
smaydeo1@student.gsu.edu

Soumya Darbha
Department of Computer Science
Georgia State University
Atlanta, GA 30302
sdarbha1@student.gsu.edu

Abstract

Steganography is a technique used to hide the secret data or secret messages within an image to avoid the actual data or message detection. The use of steganography can be combined with the encryption as an extra step of hiding or protecting the data. This technique is commonly used by cyber criminals or even the terrorists. Hence it is essential to have a method that detects the secret data or message hidden in the images and this method is called steganalysis. This method is used to recognize the criminal activities. We will introduce an efficient method for steganalysis in this project.

1 Introduction

Steganography has been used since ages and it is still popular among the cyber criminals. There are many ways to conceal information using the steganography. The most commonly used method is embedding data into digital images. We know that the digital images are stored in the form of pixels which allows some room for the criminals to embed steganographic information within the digital file. Once the targeted user downloads and opens the image file in their computer, the malware is activated. It is also difficult to identify the hidden information with naked eye and processing the large number of images every hour requires an efficient and scalable method. Hence, it is very essential to have a steganalysis technique in the security aspect for monitoring the criminal's or terrorist's communications. This helps to avoid any kind of conflicts, wars, criminal or terrorism activities, thus maintaining peace.

2 Literature Review

The various techniques of steganalysis are:

-Visual steganalysis
-Structural steganalysis
-Statistical steganalysis
-Learning steganalysis

Visual steganalysis is required to identify the differences in the visual artifacts of the images. Structural steganalysis looks at the image file's media format as it changes when there is a hidden message in it. Statistical steganalysis is used to detect the hidden information within images using the statistical models. Learning steganalysis is required to predict if the hidden data is present in the unseen image or not. As the criminals have found better and efficient ways to hide their secret message within the images, the conventional method of detecting the hidden data is no more effective.

3 Proposed Solution

The intelligent, efficient and scalable technique for steganalysis could be use of machine learning. We can train ML models on the images that are processed with different steganography algorithms and have hidden data within them and images without hidden data. These models can learn from these training images by trying to minimize classification error and adjusting model's parameters. **Convolutional Neural Networks seem to be promising solution for steganalysis.**

4 Exploratory Data Analysis

During EDA, we got to know how data looks like and got some insightful findings about the data.
We are provided with the dataset that contains a large number of unaltered images, called the "Cover" image, as well as corresponding examples in which information has been hidden using one of three steganography algorithms (JMiPOD, JUNIWARD, UERD) by Kaggle.
https://www.kaggle.com/c/alaska2-image-steganalysis/data

Files:

- Cover/ contains 75k unaltered images meant for use in training.
- JMiPOD/ contains 75k examples of the JMiPOD algorithm applied to the cover images.
- JUNIWARD/contains 75k examples of the JUNIWARD algorithm applied to the cover images.
- UERD/ contains 75k examples of the UERD algorithm applied to the cover images.
- Test/ contains 5k test set images. These are the images for

which you are predicting.

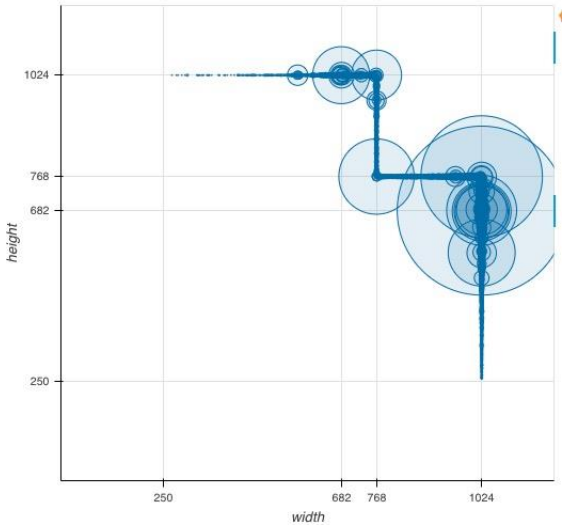

Figure 1: Test image size distribution

From this distribution of test image sizes, we can infer that we need to crop images into a frame of constant dimensions during preprocessing.

After application of these three Steganography algorithms on these images, we got following results

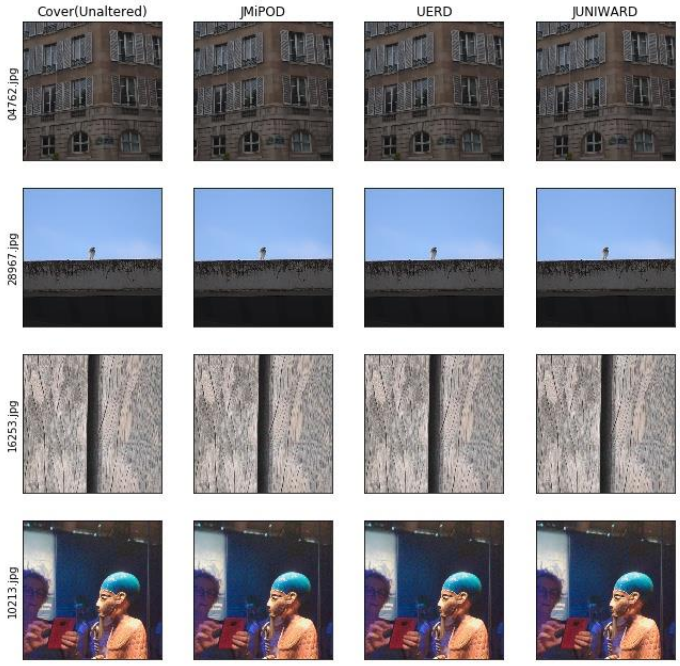

Figure 2: Different steganography algos on same images

5 Data Preprocessing

As we are working on image data, the features that we would be feeding to model are nothing but the pixels in our image.
Each pixel has RGB components with an intensity value ranging between 0 to 255. If we feed this data to model as it is, then there would be a problem of exploding gradients. To avoid this, we normalized these values between 0 and 1 by dividing these values by 255. To get rid of the class imbalance, which is there in our dataset, we performed data augmentation. We synthetically generated new examples for minority class by flipping existing examples belonging to the minority class horizontally (to right, to left) and vertically (to up, to down).

6 Modeling

You can view and run our model on the following Kaggle kernel:
[Image Staganalysis Kaggle kernal]

There are various popular CNN models out there and choosing one of them is not so easy task.

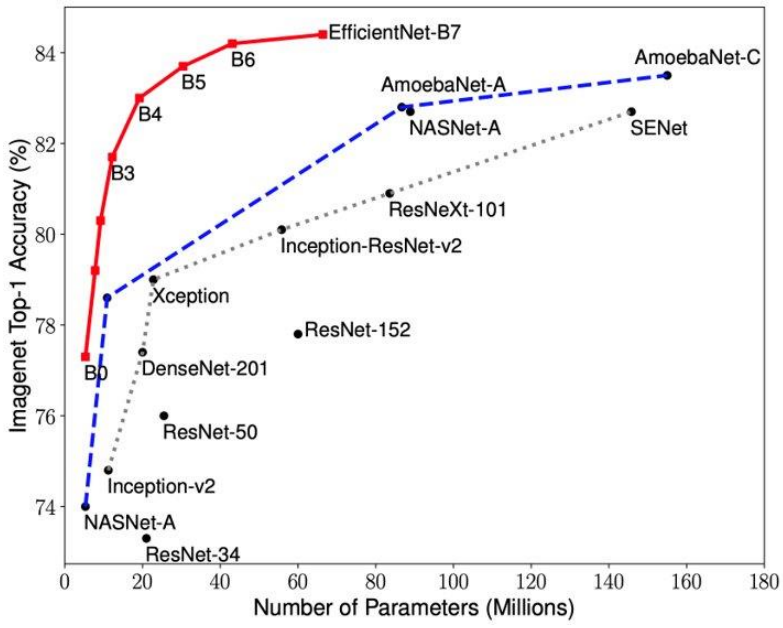

Figure 3: Comparison of CNN models

After conducting an extensive research about the models suitable for this problem, we chose to try out the state-of-the-art EfficientNetB3 model.
EfficientNetB3 is basically a group of CNN models which outperforms most of its predecessors. To explain it in simple

terms, this model's working can be divided into 3 stages – Depth wise and pointwise convolution, Inverse ResNet, linear activation to prevent information loss from ReLU.
This model reduces computation cost significantly and improves training speed. For the type of data that we are dealing with in this problem has a lot of features. To handle this issue, we thought EfficientNet could be a good choice.

It is easier to understand the architecture of the EfficientnNetB3, if we understand modules that are common to all the EfficientNets.

- **Module 1** — This is used as a starting point for the sub-blocks.
- **Module 2** — This is used as a starting point for the first sub-block of all the 7 main blocks except the 1st one.
- **Module 3** — This is connected as a skip connection to all the sub-blocks.
- **Module 4** — This is used for combining the skip connection in the first sub-blocks.
- **Module 5** — Each sub-block is connected to its previous sub-block in a skip connection and they are combined using this module.

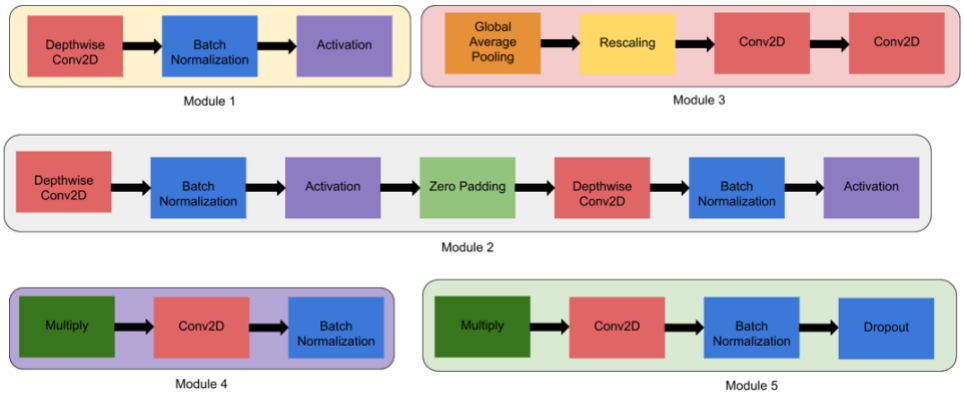

Figure 4: Modules common to all the EfficientNets

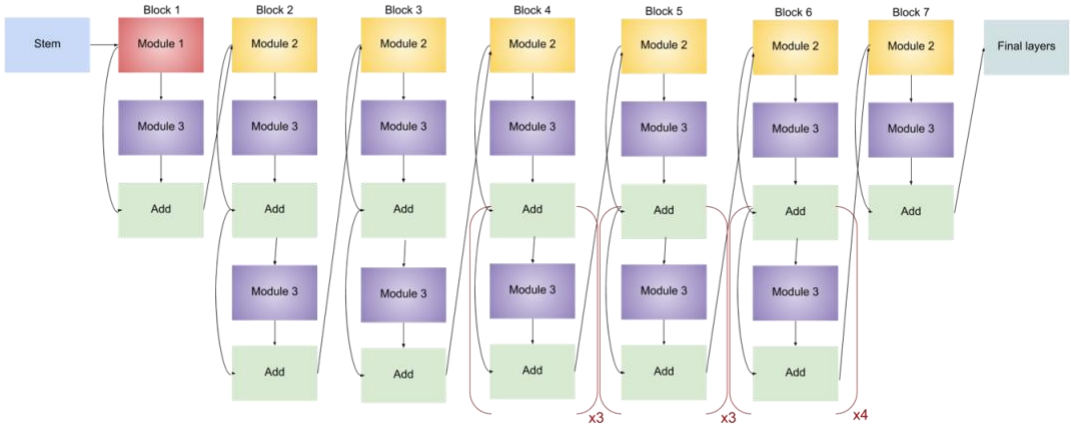

Figure 5: EfficientNet-B3 Architecture

When we connect the modules seen in the figure 4 in a certain way, we get EfficientNetB3 as shown in the figure 5. At the end we have Final Layers which look like following.

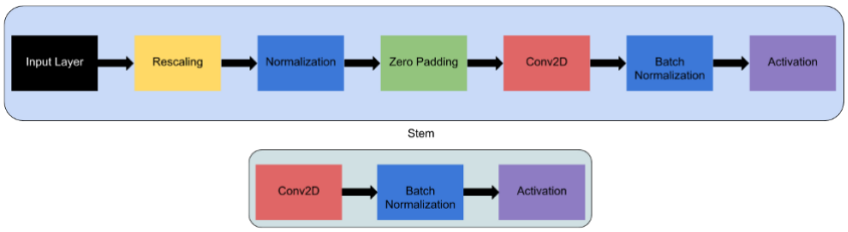

Figure 6: Final Layers

We divided the dataset into 2 parts – Training set and Validation set in such a way that we have 85% of the data for training and 15% of the data for the cross validation.

We fed the training data to EfficientNetB3 first. Then we kept a layer of global average pooling. At the last we kept 1 neuron with sigmoid as an activation function. We used Adam optimizer, binary cross-entropy as a loss function.

We ran this model on the training set for 10 epochs due to limitation of resources.

7 Results

The data in the training set was 85% of the entire data set and the data for the cross-validation was 15% of the entire dataset.

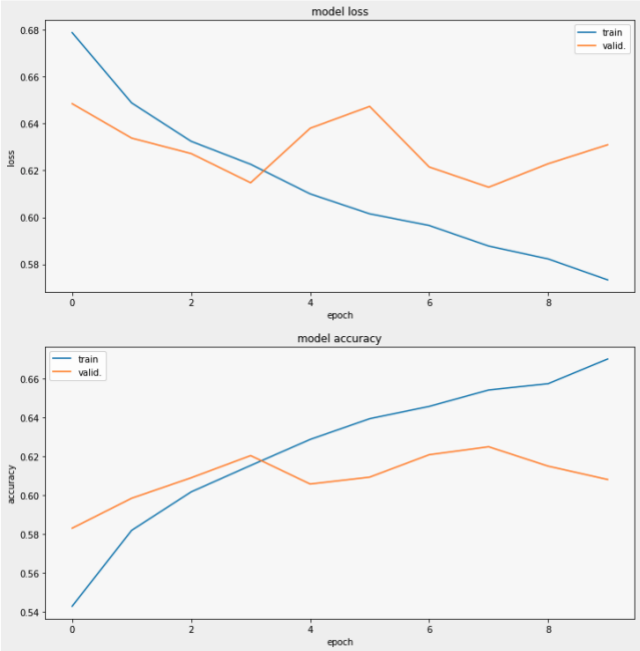

Figure 7: Loss-Accuracy w.r.t epochs for training and validation set

As we can see the accuracy on training and validation set goes on increasing till epoch 3 after which model starts to overfit the data.

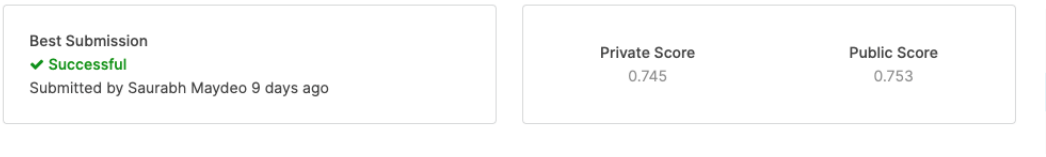

Figure 8: Weighted ROC-AUC scores on Kaggle

The Kaggle is using the weighted ROC-AUC as an evaluation metric to score our submissions. As we can see, our model got 0.745 private score and 0.753 public score.

8 Future Work

We can perform hyperparameter tuning to get better model. Moreover, this model can be deployed using some low-level programming language in switches, gateways where we can monitor all the traffic and see if the images being transferred have some hidden message in them.

9 References

[1] Mingxing Tan 1 Quoc V. Le 1: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

[2] Tang, Y.-H., Jiang, L.-H., He, H.-Q., Dong, W.-Y.: A review on deep learning based image steganalysis.

[3] Jois, A., Tejaswini, L.: Survey on LSB data hiding techniques.

[4] Songtao, W., Zhong, S., Liu, Y.: Deep residual learning for image steganalysis. Multimed.

[5] Singh, A., Singh, H.: An improved LSB based image steganography technique for RGB images.

[6] Farid, H.: Detecting hidden messages using higher-order statistical models.

[7] https://www.kaggle.com/tarunpaparaju/alaska2-steganalysis-efficientnet-b3-pytorch

[8] https://www.kaggle.com/mgornergoogle/getting-started-with-100-flowers-on-tpu

[9] https://www.kaggle.com/xhlulu/flowers-tpu-concise-efficientnet-b7