To What Extent Can Neural Networks Be Used for the Detection of the F5 Steganographic Algorithm Using a Less Statistically Complex Feature Set Than Previous Methods?

David Jay

Contents

Abstract

As techniques for hiding information (steganography) become increasing more difficult to detect along with much higher resolution carrier images being used, statistical approaches to the detection of steganography are becoming more and more complex. We propose a technique for the detection of the F5 steganographic algorithm by using a simpler set of statistics than previous methods. We do this by classifying the images using a neural network using the simpler statistics as input. We show that, the neural network can successfully detect the presence of the F5 steganographic algorithm regardless of the size of the payload that is encoded in the image and that if the complexity of the statistics is increased, the accuracy increases.

1 Introduction

Steganography is the art of hiding information inside other information. Notable examples of steganography include technologies such as invisible ink and microdots (both traditionally used by spies). More recently, there was evidence of steganography being used by Russian spies in the US to communicate with their handlers back in Russia [Pincus, 2010]. With the increasing popularity of social media and therefore number of images on the internet, the opportunity to use steganography is only going to increase.

Figure 1: *Picture with several types of flowers, used by Richard Murphy to communicate with SVR center. [FBI, 2012]*

Neural networks are algorithms that "learn" patterns in data by loosely modelling neurons in a brain, they perform exceedingly well at learning non-linear tasks. Neural networks have become recently very popular starting with AlexNet [Krizhevsky et al., 2012] at the 2012 ImageNet competition [Russakovsky et al., 2015a]. The availability of large datasets and lots of processing power (GPUs) has allowed neural networks to take off since their discovery in 1958 by Frank Rosenblatt.

Due to the fact that steganalysis (the study of discovering steganography) has become more and more statistically complex, it was decided to explore to what extent a more developed classification algorithm (neural networks) could make up for a statistically less complex feature while trying to detect the presence of the F5 steganographic algorithm [Westfeld, 2001a]. If classification algorithms can indeed make up for a less statistically complex feature set, it could mean that these techniques could be used for more difficult steganographic algorithms. This may allow current techniques, with simple classification algorithms, to become more successful when using a complex classification algorithm.

This paper will look at different aspects of the background and mechanisms behind steganography (including the JPEG compression standard), neural networks and steganalysis. Then the method will be described and justified and the data will be analysed. Finally the results and implications of them will be explained.

2 Literature Review

2.1 Steganography

2.1.1 What is Steganography?

Steganography, which derives from the compound of two Greek words: *stegos* which translates to "covered", and *grafia* which translates to "writing", meaning it literally translates to "covered writing". The field of Steganography comprises of techniques to hide messages inside inconspicuous data so that the message can be transmitted to another party without the presence of a message being detected by a third party. A steganographic message is made up of a cover and a message [Bandyopadhyay et al., 2008]. The message is hidden inside of the cover data which can be anything from images to text to TCP packets [Handel and Sandford, 1996].

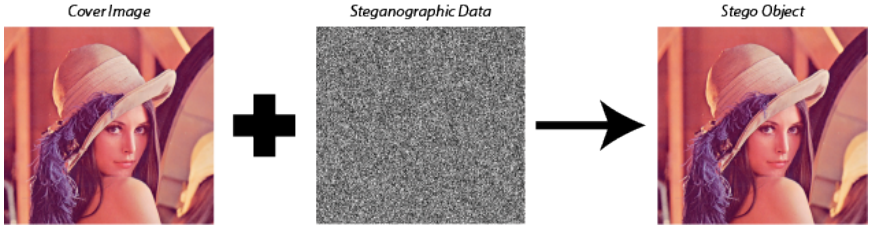

Figure 2: *Example of steganography*

An exemplary steganographic thought experiment was proposed by Simmons, called The Prisoner's Problem [Simmons, 1984], where two inmates communicate in secret to a hatch an escape plan. All of their communication passes through a warden who will throw them in solitary confinement should she suspect any covert communication [Morkel et al., 2005]. The warden has access to all of the messages that the two suspects pass each other. The warden can either be active or passive. An active warden will try and change the data that is being sent if she suspects that information is hidden in it, while a passive warden will observe and take note without blocking the message. This illustrates the main axis upon which steganographic algorithms are judged: undetectability and resistant to container change.

2.1.2 The History of Steganography.

Steganography is far from a new technique; it has been used for centuries with the first known recorded application dating back to the Ancient Greeks. Herodotus writes of a messenger who had a message tattooed onto the back of

his head. When the hair grew back, the messenger was sent thus concealing the presence of the message [Krenn, 2004].

Steganography became immensely popular at the beginning of the 20th Century. During the first world war milk, vinegar and urine were all used to write secret messages, which could be read if the letter was heated up. The microdot was developed by Germany during the First and Second World Wars. The idea behind it is to miniaturise a photo as much as possible so that it can be stored in a typographical dot (hence the name microdot). It was often used to hide vast amounts of information (normally in the full stop of a letter), that was then collected by a spy, so that it could be transmitted back to their home country. [Kipper, 2003]

More recently, steganography became increasingly popular as a deniable method to store information. It has also been used to complement cryptographic techniques to make data even more secure at it travels, allowing another layer before the data is found and decrypted. For this reason it has been speculated that terrorist organisations and governments may use it to transmit sensitive data [Krenn, 2004].

In addition to this, some steganographic techniques have been implemented in watermarking and fingerprinting to allow companies to track their documents or enforce their copyright. Even though watermarking does not necessarily conceal the knowledge of the hidden information, aside from the human senses [Johnson and Jajodia, 1998], steganographic techniques are often still used.

Throughout the history of steganography and image steganography people have tried to optimise their algorithms to increase their effectiveness. This, coupled with the large number of image types and different data compression algorithms, has led to a plethora of types of image steganography.

2.1.3 Types of Image Steganography

Image steganography techniques can be divided into two groups: Image Domain and Transform Domain [Silman, 2001].

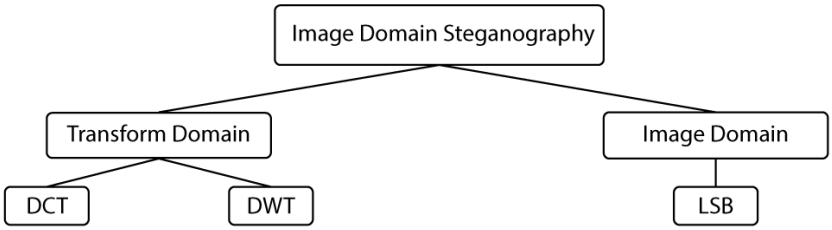

Figure 3: *Types of Image Stegonography*

2.1.3.1 Image Domain

Image domain techniques hide the messages in the pixel values themselves. The image formats most suitable for image domain steganography are lossless and the techniques are typically dependent on the image format [Abraham et al., 2004].

2.1.3.1.1 Least Significant Bit

Least significant bit (LSB) steganography is the most popular image (or spatial) domain technique and the easiest to implement. The LSB refers to the lowest significant bit of an image pixel. The concept of LSB exploits the fact that the level of precision in many image formats is far greater than that perceivable by the average human [Gupta et al., 2012]. This means that if we encode the messages in the least significant bit of the pixel values it will not be noticed by the human eye as the pixel value will change imperceptibly.

2.1.3.1.2 Transform Domain

Transform domain is a more complex way of hiding information in an image. Most of today's strong steganographic systems today operate within the transform domain. Transform domain have an advantage over spatial domain techniques as they hide information in areas of the image that are less exposed to compression, cropping and image processing [Raval et al., 2017]. This paper will mainly focus on discrete cosine transform techniques and only be a brief mention of discrete wavelet transform will be made.

2.1.3.1.3 Discrete Wavelet Transform (DWT)

Wavelet transform is used to convert a spacial or image domain into a transform domain. The wavelet transform is useful because it separates high frequency and low frequency information on a pixel by pixel basis [Reddy and Raja, 2009].

2.1.3.1.4 Discrete Cosine Transform (DCT)

This steganographic technique is almost exclusively used with JPEG images. During the DCT phase of JPEG compression, rounding errors occur in the coefficient data that are not noticeable [Johnson and Jajodia, 1998]. This property can be used to hide messages. Using LSB insertion the message can be embedded into the least significant bits of the DCT coefficients [Krenn, 2004] just the Quantization stage of JPEG compression.

2.1.4 Examples of Steganographic Algorithms

2.1.4.1 Jsteg

Jsteg hides data inside images stored in the JFIF format of the JPEG standard. Before Jsteg, JPEG steganography was seen as impossible due to the lossy

nature of the file. Jsteg works by recognising that JPEG encoding is split into lossy and non-lossy stages. Therefore Jsteg can insert the steganographic data into the image data between the lossy and non-lossy steps without risking corruption. [Upham, 1997]

The storage effectiveness for this steganographic technique is reasonable, but not astounding. An N kilobyte data file fits in the image when the resulting JPEG/JFIF file is around C*N kilobytes, where C ranges from eight to ten. [Upham, 1997]

2.1.4.2 F5

The F5 algorithm, like Jsteg, hides data inside images stored in the JFIF format of the JPEG standard. In many cases, an embedded message does not require the full capacity of an image. Therefore, a part of the file remains unused. The F5 algorithm encodes data in the image so that the density of data is as regular as possible. This means that the embedding density should be the same everywhere [Westfeld, 2001b]. F5 can hide a lot more information in an image than Jsteg.

2.2 JPEG

As has been mentioned previously, this paper is going to focus on detecting DCT-based steganography. As JPEG is the most common image compression format that uses DCT techniques as part of its compression. This section will focus on how JPEG uses DCT to compress images and explain the purpose of the discrete cosine transform.

2.2.1 What is JPEG/JFIF?

Joint Photographic Experts Group (JPEG) is an image compression standard created in 1992 by the JPEG committee [Wallace, 1992]. The JPEG standard can be used to compress either full-colour (3 channel) or grayscale (1 channel) images [Wallace, 1992].

JPEG images are almost always compressed using lossy compression (lossy file compression results in lost data and quality from the original version [Christensson, 2006]), although the JPEG standard supports a lossless encoding it is not very common. The lossy compression allows JPEG images to be very small while retaining good quality. For this reason, JPEG images are the most commonly saved format by digital cameras and have become very popular with the rise of the internet.

JPEG File Interchange Format (JFIF) is an image file format standard. Image data inside of JFIF files are compressed using the JPEG standard hence JPEG and JFIF are often used interchangeably.

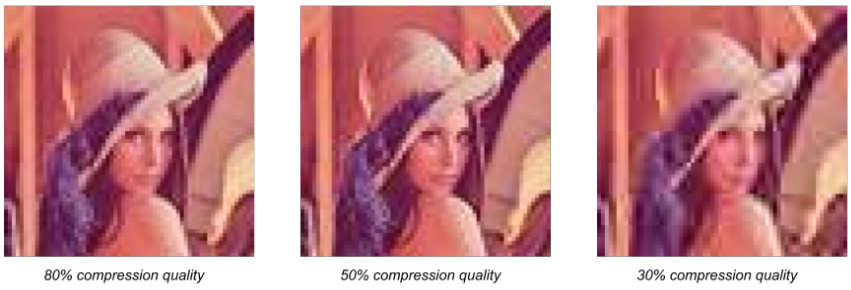

Figure 4: *Different Compression Levels of a JPEG image*

2.2.2 How does JPEG Compression Work?

JPEG compression is a complicated and multi step process so this explanation will cover standard full colour images and therefore ignores progressive encoding

and other more complex encoding techniques. Firstly, the image is converted from Red, Green and Blue (RGB) colour channels to luminance (Y) and two chrominance channels: Blue/Yellow (Cb) and Red/Green (Cr).

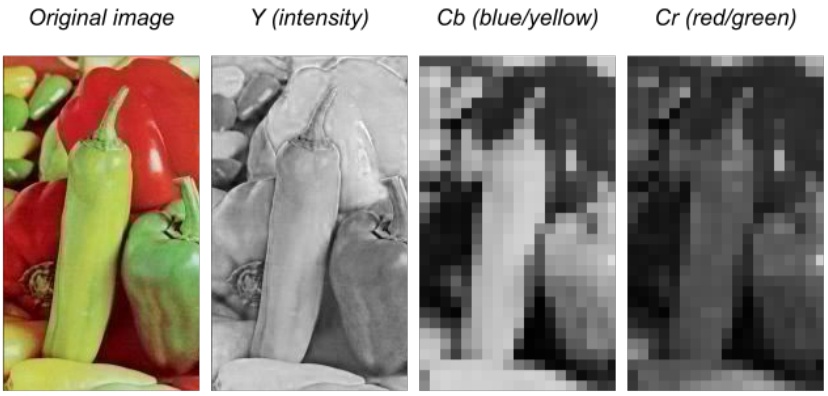

Figure 5: *Demonstration of Y Cb Cr splitting of an image [Guillermito, 2004].*

Due to the fact that the human eye is more sensitive to luminosity than it is to chrominance [Winkler et al., 2001] a step known as *downsampling* can be taken which takes adjacent pixels and combines them into one to remove information that is unnecessary for the human eye [Kerr, 2005].

The Discrete Cosine Transform (DCT) and the Quantisation process are the main parts of the compression. The channel is broken into 8x8 blocks of coefficients. The DCT process is then applied to each block and each block is then quantized. The DCT works by separating images into parts of differing frequencies [Cabeen and Gent, 1998]. The quantization process is the main reason why JPEG images are "lossy" as accuracy of the DCT value is being reduced. Quantization is performed on the DCT coefficients by dividing the matrix by a quantisation matrix that has been predetermined for maximum performance and then rounding the values to the nearest integer value. The quantised values are where most Transform Domain steganography tools store their data.

Next, the matrix is reordered in a zigzag ordering pattern 5 to allow for the maximum number of zeroes next to each other to allow for better compression during the next step of compression. Finally this process is repeated on each of the Y Cb Cr layers of the image. ([Guillermito, 2004] including the image)

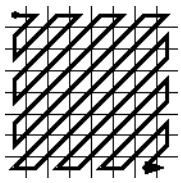

Figure 6: *The zig-zag ordering of the 8x8 block of quantised DCT coefficients.*

Finally, RLE (run length encoding) is used to compress the high frequency coefficients (such as the large number of zeroes) DPCM (Differential Pulse Code Modulation) is used to compress low frequency coefficients. Huffman coding is then used to compress everything [Wallace, 1992].

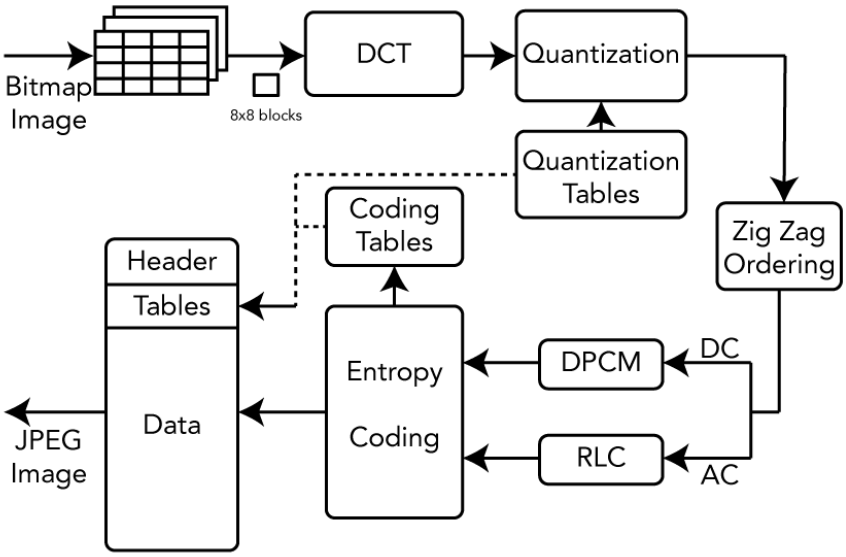

Figure 7: *Flow diagram detailing the steps of compression of a JPEG image.*

2.3 Artificial Neural Networks

Transform domain steganography is notoriously difficult to detect. For this reason this paper proposes the use of artificial neural networks to act as a classification tool.

2.3.1 What are Artificial Neural Networks?

Artificial Neural Networks, also referred simply as neural networks (NN), are built upon simple signal processing elements that are connected together [Berger, 2016] These elements form a complex and non-linear system that allow neural networks to excel at classification tasks. There are three main NN topologies: supervised learning, unsupervised learning and reinforcement learning. We will focus on supervised learning.

2.3.2 How do Feed-Forward Neural Networks Work?

Neural networks are organised in layers. These layers are made up of interconnected nodes which each apply a mathematical function called an activation function to the input. The connections are weighted so that different nodes gain different importances. These weights are what are optimised through the learning process.

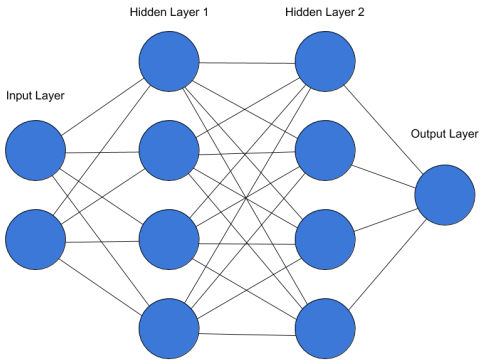

Figure 8: *An example of a simple multi-layer feedforward network.*

2.3.2.1 The Perceptron

A perceptron is a type of linear classifier used for supervised learning of binary classifiers. It was invented by Frank Rosenblatt in 1957 [Rosenblatt, 1957]. The perceptron takes in multiple binary inputs, x_1, x_2, x_3 in the case of 9, and produce a binary output.

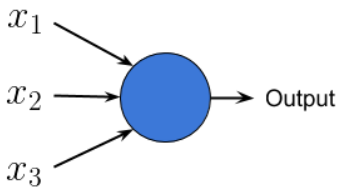

Figure 9: *An example of a 3 input perceptron.*

Weights are used to train the network and signify the importance of that input into the node. The neuron's binary output is determined by the weighted sum being less than or equal to, or greater than a threshold value. The threshold value can be moved and called a bias instead.

$$\text{output} = \begin{cases} 0 & \text{if} \sum_j w_j x_j + b \leq 0 \\ 1 & \text{if} \sum_j w_j x_j + b > 0 \end{cases} \text{Where b is the bias}$$

Layers of perceptrons which act together produce significantly more complex outputs than a single layer as each layer of perceptrons is making a decision based on the decisions of the previous layers of perceptrons. This means that decisions become more and more complex and the number of layers increase [Nielsen, 2015].

2.3.2.2 ReLU Neuron

As we want small changes to be made to the output of the neurons with small changes in the input, we add a nonlinear function $\xi(mx + c)$. Where ξ is the ReLU function which is defined by:

$$\xi(x) = \max(0, x)$$

Figure 10: *[Krizhevsky et al., 2012]*

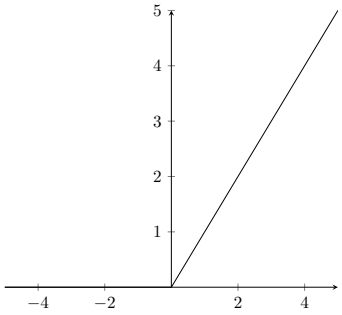

Figure 11: *Graph of* ξ

2.3.2.3 Dropout

Dropout is a technique for reducing overfitting in neural networks (regularising). With limited training data, however, many of these complicated relationships will be the result of sampling noise, so they will exist in the training set but not in real test data even if it is drawn from the same distribution. This leads to overfitting and many methods have been developed for reducing it [Srivastava et al., 2014]. Dropout work by, at each training stage, individual nodes are either dropped out of the net so that a reduced network is left as illustrated by Figure 12 [Budhiraja, 2016].

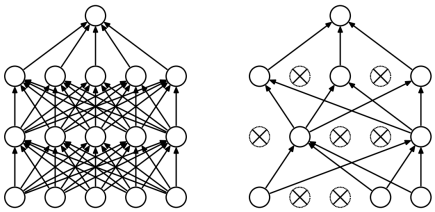

Figure 12: *An example of a thinned neural network produced by applying dropout. Crossed units have been dropped [Srivastava et al., 2014]*

2.3.2.4 Softmax Classification

The softmax classifier is applied to the output layer of the network. Its purpose is to return the probability of each class label.

2.3.2.5 Cost Function

The purpose of the loss or cost function is to gain a measurement of how well the neural network is doing. An example cost function can be defined as:

$$C(w,b) = \frac{1}{2n} \sum_x ||y(x) - a||^2$$

Figure 13: *Where $||v||$ is the length function for vector v [Nielsen, 2015]*

This particular cost function is often known as mean squared error.

2.3.2.6 Backward Propagation of Errors

Backward propagation of errors also known as backpropagation is an algorithm for supervised learning of neural networks using gradient descent. Given an error function and a neural network, backpropagation calculates the gradient of the error function with respect to the network's weights [McGonagle et al., nd]. Backpropagation is used to find a local minimum of the cost function of the network.

2.3.3 Tools

2.3.3.1 Tensorflow

TensorFlow is an open source machine learning library, which means that most of the mathematical computation of machine learning is handled by TensorFlow. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research (from: [Tensorflow, 2015]). Tensorflow is one of the industry standard machine learning libraries. For example, Ebay, Dropbox, Google, DeepMind use Tensorflow [Tensorflow, 2015] for both research and deployed machine learning.

2.3.3.2 Keras

Keras is a high-level neural network API, written in Python and capable of running on top of TensorFlow. It was developed with a focus on enabling fast experimentation. [Keras, 2018]. Using Keras allows for a very fast experimental cycle which is key to good research and the reason why it was used in this study. Keras is open source and is supported by Tensorflow.

2.3.4 Overfitting

Overfitting happens when a machine learning model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the models ability to generalise [Brownlee, 2017].

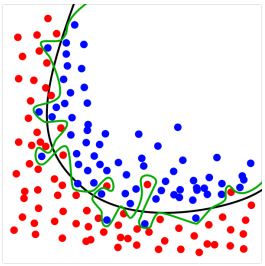

Figure 14: *The green line represents an overfitted model and the black line represents a regularised model. While the green line best follows the training data, it is too dependent on that data and it is likely to have a higher error rate on new unseen data, compared to the black line [Wikipedia, 2018].*

An example of overfitting is shown in Figure 14. Many techniques such as dropout (as discussed in section 2.3.2.3) can be used to combat overfitting.

2.3.5 Convolutional Neural Networks

2.3.5.1 What is a Convolutional Neural Network?

A convolutional neural network is a neural network that uses convolution operations to extract features of the input data. Because of the nature of convolutional operators, convolutional neural networks preserve spatial data within the data and therefore have become very popular in the image classification world [Russakovsky et al., 2015b].

2.3.5.2 Structure of a Convolutional Neural Network.

The main parts of a convolutional neural network are:

1. Convolution layers
2. Non-linearity (In our case ReLU)
3. Pooling layers
4. Classification (Standard fully connected layers as described previously)

This basic classes of layers are the building block of every convolutional neural network [cs231n, 2017]. As convolutional neural networks are mainly applied to images, network structures have 3 dimensions (width, height, depth).

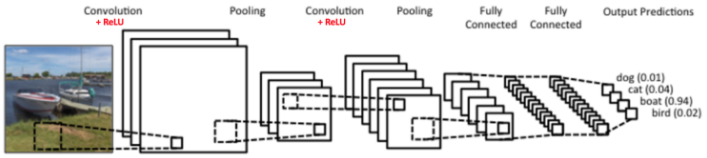

Figure 15: *A simple convolutional neural network for image classification. [Karn, 2017]*

2.3.5.3 Convolutional layer

This is where most of the computational "heavy lifting" is done in the network. The primary purpose of the convolution in our network is to extract features from the image. Every image can be considered as a matrix of pixels. There for the convolution of that matrix can be computed using a smaller matrix as our kernel.

(a) *5x5 data matrix.*

(b) *3x3 filter matrix.*

Figure 16: *An exaple of a matrix to convolute (5x5) and a filter (3x3)*

Consider this 5x5 data matrix and 3x3 convolutional matrix [Karn, 2017] figure 16. The convolution of the 5x5 data matrix and the 3x3 convolution matrix (known as the filter or kernel) can be computed by passing the 3x3 matrix over the 5x5 data matrix. The resulting product is known as the feature map.

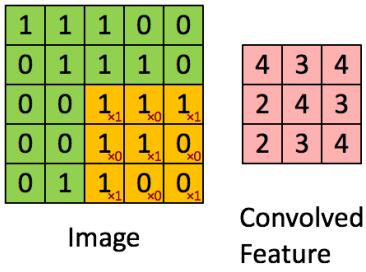

Figure 17: *An example of the convolution process. [UFLDL, 2013]*

2.3.5.4 Pooling

Spatial Pooling layers progressively reduce the spatial size of the data matrix to reduce the amount of parameters and computation in the network. The MAX operation is commonly used as a mathematical operation to pool the data [cs231n, 2017]. Different pool sizes can be used to downsample the data even rapidly but at the cost of fine tuning.

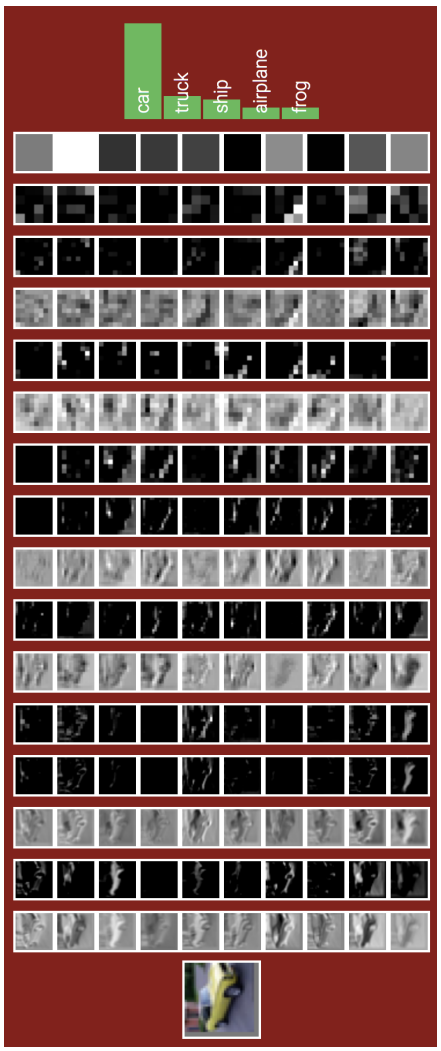

Figure 18: *An example of a simple Convolutional Neural Network in action. [Karn, 2017]*

The combination of all of these elements allows convolutional neural networks to learn from large datasets while still keeping positional elements leading to their prominence in image classification tasks.

2.4 Steganalysis

Steganalysis is the process of detecting messages hidden with steganography techniques. The goal of steganalysis is to identify whether data sets have a message encoded in them and, if possible, recover the message. As more techniques of hiding information are developed and refined, the techniques to detect them also develop.

2.4.1 Techniques for Steganalysis

Steganalysis is generally tackled using statistical approaches. Spectrum analysis can be used on lossless files but becomes less effective when dealing with lossy data formats. To combat this, when dealing with lossy formats steganalysis usually looks for inconsistencies in the way that the data has been compressed.

In these cases, methods of steganalysis look for specific signatures in the images. This can refer to a certain statistical property consistent in images that are encoded using a specific program. For example, when plotting the non-zero DCT coefficients of a JPEG image, the result is a relatively smooth graph. However, if we plot the non-zero DCT coefficients created with Jsteg (2.1.4.1) they produce a more erratic graph [Cole, 1997].

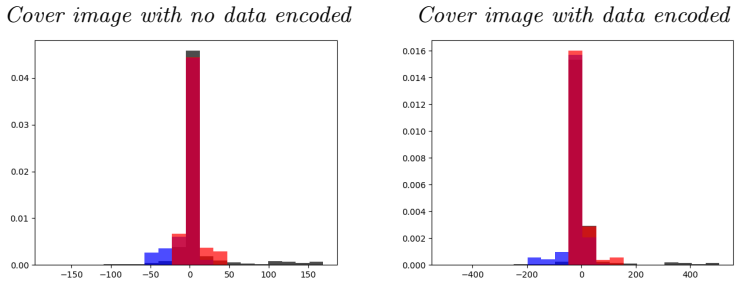

Figure 19: *DCT histograms from an image with and without data encoded into it.*

2.4.1.1 Types of Steganographic attack

- **Stego Only Attack:** Only the stego-object is available to analyse.
- **Known Cover Attack:** The original cover image is available and can be compared with the stego-object.
- **Known Message Attack:** The message is available. N.B.: The data gathered by the knowledge of the message may be of negligible use and

often Known Message Attacks are not distinguished from Stego Only Attacks.

- **Chosen Stego Attack:** The steganographic tool used to create the stego-object is known.
- **Known Stego Attack:** The steganographic tool is known and both the cover image and stego-object are available.

[Richer, 2003]

2.4.2 Steganalysis using Neural Networks

As neural networks are exceedingly good at approximations of nonlinear functions, they can be used effectively to assign importance to different statistical features. Several spatial measures based on DCT are selected to allow the network to use several measures to increase flexibility and accuracy [Shaohui et al., 2003].

3 Discussion

Due to the increasing complexity of statistical models to combat steganography, it seemed that neural networks were very well suited for the detection of complex steganographic algorithms because of their traditionally very good performance in non-linear classification tasks (2.3.2.1). As the number of layers in a network increase, the function that the network can model becomes increasingly more complex as each successive layer uses the output from the previous layer. The steganographic algorithm that was chosen to work on was the F5 algorithm [Westfeld, 2001a] due to its traditionally complex methods to break the algorithm (estimating the DCT values of the cover image to then be able to see discrepancies between the stego image and the original image) [Fridrich et al., 2002].

Initially a neural network using a simple feature set was trained and tested on the Caltech101 dataset [Fei-Fei et al., 2007] which, in preliminary tests, yielded a mean accuracy of 97%. It was decided that this dataset was too small (roughly 300 x 200 pixels) and the amount of data that could feasibly be hidden inside the images without saturating the image with data was too little to justify the F5 algorithm. Instead, it was decided that the the personal part of the Columbia Photographic Images and Photorealistic Computer Graphics Dataset (PIM Dataset) [Ng et al., 2005] should be used due to the wide variety of image subjects and high capture quality. These images were cropped to 640 x 480 pixels (cropped with reference to the centre of the image) to reduce the time to parse and analyse the images. Once we did this, the accuracy of our neural network dropped substantially to about 75%. Therefore a new network based on the same ideas as the original network was created to perform more effectively but with a more complex feature set.

In this section, both the "Old Neural Network" and the "New Neural Network" were described and evaluated on the PIM dataset. Each network section contain: a flow diagram of the topology (overall structure) of the network; which allows for easy comparing between the two networks; a description of the features that were extracted; the results of evaluation of the network (the process of which is explained in the next section) and a sample of training data to identify overfitting in the network.

3.1 Method

Python was used as the main language for all pre-processing, parsing and neural network operations due to it's ubiquity in the machine learning community (due to its incredible flexibility and very high development speed), deep learning libraries such as Keras simplify the coding of neural networks which allows greater time efficiency while doing experimental work.

3.1.1 Processing (Parsing) Images

The blank images with no steganographic data encoded in them from the PIM Dataset were loaded into a directory. Half of those images were randomly selected and encoded with random data: the amount of data was determined by counting the number of non-zero discrete cosine transform coefficients and multiplying them by a bpc (bits per coefficient ratio). This data was encoded into the image using the F5 steganographic algorithm [Westfeld, 2001a]. The discrete cosine transform values of the images were extracted using a C++ program [Ros, 2014] utilising libjpeg [Group et al., 2011]. Then labels which identify the presence of steganographic data in a particular image along with the name of the file were written to a file for use in the network.

3.1.2 Network Training

The reLU (rectified linear unit) [Nair and Hinton, 2010] activation function along with the categorical cross entropy loss function [De Boer et al., 2005] paired with the Adam optimizer [Kingma and Ba, 2014] were picked for their high performance (section 2.3). During training the dataset was pseudo-randomly split up into $\frac{2}{3}$ training data and $\frac{1}{3}$ test data.. Given that there were 800 images in the original dataset and half of them were embed with a stego payload, it is impossible for only one type of image to be present in the training (as $\frac{1}{3} < \frac{1}{2}$). Keras (using the Tensorflow backend) was used to build and train the models due to their popularity in the machine learning community as described in section 2.3.3.2. The network was then trained on the training data, and the performance of the network was evaluated using the test data.

3.1.3 Data Collection Workflow

The images were parsed as detailed in 3.1.1 using different values of the bpc coefficient to encode different amount of data, allowing an observation of whether more data encoded in the image had an effect on the accuracy of the network that will be shown in section 3.3. Then both the old and new neural networks were trained and evaluated, detailed in 3.1.2, on the generated dataset.

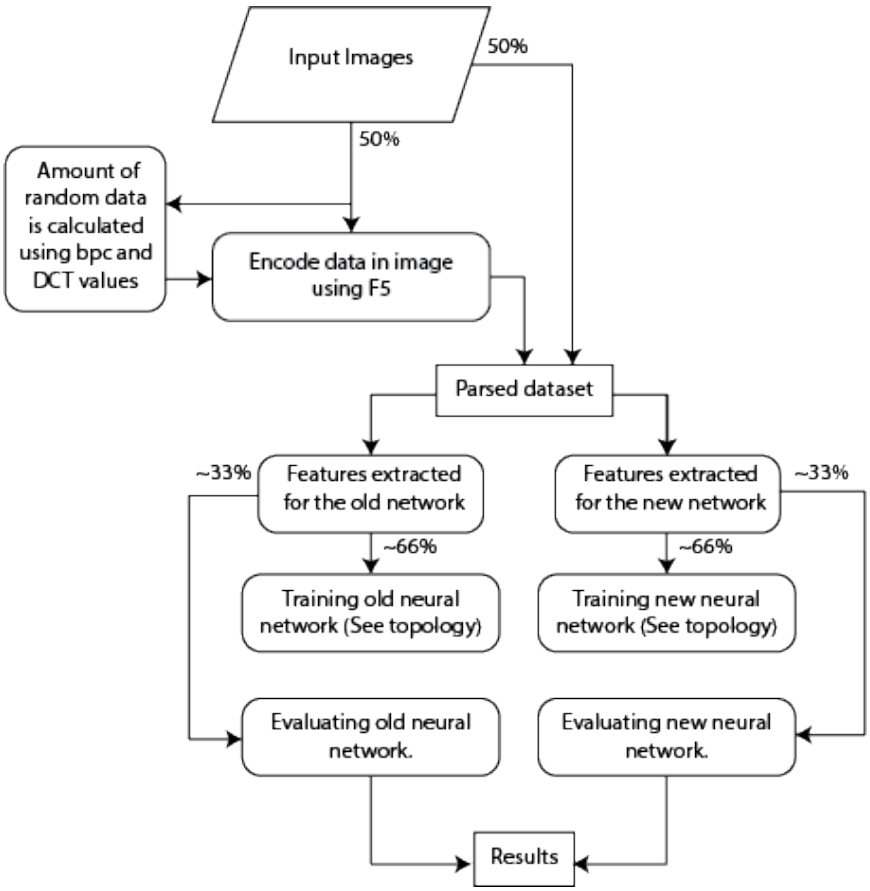

Figure 20: *A flowchart to summarise the full parsing and data collection process.*

3.2 Network Topologies

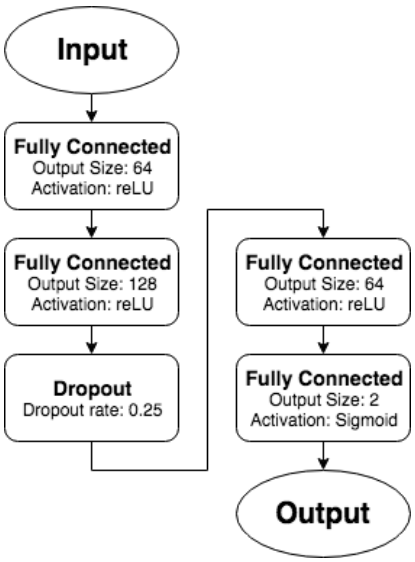

Figure 21: *Topology of the old neural network.*

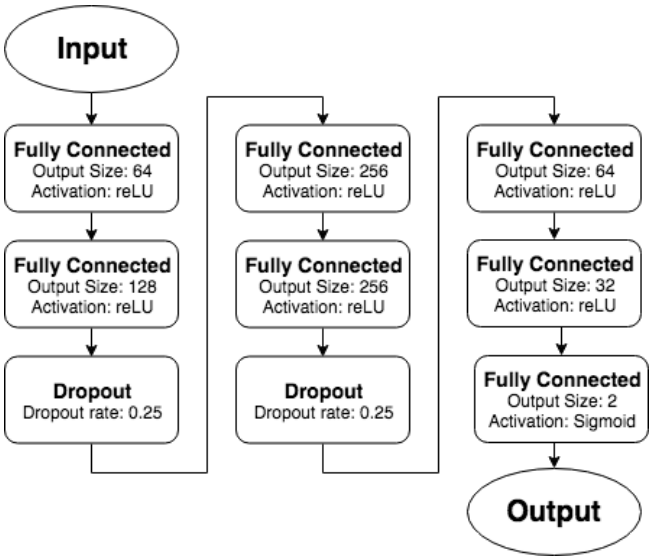

Figure 22: *Topology of the new neural network.*

3.2.1 Feature Extraction

3.2.1.1 Old Neural Network

Mean, variance, skew [Shaohui et al., 2003] and image dimensions were all extracted from the discrete cosine transform coefficients of each layer of the image. Then all of these statistics were normalised and fed into the input of the network.

3.2.1.2 New Neural Network

Mean, variance, skew [Shaohui et al., 2003] and image dimensions were all extracted from the discrete cosine transform coefficients of each layer of the image along with the same statistics across all of the raw image layers. Then all of the statistics were normalised and fed into the input of the network.

3.2.2 Training

Each epoch denotes one forward and backward pass of all of the training examples. Batch size denotes the number of images in one forward/backward pass. It was decided to use 300 epochs for the second network after experimental work, starting at 40 (the first dataset) and increasing by steps of 10 until 300 was reached upon as it allowed the network optimising well while the overfitting was handled by the dropout layers. 128 batch size was decided because it allowed for fast training (on the hardware available) without compromising the optimisation of the network. The accuracy and loss graph are closely related. Although most of the analysis is on the loss graph, the accuracy follows the inverse of the loss function i.e. if the loss function decreases, the accuracy will increase and vice versa.

3.2.2.1 Old Neural Network

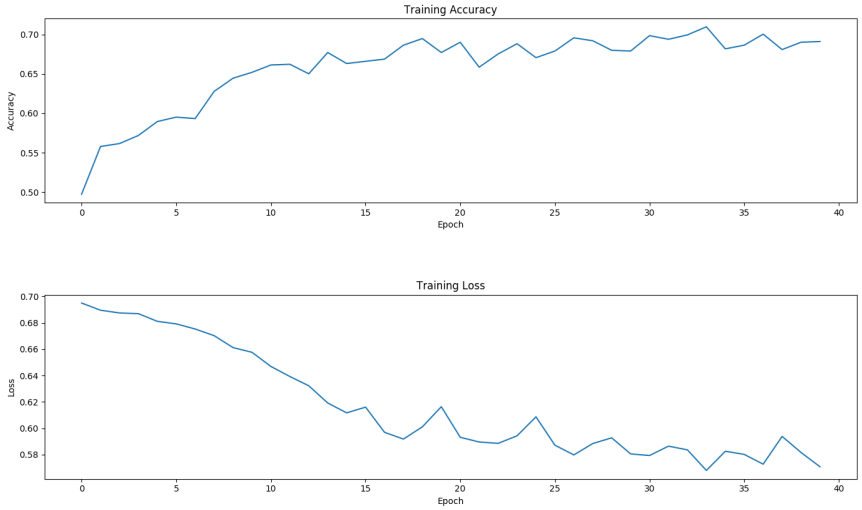

Figure 23: *Training loss and accuracy metrics of the old neural network. Trained using 40 epochs with a batch size of 128.*

The shape of the graph in Figure 23 shows that it is possible that the network has reached a minimum, but the unevenness of the data suggests that the network could benefit from more epochs of training data to allow the loss function to further descend especially as there is evidence of the dropout layers working (spikes near the end of training).

3.2.2.2 New Neural Network

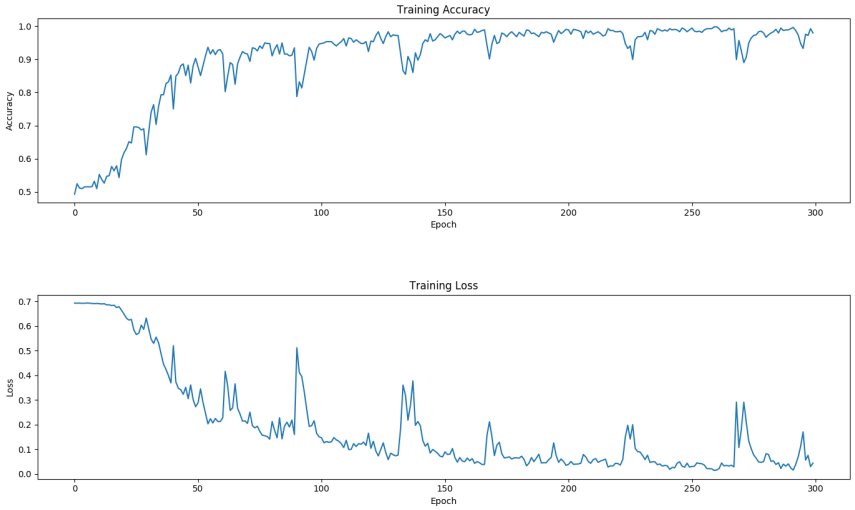

Figure 24: *Training loss and accuracy metrics of the new neural network. Trained using 300 epochs with a batch size of 128.*

The spikes in Figure 24 indicate that the dropout layers are combating overfitting as it can be seen that they are adding a penalty to the loss function. The graph also shows that the number of epochs is allowing the network to reach a local minimum (shown by the exponential decay shape), then the dropout layers are then preventing it from overfitting too much (shown by the spikes, as discussed previously).

3.3 Data

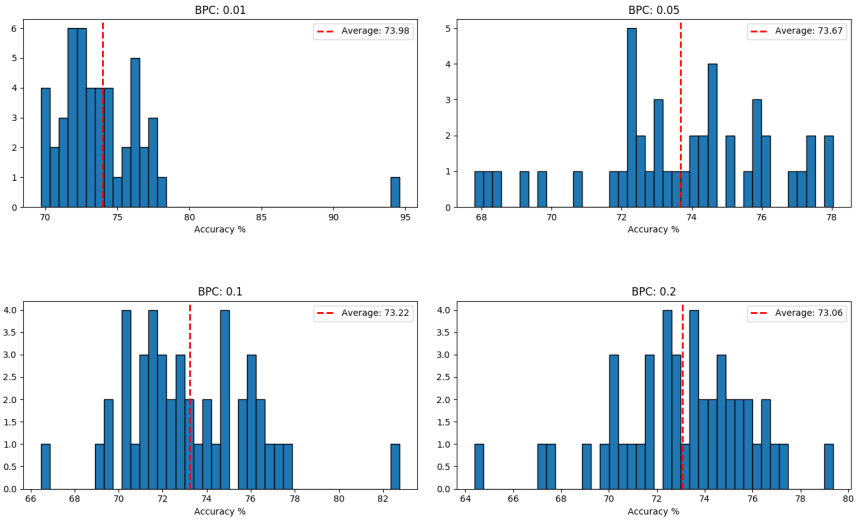

Figure 25: *Accuracy of the data collection plotted as a histogram for the old neural network along with the mean.*

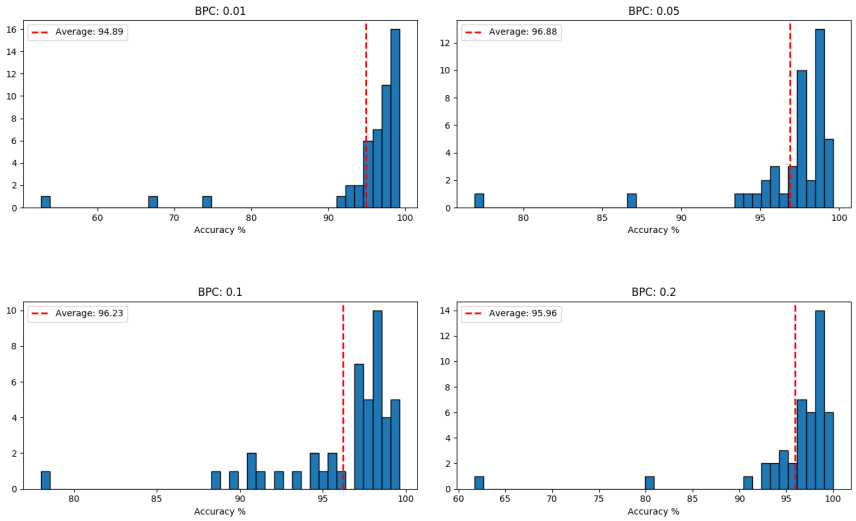

Figure 26: *Accuracy of the data collection plotted as a histogram for the new neural network along with the mean.*

3.4 Analysis

BPC	Mean	Variance
0.01	73.835%	12.716
0.05	73.556%	7.283
0.1	73.243%	7.786
0.2	73.334%	8.069

Table 1: *Summary of Old Neural Network statistics.*

BPC	Mean	Variance
0.01	94.886%	70.886
0.05	96.875%	14.329
0.1	96.230%	15.759
0.2	95.960%	37.722

Table 2: *Summary of New Neural Network statistics.*

It can been seen in the old network Figure 25 that the mean stays in a narrow range (lower: 73.06%, upper: 73.98%) throughout all of the tests regardless of the amount of data encoded (bpc level). In the same vein, Figure 26 shows that the mean doesn't seem to vary based on the amount of data encoded (bpc).

The fact that the mean seems not to vary too much (variance of 0.687 and 0.177 (3 s.f) between the averages of the new neural network and old neural network respectively from Tables 2 & 1) suggests that the neural network is detecting an artefact of the F5 steganographic algorithm which is present when any amount of data is encoded. If the network was detecting an artefact of the data encoded, it would be expected that there would be a graduation in the accuracy as the amount of data increased but it seems that this is not the case.

In the old neural network (Figure 25), it can be seen that the data is, for the most part, quite evenly spread amongst the lower and upper bounds. Whilst in the new neural network, it seems that the data is more concentrated around the mean and there are a few extreme outliers (as shown by the variance).

These outliers can be explained by looking at the training graphs of the networks (Figure 23 & 24). We claim that the outliers are caused by the dropout layers activating just before training is finished. In essence, the outliers are caused when the network stops training at one of the peaks of the loss function (when the dropout has just added a penalty to the loss function) and the network has not had time to correct it. This is backed up by the fact that it seems that the outliers are more prevalent in the new neural network, which has more dropout layers in the topology Figure 22 than the old neural network. The variance of the two graphs (Figure 2 & 1) also suggests this as the overall variance of the

new neural network (with more dropout layers) is higher than the old neural network (with less dropout layers).

4 Conclusion

It has been found that the Neural Networks created for this project can be used very effectively to detect images encoded with the F5 steganography algorithm, even when using a less statistically complex feature set as in the "old neural network". It was found that the neural network was able to successfully learn to recognise features present in the images encoded with the F5 steganographic algorithm that were present regardless of how much data was encoded in the image (Figure 26).

There is evidence that suggests that as the statistical complexity of the network features increases, the accuracy of the network increases. The old neural network (simpler feature set) had a mean accuracy of 73% while the new neural network (more complex feature set) had a mean accuracy of 97%. These very good results were achieved by utilising a relatively simple feature set compared to previous methods [Fridrich et al., 2002].

These results could result in finding a fingerprint (defined in 2.4.1) for the F5 algorithm. The fact that the network accuracy was independent of the amount of data encoded in the image is indicative that the network is learning an artefact of the process of the F5 algorithm. In future an artefact that is left behind by the F5 steganographic algorithm (regardless of the amount of data encoded) could be found.

From my reading, it seems that there is no example of a study that has reduced statistical complexity with the goal to make up for it using a more complex classification algorithm. Although there may be a similar study in the literature, the fact that it has not been found during my reading means that if there is such a study it has not had much of an impact.

5 Bibliography

References

Abraham, A., Paprzycki, M., et al. (2004). Significance of steganography on data security. In *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, volume 2, pages 347–351. IEEE.

Bandyopadhyay, S. K., Bhattacharyya, D., Ganguly, D., Mukherjee, S., and Das, P. (2008). A tutorial review on steganography. In *International conference on contemporary computing*, volume 101, pages 105–114.

Berger, C. (2016). Perceptrons - the most basic form of a neural network.

Brownlee, J. (2017). Overfitting and underfitting with machine learning algorithms.

Budhiraja, A. (2016). Learning less to learn better - dropout in (deep) machine learning.

Cabeen, K. and Gent, P. (1998). Image compression and the discrete cosine transform. *College of the Redwoods*.

Christensson, P. (2006). Lossy.

Cole, E. (1997). Steganography. *Information System Security Paper, George Mason University*.

cs231n (2017). Cs231n convolutional neural networks for visual recognition.

De Boer, P.-T., Kroese, D. P., Mannor, S., and Rubinstein, R. Y. (2005). A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67.

FBI (2012). Steganography picture.

Fei-Fei, L., Fergus, R., and Perona, P. (2007). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding*, 106(1):59–70.

Fridrich, J., Goljan, M., and Hogea, D. (2002). Steganalysis of jpeg images: Breaking the f5 algorithm. In *International Workshop on Information Hiding*, pages 310–323. Springer.

Group, I. J. et al. (2011). Libjpeg.

Guillermito (2004). Extracting data embedded with jsteg.

Gupta, S., Gujral, G., and Aggarwal, N. (2012). Enhanced least significant bit algorithm for image steganography. *IJCEM International Journal of Computational Engineering & Management*, 15(4):40–42.

Handel, T. G. and Sandford, M. T. (1996). Hiding data in the osi network model. In *International Workshop on Information Hiding*, pages 23–38. Springer.

Johnson, N. F. and Jajodia, S. (1998). Steganalysis: The investigation of hidden information. In *Information Technology Conference, 1998. IEEE*, pages 113–116. IEEE.

Karn, U. (2017). An intuitive explanation of convolutional neural networks.

Keras (2018). About keras.

Kerr, D. A. (2005). Chrominance subsampling in digital images. *The Pumpkin,(1), November*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipper, G. (2003). *Investigator's guide to steganography*. crc press.

Krenn, R. (2004). Steganography and steganalysis.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

McGonagle, J., Shaikouski, G., Hsu, A., Khim, J., and Williams, C. (n.d). Backpropagation.

Morkel, T., Eloff, J. H., and Olivier, M. S. (2005). An overview of image steganography. In *ISSA*, pages 1–11.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Ng, T.-T., Chang, S.-F., Hsu, J., and Pepeljugoski, M. (2005). Columbia photographic images and photorealistic computer graphics dataset. *Columbia University, ADVENT Technical Report*, pages 205–2004.

Nielsen, M. A. (2015). *Neural networks and deep learning*. Determination Press.

Pincus, W. (2010). Fbi spent nearly decade pursuing spy suspects in bid to gain counterintelligence. *The Washington Post*.

Provos, N. (2002). Outguess. *Software available at www. outguess. org*.

Raval, N. G., Gundaliya, P., and Rajpara, G. (2017). International journal of advance research in engineering, science & technology. *Development*, 4(5).

Reddy, H. M. and Raja, K. (2009). High capacity and security steganography using discrete wavelet transform. *International Journal of Computer Science and Security (IJCSS)*, 3(6):462.

Richer, P. (2003). Steganalysis: Detecting hidden information with computer forensic analysis. *SANS/GIAC Practical Assignment for GSEC Certification, SANS Institute*, 6.

Ros, E. (2014). Extracting jpeg dct coefficients.

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015a). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015b). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Shaohui, L., Hongxun, Y., and Wen, G. (2003). Neural network based steganalysis in still images. In *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, volume 2, pages II–509. IEEE.

Sharifzadeh, M., Agarwal, C., Aloraini, M., and Schonfeld, D. (2017). Convolutional neural network steganalysis's application to steganography. *arXiv preprint arXiv:1711.02581*.

Silman, J. (2001). Steganography and steganalysis: an overview. *Sans Institute*, 3:61–76.

Simmons, G. J. (1984). The prisoners' problem and the subliminal channel. In *Advances in Cryptology*, pages 51–67. Springer.

Solanki, K., Sarkar, A., and Manjunath, B. (2007). Yass: Yet another steganographic scheme that resists blind steganalysis. In *International Workshop on Information Hiding*, pages 16–31. Springer.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Tensorflow (2015). About tensorflow.

UFLDL (2013). Feature extraction using convolution.

Upham, D. (1997). Jsteg. *Software available at ftp. funet. fi*.

Wallace, G. K. (1992). The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv.

Westfeld, A. (2001a). F5—a steganographic algorithm. In *International workshop on information hiding*, pages 289–302. Springer.

Westfeld, A. (2001b). F5—a steganographic algorithm: High capacity despite better steganalysis. In *4th International Workshop on Information Hiding*, pages 289–302. Springer-Verlag.

Wikipedia (2018). Overfitting — Wikipedia, the free encyclopedia.

Winkler, S., van den Branden Lambrecht, C., and Kunt, M. (2001). Vision models and applications to image and video processing.