| **Image-to-Image Steganography using Convolutional Neural Networks** |
| --- |

*A Report on Study and Development of a Image-to-Image Steganography using Convolutional Neural Networks for the fulfilment of Summer Internship Project*

*by*

Chirag Agarwal

*under the guidance of*

Shri Krishna Kishore S V S R, Harkirat S Makkar



NATIONAL REMOTE SENSING CENTRE (ISRO)

# *Index of Contents*

| Abstract | 3 |
| --- | --- |
| Acknowledgements | 4 |
| Introduction | 5 |
| Architectures and Error Propagation | 7 |
| Methodology | 10 |
| Outcome | 19 |
| Conclusion | 20 |

# *Abstract*

Steganography, the practice of hiding a secret message within another, more ordinary message, is often used to inconspicuously embed small amounts of information in noisy regions of larger images. Using a novel approach, this project explores embedding a full-size 3-channel colour image within a same size higher resolution image. Deep learning with convolutional neural networks (CNNs) is utilised for this task. CNNs adapt to image characteristics, learning optimal data hiding strategies based on specific features of each image. They adjust embedding strategies dynamically, minimising distortion and enhancing camouflage.

These networks handle both encoding (hiding) and decoding (revealing) seamlessly, working together to achieve these goals. Training utilises images randomly selected from the COCO dataset, demonstrating robust performance across various natural image sources. Unlike traditional LSB-based techniques, our method compresses and distributes hidden image data across all available bits in the carrier image, reducing visible distortions and increasing capacity and robustness.

CNN-based steganography optimises encoding and decoding processes cohesively, enhancing security by avoiding static transformation rules that are susceptible to reverse engineering. This study showcases deep learning's effectiveness in image hiding, examining mechanisms and potential extensions. Our method allows trade-offs between carrier and hidden image quality, prioritising acceptable concealment over perfect hidden image reconstruction.

While statistical analysis can still detect hidden messages, our approach explores a balanced approach between detection difficulty and reconstructed image quality, offering promising advancements in steganography.

# 

# *Acknowledgements*

I would like to thank Shri Krishna Kishore S V S R (Scientist/Engineer and Group Head, SISG - NRSC/ISRO). Words cannot express my gratitude to my guide Shri Harkirat S Makkar (Scientist/Engineer - NRSC/ISRO) for his invaluable patience and feedback. This endeavour would not have been possible without the generous support from the National Remote Sensing Centre, ISRO who provided me with this opportunity.

I would like to thank Dr. Sudheer Kumar N (Director, Capacity Building and Public Outreach - ISRO) for giving me the opportunity to do an internship within the organisation. I wish to thank Smt. Jaya Saxena (Scientist/Engineer and Division Head, SPID - NRSC/ISRO), Smt. Savitha S (Scientist/Engineer - NRSC/ISRO) and Smt. Janani N (Sr. Proj.Assistant - NRSC/ISRO) for coordinating my internship period at NRSC, ISRO. I also wish to thank all the people that worked along with me at the NOC/SOC office room in Shadnagar campus - with their patience and openness they created an enjoyable and productive working environment.

Additionally, without the assistance of National Institute of Technology, Andhra Pradesh, which gave me the baseline knowledge and experience I needed to work in this specific area, I would not have been able to embark on this project.

Lastly, I would be remiss in not mentioning my family, especially my parents. Their belief in me has kept my spirits and motivation high during this process.

# *1. Introduction*

## 

## 1.1. Learning Objectives

In the realm of digital communication and data security, steganography plays a crucial role in concealing sensitive information within seemingly innocuous carrier images. Traditional methods often rely on manipulating pixel values directly or embedding information in frequency domains. With the advent of deep learning, particularly Convolutional Neural Networks (CNNs), there has been significant progress in enhancing the robustness and efficiency of steganographic techniques. This project explores the application of CNN-based models for image-to-image steganography, aiming to embed secret information into images and extract it reliably.

## 

## 1.2. Problem Statement / Motivation / Introduction

Steganography is the art of covered or hidden writing; the term itself dates back to the 15th century, when messages were physically hidden. In modern steganography, the goal is to covertly communicate a digital message. The steganographic process places a hidden message in a transport medium, called the carrier. The carrier may be publicly visible. For added security, the hidden message can also be encrypted, thereby increasing the perceived randomness and decreasing the likelihood of content discovery even if the existence of the message is detected.

1.2.1. Applications and Challenges:

Despite well-publicised misuses, such as coordinating criminal activities through hidden messages in public images, steganography is also used to embed authorship information like digital watermarks without compromising content integrity. Embedding alters carrier appearance and statistics, influenced by message size (measured in bits-per-pixel) and carrier image characteristics, impacting detection.

1.2.2. Common Approaches and Innovations:

Traditional methods manipulate least significant bits (LSB) to hide information, with statistical analysis revealing deviations. Advanced methods like HUGO preserve image statistics by modelling first and second-order statistics, typically used for smaller messages (< 0.5bpp). In contrast, this project employs neural networks to implicitly model natural image distributions and embed larger messages (e.g., full-size images), integrating hiding and decoding processes seamlessly.

1.2.3. Methodology and Security Considerations:

A neural network determines optimal embedding locations and efficient encoding across image bits, trained once independently of cover and secret images. The approach aims to hide a full RGB pixel image within another RGB image with minimal cover image distortion, prioritising acceptable trade-offs between carrier and hidden image quality over perfect reconstruction. Statistical analysis remains a challenge despite the visually imperceptible embedding of large amounts of information, paving the way for future improvements in balancing detection difficulty with reconstruction quality.

# *2. Architectures and Error Propagation*

## 2.1. About the Architectures

Steganography is often confused with cryptography, yet in our method, it draws closer parallels to image compression using auto-encoding networks. Our approach hinges on concealing a secret image within a cover image, seamlessly blending the information to minimise perceptibility. The system's architecture, illustrated in Figure 1, orchestrates this concealment process.

The components depicted in Figure 1 operate synergistically within a unified network, though each plays a distinct role that warrants individual description. Beginning with the Prep-Network positioned on the left, its primary task involves preparing the secret image for concealment. This preparatory phase serves dual objectives. Firstly, when the secret image (M × M) is smaller than the cover image (N × N), the Prep-Network scales up the secret image to match the cover's dimensions. This enlargement ensures that the secret image's data is distributed across the entire N × N grid of pixels, a crucial step for seamless integration (Details on experiments with smaller images are omitted here due to space constraints, focusing instead on full-sized images). Secondly, irrespective of size disparities, the Prep-Network transforms the pixel-based colour information into more abstract and efficient features, such as edges, as depicted in Figure 1.

The central component, known as the Hiding Network, follows the Prep-Network in the sequence. This network accepts inputs from both the transformed secret image and the cover image, synthesising what is referred to as the Container image. Here, the input space is a field of N × N pixels, incorporating concatenated RGB channels from the cover image alongside the modified channels from the secret image.

The final component, positioned at the far right, is the Reveal Network, designed for image extraction by the intended recipient. Functioning akin to a decoder, this network operates solely on the Container image, without access to the original cover or secret images. Its task is to decipher and extract the hidden secret image by effectively removing the cover image overlay.

## 2.2. Error Propagation

Our methodology draws inspiration heavily from auto-encoding networks, although our approach diverges significantly by encoding two distinct images (cover and secret) into an intermediate representation (the container image) that closely mimics the cover image's appearance. During training, the system optimises a loss function that balances the reconstruction errors of the cover (c) and secret (s) images, with a parameter β determining the weighting of these errors.



Figure 1. Architectures and Error Propagation flow

* Error term SSIM(C - C’) affects only the first two Networks
* Error term SSIM(S - S’) affects all the three Networks

Furthermore, in addition to the two aforementioned errors, we have incorporated an additional error term aimed at minimising the pixel-wise correlation between the residuals of the cover image and the secret image, denoted as Corr(Rc, S).

Here, Rc represents the Euclidean distance between the cover image and the container image, while S denotes the secret image.

Therefore the Loss Function can be defined as

Loss(C, C’, S, S’,Rc) = SSIM(C - C’) + 𝛽 \* SSIM(S - S’) + 𝛾 \* { 1 - Corr(Rc,S) }

By combining Cover SSIM loss, Secret SSIM loss, and Correlation loss, we get the total loss which provides a comprehensive measure of how well the model performs in both maintaining cover image quality and preserving secret information.

During training, the objective is to minimise total loss, thereby improving the model's ability to conceal secrets while minimising distortion to the cover image.

# 

# 

# *3. Methodology*

## 3.1. System Specifications

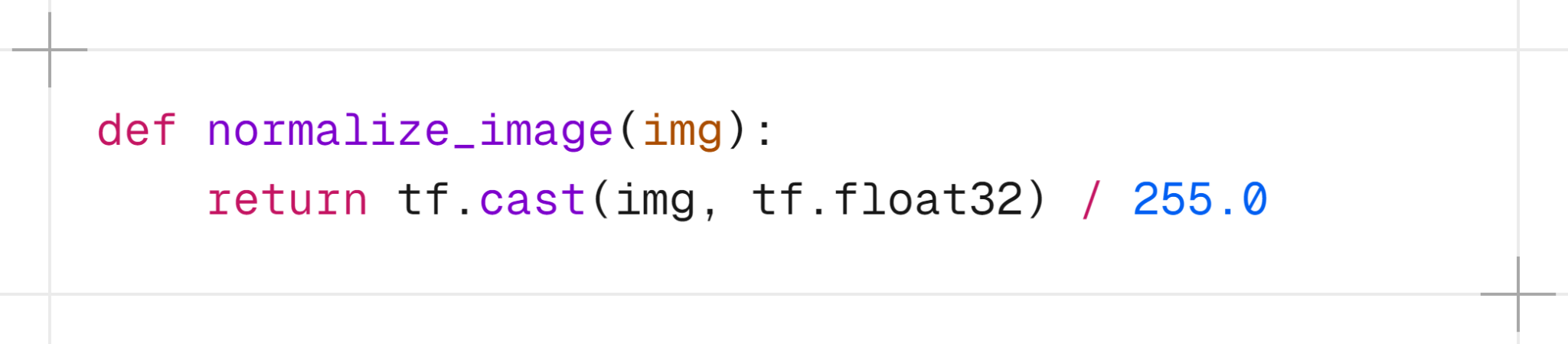
These headings succinctly summarise the hardware and software specifications of the system, providing a clear overview of the environment used for the described tasks or applications.

| Operating System | Windows 10 Pro |
| --- | --- |
| CPU | Intel(R) Core(TM) i5-10400 CPU @ 2.90GHz |
| GPU | NVIDIA A40-16Q |
| Volatile Memory | 16 GB |
| Technology Stack | Python 3.8, Anaconda 3, Tensorflow 2.10,  NumPy, cuda toolkit 11.2, CUDNN 8.1.0 |

## 3.2. Data Preprocessing

3.2.1. Normalisation:

Normalize\_image function ensures pixel values are scaled to [0, 1] for numerical stability.



3.2.2. Loading and Preprocessing:

This function loads an image from a file, ensures it has RGB channels, resizes it to a specified size (default is 224x224), and normalises its pixel values.



3.2.3. Data Loading Generator:

Load\_data function generates batches of preprocessed image pairs (batch cover and batch secret) indefinitely, each batch containing batch size images randomly chosen from files list.



## 3.3. Convolutional Neural Networks Architectures

3.3.1 Defining preparation network using Keras functional API:

The get\_prep\_network\_op function defines a neural network operation that applies multiple convolutional layers to secret\_tensor. These layers extract and combine features using different filter sizes (7x7, 5x5, 3x3) and concatenate them to form richer feature representations.



The final output is a tensor with reduced depth but preserved spatial dimensions, suitable for further processing or feeding into subsequent layers of a neural network architecture. This structure is commonly used in deep learning models to enhance feature extraction capabilities and prepare input data for subsequent layers.

3.3.2 Defining hiding network using Keras functional API:

The get\_hiding\_network\_opfunction defines a neural network operation that combines information from both the cover image and prepared feature maps . It uses multiple sets of convolutional layers to extract and process features from the concatenated input, ultimately producing an output tensor representing the hidden or modified version of the cover image.



The goal is to embed information into a cover image while maintaining the visual fidelity of the cover image.

3.3.3 Defining reveal/decoding network using Keras functional API:

The get\_reveal\_network\_op function defines a neural network operation that processes container images, likely representing a container image or data with hidden information. It uses multiple sets of convolutional layers to extract and process features from the input tensor.



The goal is to reveal or detect hidden information embedded within images or data. The final output represents the processed tensor with reduced depth but preserved spatial dimensions, suitable for further analysis or visualisation.

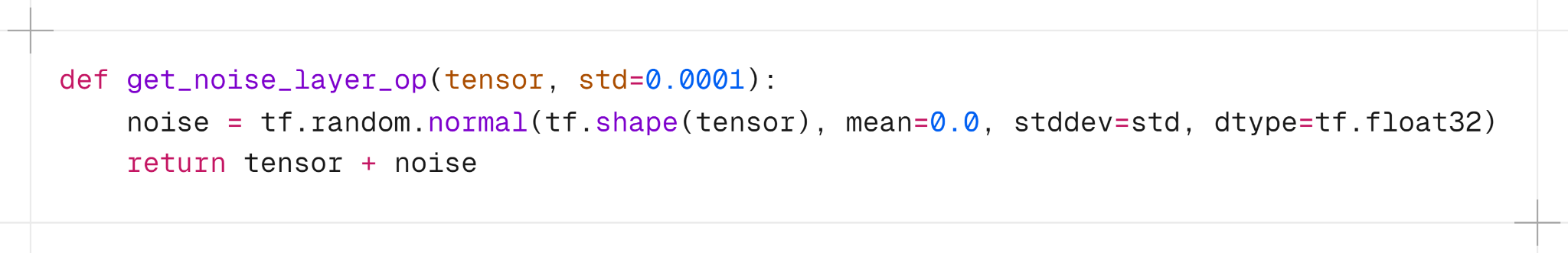
3.3.4 Loss Function:



The code structure allows for the calculation of a comprehensive loss function tailored for scenarios where maintaining image similarity (via SSIM) and ensuring correlation between cover and container images (via correlation loss) are both important metrics. The final loss is a weighted sum of these components, providing a measure of dissimilarity or error in the context of the task being addressed.

3.3.5 Noise layer Function:

The get\_noise\_layer\_op function effectively adds random noise to a given tensor. This operation can be useful in various scenarios such as regularisation during training, data augmentation, or introducing stochasticity into the model.



The amount of noise added is controlled by the std parameter, which determines the standard deviation of the normal distribution used to generate the noise.

## 3.4 Defining preparation, hiding, and reveal networks

The Steganography Model class implements a deep learning model for steganography tasks. It consists of three interconnected sub-models: the preparation network (prep\_network), the hiding network (hiding\_network), and the reveal network (reveal\_network).

The prep\_network takes a secret image input and processes it to produce a prepared output. This output is reshaped and used by the hiding\_network, along with a cover image input, to embed information.

The resulting hidden output undergoes noise addition using get\_noise\_layer\_op. Finally, the reveal\_network decodes the processed hidden output to reconstruct the hidden information.



The model's call method orchestrates these operations to perform embedding and extraction tasks seamlessly.

## 3.5 Preparing the training, testing and validation data

The code prepares datasets for training, validation, and testing using images from the COCO 2017 dataset. It first defines directories for training, testing, and validation sets, listing and filtering .jpg files accordingly.

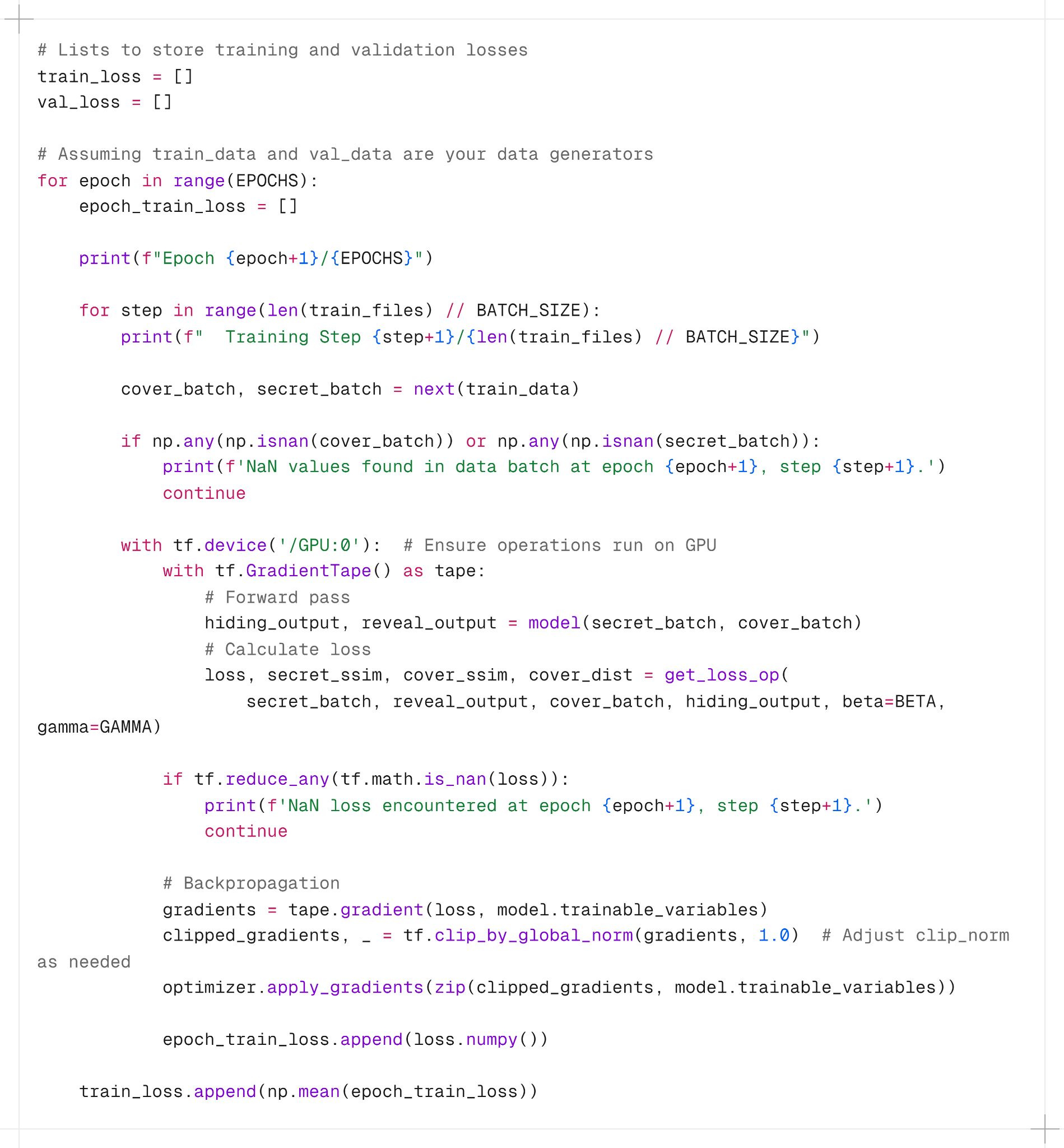


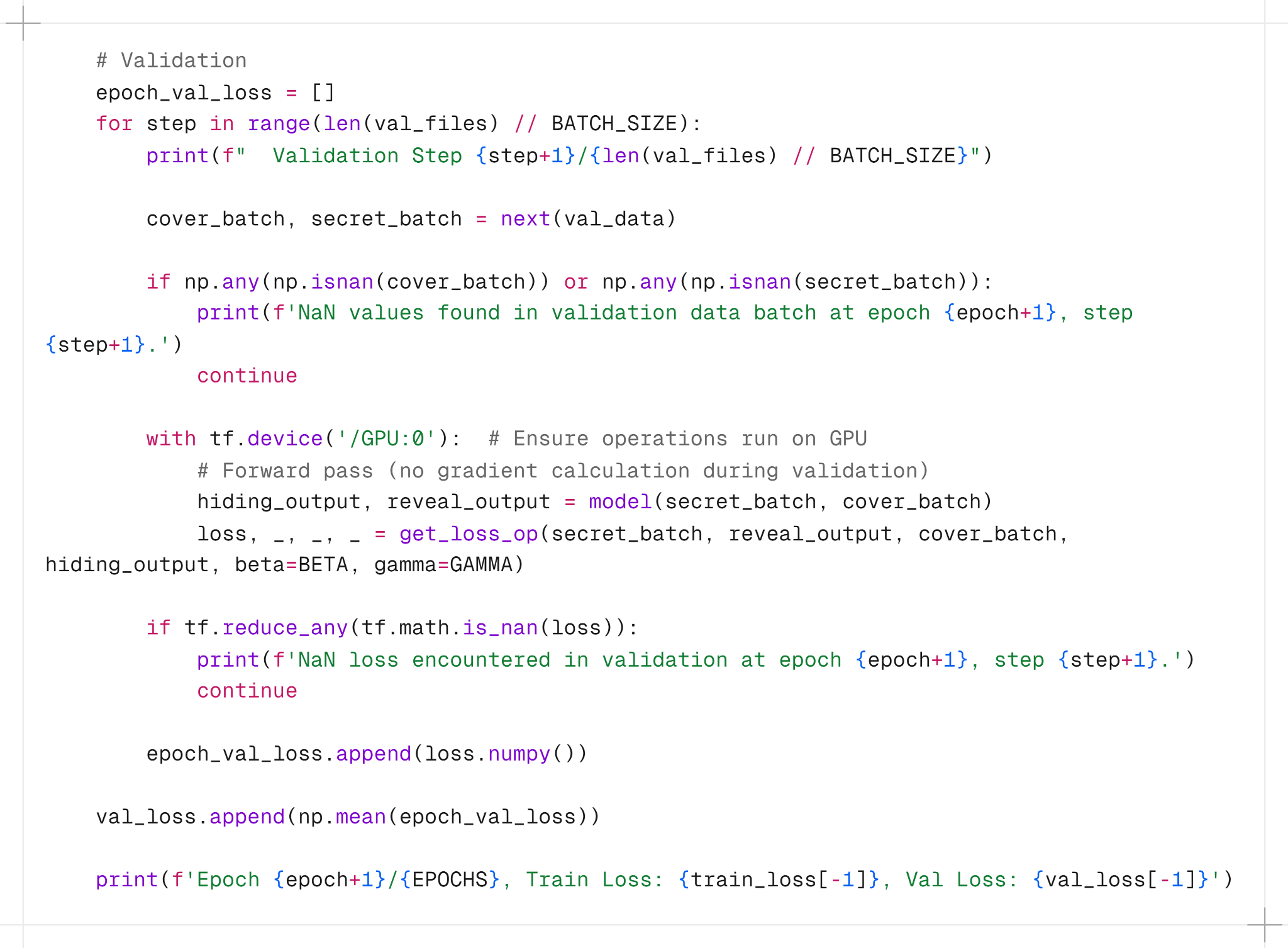
Each dataset is limited to the first 50,000 images to manage dataset size. It then loads and prepares the data using a function load\_data, creating train\_data, val\_data, and test\_data for subsequent model training and evaluation.

## 3.6 Model Training Analysis

The code snippet trains a steganography model over multiple epochs, tracking training and validation losses. It initialises empty lists train\_loss and val\_loss to store losses.

Within each epoch loop, it iterates through batches of training data (train\_data) and computes forward passes using the model.





It calculates losses using the get\_loss\_op function, handles GPU acceleration, and performs backpropagation to update model weights based on computed gradients. Validation is performed similarly using validation data (val\_data). After each epoch, it computes and stores average losses (train\_loss and val\_loss) and prints progress updates including epoch number and current losses. This iterative process facilitates model training and evaluation for improving performance over successive epochs.

# *4. Outcomes*

ASTRA integrates multiple STIX threat feeds, providing a powerful framework for detecting and mitigating cyber threats through its dashboard, visualizer tool, and log analysis capabilities.

By automating the generation of CTI and employing advanced analytics, ASTRA enhances the organizations’ ability to respond effectively to rapidly evolving cyber threats.

Some features are where the IOCs’ line of log is pinpointed by the system for ease of tracing and investigations, as in Figure 5, and it also provides a feature where the keys and passwords are stored in a secure manner protected by a master password. The keys are either only operable by a specifically arranged vendor-whitelisted public IP address, or are free of charge, so they do not require as much security features.

|  |
| --- |
| Figure 5. Log Line to pinpoint location where IOC was identified |

|  |
| --- |
| Figure 6. Master Settings Page to edit keys and passwords |

# *5. Conclusion and Future Scope*

One potential future direction involves exploring the training of a network specifically designed to recover hidden images after deployment, even in the absence of access to the original network. If adversaries manage to acquire multiple container images generated by the system, along with access to at least one component image (either the cover or the secret), they could potentially leverage this data to train a recovery network. To counter such threats, potential defences could include integrating smoothness constraints derived from classical image decomposition and blind source separation techniques.

The capability to detect the presence of hidden images, rather than precisely identifying their content, is feasible due to the significantly heightened information density compared to cover images (maintaining a 1:1 ratio). Strategies aimed at bolstering resilience include implementing techniques such as pixel permutation before concealment, which serves to diminish the perceptual similarity between the cover and hidden images. Implementing such measures necessitates retraining the system to prevent exploitation of local structures within the secret image.

This study not only expands the horizon of possibilities in steganography but also opens avenues for incorporating supplementary information into images. The system, versatile enough to handle text or audio by utilising spectrograms, consistently delivers superior visual results when embedding large, colour images. Future endeavours are aimed at evading statistical analyzers, adapting methodologies for lossy formats like JPEG using DCT coefficients, and refining error metrics such as the Structurally Similar Index Metric (SSIM) to better align with human visual perception and during training of the model.