Image Steganalysis using CNNs

Deep Vision Project

06.08.2020

Supervised by: *Prof. Dr. Björn Ommer*

Authors: **Michael Dorkenwald and Sebastian Gruber**

1 Introduction

1.1 Stegonography

Stegonography is used to initially hide the information. This information can basically be anything such as plain text, another image etc. Ideally the information is hidden in a way that makes it very hard to detect for humans but also for machine learning. There are various techniques in use, in our case JUNIWARD, UERD and JMiPOD were used, however their algorithms are not available publicly. However, all of these methods make use of JPEG image compression. They somehow encrypt the information and the apply it to the high frequencies in the frequency space converted image.

1.2 Motivation

Image Steganalysis is the task of detecting hidden information in images. The technique is largely used in law enforcement, for which it is important to have a low rate of false positive detection. This is because it it quite difficult to decrypt the information hidden in the image and thus would be a wasteful undertake if there was actually nothing to be found in the image.

1.3 JPEG Compression

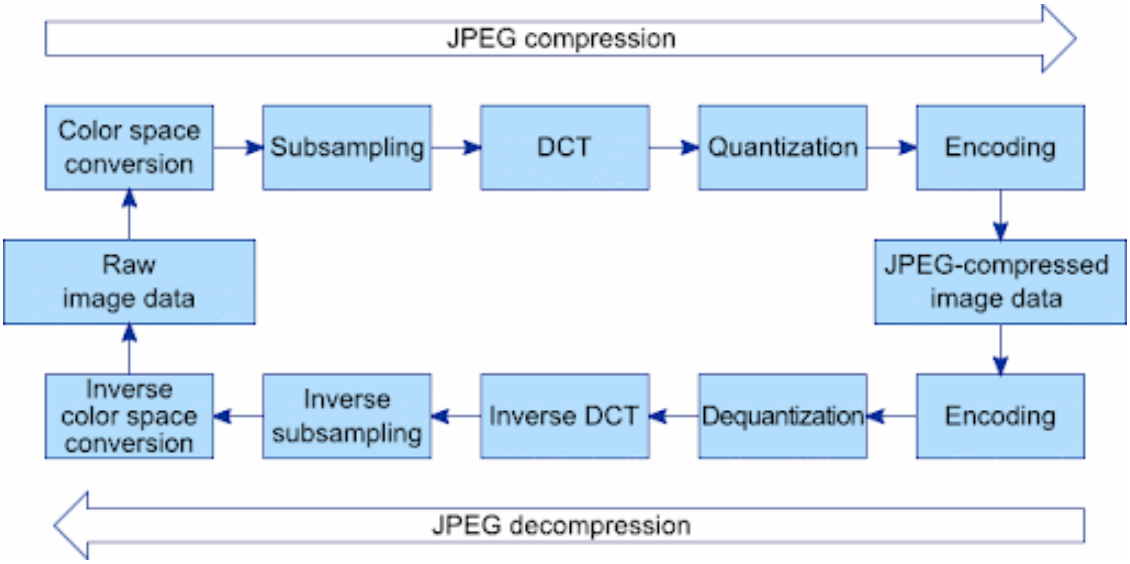


Figure 1: JPEG compression flowchart. The top half indicates the compression process. Raw image data is converted from RGB to YCbCR, then image patches are subsampled, a DCT is applied and multiplied by a quantization table that aims to retain low frequencies and discard high ones. Lastly The image itself is encoded. The loss during this compression mainly stems from the choice of quantization table. The bottom half simply inverts this process to get the (slightly changed) initial image. Retrieved from [1].

In JPEG compression, after color space conversion, a Discrete Cosine Transform (DCT) is applied. It works analogous to the Fourier Transform excluding the sin functions. Thus the image will be represented by frequencies rather than pixel values. Because to the human eye lower frequencies correspond to larger objects, higher frequencies are usually not as important to reconstruct the image up to a certain quality. The three algorithms (JUNIWARD, UERD, JMiPOD) all change the higher frequencies slightly in order to hide the information. If the image is decompressed again from frequency to real space, no change is noticeable to the bare eye. Aside from that, JPEG compression will also decrease image quality by a given amount (this

is done by zeroing out some of the higher frequencies in practice). This lossy compression can also influence how hard it is to detect a payload.

1.4 Task

Our task is to classify, given a single image, if it contains a payload or not. Consequently, our method takes an image as input an returns a binary classification. We aim to have a low rate of false positives while ideally also a high overall accuracy.

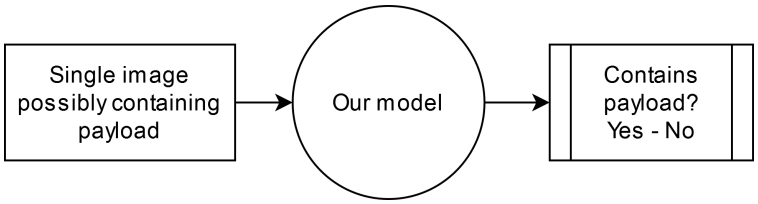
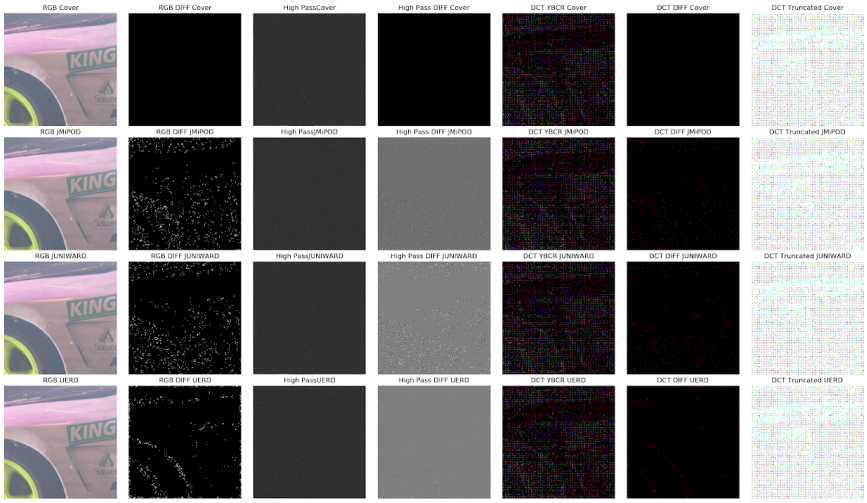


Figure 2: Simple flowchart of our task.

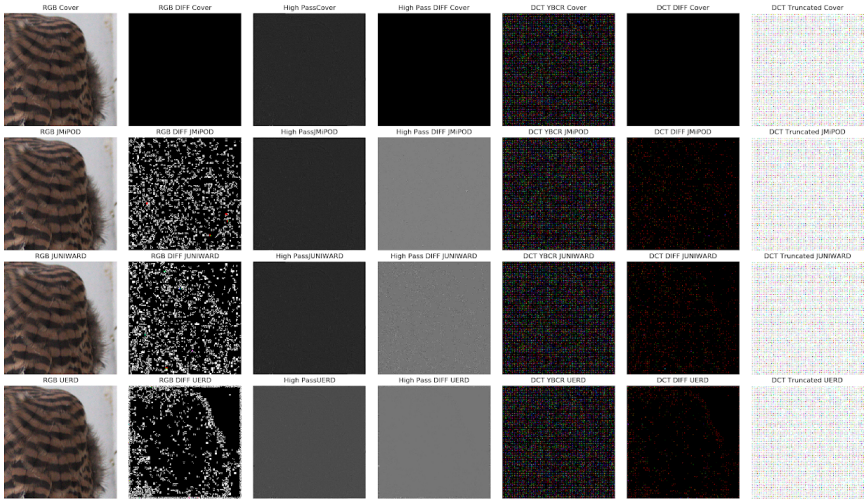
1.5 Data Set

Our data set is from the Alaska2 Image Steganalysis kaggle challenge [2]. It 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 the three mentioned steganography algorithms (JMiPOD, JUNIWARD, UERD). Aside from that, no information is provided. Furthermore:

- Each embedding algorithm is used with the same probability
- The images are all compressed with one of the three following JPEG quality factors: 95, 90 or 75 (these are also used with the same probability)
- The payloads were all similar in size and decryption difficulty



(a) A racing car with part of the wheel visible. As can be seen in the second column, mainly corners and high frequency patterns contain the hidden payload.



(b) Fur containing a periodic pattern. Here the pixel difference images seem rather noisy as most of the image contains high frequency parts due to the texture of the fur.

Figure 3: Visualization of data pairs. From left to right the columns show the original image, pixel difference to the cover image, a high pass filtered version and the respective pixel difference to the cover image, and the last three rows are composed of the DCT coefficients and different ways to visualize them. Row wise, the upper most row shows the cover image containing no payload, and the lower three are the cover image with one of the three stegonography algorithms applied.

2 Methods

In this section we will introduce the methods we used to solve the task at hand.

2.1 Data Distribution Analysis

For better data comprehension we aim to classify also which type of stegonography algorithm and what JPEG compression is used. As these are subsets of the initial binary classification the final result will still be whether a payload is present or not. In table 1 the mean image difference is shown. As these means do vary across different values of compression, it is useful to treat them as separate classes.

	JMiPOD	JUNIWARD	UERD
JPEG 75	45.1	29.5	26.6
JPEG 90	50.5	35.5	35.0
JPEG 95	37.2	35.7	36.3

Table 1: Mean image difference for each of the three stegonography algorithms and JPEG compressions.

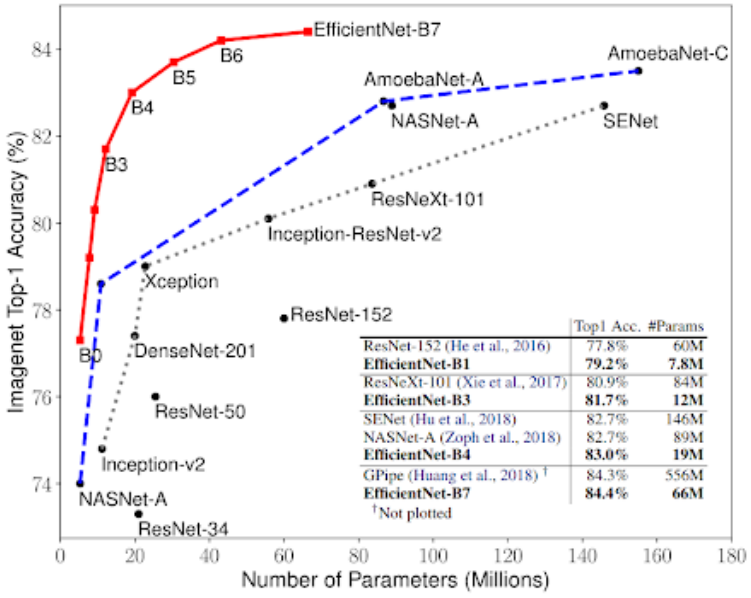
2.2 Preprocessing

In order to tailor our method to the problem, different types of preprocessing were used. Because this problem is highly correlated to high frequencies in the frequency domain of the image, a highpass-filter (attention) was used for some experiments. Also we used the direct DCT coefficients to train, transforming the input image from real to frequency space. Lastly, an attention block as described in [5] was used. The attention block was thought to attain similar qualities as a high pass filter would, but with a higher degree of freedom.

2.3 Model

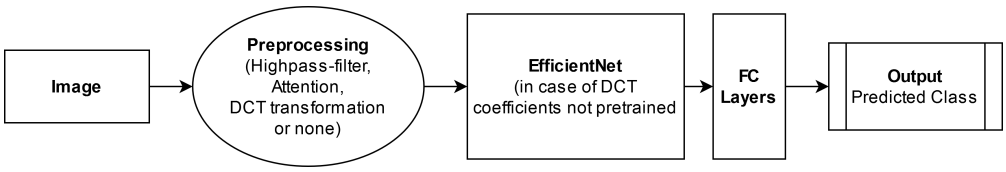
For our model we chose to use EfficientNet [4] as the core architecture. It is a convolutional neural network pretrained on ImageNet. It uses neural architecture search to design its architecture, while being 8.4x smaller and 6.1x faster than best existing networks. Moreover, it transfers well to other data sets, mainly contributed to its smaller amount of parameters and thus better abstraction. In figure 4 a comparison between different EfficientNet architectures in comparison to other popular networks are shown. B0 - B7 indicate the amount of parameters as can be seen from the x-axis of the plot. We mainly used the B0 type, as it transferred best to our task, however also tried versions with more parameters.

Figure 4: EfficientNet in comparison to other state of the art networks. All networks are trained on ImageNet while having different amounts of model parameters. Retrieved from [4]



We used fully connected layers to predict the desired classes from the image embeddings obtained from Efficient Net. Moreover, we used different types of preprocessing, such as highpass-filter and attention-blocks. We also used the DCT coefficients instead of the image data, for we which we then used a blank Efficient Net. The schematic for this is depicted in figure 5.

Figure 5: Overall schematic of our proposed model.



We use cross entropy loss to train the model and also to be able to apply label smoothing:

$$H(p, y) = \sum_{k=1}^{K} -y_k log(p_k) \tag{1}$$

2.4 Label Smoothing

Label smoothing replaces one-hot encoded label vector with a mixture of itself and a uniform distribution of the form:

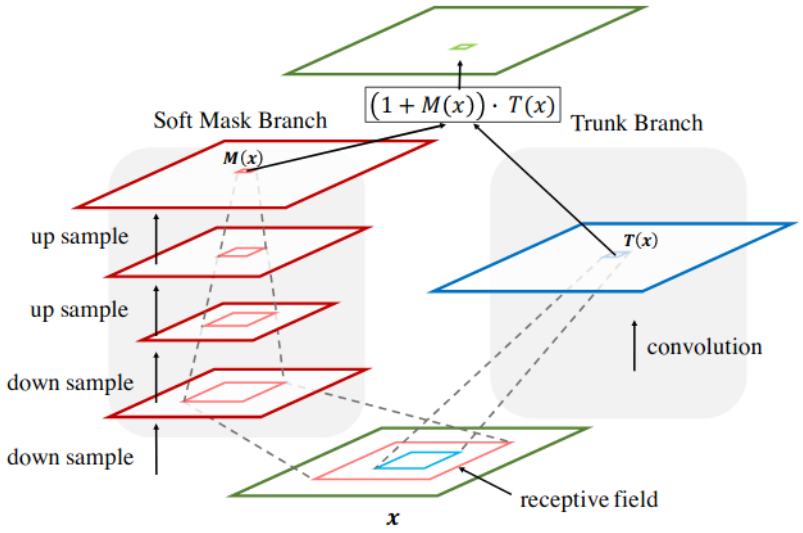
$$y_k^{LS} = y_k(1 - \alpha) + \alpha/K \tag{2}$$

where α is a hyperparameter. This technique improves learning speed and generalization. Furthermore it trains the network to not be overconfident in its predictions [3], which is helpful in avoiding a high false positive rate.

2.5 Attention

Attention in our case is used to improve image classification results, by focusing network attention on areas of interest. This is equivalent to decreasing the perceptive field of the network [5]. This approach can schematically seen in figure 6. However, instead of learning the attention mask (left) we use a highpass-filtered image.

Figure 6: Attention block schematic. Instead of learning the attention mask on the left, we chose to use a highpass-filtered image. Retrieved from [5]



3 Evaluation

In this section we will present how we evaluated our model and the final results. The code for our model, the evaluation and the visualizations can be found at: https://github.com/mdork/Steganalysis-DeepVision-Project.

3.1 Evaluation Metrics

As described in the Kaggle challenge [2], our main evaluation metric is a weighted AUC.
"In order to focus on reliable detection with an emphasis on low false-alarm rate, submissions are evaluated on the weighted AUC. The area between the true positive rate of 0 and 0.4 is weighted 2X, the area between 0.4 and 1 is now weighed (1X). The total area is normalized by the sum of weights such that the final weighted AUC is between 0 and 1."
We chose to also consider the classification accuracy in order to better understand the models performance and its shortcomings. Aside from that, we also tried to qualitatively evaluate our model with attention mapping.

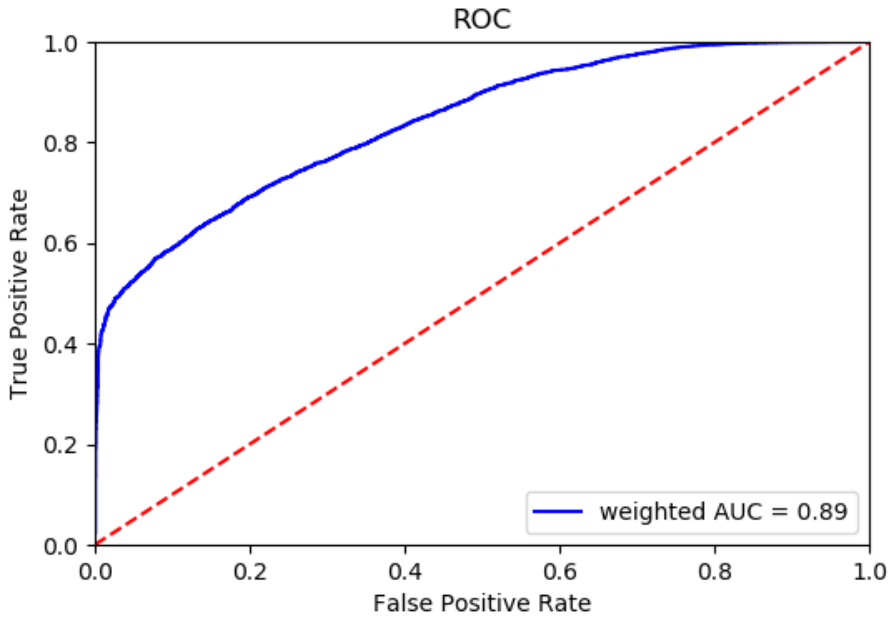
3.2 Overall Results

	Pretrained	Smoothing	Attention	Weighted AUC		Binary Accuracy	
				4 classes	12 classes	4 classes	12 classes
EfficientNet B0	x	x	x	0.58	0.57	54%	53%
EfficientNet DCT	x	x	x	0.56	0.55	53%	54%
EfficientNet B0	✓	x	x	0.876	0.884	72.4%	73.1%
EfficientNet B0	✓	0.05	x	0.870	0.885	71.6%	73.3%
EfficientNet B0	✓	0.1	x	0.865	0.887	71.1%	73.4%
EfficientNet B0	✓	0.2	x	0.843	0.891	69.2%	74.5%
EfficientNet B0	✓	0.2	✓	-	0.877	-	72.8%
EfficientNet B3	✓	0.2	x	-	0.888	-	74.2%

Table 2: Overall results for all trained architecture and class combinations. On the left the different architectures are listed.

The results for all our trained architectures can be seen in table 2. Our best result was achieved for a pretrained EfficientNet B0 with label smoothing with $\alpha = 0.2$ and without attention. We achieved a weighted AUC of 0.891. The AUC curve for this result can be seen in figure 7.

Figure 7: AUC curve for our best result.



3.3 Classification Accuracy per Class

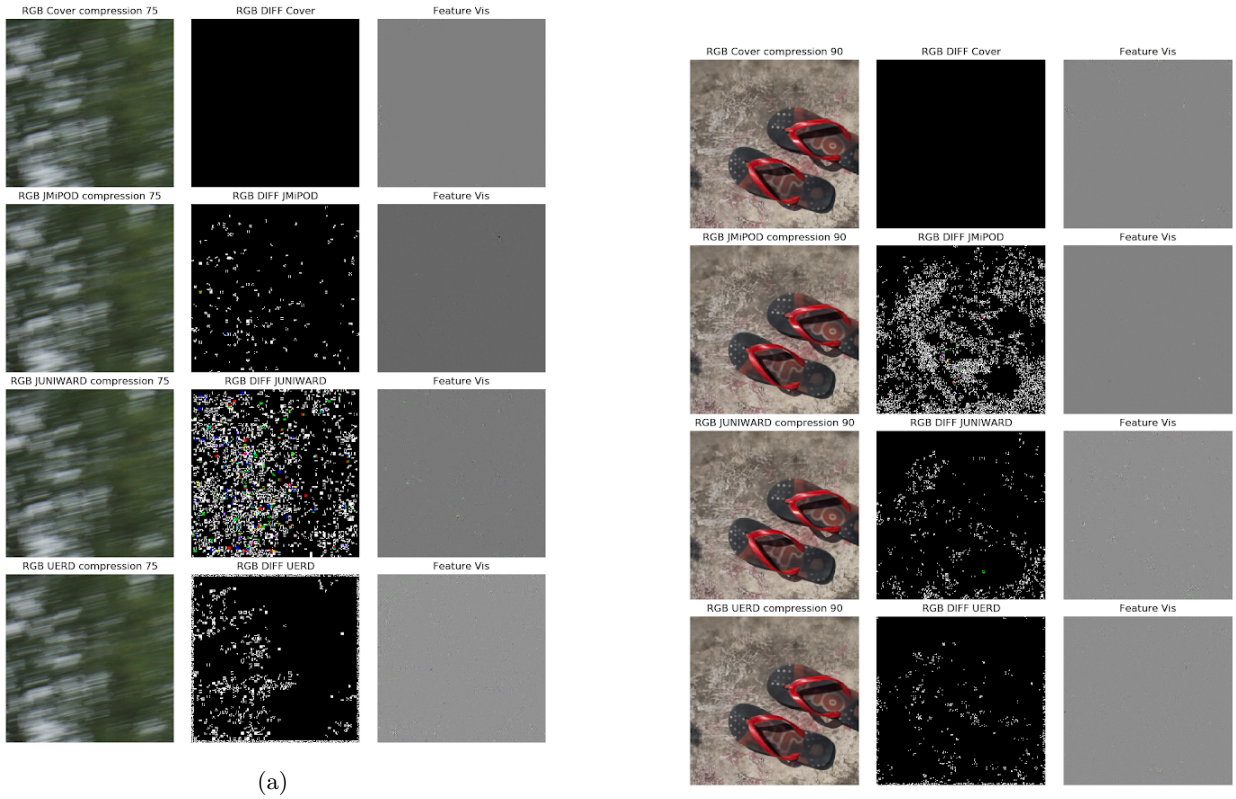
	Cover			JMiPOD			JUNIWARD			UERD		
JPEG quality	75	90	95	75	90	95	75	90	95	75	90	95
Model with 4 classes	93%			20%			61%			63%		
Model with 12 classes	96%	93%	94%	25%	31%	26%	92%	85%	39%	64%	70%	59%

Table 3: Accuracy of all the different classes with respect to the used stegonography algorithm and JPEG encoding quality.

Table 3 shows the classification accuracy for each class in respect to the JPEG encoding quality. For 4 classes, of course all JPEG qualities were treated as one.

3.4 Feature Visualization

The model attention we obtained can be seen in figure 8. As can be seen, this was not necessarily successful, as the model does not seem to pay more attention to certain parts of the image more than others.



(a)

Figure 8: Feature visualizations obtained using attention. The left column shows the original image, the middle one the image pixel difference in respect to the cover image and the right column shows obtained model attention.

4 Discussion

In this section we will discuss in detail how our model performed and try to interpret some of the findings.

4.1 Data Set

As mentioned before, the data set itself consists of images of different quality, scenery and obtained with different modalities. Also, some images contain few big structures and seem thus more noisy in respect to the hidden information. As can be seen in 3, the important parts of the image for steganalysis were sometimes clear edges, but could also be spread over the image as seemingly random noise. Because of this, making the model learn a structure to differentiate normal image noise from fluctuations arising from a payload becomes more difficult.
As seen in table 1, there is an overall trend to also cluster different JPEG qualities together. This is underlined when compared to table 3, where there is a tendency for lower JPEG qualities to be identified more easily. We attribute this to the fact that a lower compression quality will result in more of the high frequencies of the image being zeroed. In consequence, when decompressed, there are less high frequent contributing to the image, meaning the amount of frequencies carrying information of the payload must correspond to more of the image. So for one, there are more real space pixels corresponding to this information and secondly they will likely contribute more to the pixel intensities overall. We argue this analytically makes detection easier.

4.2 Model Performance

Comparing to table 2, the best performance was reached while training on real space images, using the pretrained EfficientNet. using label smoothing and training on 12 classes.
Using a pretrained network was crucial, as otherwise the task was too difficult for the network to grasp. Consequently, training on the DCT coefficients directly also proved to be unsuccessful because we had no pretrained network on a DCT transformed ImageNet. This was a shortcoming mainly based on computational effort, however we would have liked to further pursue this direction.
The attention mechanism we used also did not prove to better our model. We attribute this to the fact that there was no consistency of where exactly the payload information would correspond to on the image pixels. Sometimes they were encoded in rather monochrome surfaces (that however still inhabited small intensity fluctuations), while sometimes most of the information was condensed to e.g. and edge in the image. Because of this, excluding parts of the image could have always served detrimental to the overall model performance.
When comparing the results for 4 and 12 classes, it is apparent that training on 12 classes lead to consistently better results. This is in line with prior argumentation on how there are statistical differences in the data set correlated to JPEG compression quality. This is further underlined when comparing to table 3.
An overall shortcoming can also be deducted from table 3. As can be seen, the classification accuracy for JMiPOD was rather low compared to the other classes. When looking at e.g. figure 3, in terms of pixel difference we could not deduct any special quality to the JMiPOD algorithm when compared to the other two, especially JUNIWARD. A way to combat this would be to train an ensemble of networks, one for each of the stegonography algorithms. We chose not to do this because of computational limitations, however are convinced we could have improved our results with this.

5 Conclusion

We achieved an overall weighted AUC of 0.891. This is a sufficient result for the goals we set out and in comparison to other kaggle submissions. Overall we managed to solve the task of detecting whether a stegonography payload is present in an image. We employed multiple architectures in an ablation study to tailor to our problem and used different evaluations with feature visualizations to gain in depth understanding.

6 Outlook

With more computational power and time we would have liked to train EfficientNet on the DCT transformed ImageNet to be able to directly use the DCT coefficients for our method. We believe this would also enable us to use attention approaches as the payload information would be more localized. Furthermore, we would have trained an ensemble of networks for each of the stegonography algorithms. From a theoretical standpoint, we believe different types of preprocessing should have a significant impact on model performance, as we should be able to distill the needed frequencies better. This is the same reasoning that led us to use a highpass-filter in real space, whereas we would be able to employ better tailored or handcrafted filter in frequency space.

References

[1] Jpeg compression, https://www.graphicsmill.com/docs/gm/working-with-jpeg.htm.

[2] Kaggle alaska2 challenge, https://www.kaggle.com/c/alaska2-image-steganalysis.

[3] Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? *CoRR*, abs/1906.02629, 2019.

[4] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[5] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. *CoRR*, abs/1704.06904, 2017.