Audio Steganography: a Deep Learning Approach

November, 2020

Philippe Partarrieu
M.Sc. Security and Network Engineering
University of Amsterdam
Amsterdam, The Netherlands
philippe.partarrieu@os3.nl

Davide Pucci
M.Sc. Security and Network Engineering
University of Amsterdam
Amsterdam, The Netherlands
davide.pucci@os3.nl

***Abstract*—Digital steganography is the technique of hiding secret information into a file, known as the carrier. Traditional approaches lead to highly detectable steganography with a low ratio of information hidden to information carried over. Novel methods based on Artificial Intelligence (AI) are recently emerging, revamping steganography and steganalysis leading to an increase in hiding capacity. In this paper, we show how deep learning can be a viable approach to achieve audio steganography. We build a Convolutional Neural Network (CNN) inspired by the encoder-decoder model, that is capable of hiding a spectrogram representation of a secret audio file into a carrier spectrogram of the same length. We find that this method produces a lossy output but that the words from the secret and cover audio are audible. We also find that this method is resistant to noise.**

***Index Terms*—digital forensics, steganography, artificial intelligence, neural networks**

I. Introduction

The first written account of steganography dates back to the 15th century and appears in Steganographia [1] by Johannes Trithemius. Information was embedded into images, articles or shopping lists with the purpose of making a hidden message as hard as possible to be identified.

Modern digital steganography, uses the same approach to hide generic payloads, called secrets, into transport mediums, called carriers. These containers are usually well-known media formats such as images, video or audio files. The powerful idea behind digital steganography is that the output carrier file, which is carrying the secret data, not only highly resembles the original one before the secret data is inserted, but it is also a perfectly legal media file which is not affected in any way in terms of human perception. The result is that the carrier file can be safely and publicly exchanged between parties.

On the other hand, steganalysis describes the techniques to detect and possibly identify a message hidden within a carrier.

The practice of steganography is agnostic to the media format used such that the hidden data can be improved by encrypting it and thus reducing the probability that steganalysis approaches could detect the data.

Steganography is sometimes used in criminal contexts, allowing files to be publicly released with reasonable confidence that they will not be discovered [2]. On the other side, steganography finds legitimate usage as a way to embed authorship information as in digital watermarks.

Modern steganalysis approaches easily detect if a file is carrying hidden data, as the insertion of this secret data alters the statistics of the medium itself. This is why common digital steganography hides small messages in big mediums, so that carriers are not heavily and evidently altered. In the case of images used as mediums, this is usually referred to as the bits-per-pixel (bpp): in traditional approaches, the amount of information is set to 0.4bpp or even lower. On the other side, depending on the type of carrier media format, information can be hidden in different file areas, depending on the specific media format internal structures.

The traditional approach deals with the medium's Least Significant Bit (LSB)(s) to put secret information, as alterations at that level will less likely be visible. There exist many advanced methods to distribute the secret information in the carrier message in order to preserve the image's first and second order statistics, HUGO [3] being the most popular one.

Advances in the field of Artificial Intelligence (AI) has largely improved steganalysis, the practice of finding hidden messages. In this area, a very basic approach consists of training a Deep Neural Network (DNN) to decide in which LSB to place the binary representation of a text message. A DNN is used to select which bits to extract from the container image, for instance.

There are many neural network architectures each with many different applications. The most popular architecture for embedding information is the encoder-decoder neural network, introduced by N. Kalchbrenner in his PhD thesis *Encoder-Decoder Neural Networks*. In his words [4],

> Encoder-decoder neural networks are probabilistic conditional generative models of high-dimensional structured items such as natural language utterances and natural images. Encoder-decoder neural

networks estimate a probability distribution over structured items belonging to a target set conditioned on structured items belonging to a source set. The distribution overstructured items is factorized into a product of tractable conditional distributions over individual elements that compose the items. The networks estimate these conditional factors explicitly.

This architecture has been applied to solve complex learning problems involving linguistics, natural images and videos. The Recurrent Neural Network (RNN) variant is currently what powers Google's popular automated translation system [5]. In his thesis, Kalchbrenner proposes an encoder-decoder variant based solely on convolutional layers which performs well in the domain of images. When applied to the problem of embedding information, the architecture learns a distribution of secret and cover assets which results in an encoding and decoding process which, in simple terms, allows to extract the important features from the secret, encode them into the carrier while maintaining a visually unchanged carrier and finally decoding the secret from the carrier.

While steganography applied on DNN paradigm has been heavily researched for images, which are most likely the best media format to be used in an encoder-decoder model, the same can not be said for audio.

We decided to work with an uncompressed and lossless audio format WAVEform audio file format (WAV), since each file carries a large amount of information, which will translate to better feature extraction by the network and ultimately a better quality output cover and secret. This network can also be trained on compressed and lossy file formats such as Moving Picture Expert Group-1/2 Audio Layer 3 (MP3).

Specifically for audio signals, carriers have to be compliant to several properties, in order to be able to hide a secret message without impacting the human auditory perception:

1) *Inaudibility of distortion* (or *Perceptual Transparency*) means that the audio file carrying the secret should not sound distorted when compared to the unmodified file.
2) *Robustness* is defined as the relation between inaudible inadvertent, e.g., resizing, rescaling or expansion of commotion, and advertent information, e.g., such as control information.
3) *Data Rate* (or *Capacity*) refers to the audio codec properties that have to be maintained, such as the original medium sample rate.

II. Related Work

In recent years, we have seen considerable steganographic improvements since LSB encoding. Most frameworks revolve around image hiding, such as HUGO (Highly Undetectable steGO) [3] from *Using High-Dimensional Image Models to Perform Highly Undetectable Steganography* which preserves the statistics of an input image by relying on the principle of minimal impact embedding [6].

When looking at digital audio steganography, most methods exploit physical properties of sound waves and human hearing, i.e., tone insertion place merges the cover audio signal with two tones of known frequencies and low power level. S. Mishra *et al.*, in *Audio Steganography Techniques: A Survey* [7] exhaustively describe such methods. A *Comparative study of digital audio steganography techniques* [8] defines comparison criteria such as hiding rate, imperceptibility and filtering that can be used to evaluate the different steganographic systems.

Y. Qian *et al.*, in *Deep learning for steganalysis* [9] show that DNN can be used for steganalysis. They propose a new paradigm, shifting away from common statistical detection models, for steganalysis to learn features automatically by using a Convolutional Neural Network (CNN). This research yields outstanding results and introduces the effectiveness of AI techniques in steganography.

HiDDen: Hiding Data With Deep Networks [10], by J. Zhu, shows that neural networks can learn to use invisible perturbation for data hiding. They borrow from the auto-encoder network model by jointly training an encoder and decoder network and find that the algorithm is robust to noise.

In a similar way, *Hiding images in plain sight: Deep steganography* [11], by S. Baluja, shows that it is possible to hide an image of dimensions $N \times N \times RGB$ into another image of the same dimensions, with very little discrepancy, achieving an impressive 1:1 ratio of information hidden to the cover image's data. Unlike previous studies in which the hidden information must be received with intact integrity, the requirements are relaxed to allow the secret image to be received with some degradation — the paper describes an acceptable trade-off in image quality. Since the amount of information hidden is significantly higher than other techniques, the model does not conceal the fact that a secret image may be present, as can be shown from statistical analysis.

III. Problem

Steganography is a well-known topic that has been researched in a very exhaustive way. The shortcomings of the traditional approaches, which were easily detectable and poorly performing, led to an obvious and inevitable skepticism on steganography use-case scenarios. And even though the AI-based techniques are applying a very novel and modern approach, revamping steganography and steganalysis once more, for what steganography is concerned, it stays highly detectable. As a matter of fact, increasing the ratio of information hidden in a single medium to a high boundary inevitably increases the chances of steganography being detected, as highly interfering with original medium statistics.

Nonetheless the very same high ratio makes the technique intriguing. Even though an altered carrier asset can be detected, it does not necessarily mean that the secret information can be unveiled. In fact, without the exact same neural network used to inflate the medium with the secret information, it is not possible to extract the asset. This can only be done with the very same model composed of internal layers, filters, along with the same weights and biases associated each filter. More importantly, applying the learning capabilities of a DNN to the

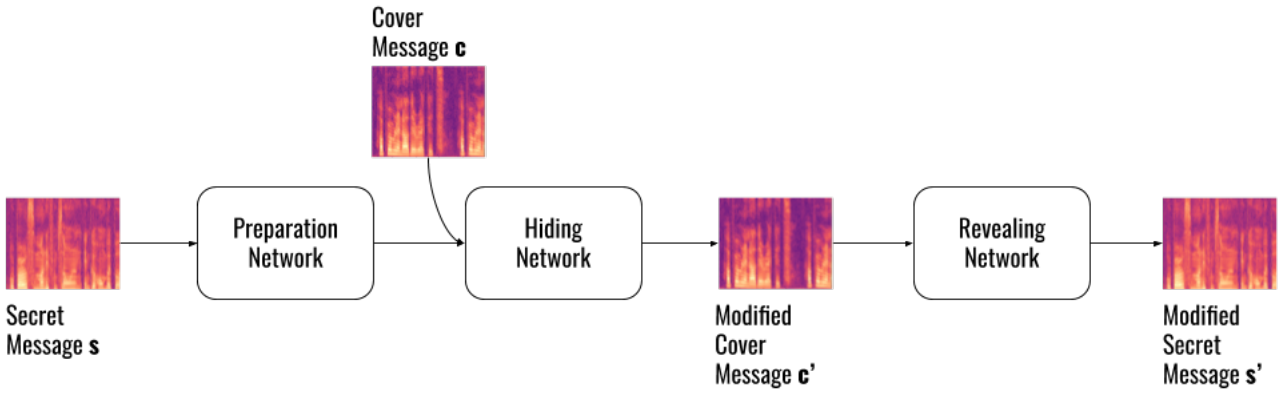

Fig. 1. Network model composed of Preparation, Hiding and Revealing networks

problem of hiding as much information as possible into an audio frame has not been explored yet. Thus, this research aims to apply the novel AI-based image steganography approach against audio signals.

IV. Research Question

This section presents our main research question and sub-questions that follow from it.

> Can we improve audio steganographic performance, in terms of inaudibility of distortion, robustness, and data rate, by using an DNN-based approach to embed a secret audio message into a carrier audio file?

This question can be divided into these sub-questions:

1) What audio pre-processing steps must be taken in order to find an image representation suitable for input to the neural network?
2) Given that encoder-decoder based neural networks behave efficiently for natural images, what is the best neural network architecture and structure to hide a sound into a sound, both represented as images?

V. Experiment Design & Implementation

Encoder-decoder (also known as auto-encoding) networks are effectively data compression algorithms which are designed for a specific type of data. The main difference with classical compression algorithms is that the network learns from examples. Typical encoder-decoder networks are designed to receive a single input and output. By mathematically comparing the distance between the output and the input they are able to adjust the weights and biases between layers, such that the output will more closely resemble the input. This process of calculating the distance is done by a loss function, which outputs a number in the $[0, 1]$ interval, with 0 representing a perfect match between the output and the input. The value of the loss function is a good indicator of the performance of any given model.

In the following section, we represent our secret and carrier as a sequence such that a given audio signal x of length l in seconds, recorded at a sample rate S of 16000kHz can be represented as $x = (x_0, x_1, ..., x_t)$ where $x_i \in \mathbb{R}$ represents an arbitrary amplitude value.

A. Model

There are three components that make up our network, as illustrated in fig. 1. The *Preparation* network P is tasked with encoding the secret spectrogram all the while extracting its most recognisable features. P takes in the secret s and encodes it into s_e such that $s_e = P(s)$.

The *Hiding* network H takes in the output of the *Preparation* network as well as the cover spectrogram c and hides the secret into a modified carrier image c' such that $c' = H(P(s), c)$.

The third and final network, the *Revealing* network R, receives the output of the *Hiding* network and decodes the secret image s' from it such that $s' = R(c')$

From H, we can calculate a mean squared error loss $||c - c'||$ that indicates the difference between the cover and the modified cover.

Similarly, R gives us $||s - s'||$ which tells us how closely the output secret resembles the input secret.

The attentive reader will notice that if the networks are trained separately, the *Hiding* network would simply discard s_e and perform the identity operation on c. This is why all three networks are trained simultaneously with the goal of minimising the following loss function,

$$L(s, c) = \lambda_c||c - c'|| + \lambda_s||s - s'||$$

where λ_c and λ_s are loss weight for the carrier and secret loss, respectively. They define how much the output loss of each model contributes to the overall loss value.

In practice, the sender only requires the trained *Preparation* and *Hiding* network, and the receiver only needs the *Revealing* network.

The network is made up of 3 convolutions layers at the *Preparation* network, 5 at the *Hiding* network and 7 at the *Revealing* network. We tried a varying number of layers but

found that this configuration produced the best output. Each layer is made up of 64 neurons, and we use a kernel size.

As is common for convolutional networks, we used Rectified Linear Unit (ReLU) as an an activation function for our neurons. ReLU is an activation function that is the identity function for all positive values and zero for all negative values. In practice, this makes it easy to compute, which speeds up training.

B. Dataset

In our experiment, we use the TIMIT Acoustic-Phonetic Continuous Speech Corpus [12]. This collection is made up of short readings of phonetically rich sentences by 630 different American-English speakers. Along with the waveforms, the dataset is collecting the time-aligned orthographic, phonetic and word transcriptions. For what the experiment is concerned, only the speech waveform files are used.

The dataset comes already split in TRAIN and TEST subsets: the first one contains 4620 audio files, while the latter 1680. Each sample comes in the shape of a WAV file. Each file has a frame rate of 16kHz.

As each sample represents a variable sentence, there is no common factor in the dataset in terms of audio length: the collection varies from very short audio waveforms of 0.92 to longer ones of 7.79 seconds. To reduce the study complexity, the variety in length has been removed by fixing samples to 2.5 seconds, with samples being either cut or looped depending on the original length. Below the command used to apply the constraint:

```
ffmpeg -stream_loop -1 -i "${src}" "${dst}"
    -ss 00:00:00.00 -to 00:00:02.500
```

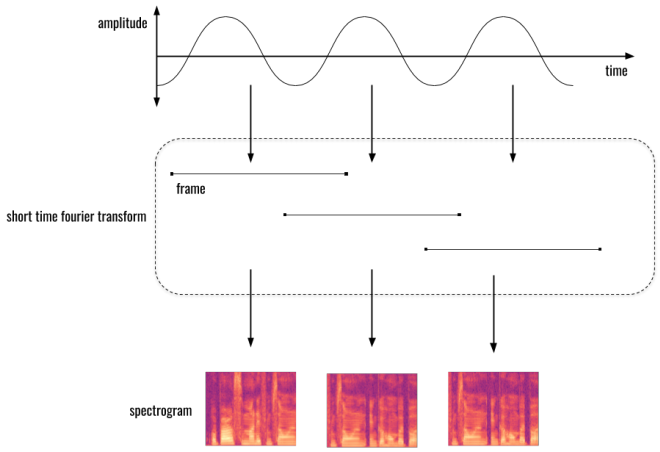

Fig. 2. Applying a STFT transformation to an audio signal

C. Samples representation

Since convolutional encoder-decoder networks and their variants perform well with images but not sound, we need to convert our audio signal into an image. There are various approaches to do this, depending on the type of audio we are trying to represent. One way is to convert the sound wave given by the audio stream from the time domain into the frequency domain. This allows us to make a spectrogram representation of an audio signal. We apply a Fourier transform to our audio signal and therefore convert our the signal from amplitude over time to frequency over time, as illustrated in fig. 3.

Originally, every audio signal is represented as a sound wave, where every sample has an associated amplitude which can be placed on a x-axis representing time. We divide the sound wave into small windows (of the order of 10^-2 seconds) and apply a Fourier transform to each window using the Short Time Fourier Transform (STFT) algorithm. We use a small analysis window to reduce the dimensionality of our data. The window is progressively shifted according to a given value known as the step, such that multiple windows and spectrograms overlap in time: this step is set to 25% of the sliding window. This gives the algorithm multiple viewpoints on the audio stream which increases our ability to extract the sounds of interest. In layman terms, the STFT algorithm will decompose an audio wave into a given number of sinusoidal waves, being the number of Fast Fourier Transform (FFT). Setting the FFT is a compromise between having an accurate spectrogam or having good compression. Therefore representing complex audio signals such as music requires a high FFT value (usually 2048) whereas speech processing is 512 [13].

D. Training

After processing a given pair of secret and cover images as explained in the paragraph above, the network will calculate the loss. In an attempt to minimise this loss in the upcoming runs, the networks will adjust the weights between connected neuron of the various layers through a process called *back-propagation*. Since the dataset is too large to fit in memory the network is trained by processing subsets of the data, also known as batches. A single iteration over the entire dataset is called an epoch and in order to learn a probability distribution over structured items, such as images, the network must iterate over the dataset many times. In the specific case, 100 epochs have been done, using the whole dataset of 6300 samples divided into 32 batches.

Furthermore, we use a custom learning rate schedule to adapt the weight that is given to the loss based on the amount of epochs for which the model has been training. This modification brings the benefit of making breaking changes in the network at the beginning of the training, when bigger rates get used, while the more the training is going, the more this rate decreases, resulting in a more granular tuning towards the end. Below the schedule used, in pseudo-code:

```
function scheduler(epoch)
    if epoch < total_epochs / 4 then
        return 0.001
    else if epoch < total_epochs * 2 / 4 then
        return 0.0003
    else if epoch < total_epochs * 3 / 4 then
        return 0.0001
    else
        return 0.00003
```

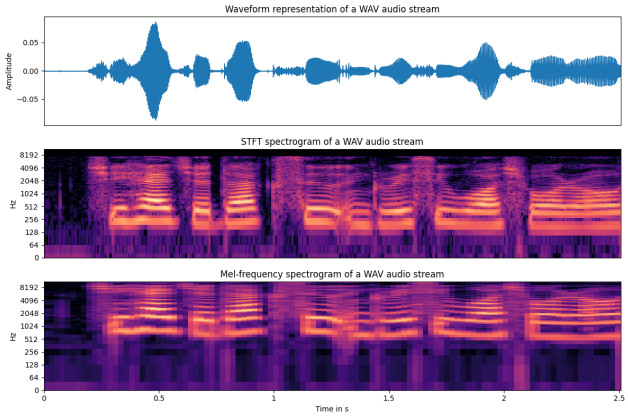

Fig. 3. Different representations of the same audio signal

VI. Results & Discussion

The first noticeable difference between the spectrograms in fig. 4 is that the output is significantly darker. In the audio files, this difference in gain means that the the output secret and cover are less loud the their input counterparts. The output spectrograms clearly show that the output has loss at certain frequencies, this manifests itself as black bands that run from the right to the left of the spectrogram. The darker bands are clearer at the top of the output spectrograms which indicates that we have more loss at higher frequencies. Despite the clear loss, the output spectrograms still display bright bands running from the top to the bottom. These are words spoken by the speakers and are remain intelligible in the output audio. Note that the spectrogram shows that there is no loss in the time domain meaning the length of the audio remains the same, unlike other audio steganography techniques which lengthen the carrier.

It is important to realise that our audio from the dataset was made in a noise free environment which results in particularly clean looking input spectrograms. Real-world recordings will most likely have background noise which will impact the quality of the audio. In practice, we could modify the dataset to overlay noise into each audio file which may have resulted in less noticeable differences between the input and ouput audio files and spectograms.

Since the network could learn to perform LSB encoding, we experimented with adding a noise layer of 0.01 to the modified carrier and found that the output audio was still comprehensible.

Inspired by audio classification networks, we attempted to apply a Mel-scale transformation to the resulting STFT spectrogram. Human hearing ranges from 20Hz to 20kHz. This being said, we are capable of distinguishing between lower frequency sounds better than higher frequency sounds. The mel scale applies a logarithmic transformation from frequencies into mel (the unit's name comes from the word melody). We applied this transformation in the hope that the encoded spectrograms would have a representation closer to the way human hearing works. Librosa, the audio library we used to convert back from a mel spectrogram to audio use the Griffin-Lim algorithm which estimates a signal from its STFT whereas the Inverse Short-Time Fourier Transform (ISTFT) allowed us to specify the number of FFT used to decompose the signal in the first place. In summary, the transformation from mel-scale spectrogram to audio was lossy where the transformation from a STFT based frequency spectrogram to audio was not, for our particular audio library.

A. Limitations

Since the carrier message should be inconspicuous we cannot say that our network produces good audio steganography. Improvements and tweaks to the number of FFTs as well as the number of layers, and epochs that the network is trained for could theoretically significantly reduce the amount of loss we are experiencing.

Our final trained model weighs 25MB which is quite sizeable. This model contains the weights connecting every neurons from every layer for all three networks. If this approach is to be implemented in a disk space restrained environment we could employ the following techniques to reduce the size of the model:

1) The optimal brain damage algorithm [14] can be used to remove connections between neurons that are rarely used, or neurons that rarely fire to begin with.
2) We can split the model into a *Preparation* and *Hiding* network to be used by the sender and a *Revealing* network to be used by the receiver.

Since our network is trained on a speech dataset it would not perform well with other types of sound. Attempting to encode more complex and layered audio such as music could theoretically be done with a large enough dataset of said audio type. We would also need to reconsider certain parameters such as the number of FFTs used in the STFT.

VII. Conclusions

We find that audio steganography using deep learning has several interesting properties compared to other methods. Namely, it is noise resistant and provides a comparatively high hiding capacity thanks to the encoding and decoding networks. The already mentioned LSB-based approach, although representing a very simple and easy way to hide content with high bit rate, has the drawback of being highly detectable, as well as extractable and, thus, destroyed. Furthermore, what is in common with all the well-known approaches to audio steganography [8] resides in the fact that several areas in audio representations have less value than others, in terms of impact in human hearing perception, and thus are more suitable for being inflated with additional data. This method, although effective, has a huge limitation: those areas have an upperbound beyond which the resulting audio signal would be damaged or highly distorted. Such a constraint imposes a very low hiding ratio.

On the other side, in DNN-based steganography, regardless of how a specific area in a carrier asset is critical in terms of impact to the perceived human hearing, the criteria used to

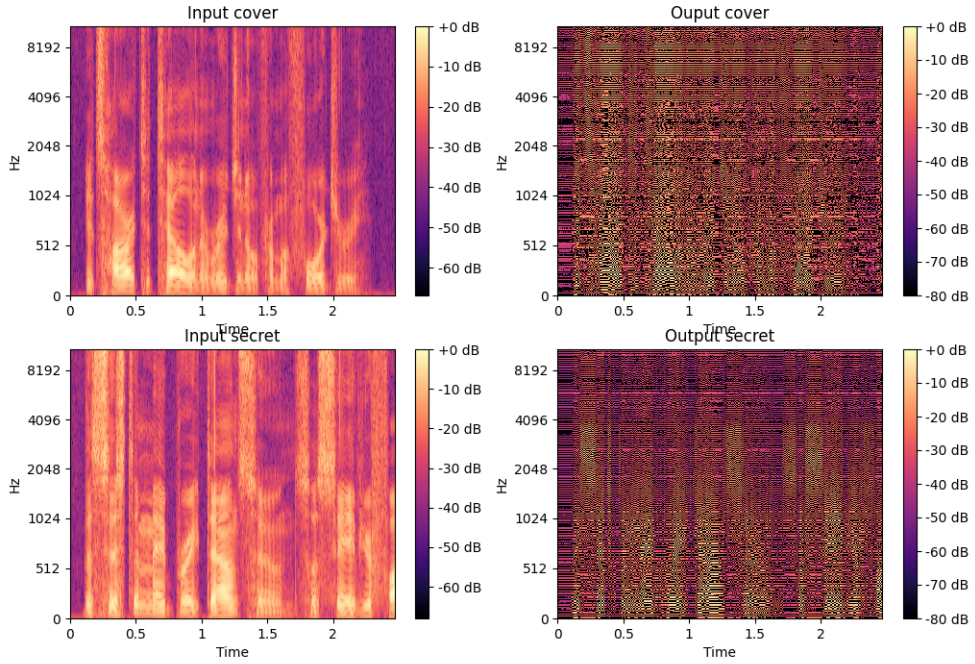

Fig. 4. Input and output cover and secret spectrograms

hide an information is based on statistics: after the network has been trained with thousands samples, it will be able to tell how a bit replacement will affect the similarity between the original asset and the one given in output, in terms of loss. Such a method of calculating what is worth to be replaced or kept as it was, leads to a model completely agnostic of the underlying data type. As a drawback, though, the audio files produced from the conversion of the output spectrograms are noisy and therefore are not viable for real-world steganographic applications which require that modified cover are inconspicuous.

A. Future work

For simplicity reasons, the research has been scoped down to deal only with utterances of the same 2.5s length: this constraint could be removed by introducing pre-processing padding measures, with which samples length could be stretched either by putting fixed zeros or — probably more suitable for the purpose — by looping the sample itself. Such an approach, would possibly enable the chance to be able to combine cover-secret pairs of different sizes.

Furthermore, the selected dataset is made up of a collection of recorded sentences. This represents not only a constraint in terms of semantic value, but more likely in terms of internal data structure. In fact, the samples used are *mono* recordings, i.e., audio assets composed of a single signal channel. The model described in this research relies highly on the audio samples length and number of channels. The network could theoretically be used for two-channel samples but would require modifications in order to support a higher number of channels.

B. Model & Code

The code for the project can be found at https://github.com/ppartarr/audioSteganography alongside a pre-trained model and audio samples.

References

[1] Trithemius, J. *Steganographia*. 1499.

[2] Fridrich, J. and Goljan, M. “Practical Steganalysis of Digital Images - State of the Art”. In: *Proceedings of SPIE - The International Society for Optical Engineering* 4675 (June 2002). DOI: 10.1117/12.465263.

[3] Pevný, T., Filler, T., and Bas, P. “Using High-Dimensional Image Models to Perform Highly Undetectable Steganography”. In: *Information Hiding*. Ed. by Böhme, R., Fong, P. W. L., and Safavi-Naini, R. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 161–177. ISBN: 978-3-642-16435-4.

[4] Kalchbrenner, N. “Encoder-decoder neural networks”. PhD thesis. University of Oxford, 2017.

[5] Wu, Y. et al. “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”. In: *CoRR* abs/1609.08144 (2016). URL: http://arxiv.org/abs/1609.08144.

[6] Fridrich, J. and Filler, T. “Practical methods for minimizing embedding impact in steganography”. In: *Security, Steganography, and Watermarking of Multimedia Contents IX*. Vol. 6505. International Society for Optics and Photonics. 2007, p. 650502.

[7] Mishra, S. et al. “Audio Steganography Techniques: A Survey”. In: Jan. 2018, pp. 581–589. ISBN: 978-981-10-3772-6. DOI: 10.1007/978-981-10-3773-3_56.

[8] Djebbar, F. et al. “Comparative study of digital audio steganography techniques”. In: *EURASIP Journal on Audio, Speech, and Music Processing* 2012.1 (2012), p. 25.

[9] Qian, Y. et al. “Deep learning for steganalysis via convolutional neural networks”. In: *Media Watermarking, Security, and Forensics 2015*. Ed. by Alattar, A. M., Memon, N. D., and Heitzenrater, C. D. Vol. 9409. International Society for Optics and Photonics. SPIE, 2015, pp. 171–180. DOI: 10.1117/12.2083479. URL: https://doi.org/10.1117/12.2083479.

[10] Zhu, J. et al. “HiDDeN: Hiding Data with Deep Networks”. In: *The European Conference on Computer Vision (ECCV)*. Sept. 2018.

[11] Baluja, S. “Hiding images in plain sight: Deep steganography”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 2069–2079.

[12] Garofolo, J. S. et al. *TIMIT Acoustic-Phonetic Continuous Speech Corpus*. 1993.

[13] Mller, M. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. 1st. Springer Publishing Company, Incorporated, 2015. ISBN: 3319219448.

[14] LeCun, Y., Denker, J. S., and Solla, S. A. “Optimal brain damage”. In: *Advances in neural information processing systems*. 1990, pp. 598–605.