SOFA - Steganographic Security on an FPGA

Dev Patel

Thomas Jefferson High School for Science and Technology

Objectives

This project had several objectives:

- Implement Image Steganography (Hiding either text or an image) with a Neural Network
- Deploy the model on an FPGA
- Benchmark Model Execution times

Introduction

The Internet of Things (IOT) is a rapidly growing industry, with around 31 billion devices connected and transmitting data across the internet. However with all of these devices come security risks. Current software based encryption models use well known number systems and key pairs that can be cracked. Neural cryptography doesn't use number systems or standard public private key pairs, and are synced completely by neuron weights. One subset of Neural cryptography is Steganography, the science of hiding data within other mediums. By combining the benefits of Neural Cryptography and Hardware acceleration, this project aims to create a "plug and play" encryption model for video applications on the "edge".

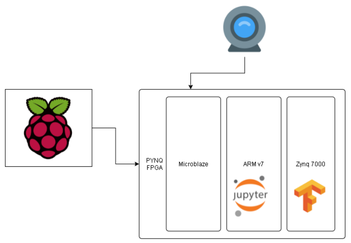

Figure 1:Original Model Overview

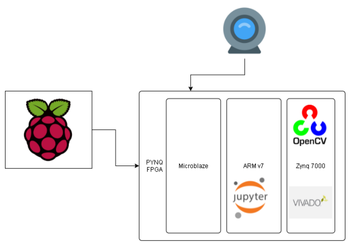

Figure 2:Final Model Overview

Materials

The following materials were required to complete the research:

- Xilinx PYNQ Z2
- Raspberry Pi 4
- MicroHDMI to HDMI Cable
- SD Card flasher

The materials were prepared according to the steps outlined below:

1. Install Vivado
2. Flash the RPi to be headless
3. Flash the PYNQ Z2 to Board version 2.4

Important Result

The Xilinx Vitis Software only had beta support for the PYNQ z2, and as a result the network couldn't be accelerated with Programmable Logic.

Neural Network Architecture

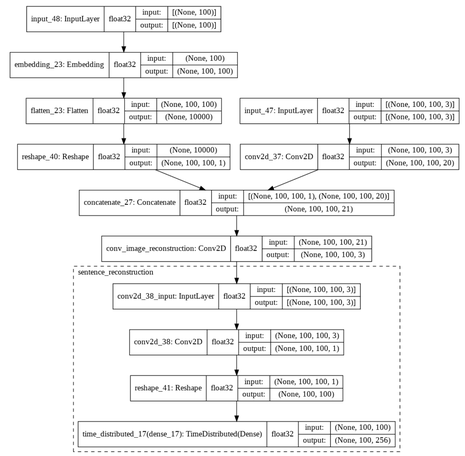

Figure 3:Figure caption

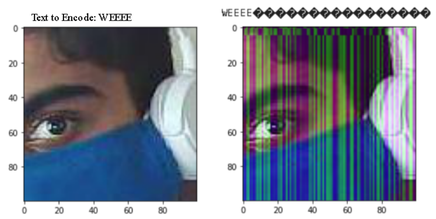

Figure 4:Neural Network Results

Methods

The first step was to design and train the Neural Network. The network was based on existing implementations of Image in Image and Text in Image Steganography models. These implementations used Tensorflow, and as a result the project was constrained around Tensorflow and the tensorflow design process. The template code was incomplete and lacked a proper data set, so a data generator was used in absence of a dataset. Due to dependency issues on the FPGA, the Neural network was abandoned, and Least Significant Bit Steganography with accelerated scalar functions to encrypt the images.

Results

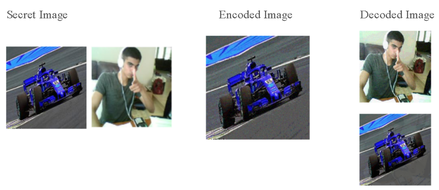

Figure 5:LSB Encryption Results

Execution Times for the LSB and the Neural Network:

Model	Response 1
LSB Encrypt	8.313
LSB Decrypt	16.097

Table 1:Execution Times

Model	Image Loss	Sentence Loss
Training	0.05	0.0001
Validation	.39	0.0001

Table 2:Loss of the Neural Network

Conclusion

Hardware acceleration can be outweighed by the cost of Data Transfer. The accelerated functions were faster than the software implementations, but the data transfer times ended up making it slower. In addition, the PYNQ z2 board had limited support, making the quantization and deployment of the Neural Network difficult and time consuming.

Mathematical Section

The Least Significant Bit Encryption Algorithm works as follows:

Algorithm 1 LSB Encryption for two NXN Images

```
for pixel − x = 1, 2, ..., N do
    for pixel − y = 1, 2, ..., N do
        for Color = 1, 2, 3 do
            Convert Cover Image at (pixel-x,pixel-y,Color) into Bit Array
            Convert Secret Image at (pixel-x,pixel-y,Color) into Bit Array
            Replace 4 Least Significant bits of Cover Image with 4 Most Significant Bits of Secret Image
        end for
    end for
end for
Return New Encoded Image
```

Loss for the Neural Network was calculated using Mean Absolute Error and Categorical Cross Entropy

$$Loss = \frac{1}{n}) \sum_{i=1}^{n} |y_i - x_i| \quad (1)$$

References

[1] Martín Abadi and David G. Andersen.
Learning to protect communications with adversarial neural cryptography.
CoRR, abs/1610.06918, 2016.

[2] Shumeet Baja.
Hiding images in plain sight: Deep steganography, 2017.

[3] Dylan Modesitt, Tim Henry, Jon Coden, and Rachel Lathe.
Neural cryptography: From symmetric encryption to adversarial steganography, 2018.

[4] P.W.D. Shshemi.
Neural cryptography.
https://github.com/shshemi/NeuralCryptography, 2020.

Acknowledgements

Thank you to Mr. Bell, Mr. DC, and Mrs. Ahn for helping me throughout my project and making sure I got my parts on time. Thank you to Dr. Schaumont, Charlotte, Shashank and Kari for helping me troubleshoot my board. Thank you to Akash for explaining Neural Networks