Deep Steganographic Technique to Hide an Audio Signal in an Image

Pulkit Garg (P21CS011) Chiranjeev (P21CS007)

Abstract

In today's data-driven world, we are required to share information with each other digitally. And with the advancements in technology and digitization of various processes, especially after the covid pandemic, usage of digital mediums for information sharing has increased significantly. These digital mediums might not be secure and may result in unauthorized access to sensitive information. And as we know, unauthorized access can have significant impacts on organizations and individuals including huge monetary losses. Hence it is very crucial to the confidentiality of sensitive information. To achieve data confidentiality, we have developed Convolution Neural Network-based technique to embed secret audio in a cover image. The solution uses the Luminance chrominance color space (YCbCr) of the cover image to encode the audio instead of the Red-Green-Blue (RGB) Color channel. Finally, we evaluate the performance of the developed solution using the UTKFace dataset for the cover image and the Audio Speech Sentiment dataset for the secret audio signals.

Introduction

In today's data-driven world, we are required to share information with each other digitally. Our day-to-day process relies heavily on such data transactions. Also with the advancements in technology and digitization of various processes, many services are being provided at the comfort of our homes. Resulting in the increased usage of digital mediums for information sharing. But sometimes the information to be shared is sensitive and any unauthorized access can have a significant impact. For instance, if an organization's secrets leak to competitors, it can result in huge monetary loss and can also lead to the closing of that particular organization. Hence every organization tries to ensure the confidentiality of their sensitive information.

Considering the huge impacts associated with unauthorized access, the first option that comes to our mind is Cryptography. In cryptography, the original data is first encrypted and then shared. But there are some limitations associated with this. First of all, if we share encrypted data eavesdroppers are aware that some sensitive information is being passed on because encryption alters the data structure and makes it unreadable. Secondly, the encrypted data is larger than the original data, hence requiring more data storage space.

To cope with the problem of unauthorized access to sensitive information, steganography is generally used in combination with cryptography. Steganography is the practice of concealing a message within another message or a physical object. In computing/electronic contexts, a computer file, message, image, or video is concealed within another file, message, image, or video. Steganography requires a carrier in which sensitive information is hidden without noticeable changes and then the resulting carrier is shared. The resultant carrier and initial carrier are so similar, that it is very challenging to differentiate between them visually. This helps to ensure that the eavesdropper doesn't even know that some sensitive information is being shared is not. Resulting in reduced chances of unauthorized access.

Researchers have developed various steganographic techniques to hide critical information in documents, images, audios, etc. Of them, the most commonly used technique is to hide the information in the least significant bit where the significant bits of each pixel in the cover image are retained and the less significant bits are replaced with significant bits of the secret image. And since humans perceive majorly the significant bits, the changes are unnoticeable to the naked eye. But this technique is continuously challenged by the recent advancements in technology and is now not secure enough to be used. So the use of deep learning was introduced for the steganography process. Initially, the researcher tried to embed an image in an image using Convolution Neural Networks. After its success, this technique was further extended to embed audio in audio, the image in audio, etc. But little work is done to embed audio into an image because the range of audio frequencies is comparatively very large ass compared to the image intensity range. So to tackle the problem of embedding audio in images, we have developed a solution based using Luminance Chrominance Color space (YCbCr) instead of Red Green Blue (RGB) color space. It is known that the human visual system is very less sensitive to chrominance red and chrominance bleu channel, hence changes in these color channels will be far less noticeable than changes in RGB Color channel space.

In the developed solution we have used a combination of three CNN models referred to as Prepare Model, Hide Model and Reveal model. These three models are combined and work as a single model instead of three. This model accepts secret audio and a cover image as an input and produces a container image and recovered audio. Here container image is similar to the cover image but contains an audio signal within it as minimal changes that are not distinguishable visually. Specialized tools are required to extract the hidden audio signal as recovered audio. Currently, the model is combined but once trained these models can be splitted into their individual models while retaining the same trained weights. For the purpose of training and testing the developed solution, we have used a publicly available UTKFace dataset as a cover image and another publicly available Audio Speech Sentiment dataset (Kaggle dataset) as secret audio.

Dataset Description

In order to train a Convolution Neural Network that can conceal a secret Audio in a Cover image, we are required two datasets. For training and testing purposes, we are using the UTKFace dataset that contains over 20,000 facial images with annotations of age, gender, and ethnicity. Along with that, we are using the Audio Speech Sentiment dataset from Kaggle that comprises of 250 audio signals with three different emotions.

In this project, we have selected only the first 1000 facial images as cover images. 800 images of these images are considered to be a part of the training set and the rest 200 are considered to be a part of the test set. Similarly, for the audio dataset with 250 audio files, 200 are selected for training purposes and the rest 50 for testing purposes. Now to generate image and audio pair, we hance joined every facial image with every audio signal in the training set as well as a test set i.e. a total of 1,60,000 training image and audio pairs and 10,000 testing pairs as shown in Table 1.

Table 1. Dataset Description

	Number of Training Images	Number of Test Images
UTKFace	800	200
Audio Speech Sentiment	200	50
Total	800 x 200 = 160000	200 x 50 = 10000

Proposed Work

Steganographic techniques are used to conceal secret data into ordinary files/ images and secure it against unauthorized access. For every steganographic technique, it is very crucial to ensure that the resultant changes are minimalistic and are hardly noticeable. This helps the sender to obscure the information that the image contains some secret data. Now considering the importance of such techniques in today's data-driven world, we have developed a solution to conceal secret audio (S) into a cover image (C) using Convolution Neural Networks (CNN). The aim of the solution is that the resultant image(R) should be very similar to the cover image and secret audio can be extracted(O) correctly by using specialized tools. Figure 1 shows the complete architecture of the developed solution. The detailed steps are discussed below:

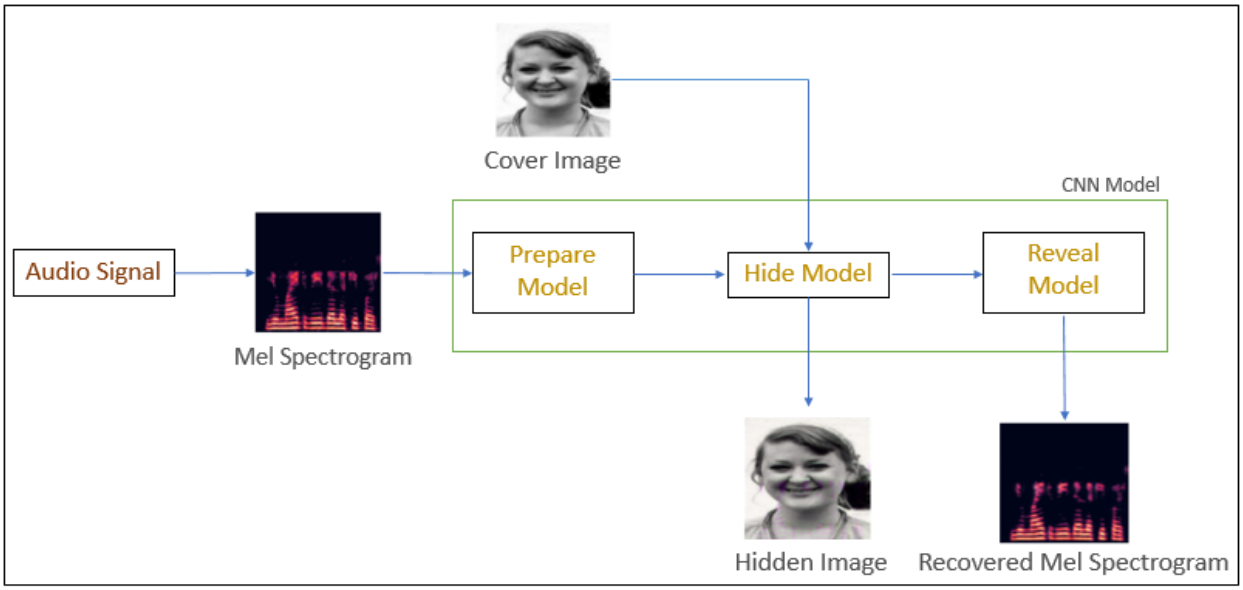

Fig 1. Overall Architecture of the Developed Solution

The solution is divided into 3 parts i.e. Data Preprocessing, Data hiding and Data Extraction

Data Preprocessing

- Data Preprocessing: Given facial images i.e. cover images are first resized to a 128 x 128 x 3 as a standard step. Now since we wish to encode audio files in YCbCr color space and RGB color space and compare the results. We train two models separately. We first convert the received cover images to YCbCr. And then to normalize we divided the image by 255 (max possible intensity value of an image) such that now images lie in the range 0-1.

- Audio Data Preprocessing: An audio signal is just the variation in air pressure over time and hence can be easily represented as a 1-D array of observed amplitudes (corresponding to a particular sampling rate). But these audio signals can't be directly passed to CNN. The first barrier is that different audio signals are of different lengths. And since audios are non-periodic we clipped the audio signals till the average length of all arrays formed on loading all audios. And if the audio signals are smaller as compared to the average length (average of unmodified length of all audio signals) then we have padded the remaining length. As a result, all audio signals are now of the same length.

  Further, the raw audio files are in the temporal domain and its amplitude range is much greater than the intensity range of an image. So to mitigate this issue a preprocessing step is carried out as shown in Figure 2. First, the audio signal is converted to the frequency domain with respect to time by doing Short-time Fourier Transform (STFT) to make its

representation of a spectrogram. It is like a bunch of FFTs computed on various window segments and are stacked on top of each other. This spectrogram represents the signal's loudness. The y-axis tells the audio's frequency and the x-axis tells the time. Now, to make it a Mel-spectrogram the log scale is computed on the y-axis and is mapped to Mel scale and colour dimension (amplitude) is converted to decibels because humans can perceive small audio frequencies and amplitude. Finally, audio is saved in the image format and reshaped to size of (128x128x3) so that it can be used in the Neural Network with the combination of cover image to form container image having mel-spectrogram's image and cover image both of similar dimensions of (128x128x3).

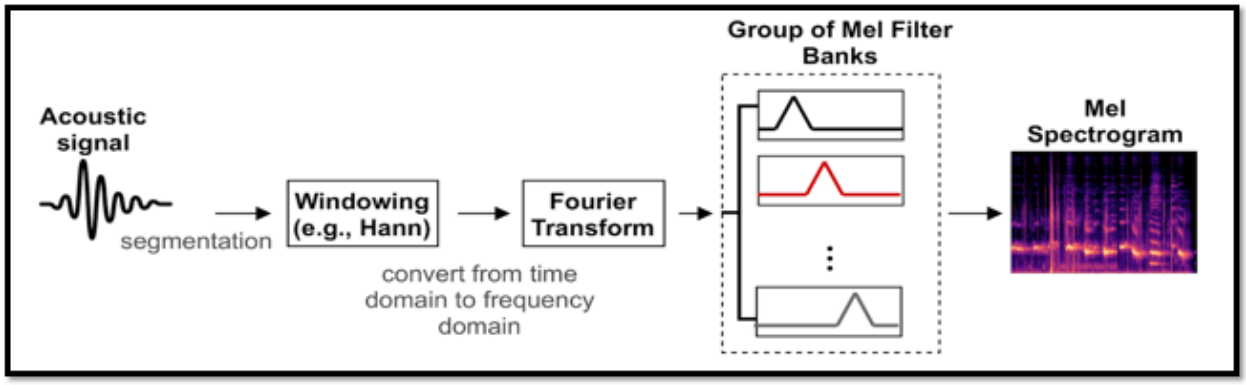

Fig 2. Illustrates Audio Pre-Processing Steps

Data Hiding

The solution uses a CNN model comprising 3 individual models i.e Prepare Model, Hide Model and Reveal Model. Out of these 3, Prepare and Hide models are responsible to hide audio in an image. But for training purposes, these models are treated as one single unit. This model works similarly to autoencoders where instead of one input, two inputs are passed (i.e. audio Mel spectrogram and cover image). The detailed architecture of the Prepare and Hide models are defined below:

- Prepare Model: Given the Mel spectrogram (audio signal in frequency domain) as an input, the objective of this model is to convert it into feature vector space. This conversion helps to remove the redundant information present in the input and only retain the useful characteristics. The model takes a 1x128x128x3 dimensional array as an input and passes it to 3 parallel blocks comprising 5 convolution layers each. Convolutions layers in these parallel blocks use different sized kernels i.e. 3x3x3, 4x4x4 and 5x5x5 so that the model can learn diverse features and adapt to complex spatial dependencies. The resultant feature vector is of the shape 1x128x128x150 and it can be uniquely associated with the given input.

One important thing to note here is that, in scenarios where the input size is not same as the size of the cover image, prepare model can be modified a little to gradually increase/decrease the input size.

- Hiding Model: This model has the responsibility to hide the feature vector generated above into the cover image with minimalistic changes. The model receives two inputs i.e feature vector of size 1x128x128x150 and a cover image of size 1x128x128x3. Here the 3 layers in the cover image represent Luminance, Chrominance Blue and Chrominance Red color channels respectively i.e. the cover image is in YCbCr color space. Initially, the model concatenates the two inputs and again proceeds with similar architecture as defined in Prepare Model. Towards the end, the model defines another convolution layer just to map the processed output to the required shape i.e. 1x128x128x3. The resultant image is called the Container Image.

Data Extraction

- Reveal Model: This model behaves as a decoder model of the autoencoders as it is the network that returns us the audio hidden in the container image. Here the container image is an input to this network and output is the audio's mel spectrogram of dimensions 128x128x3. Reveal Model's network architecture is the same as Hide model's network as Reveal model is the decoder model of Hiding Network (encoder model).

Once the joint model is trained (Prepare + Hide + Reveal), we can separate it into two parts where one part contains Prepare and Hide models. So given any image and audio pair, the first part can successfully encode audio into audio and provide with the hidden image. And the second part contains only the reveal model. So for instance, if the receiver receives a hidden image, he can pass that image to the second part and it will provide the embedded audio signal to the receiver. As a result, the sender can now share the container image with the receiver without worrying about unauthorized access.

Loss Function

The objective of the loss function is to iteratively minimize the loss between the model and input such that the hidden image (R) and cover image (C) are similar to each other. Also, if needed we can extract the embedded audio signal with minor distortions. In this project, we have used Mean Squared Error(MSE) loss function for comparison between both cover image and hidden image and input audio mel spectrogram (S) and output mel spectrogram (O). Since we have two work with two individual loss, we have combined them to obtain the total loss. Here we are assigning certain weights to these loss functions because the visual quality between the cover image and hidden image needs to be similar and we have to preserve the secret audio as well. So, first loss function is given the weightage "α" and second loss function is given the weightage "β". In order to determine the optimal weights, we tried various "α" and " β" values

and finally come up with α= 1.0 and β= 0.8 for RGB color space model and α= 0.9 and β= 0.7 for YCbCr color model. So, as to extract the audio in a good resemblance to input audio and also the image visual quality doesn't change much so that attacker can't visualize that there is something inside the image and audio can be passed from one person to another effectively with very less intervention from an intruder.

$$Loss = \alpha\, MSE(C, R) + \beta\, MSE(O, S)$$

Evaluation Metrics

We are using two metrics to analyze the performance of trained models on the test dataset are described in Table 2.

Table 2. Evaluation Metrics used to compare inputs and generated outputs

Metrics	Formula	Explanation
Mean Squared Error (MSE)	$MSE = \frac{\sum_{M,N} [I_1(m,n) - I_2(m,n)]^2}{M * N}$	It computes the mean squared between 2 images for each pixel.
Structural Similarity Index Metric (SSIM)	$SSIM(x,y) = \frac{(2\mu_x \mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$	It is the metric used to tell the structural similarity between two images and gives the similarity score.

Results

We have trained two models individually to hide audio data in images (one using RGB color channel and the other using YCbCr color channel). Figure 3(a) and (b) show some examples of the results obtained by model trained on the RGB color channel and Fig 3(c) and (d) shows the examples of the results obtained by the model trained on YCbCr color channel. And it can be observed that both models perform well in terms of retaining audio Mel spectrogram. As well the cover image and hidden image are not distinguishable easily.

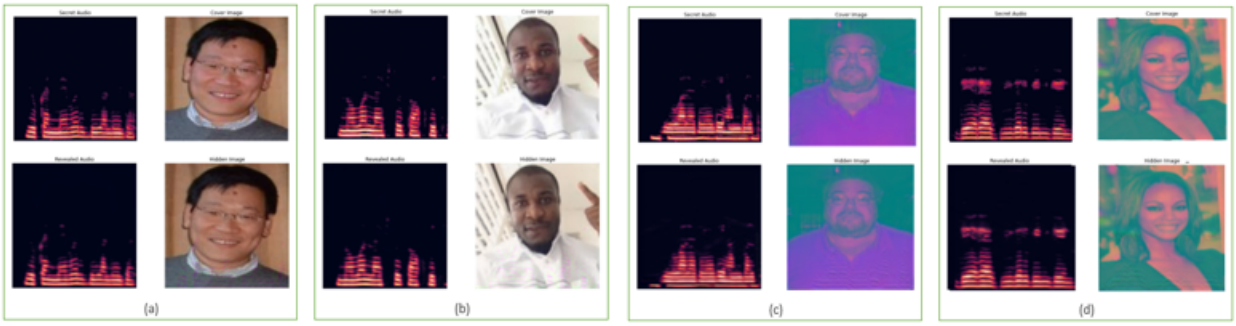

Fig 3. (a) and (b) illustrates examples of the results obtained by the model trained with RGB images. (c) and (d) illustrates examples of the results obtained by the model trained with YCbCr images. The first row represents in each example represents secret audio Mel spectrogram and cover image respectively and the second row represents the hidden image and extracted Mel spectrogram respectively.

On examining carefully it was observed that the Hidden image losses its natural tone and becomes a little bit white as compared to the Cover image.
Figure 4, shows the training loss over 15 epochs for both the models and it can be observed the model trained with RGB color channel outperforms the other model.

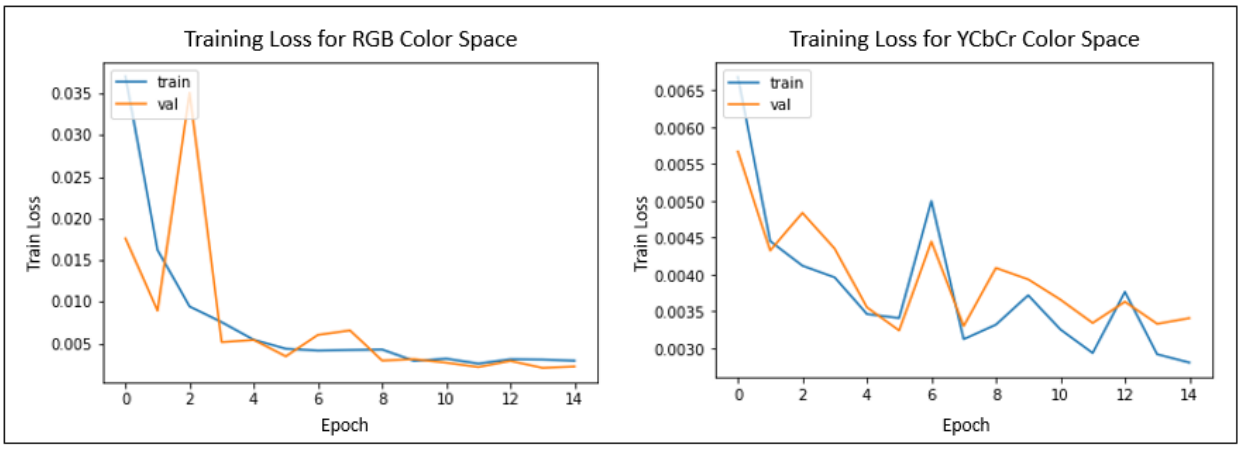

Fig 4. (a) Illustrates the variation of training loss with the number of epoch for the model trained with RGB colored image (b) Illustrates the variation of training loss with the number of epoch for the model trained with YCbCr colored image

Finally, In order to test the effectiveness of the YCbCr color space and compare it with RGB color space we have used the above two metrics i.e. Mean Squared Error (averaged over test set) and Structural Similarity Index Measure (average over test set) and the results are as follows:

Table 3. Metric for Model trained with RGB on Training Set

	MSE	**SSIM**
Audio Mel Spectrogram	$2.4308667 * 10^{-3}$	0.892
Cover image	$8.21204 * 10^{-4}$	0.9466

Table 4. Metric for Model trained with YCbCr on Training Set

	MSE	SSIN
Audio Mel Spectrogram	$1.457877 * 10^{-3}$	0.922257
Cover image	$3.5541662 * 10^{-4}$	0.922569

From Table 3 and Table 4 it can be concluded that the steganographic technique works well in cases where the cover image are provided in YCbCr color space.

Conclusion

In this project, we have analyzed the effect of different color channels in Audio in the Image Steganographic technique. Currently, we have tried to compare two color spaces i.e. RGB and YCbCR. To accomplish the same, we have used an overall loss function which is a weighted combination of two loss individual loss functions. One loss tries to preserve the secret audio and the other tries to preserve the cover image. We have tried assigning different weights to individual losses and finalized the one with minimum Mean Squared Error Loss.
From the above results, it is observed that YCbCr color space not only preserves its visual appearance but also the audio content better as compared to RGB color space.

References

1. Wu, P., Yang, Y., & Li, X. (2018, September). Image-into-image steganography using deep convolutional network. In Pacific Rim Conference on Multimedia (pp. 792-802). Springer, Cham.
2. Morkel, T., Eloff, J. H., & Olivier, M. S. (2005, June). An overview of image steganography. In ISSA (Vol. 1, No. 2, pp. 1-11).
3. Hamid, N., Yahya, A., Ahmad, R. B., & Al-Qershi, O. M. (2012). Image steganography techniques: an overview. International Journal of Computer Science and Security (IJCSS), 6(3), 168-187.

4. Purwins, H., Li, B., Virtanen, T., Schlüter, J., Chang, S. Y., & Sainath, T. (2019). Deep learning for audio signal processing. IEEE Journal of Selected Topics in Signal Processing, 13(2), 206-219.
5. Ye, D., Jiang, S., & Huang, J. (2019). Heard more than heard: An audio steganography method based on gan. arXiv preprint arXiv:1907.04986.
6. Chen, L., Wang, R., Yan, D., & Wang, J. (2021). Learning to Generate Steganographic Cover for Audio Steganography using GAN. IEEE Access.
7. Aagarsana, B. G., Anjali, T. K., & Kirthika, M. S. S. (2018). Image Steganography using secured force algorithm for hiding audio signal into colour image. IRJET access, 5.
8. Khalil, M. I. (2011). Image steganography: hiding short audio messages withidin digital images. Journal of Computer Science and Technology, 11(02), 68-73.