Feasibility and Effectiveness Analysis of Deep Learning Vision Classification Models for CameraCommunication

AbdulHaseeb Ahmed

Follow this and additional works at: https://scholarworks.gsu.edu/cs_theses

Recommended Citation
https://scholarworks.gsu.edu/cs_theses/101

This Thesis is brought to you for free and open access by the Department of Computer Science at ScholarWorks @ Georgia State University. It has been accepted for inclusion in Computer Science Theses by an authorized administrator of ScholarWorks @ Georgia State University. For more information, please contact scholarworks@gsu.edu.
Feasibility and Effectiveness Analysis of Deep Learning Vision Classification Models for Camera Communication

by

AbdulHaseeb Ahmed

Under the Direction of Ashwin Ashok, PhD

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in the College of Arts and Sciences

Georgia State University

2021
ABSTRACT

This thesis studies and evaluates Deep Neural Network models for data demodulation and decoding in a camera-based Visible Light Communication system. Camera communication is an emerging technology that enables communication using light beams, where information is modulated through optical transmissions from light-emitting diodes. This work conducts empirical studies to identify the feasibility and effectiveness of using Deep Learning models to improve signal reception in camera communication. The key contributions of this work include the investigation of transfer learning and customization of existing models to demodulate transmitted signals at the receiver end. The work expounds from a binary quantized system to a 3-bit and 4-bit quantized system. In addition to leveraging Deep Learning methods for demodulating a single VLC transmission, this thesis has developed a pipeline for integration of Deep Learning in a visual multiple-input multiple-output system where transmissions from an LED array are decoded by a camera receiver.

INDEX WORDS: Visible Light Communication, Deep Learning, Deep Neural Networks, Optical Camera Communication, Quantization, Multi-Input Multi-Output
Feasibility and Effectiveness Analysis of Deep Learning Vision Classification Models for Camera Communication

by

AbdulHaseeb Ahmed

Committee Chair: Ashwin Ashok
Committee: Anu Bourgeois, Xiaojun Cao

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University

May 2021
DEDICATION

I would like to express my sincere thanks and appreciation to my entire family but especially to my parents who have been alongside me on this journey. I know they have gone through as much of the stress of this degree as I have, therefore even the words written here cannot express the magnitude of thanks that I owe them for their support and guidance not just through this degree but throughout my life. I remember the first day I started and after the first week I wanted to quit but they advised me not to and told me that it will get better and that I will start to enjoy it soon. In the beginning, I didn’t believe them, but I kept moving forward, doing the best I could, and now I can’t deny anything they had told me. Looking back, it has been one of the scariest roller coasters of my life but the thing that prevented me from falling off has always been my parents and if it weren’t for them, I would never have been able to get to this point. Along the way, I have learned a great many lessons and I just wanted to highlight two of them here. The first being that your parents are indeed the ones that love you the most and will always give you the best advice, so just take it, and the second is that the unseen road ahead is not as scary and gloomy as it may seem, but the important thing is to never give up or lose hope and to always have faith and know that whatever happens is by the Will of Allah.

To my parents Abu and Ammi – Thanks for always being there for me and may Allah grant you the best in this world and next. Ameen

To my grandparents Mom, Dad, Nani, and Babi – Your prayers have never failed me and
I also pray to Allah to give you the best in this world and the next. Ameen.
ACKNOWLEDGMENTS

I would like to thank Dr. Ashwin Ashok for all the help and guidance he has given during my Thesis work. If I were to redo my Masters degree I would have sought out Dr. Ashok from day one because looking back I know I could have done so much more with him. Since, the first time I met Dr. Ashok, I knew he was a serious researcher who also was very compassionate and looked after his students. In the short time that I have known him, I have found him to be very helpful both on a personal level as well as on an academic level. His level of meticulousness has been something I admired because he aims to not leave any stone unturned when researching, therefore his conclusions are solid. Another thing that I would say is unique about Dr. Ashok is that he really wants you to excel; instead of “hiding” information Dr. Ashok has been very open in terms of divulging knowledge not just for the sake of sharing but also to help sharpen and refine my own skills. He has always taught me to look from the perspective of what is impossible and make it possible rather than see what is already possible and tweak it. It was actually Dr. Ashok who allowed me to join his lab without me even having to ask which is not a very common thing in a graduate-level setting. Another benefit that I derived while working under Dr. Ashok, is that given the nature of his research I have been able to learn so many different things from the practical aspects of an experimental system to the physics behind the environment in which a system works. And when you start looking at systems from that perspective you really start to appreciate the complexity. So, had it not been for Dr. Ashok, I really would not be where I am now.
Another professor I would like to thank is Dr. Anu Bourgeois. She was my first professor at Georgia State University, the rest were graduate students teaching the courses I was in. She was someone I found to be a great conveyer of knowledge as well as a great advisor. She was kind of the first one to help me navigate the road ahead at GSU and helped to get a sense of direction as to what I wanted to do and how to go about it. And like Dr. Ashok she was also someone, who on several occasions, helped me both academically and personally. So again, I would just like to thank these two amazing faculty members at GSU for making my experience as smooth as possible and for teaching me so much.
# TABLE OF CONTENTS

ACKNOWLEDGMENTS ......................................................... vi

LIST OF TABLES ........................................................ xi

LIST OF FIGURES .......................................................... xiii

LIST OF ABBREVIATIONS ................................................ xv

1 INTRODUCTION .......................................................... 1

1.1 Motivation ............................................................ 1

1.2 Problem Statement .................................................. 3

1.3 Contributions ........................................................ 4

2 RELATED WORK .......................................................... 6

3 PRELIMINARY STUDY ..................................................... 11

3.1 Setup and Data Collection .......................................... 12

3.2 Basic Thresholding based Approach ............................... 17

3.3 Deep Neural Network Background ................................. 22

3.3.1 Evaluation Overview ............................................ 22

3.3.2 VGG16 Model ...................................................... 27

3.3.3 Resnet50 Model .................................................... 29

3.3.4 Inception V3 Model ............................................... 31

3.3.5 3 Layer Convolution Model .................................... 32

3.3.6 K-means Clustering ............................................... 34

3.3.7 Convolutional Autoencoder .................................... 35

3.4 Supervised Model Results .......................................... 36

3.4.1 2 Level Full Color Model Analysis ............................ 36

3.4.2 3 Level Full Color Model Analysis ............................ 38
3.4.3 4 Level Full Color Model Analysis ..................................... 38
3.4.4 Full Color Model Analysis As a Whole ............................... 40
3.4.5 2 Level Red Channel Model Analysis .................................. 41
3.4.6 3 Level Red Channel Model Analysis .................................. 41
3.4.7 4 Level Red Channel Model Analysis .................................. 42
3.4.8 Red Channel Model Analysis As a Whole ............................. 43
3.4.9 Analysis as a Whole ..................................................... 43
3.5 Unsupervised Model Results .............................................. 45
  3.5.1 K-means Clustering Algorithm ....................................... 45
  3.5.2 Convolutional Autoencoder ......................................... 46

4  EMPIRICAL EVALUATION WITH MIMO SETUP .......................... 50
  4.1 Methodology ............................................................. 50
  4.2 Hardware Setup ......................................................... 51
  4.3 Transmitter Design ...................................................... 52
  4.4 Receiver Design ........................................................ 56
  4.5 Preliminary Results ..................................................... 59
  4.6 Model Selection ......................................................... 62
    4.6.1 Model I Data ...................................................... 63
    4.6.2 Model II Data ...................................................... 66
    4.6.3 Model III Data ...................................................... 67
  4.7 Model Training ........................................................ 68
  4.8 Model Validation ....................................................... 68
  4.9 Revised Experiment .................................................... 69

5  DISCUSSION ................................................................. 79
  5.1 Preliminary Study ....................................................... 79
  5.2 Empirical Experiment .................................................. 81
  5.3 Insights ................................................................. 82
5.4 Future Work .................................................. 83

6 CONCLUSIONS .................................................. 84

7 APPENDICES .................................................... 85
  7.1 Appendix A .................................................. 85
  7.2 Appendix B .................................................. 92
  7.3 Appendix C .................................................. 98

REFERENCES ..................................................... 108
LIST OF TABLES

Table 3.1 Raw Camera Data ................................................. 14
Table 3.2 Number of Data Points For Each LED Transmit Level in the Dataset . 16
Table 3.3 Basic Thresholding based Approach Accuracy and F1-Score Results . 21
Table 3.4 Average Basic Thresholding Based Approach Results ................. 21
Table 3.5 2 Level Full Color Model Evaluations .......................... 36
Table 3.6 3 Level Full Color Model Evaluations .......................... 38
Table 3.7 4 Level Full Color Model Evaluations .......................... 39
Table 3.8 2 Level Red Channel Model Evaluations ....................... 41
Table 3.9 3 Level Red Channel Model Evaluations ....................... 41
Table 3.10 4 Level Red Channel Model Evaluations ..................... 42
Table 3.11 K-means Model Evaluation on Raw Data ....................... 45
Table 3.12 Autoencoder/K-means Model Evaluation on Latent Space Data .... 47

Table 7.1 4 Level Full Color Background Frames Average Pixel Intensity Threshold Ranges ................................................. 99
Table 7.2 4 Level Red Channel Background Frames Average Pixel Intensity Threshold Ranges ................................................. 99
Table 7.3 3 Level Full Color Background Frames Average Pixel Intensity Threshold Ranges ..................................................... 100
Table 7.4 3 Level Red Channel Background Frames Average Pixel Intensity Threshold Ranges ..................................................... 100
Table 7.5 2 Level Full Color Background Frames Average Pixel Intensity Threshold Ranges ..................................................... 101
Table 7.6 2 Level Red Channel Background Frames Average Pixel Intensity Threshold Ranges ..................................................... 101
Table 7.7 4 Level Full Color Cropped and Resized Frames Average Pixel Intensity Threshold Ranges ................................. 102
Table 7.8 4 Level Red Channel Cropped and Resized Frames Average Pixel Intensity Threshold Ranges ................................. 102
Table 7.9 3 Level Full Color Cropped and Resized Frames Average Pixel Intensity Threshold Ranges ................................. 103
Table 7.10 3 Level Red Channel Cropped and Resized Frames Average Pixel Intensity Threshold Ranges ................................. 103
Table 7.11 2 Level Full Color Cropped and Resized Frames Average Pixel Intensity Threshold Ranges ................................. 104
Table 7.12 2 Level Red Channel Cropped and Resized Frames Average Pixel Intensity Threshold Ranges ................................. 104
Table 7.13 4 Level Full Color Cropped As Is Frames Average Pixel Intensity Threshold Ranges ................................. 105
Table 7.14 4 Level Red Channel Cropped As Is Frames Average Pixel Intensity Threshold Ranges ................................. 105
Table 7.15 3 Level Full Color Cropped As Is Frames Average Pixel Intensity Threshold Ranges ................................. 106
Table 7.16 3 Level Red Channel Cropped As Is Frames Average Pixel Intensity Threshold Ranges ................................. 106
Table 7.17 2 Level Full Color Cropped As Is Frames Average Pixel Intensity Threshold Ranges ................................. 107
Table 7.18 2 Level Red Channel Cropped As Is Frames Average Pixel Intensity Threshold Ranges ................................. 107
LIST OF FIGURES

Figure 3.1  LED Transmitter Setup ........................................ 13
Figure 3.2  4 Level Data Representation .................................. 15
Figure 3.3  Different Frame Croppings For Basic Thresholding Based Approach Evaluation .................................................. 18
Figure 3.4  Appended Layers for the Deep Neural Networks .......... 25
Figure 3.5  Effects of Staking Red Channel Upon Itself ............. 26
Figure 3.6  VGG16 Model Architecture - Image adapted from (1) .... 29
Figure 3.7  Resnet50 Model Architecture - Image adapted from (2) .. 31
Figure 3.8  Inception V3 Model Block - Image adapted from (3) .... 32
Figure 3.9  3-layer Convolutional Neural Network Model Architecture .............................. 33
Figure 3.10 Mapping of Original Data and Latent Space Representation of Data on 2D Map .................................................... 48
Figure 4.1  MIMO Setup ...................................................... 53
Figure 4.2  Packet Structure ................................................... 54
Figure 4.3  Transmitter Design Flow Chart ............................... 55
Figure 4.4  Receiver Design Flow Chart .................................... 57
Figure 4.5  Consecutive LED ON Frames ................................. 61
Figure 4.6  Hardware Versus Model Error ................................ 62
Figure 4.7  Dark Versus Light LED Frames ............................... 64
Figure 4.8  LED Average Pixel Intensity .................................. 66
Figure 4.9  Model Training Curves ......................................... 69
Figure 4.10 Model Validation Results ..................................... 70
Figure 4.11 Model Errors ..................................................... 72
LIST OF ABBREVIATIONS

VLC : Visible Light Communication
DL : Deep Learning
DNN: Deep Neural Network
OCC: Optical Camera Communication
LED: Light Emitting Diode
ML: Machine Learning
ANN: Artificial Neural Network
BER: Bit Error Rate
MIMO: Multi-Input Multi-Output
FPS: Frames per Second
Hz: Hertz
YOLO: You only look once
1 INTRODUCTION

1.1 Motivation

Though visible light communication (VLC) systems have been around since the late 19th century they had not received much research attention until the early 2000s [4]. The fundamental idea behind VLC systems is to use a light source, such as an light emitting-diode (LED), to transmit the binary representation of data by varying either the intensity or state of the LED (ON or OFF). On the receiving end is a photodetector, usually, a photodiode, which absorbs photons from the light source and converts it to an electrical current which can then be correlated back to a binary representation [5]. In this regard, the photodiode serves to demodulate the transmitted data from the LEDs at the receiver end of the system. However, because of the ubiquity and prevalent use of cameras, Optical Camera Communication (OCC), which can be considered a parallel form of VLC systems, is being considered as a replacement/additional integration in traditional VLC systems [6]. Thereby replacing the photodiode with a camera. Before divulging into the technical aspects it is important to realize the recent trend in VLC research and what that means for the future of communication networks. One of the biggest benefits of VLC systems is the huge bandwidth available, which in addition to being large is also unregulated. The reason being that the visible light spectrum operates over the frequency spectrum of 430 THz to 790 THz, compared to the radio frequency spectrum which is from 3kHz to 300MHz [7]. In addition, hardware costs are reduced as most of the infrastructure for VLC systems is already in
place, such as the LEDs. Another financial incentive is that the operating cost of LEDs is significantly lower compared to some radio hardware [8]. And not only does VLC provide financial and operational incentives, but in terms of security VLC systems also have some very unique qualities. This relates to how the light source is unable to cross objects or walls thus providing a level of security by only allowing the transmission to reach receivers that are in line of sight. Yet another benefit of VLC systems is that because they operate in in a different frequency band of the electromagnetic spectrum, they do not interfere with already in use radio waves, thus they can operate in tandem with the current radio communication infrastructure to possibly even create hybrid systems. Lastly, one of the biggest potentials of VLC systems is their ability to have much faster data speeds in the order of 10s of 100s of Gbits/sec [7]. The speed of the VLC system is directly linked to how fast both the transmitter, i.e. LEDs, and the receiver, i.e. camera, can operate. All these features are the principal catalyst for the current interest in VLC systems in the data communications field. As such, much of the effort is concentrated on optimizing these systems with better algorithms or hardware, such as the use of Machine Learning (ML) and Deep Learning (DL).

Having said that, the current state of the research still has a lot of areas unexplored or are currently under research, especially in the domain of camera-based VLC systems. Therefore the potential application use of such systems is continually being investigated. Some of the more common/current applications include simple indoor communication between LEDs and a mobile camera to vehicle to vehicle camera communication, 3D positioning, etc. However, given that cameras provide a rich feature set when capturing a frame, not to mention how
varying the individual features become when in a dynamic setting, the problem of OCC boils down to interpretability and robustness of the demodulation from the frames captured. This is where Deep Neural Networks (DNN)s come into the picture. DNNs have gained a lot of traction within the scientific and industrial community as they have been able to approximate very complex problems and can be robust across varying situations. They have allowed for the quick learning of many types of functions including nonlinear functions which makes them robust to many challenges that a regular algorithm might have when performing the same operation. This is why DNNs are also known as universal function approximators [9]. One of the biggest benefits of DNNs is their ability to internalize large input feature space vectors, like images, and compute accurate predictions. Therefore the combination of DNNs and camera technology is a frontier that this thesis aims to study.

1.2 Problem Statement

The problem that this thesis aims to address is, can Deep Neural Networks be used to successfully demodulate LED signals in an Optical Camera Communication system.

This thesis aim is to first study different DNNs and establish whether or not DNNs provide a capability in classifying an LED’s state based on the captured frame. Then the goal is to implement an camera based VLC system in which demodulation occurs through the use of a DNN. The empirical study can then be used to analyze the feasibility of DNNs to demodulate the received LED signal.

In doing so, the preliminary study is conducted to verify DNNs performance versus
a basic thresholding based approach in relation to approximating the LED signal from a binary case to a 4 symbol case. The goal is to form a baseline from which the camera based VLC implementation study can proceed. The hypothesis is that because of the nature of DNNs as universal function approximators they should be able to accurately classify the state of an LED with accuracy close to 100%. The reality of which may be different but the ultimate goal is to provide Bit Error Rates (BER) close to the standard minimum for data communication which is in the order of $10^{-13}$ [10]. This implies that out of 10 trillion bits, only 1 bit is in error.

### 1.3 Contributions

The contributions of this thesis have been in realizing that DNNs are in fact a feasible approach towards OCC system and that as a universal function approximator, DNNs have the potential of learning an LED’s state. In the binary case, DNNs can achieve close to 99% accuracy in classifying the LED state but as the number of symbols increases, the accuracy dips. Though this is far from the minimum standard of BER, the potential is there for DNNs to approximate the LED state with further training and perhaps model architecture restructure/refinement. Furthermore, this thesis has shown that VLC systems with a camera as the receiver are an application that can be contrived and considering the continual growth of camera technology, OCC with DNNs can become more efficient. However, as the empirical study proved, the hardware specifications of both the LED transmitter and the camera receiver need to be optimized to achieve better BER results. The system, as a proof
of concept, demonstrates that DNNs that are trained on an LED dataset can effectively
demodulate the LED signal but because of hardware related issues, such as synchronization
and processing power and real time inconsistencies, the system can suffer miserably. There-
fore to produce a system that is within acceptable standards requires specialized hardware
and further investigation into the DNN interpretability in order to optimize the capability
of the DNN in deciphering the LED transmission.
2 RELATED WORK

As mentioned earlier, the integration of camera technology with DNNs for the demodulation of a LED transmission has not seen much research focus. Because of this, there were only a limited number of resources available for gathering background information for this thesis. Therefore the work presented below is the closest literature found that corresponds to the area of study this thesis explores. Specifically the incorporation of any DL or ML algorithm in VLC systems, whether that be a traditional VLC system with a photodiode or an OCC system.

Much of the research relating to OCC systems that utilize DNNs relates to using the DNN as an auxiliary component rather than the means by which the demodulation of the LED signal is done. Most of the concentration is related to the synchronization between the LED and the camera as well as the technology with which the camera operates. In regards to the camera technology, the camera image sensor which usually operates in 1 of 2 ways, as a global shutter effect or a rolling shutter effect is usually studied. The difference between the two is essentially the period of time and the method of capturing a frame. Because rolling shutter cameras are more common much of the research with OCC systems is in regard to this type of camera image sensor. Because rolling shutters capture a frame row by row there is a challenge in the timely capture of the LED signal before the modality of the LED changes. To circumvent this, researchers have come up with several techniques. For instance, if the modality of the LED is faster than frame rate but lower than row by row sampling rate of the camera sensor then an enhancement of demodulation can be achieved
by interpreting the different intensities within the same frame [11]. Effectively, the frame is able to capture more data per frame. Similarly, authors of [12] show that by fitting a second order polynomial to reduce the saturation of pixels and a third order polynomial to define threshold ranges, they can effectively reduce the BER when utilizing a mobile receiver with rolling shutter effect [12]. Their system showed that within their proof of concept, the BER reduced from $3.25 \times 10^{-2}$ to $6.05 \times 10^{-4}$.

In regards to DNNs that are employed with OCC but mainly serve as auxiliary components, there exists research related to detecting the region of interest with the DNN first and then applying other techniques to interpret the LED signal. Such a system was experimented with by authors of [13] who used DNNs to find the region of interest in a vehicle to infrastructure setting. Essentially they were identifying traffic lights with DNNs to select a more stable image of the transmitter in a driving setting. The specific DNN they used was a YOLO (You only look once) model [14]. The experiment was conducted on an actual road with the vehicle moving at 20km/hr and 50 km/hr with the light source being 15 meters away before capturing the data. What they empirically showed was that the data packet reception rate increased with the use of DNNs for first processing the region of interest. The results they showed included an 8.2% increase in packet reception rate at 20km/hr and 14.7% increase in packet reception rate at 50km/hr. A lot of other research in the field of DNNs incorporated with OCC systems also parallel this area in which DNNs are more often used to detect the LED object but not serve to interpret the LED signal. Research in the area of vehicle to vehicle communication is such an avenue. For instance, research conducted
to overcome non-line of sight perception between vehicles to ensure safety measures on a roadway utilized a camera based VLC system that used DNN models for the localization of the light source rather than demodulating the light signal [15]. The researcher developed a camera based VLC system between vehicles where the vehicle in front serves to evaluate the oncoming situation and transmit codes to the vehicle behind it by modulating its own brake lights. The modulation of the front vehicle’s brake light is captured by a dashboard camera in the rear vehicle and the frames are demodulated to interpret the sent transmission. In doing so, they used a YOLO model to localize the vehicle’s brake light to identify the transmitter region of area, and then using an adaptive thresholding algorithm, discerned the LED state so that it could be mapped to bit 0 or bit 1. The transmissions were packetized and were essentially transmitting data in the form of predefined safety codes so that the rear vehicle could be aware of the events transcribing in-front of the front vehicle. This allowed for a non-line of sight perception for the rear vehicle.

As such, not a lot of research has been conducted explicitly with DNNs used to demodulate the LED signal in an OCC system. Similarly, within the scope of traditional VLCs, that don’t utilize cameras, there is a similar trend of DNNs and ML models used as auxiliary components to assist in demodulating the LED signal. Much of research in the field of VLC systems integrated with ML/DL models corresponds to positioning systems. [16] showed that with the assistance of an artificial neural network (ANN), their proposed 3D VLC based positioning system can resolve average 3D positioning error to 0.9 cm. The architecture design is a grid of multiple trilateral positioning cells and a photodiode that captures
the received signal strength from each LED and then using an ANN with two hidden layers, it determines the location of the receiver. Similarly, [17] used a photodiode for positioning schemes but instead of using an ANN they used a ML algorithm, specifically, KNN. In doing so, they found the cluster that matched closest to the measured received signal strength. The authors employ the same trilateral LED grid for their setup. In addition, they also employ ANNs, clustering algorithms, and also try a fusion of different models. The purpose of the different models/algorithms was to compare the speed and accuracy of the algorithms and in doing so they achieved accuracy in the order of a single-digit centimeter. [18] also uses ML algorithms for accurate positioning with a VLC system that uses a trilateral LED setup.

Other research that utilizes DL models with VLC systems includes the work of [19], who used Autoencoders to mitigate the LED nonlinearity for orthogonal frequency division multiplexing (OFDM)-based VLC systems. The authors performed simulations of adjusting the Field of View from 70 degrees to tilts of 5 degrees. Their model is trained to minimize the error in the recovery of the transmitted data. The closest research related to this thesis was that of [20] where the authors use Convolutional Neural Networks to filter out ambient light from a VLC receiver. Their proposed system showed an improved BER by 40% compared to a system with the same signal-to-noise ratio. [8] summarize some of the other areas of VLC research where ML/DL models are being incorporated. They describe that there are four main applications of ML/DL models in VLC systems, which include nonlinearity mitigation, jitter mitigation, modulation discrimination, and phase estimation. Models and
algorithms such as K-means, Support Vector Machines, DNNs, etc. can all be used to achieve these tasks. That being said there isn’t much else available for academic researchers to analyze before undertaking a thesis such as this. It is for this reason that the extent of the following thesis began by analyzing various models and trying to understand the underlying reasoning for why certain models worked and others didn’t and then choosing the best model for implementation in a camera-based VLC system. After further background research, no other research was found that paralleled this study which indicates that this research is unique and has the potential of opening further exploration into this niche of VLC systems.

That being said, there are other aspects of camera based VLC systems that other researchers have explored and presented results that increase the performance of camera based VLC systems. One of the main disadvantages of OCC systems is that because of the current state of camera technology the cost of hardware that can be utilized to support high speed data communication in the range of 10s of 100s of Gbits/second is very costly. Therefore techniques to parallelize the data transmission like Multi-Input Multi-Output (MIMO) systems can be developed to increase the data transmission capacity. Researchers have shown that with a MIMO OCC setup the effective transmitted data per unit time can be increase manifold, however, based on the type of camera image sensor different requirements for the setup and processing of the data are needed. [21]. Basically, using multiple LEDs to distribute the transmission can serve as effectively sending the same amount of data in a high speed VLC system with a receiver other than a camera. For this reason, this thesis has also decided to pursue this avenue in its empirical study.
3 PRELIMINARY STUDY

To determine if DNNs have any practical benefit in a camera-based VLC system, the first objective was to understand what the purpose of a DNN in a VLC system would be. As mentioned earlier, many VLC system implement DL models for positioning schemes or as an auxiliary component but the focus of this thesis was to use DNNs for demodulating the LED signal. With that in mind, the best way to utilize the DNN in a VLC system as an image classification model which was given frames of the LED from a camera. The DNN would then classify the LED within the frames by the state or intensity of the LED and convert that prediction to the received transmission. With that established the following were the goals of the preliminary study:

- Evaluate different DNNs, including transfer learning, with LED data
- Evaluate DNN model’s with binary LED data as well as varying intensities of the LED
- Compare the DNN results to a basic thresholding based approach way of distinguishing the LED state and intensity

The reason why these were the goals that the preliminary study set out to accomplish was mainly because the literature survey illustrated that no one else was exploring this area. So to thoroughly examine DNNs in this regard the breadth of the research had to be expanded. The reason transfer models were explored was because they have the ability to learn faster as they have already been trained to learn low level features that are more or less
consistent across domains. Similarly, the reason why binary LED data and quantized LED data was explored was to study how robust the DNN models are to slight changes in the LED intensity, not to mention there capability in scenarios were transmission of multiple symbols is necessary. Finally to make sure that DNNs are worthwhile, they had to be compared to a traditional or naive way of performing the same task. Which is why the basic thresholding based approach was setup and used as a reference to judge the model’s potential.

3.1 Setup and Data Collection

With those goals in mind, the next steps were to collect the data, set up the experiments, and begin processing the data. Because most of the applications of VLC systems are indoor based, it was decided to collect the LED data in an indoor setting. In doing so, a 3rd party collected the LED data for the preliminary study phase. The reason why the data had to be collected was because there were no publicly available LED datasets that are either available separately or part of another dataset. This included ImageNet which has 1000 labels yet none of them correspond to an LED or for that matter any type of light [22]. However, ImageNet does have traffic lights but the region of interest, in that case, is of the entire traffic light, i.e. housing, etc., not just the light source. The interest of this study, however, was to focus on the LED diode itself and therefore data had to be manually collected. To collect the data a prototype camera based VLC system was constructed from which frames of the LED blinking were collected. Figure 3.1 illustrates the LED setup in which a Raspberry Pi was used to control a red LED and the emission of the LED was recorded on a handheld
mobile device.

The data consisted of frames from videos of the LED which was operated at 15Hz and captured by a camera recording at 30 frames per second (FPS). Each video was recorded for approximately 15 seconds and at distances ranging from 1 meter – 7 meters in increments of 1 meter. The original frame size was 1920 x 1080 x 3 pixels, which were cropped to 128 x 128 x 3 and 32 x 32 x 3 pixels for the experiments. For the remainder of the study, 128 x 128 x 3 frames represent frames of the LED with some background and are thus called full frames, and frames of size 32 x 32 x 3 represent cropped frames of just the LED and are thus called cropped frames. Table 3.1 presents the number of raw frames in each of the 7 videos.

As mentioned in the goals, one of the aspects of the DNN exploration was to evaluate the ability of the DNN to discern between different LED intensities. What this amounts to is
Table 3.1: Raw Camera Data

<table>
<thead>
<tr>
<th>Distance</th>
<th>Number of Frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>1m</td>
<td>459</td>
</tr>
<tr>
<td>2m</td>
<td>447</td>
</tr>
<tr>
<td>3m</td>
<td>461</td>
</tr>
<tr>
<td>4m</td>
<td>447</td>
</tr>
<tr>
<td>5m</td>
<td>445</td>
</tr>
<tr>
<td>6m</td>
<td>474</td>
</tr>
<tr>
<td>7m</td>
<td>462</td>
</tr>
<tr>
<td>Total</td>
<td>3200</td>
</tr>
</tbody>
</table>

what is referred to as quantization. Basically, it means to map, or assign, more symbols to a transmission’s state than just 2 states. So for example, LED ON and LED OFF correspond to 2 states which can be mapped to bit 1 and bit 0 respectively. Similarly, if the LED intensity is subdivided then more states can be represented like LED OFF, LED slightly ON and LED fully ON. This would correspond to bits 00, bits 01, and bits 10 respectively. The point of quantization is to pack more information into the transmission or increase the relative number of characters that can be transmitted [23]. Because this part of the study was to just establish a baseline, 3 different ”Levels” were evaluated including 2 Level, 3 Level, and 4 Level. Just for reference, 4 Level corresponded to the following states, LED OFF was state 0, LED just slightly ON was state 1, LED half ON was state 2, and LED fully ON was state 3. Similarly, 3 Level corresponds to the following states, LED OFF was state 0, LED just slightly ON was state 1, and LED half ON and fully ON was state 2. Similarly, 2 Level corresponds to the following states, LED OFF and just slightly ON was state 0, and
LED half ON and fully ON was state 1. The methodology to label each frame was done by perceived intensity. The labeling of the frames was crucial for training the supervised DNNs. This process was done for each distance, 1 meter to 7 meters. Figure 3.2 depicts how 4 Level data was labeled and what each intensity/class was at each distance. The blur effects are seen as the region shown is cropped from the original image and resized to 32x32 resolution.

Once the raw data was collected and labeled it was analyzed. In doing so, it was found that the dataset was imbalanced. That there was an uneven representation of each class label, i.e. each intensity of the LED. Therefore balancing the data was crucial as it would ensure...
Table 3.2: Number of Data Points For Each LED Transmit Level in the Dataset

<table>
<thead>
<tr>
<th>Data Level</th>
<th>Raw Frames per Class Label</th>
<th>Balanced Frames per Class Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Level</td>
<td>0s = 406</td>
<td>0s = 1624</td>
</tr>
<tr>
<td></td>
<td>1s = 309</td>
<td>1s = 1545</td>
</tr>
<tr>
<td></td>
<td>2s = 940</td>
<td>2s = 1540</td>
</tr>
<tr>
<td></td>
<td>3s = 1545</td>
<td>3s = 1545</td>
</tr>
<tr>
<td></td>
<td>Total = 3200</td>
<td>Total = 6254</td>
</tr>
<tr>
<td>3 Level</td>
<td>0s = 406</td>
<td>0s = 2436</td>
</tr>
<tr>
<td></td>
<td>1s = 309</td>
<td>2s = 2472</td>
</tr>
<tr>
<td></td>
<td>2s = 2485</td>
<td>2s = 2485</td>
</tr>
<tr>
<td></td>
<td>Total = 3200</td>
<td>Total = 7393</td>
</tr>
<tr>
<td>2 Level</td>
<td>0s = 715</td>
<td>0s = 2860</td>
</tr>
<tr>
<td></td>
<td>1s = 2485</td>
<td>2s = 2795</td>
</tr>
<tr>
<td></td>
<td>Total = 3200</td>
<td>Total = 5654</td>
</tr>
</tbody>
</table>

that the DNN metric results were not biased. To do this, upsampling via data augmentation was used to increase the sample of the minority classes within each data Level. Just as a perspective, for 4 Level data, the minority classes were 0, 1, and 2; for 3 Level data, the minority classes were 0 and 1; and for 2 Level data, the minority class was 0. However, specific data augmentation techniques needed to be employed to preserve the information of each image. Therefore, only flipping and rotating the images of the minority class was performed, no shearing, darkening, etc. was performed. Table 3.2 illustrates the balanced version of each dataset for 4 Level, 3 Level, and 2 Level data.
3.2 Basic Thresholding based Approach

Before evaluating the DNN models, the basic thresholding based approach was performed to serve as a reference. The idea of the basic thresholding based approach was to compute some sort of function that could be used to effectively classify each type of frame based on the state of the LED within that frame. Essentially the idea was to have a function that could set thresholds which corresponded to each LED intensity. There were many ways of performing this analysis but just to serve as a baseline, the average pixel intensity of the image was used as the thresholding metric. Other means by which this analysis could have been done was to take the max pixel intensity or perform a masking operation over the frame. But for the purposes of this study the average pixel intensity was chosen.

The way the thresholds were set was by collecting the average pixel intensity of all the frames in a video, and then analyze the results so that groups can be formed based on the intensity of the LED. For instance, the average pixel intensity range from 0 - 125 could correspond to LED OFF and 126 - 255 could correspond to LED ON. This would be computed similarly for 3 Level and 4 Level, but with 3 threshold ranges and 4 threshold ranges respectively. Another point to mention is that the basic thresholding based approach did not use the balanced dataset, rather it worked with the raw data because the threshold was independent of the balance of the dataset. Also, the basic thresholding based approach was applied at each distance instead of being applied on all distances at once because the threshold ranges were found to be different for different distances. However, in the analysis the average across all distances is also present. Because this research area has not been fully
Figure 3.3: Different Frame Croppings For Basic Thresholding Based Approach Evaluation

researched there are many avenues of exploration that need to be assessed to make solid conclusions. Therefore, to add rigidity to the basic thresholding based approach, the color channels of the frames were also explored. In particular, the full-color frame (3 channels) was compared against just the red channel (1 channel) of the frame. In doing so, a repetition of the analysis was conducted to set the threshold ranges for just the red channel of the frame.

Another aspect that was incorporated into this analysis was comparing the LED with some background in the frame versus just a cropping of the LED. However, because the LEDcroppings were different for different distances, i.e. size of the cropped image, a 3rd case was also evaluated in which the cropped image was resized to 32 x 32 pixels. An example of each image type at each distance is presented in Figure 3.3.
The method used to get the threshold values, i.e. the intensity range, for each class label was calculated as follows. For each distance, a separate matrix was formed where each class label was listed along with its associated intensity range. To figure out what the intensity range was for each class label, a simple script was used. In this script, which was used separately for each distance, five images of each class were sampled and their average pixel intensity was calculated. This averaging method was as simple as adding up all the pixel intensities of the image and then dividing by the area of the image. This method was done for both red channel images and full-color images for all three frame types. These threshold values were then set for each class label and the entire dataset, i.e. all the frames of a video, were processed. Once the evaluation was complete an accuracy score of how well the threshold ranges fit, with respect to each class label, was generated. To get the most optimum threshold values, a loop was used to run the above-mentioned process 10 times to determine which set of threshold values produced the greatest accuracy. This was to ensure that the five random images that were sampled were not biased. Afterward, a manual evaluation of the threshold values on the data was conducted to ensure that the results were reproducible as well as the most optimal. The F1 score was also used to analyzed to determine how well the threshold values fit each class label. The use of the F1 score was a key factor in determining if the threshold range values were meaningful or not because the basic thresholding based approach utilized the raw frame data rather than the balanced dataset. This was a crucial tool as often the threshold values had to be tweaked because whole classes were misclassified even though the accuracy was nearing 100%. The accuracy
and F1 score computation were both utilized from the scikit-learn library as functions and were computed as shown in equations 3.1 and 3.2 respectively [24].

\[
Accuracy = \frac{\text{of correct prediction}}{\text{total observations}} \quad (3.1)
\]

\[
F_1 \text{ score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (3.2)
\]

The results for each distance in terms of accuracy and the F1 score are presented in Table 3.3. And as a quick summary of the basic thresholding results, Table 3.4 shows the average accuracy across all distances for each Level with each type of frame. In addition, the average pixel intensity threshold ranges for each type of frame at each distance are given in Appendix C.

Before concluding here, some points to address are that in an ideal case there should be clear distinctions between the different intensity levels of the LED, whereby clear differences are visible in the different threshold values for each class. Another intuition for the ideal case is that at any distance, these intensity distinctions should be clear. However, if there are other conditions such as environmental factors or possible resolution issues, then it would be expected that the accuracy would be best for the closest LED and worse for the furthest LED. The situations just described relate to the most ideal case and would require a pristine dataset, both in terms of how it was collected and the actual information held in the data. As the results above and the latter show, this is not the case for the dataset that was collected.
Table 3.3: Basic Thresholding based Approach Accuracy and F1-Score Results

<table>
<thead>
<tr>
<th>Data Level</th>
<th>Distance</th>
<th>Full Color Frames</th>
<th>Red Channel Frames</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Background</td>
<td>Cropped and Resized</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Background</td>
<td>Cropped and Resized</td>
</tr>
<tr>
<td>2 Level</td>
<td>1m</td>
<td>Acc - 96.08%, F1 - 96%</td>
<td>Acc - 96.51%, F1 - 97%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Acc - 96.51%, F1 - 97%</td>
<td>Acc - 97.99%, F1 - 98%</td>
</tr>
<tr>
<td></td>
<td>2m</td>
<td>Acc - 97.18%, F1 - 97%</td>
<td>Acc - 95.88%, F1 - 96%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Acc - 98.88%, F1 - 99%</td>
<td>Acc - 98.21%, F1 - 98%</td>
</tr>
<tr>
<td></td>
<td>3m</td>
<td>Acc - 96.06%, F1 - 96%</td>
<td>Acc - 98.78%, F1 - 98%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Acc - 97.55%, F1 - 97%</td>
<td>Acc - 96.97%, F1 - 96%</td>
</tr>
<tr>
<td></td>
<td>4m</td>
<td>Acc - 92.19%, F1 - 92%</td>
<td>Acc - 91.72%, F1 - 92%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Acc - 87.69%, F1 - 90%</td>
<td>Acc - 89.33%, F1 - 90%</td>
</tr>
<tr>
<td></td>
<td>5m</td>
<td>Acc - 98.35%, F1 - 98%</td>
<td>Acc - 91.98%, F1 - 98%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Acc - 84.14%, F1 - 86%</td>
<td>Acc - 80.98%, F1 - 80%</td>
</tr>
<tr>
<td></td>
<td>6m</td>
<td>Acc - 99.99%, F1 - 99%</td>
<td>Acc - 99.99%, F1 - 99%</td>
</tr>
<tr>
<td></td>
<td>7m</td>
<td>Acc - 95.45%, F1 - 95%</td>
<td>Acc - 96.97%, F1 - 96%</td>
</tr>
</tbody>
</table>

Table 3.4: Average Basic Thresholding Based Approach Results

<table>
<thead>
<tr>
<th>RGB With Background</th>
<th>RGB Cropped/Resized</th>
<th>RGB Cropped</th>
<th>Red With Background</th>
<th>Red Cropped/Resized</th>
<th>Red Cropped</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Level</td>
<td>97.37%</td>
<td>97.20%</td>
<td>97.10%</td>
<td>98.19%</td>
<td>97.55%</td>
</tr>
<tr>
<td>3 Level</td>
<td>96.90%</td>
<td>97.18%</td>
<td>96.98%</td>
<td>96.33%</td>
<td>97.17%</td>
</tr>
<tr>
<td>4 Level</td>
<td>90.41%</td>
<td>83.11%</td>
<td>83.17%</td>
<td>93.07%</td>
<td>83.44%</td>
</tr>
</tbody>
</table>

One issue with the collected dataset, which is apparent from the images in Figure 3.2, is that as distance increases the resolution degrades, which means that the information in the image becomes distorted. Another issue that was realized was that the video collection method was not as optimal as it could have been. The videos were not captured in a stationary fashion, rather the videos were captured by holding the recording device. This caused some movement in the LED’s position in each of the frames. The problem with this is that to crop the images, the LED was more often than not, uncentered which caused the pixel intensity
values to shift because there was a lack of captured information. These discrepancies in the cropping of the images can cause radical changes in accuracy and perceived light intensity. With these factors in mind, it can be asserted that the collected dataset could never fall within the ideal intuition characteristics. If none of the ideal conditions are possible, the following results are not surprising. This is further supported by observing the results, presented in Table 3.3 in which the accuracy is all over the place at each distance and is not consistent across different image types, data levels, or color channels. However, given the overall results, they do indicate that our results could reach close to 100%, both in terms of accuracy and F1 score, if they are evaluated on a pristine dataset. To correct these would mean using a pristine dataset, but for a baseline approach, it is sufficient to compare with the DNN results. With that being said, in a practical scenario our collected data is justifiable and appropriate given that in a real-world VLC system scenario, the possibility of a shaky receiver, such as a mobile device, or poor resolution is entirely possible.

3.3 Deep Neural Network Background

Before diving into the setup and experiments conducted with the DNNs, it is worthwhile to understand some of the underlying functions of these models and to understand why they were chosen.

3.3.1 Evaluation Overview

Six different models were implemented and evaluated on the collected LED data, 4 of which were supervised models and the other 2 were unsupervised models. The supervised models
included the VGG16 model, Resnet50 model, Inception V3 model, and a simple 3 layer convolution model. The reason why the first three models were chosen was because these models have shown great performance in many image classification problem. A detailed explanation of each will be given below. In order to touch all bases, the simple 3 layer convolution model was also implemented to serve as a reference between a naive model versus the three tried and tested models. The first three supervised DNN models were utilized as a transfer model, meaning that they were trained on a separate data distribution whose weights were saved with the model. The transfer models had all their previous layers frozen and the last 3 dense layers removed. This was then followed by adding 3 new dense layers and 2 new dropout layers to the end of each of these models. For reference, these models were implemented with the Keras library and the pre-trained weights were from when these models trained on the ImageNet database. Figure 3.4 illustrates the graphical representation of the VGG16 model, Resnet50 model, and Inception V3 model architecture with the appended layers. Though the input of this network is a 128 x 128 x 3 frame, which represents the full frames, this same architecture was applied for the 32 x 32 x 3 frames. Therefore, only two types of frames were evaluated with the DNN models, the full-frame which was a 128 x 128 x 3 image, and a cropped version which was resized to 32 x 32 x 3. This is different from the basic thresholding based approach in which cropped frames that were not resized to 32 x 32 x 3 were also evaluated. This was a conscious decision as the input to the model needed to be consistent and therefore without resizing the cropping, the model would simply not accept the frame. However, cropped frames used for the Inception
V3 model were resized from the 32 x 32 x 3 cropped versions to 75 x 75 x 3 because that was the smallest image size that the Keras library allowed when using their Inception V3 model as a transfer model. It should also be noted that this same architecture was used for both 4 Level, 3 Level, and 2 Level data analysis as well for full-color frames (RGB) and red channel frames. However, for the red channel analysis, the models required a 3-tuple image input and the red channel of the frame was just a single tuple, therefore the red channel frames had to be duplicated and stacked to span across all three channels. Thus, each channel had the same values, all of which were of the red channel. This did cause some changes in the image which can be seen in Figure 3.5 in which it can be observed that stacking the same channel on top of itself causes a brightening effect. Both the unsupervised and supervised models were evaluated on the balanced datasets.

As mentioned earlier, only the appended layers of the transfer models were trainable because the previous layers for the VGG16 model, Resnet50 model, and Inception V3 model were all frozen. As for the simple 3-layer convolutional model, it had all its layers set to trainable. Each time the model is compiled, the trainable layers receive a new set of weights. The new weights affect the way the model performs because of how the gradient adjusts the parameters of the model as the backpropogate. This is why if at compile-time, a “good” set of weights are acquired then the model can quickly learn the features and if the weights are “bad” then it takes longer for the model to figure out the features. But because the weights are randomly chosen at compile-time, there is no guarantee of ”good” weights. To circumvent this, each model was compiled and ran 3 times to get an average.
For each model, both Softmax and Sigmoid activations were experimented with as the last activation function for the output layer. In addition, 3 different optimizers and 3 different learning rates were explored. The optimizers included RMSprop, SGD, and Adam and for the learning rate, 0.001, 0.0001, and 0.00001 were used. The F1 score was also taken to get a more intuitive understanding of the model’s performance. Each model was trained
for 10 epochs and dropout was set to 0.2, though other dropout values were used to reduce overfitting of some models. However, the results presented later are for dropout set to 0.2. Each of the data levels had 500 frames set aside for validation and 500 frames set aside for testing, the rest were used for training. This was done for the 4 Level, 3 Level, and 2 Level balanced datasets.

As for the unsupervised models, there were only 2 models that were utilized. One was a ML algorithm and the other was DNN model whose “output” was fed back to the ML algorithm used earlier. The ML algorithm was K-means Clustering Algorithm and the DNN was a Convolutional Autoencoder whose latent space vector was fed to the K-means algorithm. In both cases, the model was just given the data, i.e. no labels because the models follow an unsupervised learning methodology. The main parameter of the K-means algorithm was the number of clusters, which corresponded to the number of intensities in the data, i.e. 4 Level, 3 Level, and 2 Level. After the algorithm ran, each cluster was assigned
an expected label by considering how much of the overall data it grouped and which class it best clustered. Then the results were tabulated by counting the number of correct samples in each cluster and dividing by the total count in the dataset. As for the Autoencoder, a Convolutional Autoencoder was used considering the data was images. To get different latent space vector representations, different architectures for the encoder part of the Autoencoder, in terms of number of layers, filter numbers, etc were used. The main idea was to determine if a particular latent space representation captured the underlying features better than the raw image. After generating the latent space vectors, the K-means algorithm was applied on the latent space vector, as before.

As a quick summary of the models used, the following sections relate some of the main features of each of the models. The information was taken primarily from the papers which first introduced these models to the ML/DL community. There is no real need to describe the simple 3-layer convolutional model except to relate that it served as more of a baseline model for the supervised models than anything else.

### 3.3.2 VGG16 Model

As per the description of [25], VGG stands for visual geometry group and 16 represents the number of layers in this version of VGG. VGG16 consists of 13 convolutional layers and 3 dense layers in addition to the input layer. Besides VGG16, there is also VGG19 which consists of 16 convolutional layers and 3 dense layers. This model was proposed by K. Simonyan and A. Zisserman from the University of Oxford. The model achieved 92.7% top-5 test accuracy with the ImageNet dataset and was submitted to the ILSVRC-2014. The
main feature of this model is its use of only 3x3 filters which allows for a deeper model due to less computational cost. Originally the VGG16 was trained on 224 x 224 x 3 images but with the Keras library, an image size above 32 x 32 x 3 can be used if the model is used as a transfer model and the last 3 dense layers are replaced. The use of smaller filter sizes consecutively gives the same result as a larger filter. For example, 2 back-to-back 3 x 3 filters is the same as using a single 5 x 5 filter, and 3, 3 x 3 filters back-to-back is the same as using a 7 x 7 filter. The benefit of this technique is that instead of using a single large filter and only being limited to one non-linear activation, there can now be 2 or more non-linear activations incorporated with a small filter size which is equivalent to a single non-linear activation with a large filter size. Another benefit of using smaller filter sizes is that the number of parameters is decreased. For instance, the authors who developed the VGG16 model cite that if the input and output of a 3 x 3 convolution stack has C channels then the stack is parameterized as $3(3^2C^2) = 27C^2$ weights. Whereas a single 7 x 7 convolution layer will require $7^2C^2 = 49C^2$ weights. The authors of the VGG16 paper cite that this action can be seen as imposing a regularization on the 7 x 7 convolution layer by forcing it to decompose through 3 x 3 filters. The main idea behind the smaller filter size is that the receptive field decreases which means more information is preserved in the output. Receptive fields are a quintessential part of convolutional neural networks as it is the region of the input space that affects a particular unit of the network, i.e. current area of the image where the kernel filter is passing over. With that in hand, the main idea behind the 3 x 3 filters is that the receptive field is smaller, fewer parameters are needed, and most importantly, more
non-linearity activations can be added by the stacking of the convolution layers which results in greater discrimination of the image. Figure 3.6 is the visual architecture of the VGG16 model.

### 3.3.3 Resnet50 Model

In [2], the authors of the Resnet50 model, discuss that Resnet stands for Residual Networks and the 50 stands for 50 layers deep. It was the winner of the ILSVRC 2015 which achieved 3.57% error on the ImageNet test set. The principle idea behind this model was the intuition that a deeper model should perform better. The idea is that if you have 2 networks, one network with say 5 layers and another with 10 layers and the first 5 layers of the 10 layer network are the same as the 5 layer network and the rest are just identity layers, then the 10 layer network should perform better because it is a deeper model and should be able to learn the features much better than the 5 layer model. However, in practice, the m layer model performs worse. This is something the authors of the Resnet paper make note of and describe in greater detail. In theory, as you pass through a DNN the initial layers learn lower level features like lines and edges, and then intermediate layers learn medium level features like shapes, and then upper layers learn high-level features such as whole faces, or entire objects. However, the problem with deep models is the vanishing/exploding
gradient issue and also the degradation problem. The vanishing gradient problem is that as 
backpropagation occurs, the update to each of the weights becomes smaller and smaller to 
the point that the initial layer’s weights are so insignificantly updated that the model does 
not improve. The degradation problem is when deep models start to converge, and accuracy 
gets saturated and then degrades rapidly, however, this problem is not due to overfitting.

The main concept of Resnets is the skip connection, also known as a residual block. The 
purpose of the skip connection is to mitigate the vanishing gradient problem by providing 
an alternate path for the gradients to pass back through. Also, it allows the model to 
learn an identity mapping which makes sure that higher layers do just as good as the lower 
layers and not worse. Basically, the purpose of the skip connection is to create an identity 
mapping whereby the input of the first convolutional layer is passed to the output of the 
second convolutional output but before the non-linear activation is applied. Therefore, the 
idea is to learn $F(x)$, i.e. the residual function, rather than $H(x)$ which is equal to $F(x) + x$. By letting the residual block learn the function $F(x)$ the accuracy increases. The concept 
of skip connections itself is not new, they have been used in highway networks, but the 
implementation is different in Resnet. Another technique employed with Resnet50 is batch 
normalization used after each convolution and before each activation. Batch normalization 
allows each layer of a network to learn by itself a little bit more independently of other 
layers. It reduces overfitting because it has a slight regularization effect. Similar to dropout, 
it adds some noise to each hidden layer’s activations. The use of identity connections does 
not carry with it extra parameters which means the computation complexity for simple deep
networks and deep residual networks is almost the same. Figure 3.7 is the model architecture of Resnet50. The solid arrows are for an identity connection. The dashed arrows are for convolution operation in the Residual Block where the stride is 2.

### 3.3.4 Inception V3 Model

In [3] describe the Inception V3 model which is the 3rd version of the Inception model. It has 42 layers and is a sparsely connected network and uses a special technique that not only makes the model deep in length but wide as well. The idea is to have the model adapt by trying different filter sizes on the dataset. The principle component is called the inception block which takes the previous layer’s output and passes them through several filter sizes before it concatenates them all and passes it on to the next layer. However, unlike traditional methods, where the data passes right through the filter size no matter the size, the inception block passes the previous layer’s data first through a 1 x 1 filter which acts as a bottleneck, then through the different filter sizes like 3 x 3 and 5 x 5, etc. This effectively reduces the number of parameters that need to be trained. This property of the inception model is called factorized convolutions. This process also employs the concept of using smaller filter sizes to produce the same effect as larger filters as VGG models do. In addition to having different filter sizes, a max-pooling layer is also placed within these inception blocks. At the end of
each inception block, the results of each filter operation are concatenated. Each inception block computes a different filter size but with the same dimension, including the pooling layer which uses padding to produce the same size as the input. Batch Norm and ReLU are used after each convolutional operation. Another feature of the Inception V3 model is the use of auxiliary classifiers which act as a small CNN. They are inserted between layers during training, and the loss incurred is added to the main network loss. Figure 3.8 depicts the architecture of the Inception V3 block which is where the model adapts via different filter sizes.

### 3.3.5 3 Layer Convolution Model

The simple 3-layer convolutional neural network architecture that was implemented consisted of 3 convolutional blocks, which utilized the ReLU activation function, as well as a max-pooling layer. The output of the convolutional blocks is flattened and then fed through 2
dense layers with 1 dropout layer. This model was the baseline model for the supervised model analysis. The architecture for this model can be seen in Figure 3.9 which was used for 4 Level, 3 Level, and 2 Level data, full frames (128 x 128 x 3), cropped frames (32 x 32 x 3), color frames, and red channel frames.
3.3.6 K-means Clustering

K-means Clustering is by no means a new algorithm. It was first proposed by Bell Labs in 1957 but wasn’t published until 1982, but by 1965 Edward W. Forgy published the same algorithm [26]. K-means is an unsupervised learning technique whereby the data passed to it is entirely clustered into how many ever clusters are established [27]. Meaning that all data points in the data set are grouped around a cluster based upon some similarity metric to the cluster centroid location, usually distance calculated via euclidean distance. Initially, the cluster centroid locations are randomly picked. Then the distance of each data point to each centroid is calculated. The data point is then grouped with the centroid closest to it, relative to the other centroids. Then the mean of each cluster is computed, relative to all data points belonging to that cluster, which is the new location of the cluster centroid. Then the process repeats till the centroids are within some epsilon of the old value. Since the centroids are initially located randomly, the variance relative to the data points in each cluster can be asymmetric, therefore K-means clustering algorithm is usually repeated several times to find the best locations for the centroids, i.e. best variance spread amongst all K-means runs. Since none of the labels are passed to this model, the cluster IDs don’t actually correlate to the actual value of the true label, therefore post-processing is needed to relate each cluster ID to a true class label.
3.3.7 Convolutional Autoencoder

Autoencoders in general are a data compression algorithm and have been around for decades for the use of data compression [28]. They are a special type of neural network that both compress and decompress an input. Autoencoders are made up of 2 main parts, the encoder and the decoder. The encoder compresses an input to some latent space vector representation, and the decoder tries to reconstruct the original input from the latent space representation of the input. Autoencoders are data-specific, meaning they only work with similar data with which they were trained, lossy meaning the compressed representation is degraded, and they learn automatically from the data as long as the data is well defined. There are many variations of Autoencoders from simple Autoencoders, Sparse Autoencoders, Deep Autoencoders, Convolutional Autoencoders, to Variational Autoencoders. Since images are the primary object in this research the use of Convolutional Autoencoders made most sense. The Convolutional Autoencoder consists of two neural networks, a network of convolutions and max-pooling to compress the input and a network of convolutions and upsampling to reconstruct the image [29]. However, with all Autoencoders, the idea is to properly compress the input, the compression can be anything the user defines. So in the case of images, the encoder is fed an image that has dimensions equal to the number of pixels. So given a 10 x 10 x 3 image it means the data is 300 dimensional. And the compression of this input can be anything for the most part. Usually, you don’t want to compress too much, such as a 500:1 compression but something reasonable such as 5:1, but even that is dependent on the data. Another point to mention here as well is that before training the Convolutional
Autoencoders, normalize the data, thereby representing each pixel in the image as a value between 0 and 1. Then train the model and set the loss function to binary cross-entropy. This may seem odd but it is an acceptable loss function for many Autoencoders architectures as long as the data is normalized. The loss will usually never reach close to 0 or even 0 if the values of the input are not 0 or 1 exactly. If the values fall between 0 and 1, then the value of the loss function would generally never be 0. Therefore, the loss value serves more as a measure of when training is complete, which is indicated by the plateauing of the loss value. Therefore, the main metric for evaluating the Autoencoder is the reconstructed image versus the input image. Metrics such as peak signal to noise ratio (PSNR) and structural similarity index (SSIM) can be used to evaluate the two metrics.

3.4 Supervised Model Results

The following sections describe the supervised model results for both the full color frames as well as the red channel frames.

3.4.1 2 Level Full Color Model Analysis

Table 3.5: 2 Level Full Color Model Evaluations

<table>
<thead>
<tr>
<th>Model</th>
<th>Softmax Full Image</th>
<th>Softmax Cropped Image</th>
<th>Sigmoid Full Image</th>
<th>Sigmoid Cropped Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>98.89%</td>
<td>98.69%</td>
<td>97.99%</td>
<td>96.60%</td>
</tr>
<tr>
<td>Resnet50</td>
<td>95.60%</td>
<td>68.20%</td>
<td>81.00%</td>
<td>57.00%</td>
</tr>
<tr>
<td>Inception V3</td>
<td>76.80%</td>
<td>51.23%</td>
<td>63.30%</td>
<td>39.34%</td>
</tr>
<tr>
<td>3 Layer Convolution</td>
<td>97.56%</td>
<td>93.24%</td>
<td>94.20%</td>
<td>99.00%</td>
</tr>
</tbody>
</table>
From the results presented in Table 3.5 it is clear that for the most part, the VGG16 model performed best on 2 Level, full-color data, followed by the 3 layer convolution model. The results for the VGG16 model show that the accuracy was nearing 99% consistently and was well above 95%, with the different frame types and activation function for the output layer. In terms of which activation function performed best in the output layer, it is clear that the Softmax function performed better than the Sigmoid function, across all model types, though technically in a binary case, Sigmoid is just a special case of Softmax. In terms of what type of frame performed better, full frames or cropped frames, the full frames evaluated better. The reason why this might be the case is that the full frames distinguished the LED clearly whereas the cropped frames were completely polarized by the LED. Therefore, the LED was more clearly segregated in the image and its feature was, therefore, more easily learnable. However, further exploration into what the models are actually learning in each layer is needed to verify this intuition. Considering that the common bit error rates (BER) for data communication systems are $99 \times 10^{-13}\%$ or greater, it would be more appealing to see results closer to 99.999% and above. However, considering the models achieved this level of accuracy with only 10 epochs of training and less than 10,000 frames of training data, begs the question of what the potential could be. By tweaking the hyperparameters and other parameters and perhaps training the model for a longer duration than 10 epochs, as well as training on a larger and more pristine dataset might allow the BER to be closer to the ideal.
Table 3.6: 3 Level Full Color Model Evaluations

<table>
<thead>
<tr>
<th>Model</th>
<th>Softmax Full Image</th>
<th>Softmax Cropped Image</th>
<th>Sigmoid Full Image</th>
<th>Sigmoid Cropped Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>98.00%</td>
<td>96.79%</td>
<td>95.99%</td>
<td>96.60%</td>
</tr>
<tr>
<td>Resnet50</td>
<td>77.60%</td>
<td>53.20%</td>
<td>79.00%</td>
<td>50.00%</td>
</tr>
<tr>
<td>Inception V3</td>
<td>42.80%</td>
<td>32.23%</td>
<td>31.60%</td>
<td>32.82%</td>
</tr>
<tr>
<td>3 Layer Convolution</td>
<td>94.99%</td>
<td>94.40%</td>
<td>91.20%</td>
<td>93.00%</td>
</tr>
</tbody>
</table>

3.4.2 3 Level Full Color Model Analysis

From the results presented in Table 3.6, it is clear that the VGG16 model performed best with accuracy well above 95% for both cropped and full-sized frames and for both Softmax and Sigmoid activations in the last layer of the VGG16 model. As for the second-best model, it is the 3-layer convolution model with an accuracy well above 90% and below 95%. As a whole, full-size frames performed better than cropped frames, especially with Softmax as the activation function in the last layer. In comparison to 2 Level, full-sized frames, the results for 3 level, full-color frames, model evaluations were much less accurate than 2 level data. This was actually something that was expected because as you increase the quantization of the system you are also increasing the risk of mistakes in interpreting the received signal. However, 3 level data seems promising given a more refined model with perhaps a large training session and larger training data size.

3.4.3 4 Level Full Color Model Analysis

From the results presented in Table 3.7, it is clear that the VGG16 model again performed the best, however, the VGG16 model only performed over 90% when the activation function
Table 3.7: 4 Level Full Color Model Evaluations

<table>
<thead>
<tr>
<th>Model</th>
<th>Softmax Full Image</th>
<th>Softmax Cropped Image</th>
<th>Sigmoid Full Image</th>
<th>Sigmoid Cropped Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>93.50%</td>
<td>95.99%</td>
<td>75.59%</td>
<td>73.79%</td>
</tr>
<tr>
<td>Resnet50</td>
<td>69.99%</td>
<td>35.60%</td>
<td>66.00%</td>
<td>34.40%</td>
</tr>
<tr>
<td>Inception V3</td>
<td>37.59%</td>
<td>35.19%</td>
<td>28.00%</td>
<td>23.00%</td>
</tr>
<tr>
<td>3 Layer Convolution</td>
<td>50.59%</td>
<td>75.00%</td>
<td>70.99%</td>
<td>74.59%</td>
</tr>
</tbody>
</table>

of the output layer of the model was set to Softmax. Interestingly, the pattern of the VGG16 model performing the best regardless of the activation function of the output layer did not hold for this dataset as it did with the previous 2 datasets. There was about a 20% reduction in accuracy when Sigmoid was used as the activation function for the output layer of the VGG16 model, whereas for the 2 previous datasets, 2 Level and 3 Level full-color frame datasets, the accuracy reduction was roughly 5% or so between Softmax and Sigmoid. This is probably because Sigmoid is not meant to handle multiclass classification. Similar to 3 Level and 2 Level data, the second-best model was the 3 layer convolution model. Also similar to 3 level and 2 level model evaluation, full frames did better than cropped frames, with the exception of the 3 layer convolution. Like 2 Level and 3 Level data, Softmax in the output layer performed better than Sigmoid in the output layer, on average across all models. In terms of quantization, 4 level data will need to be further reviewed and refined to guarantee at adequate data transmission.
3.4.4 Full Color Model Analysis As a Whole

As for all the color frame results as a collective, 2 Level data performed the best followed by 3 Level data, followed by 4 Level data. There are several reasons why but the one that is the most compelling is that the intensity level between some class labels might have been too minute that it was difficult for the model to distinguish between them. As for Resnet50 and Inception V3 models, these two models performed pretty poorly across all 3 level types. One thing to note which is not described in the tables above is that the training accuracy for these 2 models would consistently reach an accuracy of 90+, and validation/test set accuracy of less than 40%. Basically, these models were overfitting more often than not. As mentioned earlier, these models all had their dropout layers set to 0.2, i.e. “drop” 20% of the neurons in the layer. However, after collecting the results above, a sample of dropout values were used for both Inception V3 and Resnet50 which showed as dropout increased the model stopped overfitting, but the validation/test set accuracy remained the same, which is why those results are not presented here. Another intuition/idea that explains the results for the Inception V3 model and Resnet50 model’s poor classification results is that the size of the LED dataset was too small as these model require a lot of data to learn the granularities in the data distribution, especially Inception V3 which uses adaptive learning via filter sizes. And perhaps, having a larger a dataset might improve these model’s performance.
Table 3.8: 2 Level Red Channel Model Evaluations

<table>
<thead>
<tr>
<th>Model</th>
<th>Softmax Full Image</th>
<th>Softmax Cropped Image</th>
<th>Sigmoid Full Image</th>
<th>Sigmoid Cropped Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>96.29%</td>
<td>92.89%</td>
<td>95.79%</td>
<td>95.30%</td>
</tr>
<tr>
<td>Resnet50</td>
<td>91.20%</td>
<td>62.20%</td>
<td>74.00%</td>
<td>58.00%</td>
</tr>
<tr>
<td>Inception V3</td>
<td>71.80%</td>
<td>49.76%</td>
<td>61.80%</td>
<td>35.62%</td>
</tr>
<tr>
<td>3 Layer Convolution</td>
<td>91.56%</td>
<td>91.31%</td>
<td>91.70%</td>
<td>98.00%</td>
</tr>
</tbody>
</table>

### 3.4.5 2 Level Red Channel Model Analysis

From the results presented in Table 3.8, the VGG16 model performed best overall, though the 3 layer convolution model followed closely behind. The results were well above 90% but less than 97%. This was the exact situation as the 2 Level full-color frame evaluations however, the result for the red channel frames were not as good. Similarly, the full frame evaluations performed better than the cropped frames of the LED. However, in terms of the output layer’s activation, there is not a definite answer but Softmax activation seems to have the edge. Further intuitions of why the red channel evaluations performed worse than the full color frames will be explained later.

### 3.4.6 3 Level Red Channel Model Analysis

Table 3.9: 3 Level Red Channel Model Evaluations

<table>
<thead>
<tr>
<th>Model</th>
<th>Softmax Full Image</th>
<th>Softmax Cropped Image</th>
<th>Sigmoid Full Image</th>
<th>Sigmoid Cropped Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>97.60%</td>
<td>92.00%</td>
<td>91.20%</td>
<td>96.60%</td>
</tr>
<tr>
<td>Resnet50</td>
<td>75.00%</td>
<td>43.99%</td>
<td>72.79%</td>
<td>62.00%</td>
</tr>
<tr>
<td>Inception V3</td>
<td>33.00%</td>
<td>53.79%</td>
<td>43.99%</td>
<td>42.19%</td>
</tr>
<tr>
<td>3 Layer Convolution</td>
<td>86.00%</td>
<td>92.40%</td>
<td>65.20%</td>
<td>42.19%</td>
</tr>
</tbody>
</table>
From the results presented in Table 3.9, it is clear that the VGG16 model again performed the best with accuracy well above 90% for both cropped and full-sized images and for both Softmax and Sigmoid activations in the output layer. As for the second-best model, it is the 3-layer convolution model, however, there are inconsistencies in the accuracy of the 3-layer model when different activations are used and when different frame sizes (full or cropped) are used. In addition, the best performing activation function for the output layer is Softmax for the most part, which is similar to 2 Level full-color frames. Compared to 3 Level full-color data, the red channel performed worse, and this will, again, be explained below.

3.4.7 4 Level Red Channel Model Analysis

Table 3.10: 4 Level Red Channel Model Evaluations

<table>
<thead>
<tr>
<th>Model</th>
<th>Softmax Full Image</th>
<th>Softmax Cropped Image</th>
<th>Sigmoid Full Image</th>
<th>Sigmoid Cropped Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>93.40%</td>
<td>92.00%</td>
<td>62.80%</td>
<td>78.79%</td>
</tr>
<tr>
<td>Resnet50</td>
<td>56.40%</td>
<td>39.39%</td>
<td>50.00%</td>
<td>37.99%</td>
</tr>
<tr>
<td>Inception V3</td>
<td>31.79%</td>
<td>32.19%</td>
<td>24.00%</td>
<td>33.90%</td>
</tr>
<tr>
<td>3 Layer Convolution</td>
<td>79.79%</td>
<td>78.20%</td>
<td>69.80%</td>
<td>72.20%</td>
</tr>
</tbody>
</table>

From the results presented in Table 3.10 it is clear that the VGG16 model again performed best and the second best was the 3-layer convolutional model. In terms of frame size, there doesn’t seem to be any clear winner. In addition, the best performing activation for the output layer is Softmax. Similar to 3 Level and 2 Level data for red channel analysis, the results for 4 Level red channel are also worse compared to 4 Level full-color frame results.
3.4.8 Red Channel Model Analysis As a Whole

Some key points to take away from the red channel data as a whole is that the 2 Level data performed best followed by 3 Level data followed by 4 Level data. Again, for the same reasons as stated above, it seems the data with 4 Levels is harder to differentiate since the intensities 2 and 3 are so close in perceivable intensity as well as actual intensity. Similarly, the Resnet50 and Inception V3 model performed poorly possibly because of the lack of data and other reasons stated in the full color frame analysis.

3.4.9 Analysis as a Whole

As was noted, red channel model evaluations performed poorly compared to full color model evaluations. There are many possibilities as to why but the most crucial seems to be the fact that by multiplying and stacking the red channel across all 3 channels of the image, it distorted the contrast of the image. Basically, it brightened the image much more than an image with just the red channel. This in turn seems to distort the information that the models tried to extract. The difference between the full color frames, regular red channel frame, and red frame multiplied across 3 channels can be observed in Figure 3.5.

In an ideal case, the three major DNNs should have performed well above 90% on the validation/test set. In addition, the least accurate model should have been the simple 3-layer convolution as it did not have any special techniques nor was it a deep model. In any case, if the Resnet50 and Inception V3 model are disregarded, which were hypothesized to have failed because of lack of data, the ideal case holds for VGG16 and the simple 3-layer convolution.
With that in mind, it can also be said that in terms of neural network theory the Resnet50 model should have performed the best as it is the deepest. This would be followed by the Inception V3 and then the VGG16 model and lastly the 3-layer model. This scenario is only if the criterion for success is based on how deep the network is. Another point to make is that perhaps the VGG16 model performed well because of its unique architecture wherein only 3 x 3 filters are used and the fact that more non-linear activations are used per step in the model. This would actually be pivotal because the data represents a concentrated area within the image. Thus, to extract the most relevant information would mean that a small receptive field would be best to employ. Intuitively, it would seem that the red channel evaluations should have done better as the LED was a red LED therefore the most relevant information is in the red channel of the image, however, this was not the case.

In terms of how well the DNN supervised models did compared to the basic thresholding based approach, it is somewhat unclear. The threshold method evaluated each distance where as the models evaluated all distances at once. However, on a more general scale, the full color frame model classification seemed to be on par with the thresholding method for full color frames. However, the threshold method outperformed the models when it came to the red channel analysis. This again probably relates to the scenario described above about how the red channels were evaluated, i.e. multiplying the red channel across all 3 channels. Another intuition that seemed to fail was that the models, when evaluating the cropped images, should have been more robust to the LED not being centered than the threshold method, yet they still performed worse.
Concerning quantization, the experiments conducted do show that more symbols in data transmission are possible when the receiver is a camera, however, the models have to be trained better or entirely new models need to be used. Because as the number of symbols increases the accuracy decreases. In addition, better resolution frames that can distinctly separate the different intensities of the LED need to be used. Without actually being able to separate those intensities it is not possible to perform quantization in VLC systems when the receiver is a camera. Up to this point, only 2, 3, and 4 symbols were experimented with and already the accuracy dipped about 5% - 10% when the the number of symbols increased from 2 to 4 levels. Therefore to get symbols of upto 255 or a data transmission rate of $2^8$ would be impossible without major modifications to the current model performances.

3.5 Unsupervised Model Results

3.5.1 K-means Clustering Algorithm

Table 3.11: K-means Model Evaluation on Raw Data

<table>
<thead>
<tr>
<th></th>
<th>100 Iterations Full Images</th>
<th>100 Iterations Cropped Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Level Full Color</td>
<td>59.76%</td>
<td>86.74%</td>
</tr>
<tr>
<td>2 Level Red Channel</td>
<td>54.17%</td>
<td>88.79%</td>
</tr>
<tr>
<td>3 Level Full Color</td>
<td>38.28%</td>
<td>72.47%</td>
</tr>
<tr>
<td>3 Level Red Channel</td>
<td>38.14%</td>
<td>74.54%</td>
</tr>
<tr>
<td>4 Level Full Color</td>
<td>32.49%</td>
<td>64.15%</td>
</tr>
<tr>
<td>4 Level Red Channel</td>
<td>33.99%</td>
<td>61.22%</td>
</tr>
</tbody>
</table>

As mentioned earlier the cluster IDs are not actually correlated to the true class label,
therefore post-processing is needed to identify what cluster correlates to what class label. For the purposes of experimentation, the percentage of the total data each cluster grouped is computed, then the percentage of class samples each cluster contains is computed. Then to correlate the cluster ID with its true class label the most populated cluster is evaluated and assigned the label based on which class samples are a majority in that cluster. The K-means algorithm ran for 100 iterations so that the randomness of the initial cluster centroids is not biased. From the results presented in Table 3.11, it is clear that the accuracy of the algorithm is quite poor with accuracy ranging between 30% - 55% for full-color frames and 60% - 90% for cropped frames. This is actually a different scenario from the supervised models which usually performed better with the full frames compared to the cropped frames. On the other hand, similar to the supervised models, as the number of symbols increases from 2 to 4 the accuracy decreases. In terms of red channel frames and full-color frames, there seems to be no difference in accuracy per symbol count. So as an unsupervised learning model it doesn’t compare well with the tested supervised model.

3.5.2 Convolutional Autoencoder

The results presented in Table 3.12 show the best latent space vector clustering of the Convolutional Autoencoder model. Basically, after trying several latent space representations, those in 3.12 were best classified after being fed to the K-means algorithm. Since the main objective of using the Convolutional Autoencoder was to utilize the latent space vector to determine better separation of the classes, the latent representations were fed back through the K-means pipeline and evaluated. To make sure that the latent space vector’s repre-
sentation was accurate, in the sense of capturing the features of the original image, several different architectures for the Convolutional Autoencoder model with varying dimensions for the latent space vector were tested. Compressions of 1.5, 3, 6, 12, 24, 96, and even 768 the size of the original input image with varying Autoencoder architecture were analyzed. However, none of the representations actually improved the accuracy of the clustering, i.e. no representation was able to accurately separate the different classes based on the features extracted. In addition to statistically measuring the difference in separability between the two representations, the original image clustering and the latent space vector clustering, dimensionality reduction tools such as Principle Component Analysis and t-SNE were employed to map the feature spaces to a 2D representation for analysis.

Figure 3.10 shows the mapping of the original data versus the latent space vector mapping on a 2D map. The latent space representation used here was from the architecture that performed the best when passed through the K-means pipeline. The first row shows the two representations mapped via PCA with the two largest principle axeses, the second row shows

<table>
<thead>
<tr>
<th></th>
<th>16 x 16 x 8 Latent Vector Full Images</th>
<th>4 x 4 x 8 Latent Vector Cropped Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Level Full Color</td>
<td>59.04%</td>
<td>87.83%</td>
</tr>
<tr>
<td>2 Level Red Channel</td>
<td>50.21%</td>
<td>86.68%</td>
</tr>
<tr>
<td>3 Level Full Color</td>
<td>44.62%</td>
<td>57.93%</td>
</tr>
<tr>
<td>3 Level Red Channel</td>
<td>48.23%</td>
<td>70.70%</td>
</tr>
<tr>
<td>4 Level Full Color</td>
<td>33.26%</td>
<td>66.69%</td>
</tr>
<tr>
<td>4 Level Red Channel</td>
<td>36.63%</td>
<td>63.86%</td>
</tr>
</tbody>
</table>
Figure 3.10: Mapping of Original Data and Latent Space Representation of Data on 2D Map

the mapping of the two representations with t-SNE, and the third row shows the mapping with PCA and t-SNE combined. What the figure shows is that even with the latent space representation, there isn’t any better separation of the classes. This is verified by the fact that both plots, for each data level, are almost identical. Another interesting point here is that even though the latent space representation was not accurate, the reconstructed image generated by the Autoencoder model was very similar to the input image. We verified this by using image processing metrics such as PSNR and SSIM which showed that the reconstructed image, very closely matched the input image. Overall, the results show that the latent space representation did no better in being clustered than the original images. One last important note about the dimensionality reduction mapping is that it shows that there is a lot of overlapping of the different classes which is why some of the models, especially K-mean
the Autoencoder performed poorly. This can also partially explain the reason why when performing the basic thresholding based approach, there was often overlapping between the intensity ranges which had to be manually corrected.
4 EMPIRICAL EVALUATION WITH MIMO SETUP

After completing the preliminary study, the next phase of this thesis is to empirically study the performance of DNNs in a camera based VLC. The purpose of which would expound to not only verifying the DNNs performance but also designing the architecture, the transmitter and receiver, that would be necessary if such a system were to be utilized in an actual setting.

4.1 Methodology

The goals of this empirical evaluation are as follows:

- Create a fully functioning/scalable MIMO VLC system which employs DNN vision model classification
- Analyze the effectiveness of the DNN model in the system

So now the question is what is MIMO and why implement it for VLC. MIMO is a methodology whereby multipath propagation is achieved. Effectively, transmitting more data per unit time, i.e. increasing transmission capacity. This is achieved by parallelizing the transmission over 2 or more LEDs. The total transmission time in a MIMO setup is related by \(1/n\) where \(n\) is the number of LEDs. Additionally, MIMO can be used for transmission redundancy.
4.2 Hardware Setup

For the design and setup, the empirical experiment was conducted as follows. 4 red light-emitting LEDs were placed in a 2 x 2 grid, where each LED was separated by a distance of 3 inches in both the horizontal and vertical direction as shown in Figure 4.1 (E). It was then decided to use off-the-shelf devices to control the LEDs. A single Raspberry Pi 4 was used to operate both the LEDs, which served as the transmitter and the Raspberry PiCamera V2, which served as the receiver. This can be seen in Figure 4.1 (B) and (D). Since the preliminary study was complete the best classifying model was chosen and retrained with new data and served as the demodulator for this setup. New data was collected because the setup was in a new environment and new hardware was used to capture the data. More on that in a later section. Because this was an empirical study and not a full-blown product design, the demodulation was done offline in order to process, analyze, and not overburden the Raspberry Pi. Additionally, because LED localization can be a real problem, the MIMO setup was made static and was aided with the use of ARUCO markers for consistently locating the LED within a frame. ARUCO markers are synthetic square markers made up of a black border and a generated inner portion which is a specific matrix of black and white squares [30]. They are similar to QR codes but for object detection and location. These markers work with python's OpenCV library where built-in functions can be used to identify specific markers and return the coordinates of the marker within the frame. What the markers allowed for was a means by which each of the 4 LEDs in the setup could be segregated and processed independently of the other LEDs. For this, the setup was constructed to have
2 ARUCO markers on the top right corner and bottom left corner of the LED grid, as shown in Figure 4.1 (A). Then the location of the corners of each marker can be determined from which 4 quadrants of equal size can be made around the 4 LEDs. These quadrants would serve as the cropping dimensions for each LED so that the LED was centered, more or less, within each cropping, consistently. As for the operating frequency of the LED and camera, it was decided that the camera should double sample the LED, however, to have the camera operate at a workable level with a decent resolution, it was decided that the camera would operate at 60 FPS at 1280 x 720 resolution. This would force the LED to operate at 30 Hz for double sampling at the receiver. In an ideal VLC system, the LED would operate at a frequency that the human eye would not be able to perceive, Critical Flicker Frequency, which is approximately 100 Hz, but because of the chosen hardware’s limitations, these were the best values for operation [31]. With that said, in the given case, specifically in an ideal case, this would mean that the LED would stay ON/OFF for 33.333 milliseconds and the camera would capture a frame every 16.666 milliseconds. However, as will be discussed later, this was not at all the case.

4.3 Transmitter Design

The transmitter software was written in python and ran from the Raspberry Pi 4 python IDLE. A flowchart of the transmitter design is illustrated in Figure 4.3. The goal was to provide a text file to the software script, have it converted to binary, packetize the data and transmit by operating the LED such that for every bit that is 1 the LED turns ON and
for every bit that is 0, the LED turns OFF. However, to mimic a real network application, the goal was to buffer data as transmission occurred, i.e. not to process all packets before transmission. However, it was observed that buffering was causing some delays and thus some errors at the receiver, therefore all packets were created before transmission occurred. The actual structure of the packet is shown in Figure 4.2.

The software script began by accepting a plaintext file which was checked for any non-ASCII values and if any were found they were removed. Once all values in the text file were known to be ASCII, each character in the text was converted to 8-bit binary. Then, to identify the beginning and end of the data, 2 markers were appended to the beginning of the data and end of the data. These markers could be anything but for the purposes of this setup, they were constructed as 2 bytes of 1’s. Once the data was ready the actual packetization began. First, a 5-bit Barker Code was placed at the beginning of each packet. Barker codes are a synchronization method for transmitter and receiver [32]. The Barker
Code used was "11101" After the 5-bits of Barker code, 11-bits of the packet number were appended which served to identify the packet order on the receiver end. With 11-bits for the packet number, effectively 2048 packet can be transmitted in a single transmission before repeating the packet number sequence. After appending the packet number, the next 6-bytes of the packet were for the data which is taken from the binary conversion of the plaintext data done earlier. Finally, the frame check sequence (FCS) is calculated using the Cyclic Redundancy Check (CRC) method. The CRC remainder is calculated over the payload and packet number. CRC is an error detecting technique that is commonly used in network systems [33]. The idea is to use a generator and divide the payload by it and append the remainder to the packet. Once transmitted, the receiver will use the same generator and divide the payload plus the remainder it received, and if the new remainder is all 0s then the receiver can be assured that no errors occurred in the transmission. The generator is usually

<table>
<thead>
<tr>
<th>Barker Code</th>
<th>Packet Number</th>
<th>Data</th>
<th>Data</th>
<th>Data</th>
<th>Data</th>
<th>Data</th>
<th>Data</th>
<th>CRC Remainder</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 Bits</td>
<td>11 Bits</td>
<td>1 Byte</td>
<td>1 Byte</td>
<td>1 Byte</td>
<td>1 Byte</td>
<td>1 Byte</td>
<td>1 Byte</td>
<td></td>
</tr>
</tbody>
</table>

2 Bytes

6 Bytes

1 Byte

9 Bytes = 72 Bits
1 bit longer than the FCS, therefore, in this case, the FCS was an 8-bit sequence, which meant that the CRC generator was 9-bits. The generator that was used was "100111100". Of course, there are researched and special generators employed in actual systems that have been rigorously tested, but for the purposes of demonstration, a random 9-bit CRC generator was used to get the 8-bit FCS. Once appended to the end, the packet is complete. In total there are 9 bytes per packet, where 6 bytes are of the data.

As mentioned earlier, since there is no buffer, all packets are precomputed before transmission, which means that each LED has a container that holds all the packets that it will be transmitting. Therefore, the packets for each LED are created in cycles. This is done so that the receiver has an intuitive idea of what packets to expect from each LED. Of course in an ideal case, this does not have to be true but for demonstration purposes, this is how the transmitter was structured. And just for reference, a cycle is defined as taking a 192-bit chunk from the binary representation of the data and then sequentially taking 48-bits from
that chunk and assigning it to each LED from LED 1 through 4. This way, 1 cycle cor-
responds to 4 packet creations, 1 for each LED. Another benefit of this approach is that packet
number is also easy to compute. Once the LEDs are ready to transmit, the Raspberry Pi
signals the PiCamera to start recording and allows for a brief moment for the camera to
warm up before the LEDs start transmitting.

4.4 Receiver Design

Like the transmitter design, the receiver software is also written in python however, the major
difference is that the receiver software script uses many high-level libraries like OpenCV
and Keras. This is because the receiver is responsible for locating the ARUCO markers,
cropping the LEDs from the frame, and having the DNN model predict the bit transmitted
by classifying the captured frame of the LED. A flowchart of the receiver design is illustrated
in Figure 4.4. The receiver first starts by locating the ARUCO markers in the video. If the
marker is not identified within the first 15 frames of the video, the program lets the user know
that the video can’t be processed. This is done because the markers are vital for cropping
the LEDs into separate frames and if the marker is not locatable, for whatever reason like
blurry frames, etc., then the receiver software script cannot demodulate the transmission
successfully. After having successfully located the markers, the script starts by predicting
the LED state by feeding the DNN model the frame of each LED. Each prediction by the
model is stored in a separate container for later processing. In an ideal case, there would
be some sort of approach to cross-check between consecutive frames, as the transmission is
Figure 4.4: Receiver Design Flow Chart

double sampled, but because the focus of the study was on the DNN model, every other frame was discarded. More on this will be explained later. Once all predictions are made, the first bit equal to 1 is located in each LED’s container of prediction. Once found, the previous bits are all discarded, as they were not part of the actual data transmission. The discarded frames are from the time the PiCamera starts to the time the LED’s begin to transmit.

At this point, the container of each LED’s bits starts with the first packet transmitted by that LED. Now, to identify the location of each of the rest of the packets, the Barker code indices are identified. In an ideal case, this would not be necessary if the packets are sent one after the other without delay. This is because every packet is 72 bits, thus dividing the bits into 72-bit chunks should correspond to each packet. However, because of transmission error, the Barker code identification approach is used because even within
this simple system, transmission errors exist. So, a separate function identifies the starting location of each barker code located in the container of bits. Once all indices are found, all indices which are not modulo 72, i.e. the packet size, are removed. To compensate for errors made in transmission or possibly by the DNN model in classifying, an ideal list of indices for the barker code is compared to the actual found barker code indices. This is to make sure if there are missing indices in the actual found indices then they are filled by either looking at the ideal indices list or by looking at the previous and/or next actual found index and adding in the missing indices. This compensation technique ensures that all barker code indices that should be present are indeed accounted for.

After attaining the barker code indices, the packets are parsed from the container of predicted bits for each LED. One thing to note here is that because of transmission errors or even model errors the received packet length might be larger or smaller than 72 bits, however, in this setup the packet size was found to be only occasionally larger or smaller than 72. Once the packets are parsed, the packet number is verified for correctness. Then the data is parsed out of the packet and stored in a separate container for later processing. One thing to mention here is that, unlike the barker code function, no compensation is done for the length of the packet number, 11-bits, or the data, 6 bytes, or even the CRC FCS, 1 byte. However, the CRC reminder is also calculated to check for error but if there is an error, the system does not employ any correction methodology. This is because the focus of the experiment is to determine the correctness of the model and not of the relay of the transmission itself. If there are errors in the packet number or when calculating the remainder, only a note of
it is made by identifying the packet and what type of error. This same approach will be used for demodulating the data bits, however, instead of checking bit by bit, the data will be analyzed byte by byte. This is because 1 byte corresponds to 1 character from the plaintext.

That being said, once all the data bytes from each packet from each LED are parsed out, they are concatenated together, between all 4 LEDs, in the order that they were transmitted. At this point, what the receiver has is essentially the received data transmission, with all the other packet information removed. The script starts processing every 6 bytes of data and comparing it to the ground truth which is what the transmitter actually transmitted. Again, if there is an error, a simple indication of what packet and how many bytes have an error is made. After tallying all the results a report is generated which indicates the number of packets that have data error, barker code error, packet number error, CRC remainder error, and barker code error. Finally, the received binary data is converted to ASCII, but not before locating and removing the starting and ending markers, and shown to the user.

4.5 Preliminary Results

Before diving further into the model setup, it is best to describe some of the other problems discovered with the hardware which were independent of the model. One of the biggest issues was with the LED, and to make the point clear it best to describe what the ideal case would be. In an ideal case, the LED ON would take 33.333 milliseconds, which corresponds to 30 Hz, however, what was discovered was that the LED needed extra time to energize and also extra time to turn OFF because of the current draining. In an ideal case, as soon
as the signal is sent to the LED to turn ON, it would be expected that the LED would be fully ON and because of the double sampling on the receiver end, the two frames of the LED ON would have, more or less, the same intensity. But because of the LED being nonlinear, the LED ON state would span 3 frames and LED OFF would be 1 frame. Thus to correct this, the LED ON to OFF ratio was set to 1:3 which meant that LED ON would correspond to 16.666 milliseconds of the LED ON followed by LED OFF for 16.666 milliseconds. This would be the approach for setting the LED ON for a 30 Hz period. As for LED OFF, the LED would be off for 33.333 milliseconds as usual. However, even with this adjustment another problem with the LED, again because of its nonlinear property, the 2 frames of LED ON were never the same intensity. Figure 4.5 illustrates what two consecutive frames of LED ON would look like often. To correct this was quite a challenge but the best that could be done was basically sample only the brighter LED intensity frame which is why in the transmitter script, every other frame was removed. To understand why this was an issue, some diagnostics were ran while a transmission was in progress and what was found was that the Hz rate of the LED was never actually exactly 30 Hz. The Hz rate would fluctuate between 29 and 31 Hz and sometimes the Hz rate would actually spike to around 25 Hz. Essentially what was happening was the period of delay for LED ON / OFF was inconsistent and in reality, it was actually a hardware issue, in that the Raspberry Pi OS was not meant to handle time-critical applications such as this. For accuracy and consistency, a real-time OS is needed which isn’t what the Raspbian OS is, it’s a flavor of Linux which is not real time OS [34]. Real-time OSes include NuttX, QNX and VxWorks OSes [35]. To fix this
issue only a few options were available and the best being using the most high-resolution delay function available. This function did as much as it could to sustain the 30 Hz to the point where the LED blinked within +/- 0.5 Hz. But the LED blinking issue wasn’t the only issue.

Another evident problem was that the camera would often skip frames and the way that this was discovered was by calculating the time difference between consecutive frames. In an ideal case, a frame should be captured every 16.666 milliseconds and the difference between consecutive frames should also be 16.666 milliseconds. However, it was discovered that the camera would often, inconsistently, skip a few frames for a time length in the range from a few 100 milliseconds to entire seconds. This was more of an issue for longer transmissions in the order of 15 seconds or more but for shorter transmissions, the issue was a rare case. However, this issue caused major issues in the demodulation of the transmission as will be shown later in the results. And like the LED blinking issue, the camera issue also boiled down to a hardware issue but unlike the LED, nothing much could be done to compensate for the camera. Because of these underlying hardware issues, it can be difficult to distinguish between when there is a model error or a hardware error. For instance, 4.6 shows how an
error caused by the camera, i.e. not capturing at the proper time, led to a demodulation error even though the model correctly classified the frame. So the results and discussion presented later will be an indication of the plausibility of such a system if the hardware issues were resolved.

### 4.6 Model Selection

Once hardware and software aspects of the system had been taken care of the main part of this study was analyzed, the DNN. As DNNs can be highly sensitive to the data they

![Figure 4.6: Hardware Versus Model Error](image-url)
trained on, the process that was followed to select a model for the MIMO setup was as such. 3 models with different LED data were trained and then a separate validation dataset was collected and the models were validated on that dataset. From those results, the best model was chosen to be incorporated into the MIMO setup. The model selection process began by first determining which model to use. For that, the model that performed best in the preliminary study, VGG16, was used. However, the parameters that were used in the preliminary study for the VGG16 model had to be adjusted because of the new LED data. For instance, in the preliminary study, Softmax was the best output layer activation but with the new LED data, the model performed best with Sigmoid. This might be the case because Sigmoid is often used for binary classification which is how the MIMO experiments were meant to be conducted. The process of determining these parameters in this part of the thesis has been left out because the procedure was the same as in the preliminary study and it was just a matter of sequential processing to find the best parameters. The same was done to check the best learning rate and optimizer.

4.6.1 Model I Data

The first model’s data consisted of data from an LED that was purchased from Digikey. It had a clear epoxy coating around the diode but was still a red light-emitting diode. Because the images of the LED were discovered to be different in different environmental lighting conditions, two sets of the LED blinking were collected. One was referred to as Light and the other Dark. The Light LED set was when an additional light source was added between the camera and LED to brighten the images captured. Similarly for the Dark LED set, the
additional light source between the camera and LED was removed. An example of each is presented in Figure 4.7.

The LED was captured at distances from 1 meter to 7 meters in increments of 1 meter, similar to how the LED data was collected in the preliminary study. This was done for both Light and Dark scenarios. For the data collection process the LED was just set to blink at 30 Hz to get an even number of LED ON and LED OFF states. At each distance, there were roughly 520 frames captured of the LED which in total meant about 3,700 frames of Dark LED and 3,700 frames of LED Light. For a total of 7,385 raw frames. Though the balance of LED ON versus LED OFF was consistent, the real problem was that the LED ON intensity frames ranged from slightly ON to fully ON. And even more troublesome was that the slightly ON frames were a minority within the LED ON state. As was explained earlier, in an ideal case all the LED ON frames would more or less have the same intensity but because of the underlying hardware issues the LED ON intensity ranged across a vast span. To circumvent this issue, upsampling in the form of just copying the minority class, till balance was achieved, was done. To get a more intuitive understanding of the problem,
Figure 4.8 shows what the average red channel pixel intensity of all frames in 1 meter Dark LED dataset looked like. The distribution was similar for all distance, both Light and Dark frames. The reason why the average pixel intensity of the red channel was chosen was that it distinguished best between the different frames, especially LED ON state, compared to RGB average pixel intensity or green channel or blue channel. As can be seen in the figure the first mode, illustrates LED OFF frames which are pretty consistently grouped. However, from the first LED ON state till the end of the graph is all the varying frames with LED ON. Another aspect that is visible with this representation is that most LED ON states are on the right half of the graph and very few LED ON frames are on the left side of the graph. Therefore to make sure the model was robust to the LED ON intensity, the following formula was used to select the minority LED ON frames so that they could be upsampled.

\[
\left( \frac{\text{Max}_{ON} \text{Intensity} - \text{Min}_{ON} \text{Intensity}}{5} \right) + \text{Min}_{ON} \text{Intensity}
\]

Once the minority frames from the LED ON state were collected, they were upsampled by repeatedly copying them until they made up half of all LED ON frames. For instance, if there are 200 frames of LED ON and only 15 LED ON minority samples, then those 15 samples would be upsampled to 100 frames. After the LED ON minority frames were taken care of, both the LED ON state and LED OFF state were upsampled such that for each distance in each scenario, Light and Dark, there were a total of 1,000 frames in which half were LED ON and the other LED OFF. This allowed Model I to have access to 14,000 frames for training.
4.6.2 Model II Data

Model II’s data consisted of Model I’s data plus some additional LED data which consisted of the LED from the preliminary study as well as LED data from a red diffused LED. This was done so that the model was more robust to the LED in terms of what the frame captured of the LED. The preliminary study LED data consisted of frames from 1 meter through 5 meters which totaled 2,264 frames. After upsampling the frames via augmentation, i.e. rotating, and then balancing, there were a total of 4,273 frames of just the preliminary study LED frames. For the diffused red LED, again the frames came from a distance of 1 meter to 5 meters and the raw count was 11,157 frames. After upsampling via augmentation and balancing the data, there were a total of 20,434 frames of the red diffused LED. A point of distinction here is that when balancing the red diffused LED and preliminary study LED,
the balanced wasn’t absolutely even like it was for the LED data from Model I. Often the LED ON state had 10% more frames than the LED OFF frame. For instance, the 1-meter red diffused LED had 2,200 frames of LED ON and almost 2,000 frames of LED OFF. Another point is that for the red diffused LED data and preliminary study LED, there was no balancing of the minority LED ON state within the LED ON frames as was done in Model I’s data. In total, Model II had access to 38,707 frames on which it could be trained.

4.6.3 Model III Data

Model III’s data consisted of Model I’s raw data which was reprocessed so that instead of upsampling by copying, upsampling was done via augmentation. Then the entire data for each distance, for each scenario, Dark and Light, was upsampled, again with augmentation, to 2,500 frames instead of 1,000 frames. This gave a total of 35,000 frames from Model I’s raw data. The red diffused LED was upsampled via augmentation again but this time the total number of frames per distance was closer to 4,000 frames, thus giving a total frame count of close to 20,000 frames. That is because the red diffused LED had data only from 1 meter to 5 meters in increments of 1 meter. Similarly for the preliminary study LED, the upsampling via augmentation increased to about 2,000 frames per distance, however, unlike in model II’s data, all 7 distances of the LED data were used. Thus the total number of frames from the preliminary study LED was close to 14,000 frames. In total, the model had access to close to 69,000 frames on which it could be trained.
4.7 Model Training

As mentioned earlier, the best performing model from the preliminary study, VGG16, was used for the MIMO setup. In doing so the architecture remained the same as in the preliminary study except for some minor changes. Refer to Figure 3.4 for a refresher on the model architecture with the appended layers. Also mentioned earlier, the process of selecting the model parameters was the same as in the preliminary study but one of the more major differences is that Sigmoid performed better in the output layer activation than Softmax. However, because the data is binary, in theory, both activation can be used because Sigmoid is just a special case of Softmax in that case. Softmax is usually used for multiclass classification. A few other parameters that were changed included the number of epochs, from 10 to 50 which was when the model usually plateaued. And similar to the preliminary study, 80% of the dataset was used for training of which 10% was set aside for validation and the other 20% of the data set was for testing. After finalizing the parameters the model was fully trained on the 3 LED datasets. Figure 4.9 illustrates the training phase of the model in terms of loss and accuracy. As can be seen, the models are neither overfitting nor underfitting and because the models are implemented as transfer models, there is a Higher start, Higher slope, and Higher asymptote in the accuracy curve.

4.8 Model Validation

After the models were trained, validation commenced in which a separate LED dataset was collected. This dataset was collected similar to how Model I's dataset was collected in that
Figure 4.9: Model Training Curves

there was a separation between Light and Dark LED frames. In total there were about 4,000 frames of LED Dark and 4,000 frames of LED Light. The LED used was the clear Digikey LED as that LED was the one that was decided to be the LED used for transmission in the MIMO setup. Figure 4.10 shows the validation results of each model against the validation dataset. From the results, it is clear that Model II performed the best. It only misclassified 2 samples out of nearly 8,000 frames.

4.9 Revised Experiment

Because of the underlying hardware issues, a new methodology had to be taken to show that the system, specifically the model, is capable of working even with the hardware issues. To
showcase this, the following steps were taken:

- Step 1: Find the number of characters, which will be called the chunk size for the remainder of the study, that can be successfully transmitted without any type of error (Camera or LED). As mentioned earlier, for shorter transmission the hardware inconsistencies are not prevalent.

- Step 2: Transmit a 1000 word file in a single transmission. This was the original/ideal target for the empirical study to transmit with as few errors as possible.

- Step 3: Evaluate the 1000 word file transmission by dividing it into chunks established...
from Step 1, and evaluating each chunk.

- Step 4: Transmit the 1000 word file in several videos in which each video sends one chunk of data where the chunk size is defined by the steps used in Step 1.

- Step 5: Analyze the results

The reason why the chunk size is established is to showcase the effectiveness of the model, as independently as possible, rather than the entire system. Similarly the reason why a single transmission of a 1000 word file is compared versus a fragmented transmission of a 1000 word file is again to showcase the model’s performance but more importantly to illustrate how the hardware issues can cause serious problem for this communication system.

To achieve Step 1, several videos of varying character length were transmitted at each distance. After computing the results, 100 character transmissions were found to be the best chunk size. 100 character transmission was roughly 17 packets, 18 packets if the starting and ending markers are included. The results showed that across 5 trials at each distance from 1 meter to 7 meters there are no hardware issues nor model misclassifications except for when the distance is at 7 meters. This is when a few model misclassifications start to appear. Out of the nearly 6,200 frames collected across the 5 trials at a distance of 7 meters, there was only 16 model misclassification. Figure 4.11 shows what these errors looked like. They are all of LED ON state and in each case the model predicted LED OFF. And this illustrates another point which is that as the LED move farther and farther away, the model is susceptible to more misclassifications for multiple reasons some of which include the resolution of the image becomes too degraded.
Step 2 and 3 was performed by transmitting the 1000 word file at each distance twice. Then every 17 packets, which is roughly equal to 100 characters which equals 1 chunk, was analyzed for byte errors only for the data, and not for the packet metadata. So in the 1000 word file, there were about 50 chunks or about 850 packets worth of data. After demodulating the transmission the 50 chunks were checked for the total number of packets with any number of errors and the Cumulative Average Packet Error Rate was also calculated. The reason why the cumulative average was used was because, as will be discussed later, the hardware issues would pop up randomly during the transmission and distort the transmission. So to get an intuitive idea of the success of the entire transmission, the cumulative average was utilized. The formula for the Cumulative Average Packet Error Rate is given below. Then the average of both the number of packet errors and Cumulative Average Packet Error Rate between the two trials was calculated.
For Step 4, instead of transmitting a single 1000 word file in 1 video, the 1000 word file was sliced such that approximately 100 characters were transmitted in each video. This resulted in having to transmit 50 videos to get the entire 1000 word file through. This was done twice at each distance and then the same procedure to calculate the total number of packet errors per chunk, in this case, 1 video, and the Cumulative Average Packet Error Rate was calculated with the average between the two trials also included.

Because there were 14 experiments conducted in total, the reader can refer to appendix A and B to see all the charts of the results at each distance with each type of transmission. However, for the purpose of conveying the message, the result from the 1-meter distance for both 1000 word single transmission and 1000 word divided into 100 character transmission is presented below. And before preceding, another important point to note is that unlike during training where there was a distinction between LED Dark and LED Light, there is no distinction made in the actual experiments. This is because in model training the purpose of the Dark and Light separation was to make the model more robust to the varying ways the LED could appear.

Figure 4.12 shows the result from the 1-meter distance with 1000 word single transmission. The top plot shows the Cumulative Average Packet Error with the sequence of chunks increasing. Similarly, the bottom plot shows the number of packet errors per chunk.
What is observed is that after a certain chunk all packets have some sort of error and when observing the demodulated text versus the actual text it is clear that there is 100% error in packets, so much so that every byte has an error. This phenomenon can be observed in Figure 4.13 were once the chunk with all errors starts, the rest of the message is garbage. A point to make clear here is that Figure 4.15 is just the demodulated text from Trial 1. To understand if this is a model error or a hardware error, analysis has to be done packet by packet. In doing so, to check for hardware error the transmission was checked for the time difference between consecutive captured frames. This is where the problem actually lay. In this particular transmission, there are actually 8 frames where the difference between it and the previous frame was well above 16.666 milliseconds. These 8 locations had the following time difference between themselves and the previous frame: (697.68, 1079.739, 33.224, 249.171, 33.223, 1046.517, 531.565, 647.844) in milliseconds. As can be seen, some of these values correspond to whole seconds between frame captures, which has the effect of distorting the received data by shifting the data bits and losing whole chunks of data. This is why the received text is actually less than the actual text. And the way to correlate these time difference errors back to the location in the message is by correlating the frame number of where the time difference error occurred to the chunk that it occurred in. Basically, find the first time difference error-index and then multiple the index of the error by 4, for the number of LEDs, then divide by 72, for the number bits in each packet, and then multiple by 6, which represents the number of bytes in each packet. The returning value should be equal, or close, to the number of characters that are transmitted correctly before the sequence of
Figure 4.12: 1 meter 1000 Word Single Transmission

garbage text begins. So in the current case, the previous calculation yields 520 characters correctly transmitted considering the index of the first error is 1,559th frame. This is about the same as in the demodulated text. And once the frame capture error occurs, the rest of the transmission is out of sync which is why the demodulation returns garbage characters. So what this proves is that the major issue with this system is the hardware issue specifically the camera not capturing at a regular interval.

To confirm that the issue is truly a hardware issue and not a model issue, the transmission
The results of all the trials can be found in appendix B but presented here is just the result of the videos captured at a distance of 1 meter. Figure 4.14 shows that there is indeed an improvement in transmission when the data to transfer is small. However, there are a few cases where the packet error rate is significant in comparison to the rest of the transmission. So to understand if those are a model issue or a hardware issue, the consecutive frame time difference should be checked. In doing, it is discovered that there are camera issues present for some of the chunks, not all. For instance, in the case of Trial 1, chunks 1 and 12 had camera issues near the end which is why both had 1 packet error.
Figure 4.14: 1 meter 50 Chunk Transmission

each. However, the other chunks did not have camera issues which means that those were probably caused by the model misclassifying. This can be further corroborated by viewing Figure 4.15 in which it seems to show that the model only misclassified a certain portion of the chunk rather than a series of garbage text. A point to make clear here is that Figure 4.15 is just the demodulated text from Trial 1.

The same trend in both the 1000 word single transmission and 1000 word multi-transmission can be observed in the rest of the distances from 2 meters to 7 meters. However, as distance
Figure 4.15: 1 meter 50 Chunk Transmission Demodulated increases so does the model misclassifying. But by comparing the one large transmission versus the 50 multi transmissions it is clear that the model error is secondary to the hardware issue. Nonetheless, the conclusion of the matter is that given a perfect hardware setup, the model is capable of accurately demodulated the received LED signal with consistency.
5 DISCUSSION

5.1 Preliminary Study

The goal of the preliminary study had been to explore if DNNs were capable of learning an LED state and how that compared to a basic thresholding based approach. After exploring 6 different models, 4 supervised models, and 2 unsupervised models it was certain that DNNs could approximate the LED state. However, unlike the supervised models, the unsupervised models struggled to properly classify. This begs the question of model explainability which is an actively developing field in the ML/DL community in which models are evaluated not just as a black box but on a micro-scale in terms of what the model is actually learning. This study showed that even bare bone DNNs are able to classify LED states quite well and that with further refinement and possibly more data, the models can be improved to acceptable BER standards. The takeaway points from the preliminary study are that DNNs are capable of learning the LED state whether that be a binary representation or even a set amount of intensity ranges. In the binary case, the accuracy achieved was in the order of 98% to 99% with the best performing supervised model, VGG16. However, in the case of 3 Level and 4 Level, the average accuracy was roughly in the range of 95% to 97%. Though these values would not be within acceptable standards they illustrate the potential of DNNs. As was mentioned, all the supervised models were only trained for 10 epochs and the training dataset across all 3 levels were less than 10,000 frames which leaves the door for improvement wide open. In comparison to basic thresholding based approach,
the model results, specifically the supervised model results, are on par. But reviewing
the basic thresholding based approach results and the 2D representation of the data, it is
clear that there is some fundamental challenge is clearly distinguishing between different
LED intensities/states. As for the unsupervised models, a serious investigation needs to be
conducted to confidently conclude why those models, overall, failed. Unsupervised models,
by nature, are difficult to interpret/explain so combining them with an already challenging
classification task only adds to the difficulty of understanding what is going on.

The real challenge moving forward is now to determine what actually influences the model
decision as the image propagates through the model. Basically learning what the model is
learning. With that in hand, it would be easier to train the model as the underlying properties
of the training data would be known to better teach the model. And as an extension to this
study, the next area of research would be to investigate how to teach a model to localize an
LED in any frame, in any environment, whether indoor or outdoor in a lit area or dark area.
LED localization is an aspect that can potentially provide better long-term performance for
DNNs which are trained to classify LEDs.

Before concluding upon the preliminary study some reflections on how the study went
and aspects of which, if redoing the experiments, would include a standardized way to collect
the LED data, learning what the models are interpreting at each stage in the model, and
more rigorous survey of models which may even include a fusion of different models. Another
aspect of research would be to explore how data size influences the model’s performance.
Though this is a general question in DL, the specific aspect here is in terms of the LED
intensities, i.e. 2 Level, 3 Level, and 4 Level. Also worth looking into is how does that relates to how much data the model needs to approximate the data. This is because too much data can also lead to overfitting. Overall the study has shown that DNNs can provide a decent approximation to an LED’s intensity/state and are just as good, if not better than some basic thresholding based approach that perform the same task.

5.2 Empirical Experiment

The empirical experiments showed a lot of areas in a VLC system that need to be optimized, not to mention the focus of this study which was the DNN. The results have shown that for high performance and completely reliable system the entire structure of the VLC system has to not only be synced but operating consistently. The other major conclusion is that DNNs can and do provide a performance utility for a camera based VLC system. Of course with the current setup, the transmission size is limited to get 0, or close to 0, errors. But again, the potential is there. Another point, which is the same as in the preliminary study, is that the DNN needs to be explored further from the perspective of what it is learning rather than how effective it is. This will help especially in improving the accuracy and performance of the DNN when it is exposed to an LED in an environment outside of what it was trained on. This study has also shown that distance is a major factor in the reliable transmission and demodulation of data. As distance increases so does the model’s error. To counter these, and other, issues several ideas come to mind such as specific preprocessing techniques that can alter each frame so that they all are similar in color, brightness, etc. so that the model
always gets a familiar frame. Other techniques might include having a collection of different models which then vote on the prediction rather than relying on any single model for the classification. The list can go on and on but the principle idea here, as in the preliminary study, is to explore the model training phase.

5.3 Insights

Some contributions that this thesis has made to the continual research in the area of communication networks includes the following.

- It is feasible to use DNN for demodulation in optical camera communication, however the robustness of the model is largely dependent on quality, size and diversity of the dataset. DNN cannot easily extract subtle features from LED ON/OFF signals as they appear as blobs with minimal or no structural feature. Novel spatio-temporal techniques that study the LED signal reception variability across space and time must be incorporated to build novel DNN models for VLC.

- Hardware inconsistencies in camera, computing unit (e.g. Raspberry Pi version) can severely impact the performance of DNN demodulation as transmission and reception in VLC are largely incoherent (transmission and receiver sampling are not synchronized). Identifying the exact head and tail of each packet and detection/estimation of missed camera frame (samples) are key for improving demodulation accuracy.
5.4 Future Work

Though this may seem like an extensive study the reality of the fact is that many more questions have presented themselves which are fundamental for the continuation of this area of study. There are a few questions that can be quickly answered like those related to the hardware instabilities. However, some of the more fundamental questions that relate to the core of this study like what the DNN is learning are yet to be fully answered. With that said the horizon for this area of research includes learning model interpretability and understanding the data. Particularly, interesting areas of focus would be to use Generative Adversarial Networks (GAN)s for model training and using real time OSes for proper synchronization of the transmitter and receiver. In addition, to pass the Critical Flicker Frequency, it would be interesting to use higher performing cameras and LEDs.
6 CONCLUSIONS

In conclusion, this thesis has provided a baseline for further work in incorporating DNNs for camera based VLC system that serve as a demodulation mechanism of the transmitted data from the LED. This study has evaluated the performance of various DL/ML models to explore the potential of these models in deciphering an LED state from a regular binary state up to 4 levels of intensity. Then to verify these model’s performance, a trivial method to solving the same problem was explored to compare the model’s performance. This thesis showed that with minimal training the models are able to achieve between 95% to 99% accuracy in classifying the LED intensity/state with respect to the number of intensities. After which an empirical study of an actual camera based VLC system was prototyped and experimented with to illustrate the feasibility of DNNs. The experiments showed that DNNs are indeed a very plausible solution but require an understanding of what the model is learning to effectively create a robust system. Hardware issues presented a major challenge and illustrated the need for specific hardware for this type of communication network. The results from the empirical study demonstrate the effective use of DNNs but in a set environment, therefore a dynamic setting requires further model training, perhaps large datasets, etc. Through the course of study of this thesis research work we have addressed feasibility questions for DNN usage in camera communication. However, this work has opened up new avenues for research questions across the design, development and implementation of novel DNN models and DNN-inspired algorithms for improving receiver performance in camera communication.
7 APPENDICES

7.1 Appendix A

Figure 7.1: 1 meter 1000 Word Single Transmission
Figure 7.2: 2 meter 1000 Word Single Transmission
Figure 7.3: 3 meter 1000 Word Single Transmission
Figure 7.4: 4 meter 1000 Word Single Transmission
Figure 7.5: 5 meter 1000 Word Single Transmission
Figure 7.6: 6 meter 1000 Word Single Transmission
Figure 7.7: 7 meter 1000 Word Single Transmission
7.2 Appendix B

Figure 7.8: 1 meter 50 Chunk Transmission
Figure 7.9: 2 meter 50 Chunk Transmission
Figure 7.10: 3 meter 50 Chunk Transmission
Figure 7.11: 4 meter 50 Chunk Transmission
Figure 7.12: 5 meter 50 Chunk Transmission
Figure 7.13: 6 meter 50 Chunk Transmission
7.3 Appendix C
Table 7.1: 4 Level Full Color Background Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th></th>
<th>Class 0s</th>
<th>Class 1s</th>
<th>Class 2s</th>
<th>Class 3s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Threshold Ranges</td>
<td>Threshold Ranges</td>
<td>Threshold Ranges</td>
<td>Threshold Ranges</td>
</tr>
<tr>
<td>1m</td>
<td>[0 - 191.93]</td>
<td>(191.93 - 203.93]</td>
<td>(203.93 - 211.17]</td>
<td>(211.17 - 255]</td>
</tr>
<tr>
<td>2m</td>
<td>[0 - 198.66]</td>
<td>(198.66 - 205.68]</td>
<td>(205.68 - 214.8]</td>
<td>(214.8 - 255]</td>
</tr>
<tr>
<td>3m</td>
<td>[0 - 191.4]</td>
<td>(191.4 - 199.81]</td>
<td>(199.81 - 207.99]</td>
<td>(207.99 - 255]</td>
</tr>
<tr>
<td>4m</td>
<td>[0 - 192.94]</td>
<td>(192.94 - 198.52]</td>
<td>(198.52 - 205.21]</td>
<td>(205.21 - 255]</td>
</tr>
<tr>
<td>6m</td>
<td>[0 - 176]</td>
<td>(176 - 183.36]</td>
<td>(183.36 - 188.77]</td>
<td>(188.77 - 255]</td>
</tr>
<tr>
<td>7m</td>
<td>[0 - 183.63]</td>
<td>(183.63 - 196.59]</td>
<td>(196.59 - 203.19]</td>
<td>(203.19 - 255]</td>
</tr>
</tbody>
</table>

Table 7.2: 4 Level Red Channel Background Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th></th>
<th>Class 0s</th>
<th>Class 1s</th>
<th>Class 2s</th>
<th>Class 3s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Threshold Ranges</td>
<td>Threshold Ranges</td>
<td>Threshold Ranges</td>
<td>Threshold Ranges</td>
</tr>
<tr>
<td>1m</td>
<td>[0 - 194.52]</td>
<td>(194.52 - 228.23]</td>
<td>(228.23 - 241.99]</td>
<td>(241.99 - 255]</td>
</tr>
<tr>
<td>2m</td>
<td>[0 - 202.11]</td>
<td>(202.11 - 221.91]</td>
<td>(221.91 - 238.58]</td>
<td>(238.58 - 255]</td>
</tr>
<tr>
<td>3m</td>
<td>[0 - 192.27]</td>
<td>(192.27 - 214.84]</td>
<td>(214.84 - 228.26]</td>
<td>(228.26 - 255]</td>
</tr>
<tr>
<td>4m</td>
<td>[0 - 185]</td>
<td>(185 - 208.79]</td>
<td>(208.79 - 228.2]</td>
<td>(228.2 - 255]</td>
</tr>
<tr>
<td>6m</td>
<td>[0 - 174.8]</td>
<td>(174.8 - 194.31]</td>
<td>(194.31 - 205.44]</td>
<td>(205.44 - 255]</td>
</tr>
<tr>
<td>7m</td>
<td>[0 - 166.27]</td>
<td>(166.27 - 189]</td>
<td>(189 - 206.78]</td>
<td>(206.78 - 255]</td>
</tr>
</tbody>
</table>
Table 7.3: 3 Level Full Color Background Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th></th>
<th>Class 0s Threshold Ranges</th>
<th>Class 1s Threshold Ranges</th>
<th>Class 2s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1m</td>
<td>[0 - 190.85]</td>
<td>(190.85 - 203.93)</td>
<td>(203.93 - 255]</td>
</tr>
<tr>
<td>2m</td>
<td>[0 - 196.893]</td>
<td>(196.893 - 205.205]</td>
<td>(205.205 - 255]</td>
</tr>
<tr>
<td>3m</td>
<td>[0 - 192.92]</td>
<td>(192.92 - 199.81]</td>
<td>(199.81 - 255]</td>
</tr>
<tr>
<td>4m</td>
<td>[0 - 182.89]</td>
<td>(189.89 - 195.92]</td>
<td>(195.92 - 255]</td>
</tr>
<tr>
<td>5m</td>
<td>[0 - 178.7]</td>
<td>(178.7 - 187.28]</td>
<td>(187.28 - 255]</td>
</tr>
<tr>
<td>6m</td>
<td>[0 - 175.01]</td>
<td>(175.01 - 183.36]</td>
<td>(183.36 - 255]</td>
</tr>
<tr>
<td>7m</td>
<td>[0 - 183.92]</td>
<td>(183.92 - 191.85]</td>
<td>(191.85 - 255]</td>
</tr>
</tbody>
</table>

Table 7.4: 3 Level Red Channel Background Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th></th>
<th>Class 0s Threshold Ranges</th>
<th>Class 1s Threshold Ranges</th>
<th>Class 2s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1m</td>
<td>[0 - 191.99]</td>
<td>(191.99 - 227.54]</td>
<td>(227.54 - 255]</td>
</tr>
<tr>
<td>2m</td>
<td>[0 - 202.1]</td>
<td>(202.1 - 221.75]</td>
<td>(221.75 - 255]</td>
</tr>
<tr>
<td>3m</td>
<td>[0 - 192.64]</td>
<td>(192.64 - 215.37]</td>
<td>(215.37 - 255]</td>
</tr>
<tr>
<td>4m</td>
<td>[0 - 185.13]</td>
<td>(185.13 - 208.79]</td>
<td>(208.79 - 255]</td>
</tr>
<tr>
<td>5m</td>
<td>[0 - 181.31]</td>
<td>(181.31 - 205.49]</td>
<td>(205.49 - 255]</td>
</tr>
<tr>
<td>6m</td>
<td>[0 - 174.8]</td>
<td>(174.8 - 194.3]</td>
<td>(194.3 - 255]</td>
</tr>
<tr>
<td>7m</td>
<td>[0 - 167.64]</td>
<td>(167.64 - 191.22]</td>
<td>(191.22 - 255]</td>
</tr>
</tbody>
</table>
Table 7.5: 2 Level Full Color Background Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th>Class</th>
<th>0s Threshold Ranges</th>
<th>1s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1m</td>
<td>[0 - 204.92]</td>
<td>(204.92 - 255)</td>
</tr>
<tr>
<td>2m</td>
<td>[0 - 206.10]</td>
<td>(206.10 - 255)</td>
</tr>
<tr>
<td>3m</td>
<td>[0 - 201.2]</td>
<td>(201.2 - 255)</td>
</tr>
<tr>
<td>4m</td>
<td>[0 - 196.5]</td>
<td>(196.5 - 255)</td>
</tr>
<tr>
<td>5m</td>
<td>[0 - 187.52]</td>
<td>(187.52 - 255)</td>
</tr>
<tr>
<td>6m</td>
<td>[0 - 184.16]</td>
<td>(184.16 - 255)</td>
</tr>
<tr>
<td>7m</td>
<td>[0 - 196.19]</td>
<td>(196.19 - 255)</td>
</tr>
</tbody>
</table>

Table 7.6: 2 Level Red Channel Background Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th>Class</th>
<th>0s Threshold Ranges</th>
<th>1s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1m</td>
<td>[0 - 228.87]</td>
<td>(228.87 - 255)</td>
</tr>
<tr>
<td>2m</td>
<td>[0 - 222.75]</td>
<td>(222.75 - 255)</td>
</tr>
<tr>
<td>3m</td>
<td>[0 - 214.86]</td>
<td>(214.86 - 255)</td>
</tr>
<tr>
<td>4m</td>
<td>[0 - 208.79]</td>
<td>(208.79 - 255)</td>
</tr>
<tr>
<td>5m</td>
<td>[0 - 203]</td>
<td>(203 - 255)</td>
</tr>
<tr>
<td>6m</td>
<td>[0 - 190.22]</td>
<td>(190.22 - 255)</td>
</tr>
<tr>
<td>7m</td>
<td>[0 - 190.22]</td>
<td>(190.22 - 255)</td>
</tr>
</tbody>
</table>
Table 7.7: 4 Level Full Color Cropped and Resized Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th>Class 0s Threshold Ranges</th>
<th>Class 1s Threshold Ranges</th>
<th>Class 2s Threshold Ranges</th>
<th>Class 3s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1m 0 - 184.48</td>
<td>(184.48 - 215.91)</td>
<td>(215.91 - 228.67)</td>
<td>(228.67 - 255)</td>
</tr>
<tr>
<td>2m 0 - 188.54</td>
<td>(188.54 - 224.28)</td>
<td>(224.28 - 235.22)</td>
<td>(235.22 - 255)</td>
</tr>
<tr>
<td>3m 0 - 188.31</td>
<td>(188.31 - 222.8)</td>
<td>(222.8 - 236.18)</td>
<td>(236.18 - 255)</td>
</tr>
<tr>
<td>4m 0 - 181.58</td>
<td>(181.58 - 203.02)</td>
<td>(203.02 - 213.38)</td>
<td>(213.38 - 255)</td>
</tr>
<tr>
<td>5m 0 - 174.83</td>
<td>(174.83 - 205.92)</td>
<td>(205.92 - 216.56)</td>
<td>(216.56 - 255)</td>
</tr>
<tr>
<td>6m 0 - 170.78</td>
<td>(170.78 - 196.98)</td>
<td>(196.98 - 206.73)</td>
<td>(206.73 - 255)</td>
</tr>
<tr>
<td>7m 0 - 183.61</td>
<td>(183.61 - 212.08)</td>
<td>(212.08 - 223.42)</td>
<td>(223.42 - 255)</td>
</tr>
</tbody>
</table>

Table 7.8: 4 Level Red Channel Cropped and Resized Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th>Class 0s Threshold Ranges</th>
<th>Class 1s Threshold Ranges</th>
<th>Class 2s Threshold Ranges</th>
<th>Class 3s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1m 0 - 205.2</td>
<td>(205.2 - 248.14)</td>
<td>(248.14 - 250.07)</td>
<td>(250.07 - 255)</td>
</tr>
<tr>
<td>2m 0 - 197.07</td>
<td>(197.07 - 242.77)</td>
<td>(242.77 - 250.89)</td>
<td>(250.89 - 255)</td>
</tr>
<tr>
<td>3m 0 - 191.52</td>
<td>(191.52 - 235.66)</td>
<td>(235.66 - 249.12)</td>
<td>(249.12 - 255)</td>
</tr>
<tr>
<td>4m 0 - 183.83</td>
<td>(183.83 - 215.56)</td>
<td>(215.56 - 231.84)</td>
<td>(231.84 - 255)</td>
</tr>
<tr>
<td>5m 0 - 182.83</td>
<td>(182.83 - 220.56)</td>
<td>(220.56 - 232.84)</td>
<td>(232.84 - 255)</td>
</tr>
<tr>
<td>6m 0 - 171.61</td>
<td>(171.61 - 204.25)</td>
<td>(204.25 - 222.74)</td>
<td>(222.74 - 255)</td>
</tr>
<tr>
<td>7m 0 - 164.98</td>
<td>(164.98 - 211.05)</td>
<td>(211.05 - 227.86)</td>
<td>(227.86 - 255)</td>
</tr>
</tbody>
</table>
Table 7.9: 3 Level Full Color Cropped and Resized Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th></th>
<th>Class 0s Threshold Ranges</th>
<th>Class 1s Threshold Ranges</th>
<th>Class 2s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1m</td>
<td>[0 - 183.62]</td>
<td>(183.62 - 215.68)</td>
<td>(215.68 - 255]</td>
</tr>
<tr>
<td>2m</td>
<td>[0 - 193.5]</td>
<td>(193.5 - 224.59)</td>
<td>(224.59 - 255]</td>
</tr>
<tr>
<td>3m</td>
<td>[0 - 190.5]</td>
<td>(190.5 - 224.59)</td>
<td>(224.59 - 255]</td>
</tr>
<tr>
<td>4m</td>
<td>[0 - 182.36]</td>
<td>(182.36 - 203.41]</td>
<td>(203.41 - 255]</td>
</tr>
<tr>
<td>5m</td>
<td>[0 - 176.62]</td>
<td>(176.62 - 205.92]</td>
<td>(205.92 - 255]</td>
</tr>
<tr>
<td>6m</td>
<td>[0 - 184.54]</td>
<td>(184.54 - 197.02]</td>
<td>(197.02 - 255]</td>
</tr>
<tr>
<td>7m</td>
<td>[0 - 183.62]</td>
<td>(183.62 - 211.68]</td>
<td>(211.68 - 255]</td>
</tr>
</tbody>
</table>

Table 7.10: 3 Level Red Channel Cropped and Resized Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th></th>
<th>Class 0s Threshold Ranges</th>
<th>Class 1s Threshold Ranges</th>
<th>Class 2s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1m</td>
<td>[0 - 199.07]</td>
<td>(199.07 - 246.08]</td>
<td>(246.08 - 255]</td>
</tr>
<tr>
<td>2m</td>
<td>[0 - 220.89]</td>
<td>(220.89 - 243.67]</td>
<td>(243.67 - 255]</td>
</tr>
<tr>
<td>3m</td>
<td>[0 - 191.55]</td>
<td>(191.55 - 245.07)</td>
<td>(245.07 - 255]</td>
</tr>
<tr>
<td>5m</td>
<td>[0 - 183]</td>
<td>(183 - 219.62]</td>
<td>(219.62 - 255]</td>
</tr>
<tr>
<td>6m</td>
<td>[0 - 172.06]</td>
<td>(172.06 - 210.33]</td>
<td>(210.33 - 255]</td>
</tr>
<tr>
<td>7m</td>
<td>[0 - 169.06]</td>
<td>(169.06 - 212.33]</td>
<td>(212.33 - 255]</td>
</tr>
</tbody>
</table>
Table 7.11: 2 Level Full Color Cropped and Resized Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th></th>
<th>Class 0s Threshold Ranges</th>
<th>Class 1s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1m</strong></td>
<td>[0 - 217.08]</td>
<td>(217.08 - 255]</td>
</tr>
<tr>
<td><strong>2m</strong></td>
<td>[0 - 226.28]</td>
<td>(226.28 - 255]</td>
</tr>
<tr>
<td><strong>3m</strong></td>
<td>[0 - 219.08]</td>
<td>(219.08 - 255]</td>
</tr>
<tr>
<td><strong>4m</strong></td>
<td>[0 - 204]</td>
<td>(204 - 255]</td>
</tr>
<tr>
<td><strong>5m</strong></td>
<td>[0 - 205]</td>
<td>(205 - 255]</td>
</tr>
<tr>
<td><strong>6m</strong></td>
<td>[0 - 198.57]</td>
<td>(198.57 - 255]</td>
</tr>
<tr>
<td><strong>7m</strong></td>
<td>[0 - 214.07]</td>
<td>(214.07 - 255]</td>
</tr>
</tbody>
</table>

Table 7.12: 2 Level Red Channel Cropped and Resized Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th></th>
<th>Class 0s Threshold Ranges</th>
<th>Class 1s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1m</strong></td>
<td>[0 - 245.63]</td>
<td>(245.63 - 255]</td>
</tr>
<tr>
<td><strong>2m</strong></td>
<td>[0 - 242.95]</td>
<td>(242.95 - 255]</td>
</tr>
<tr>
<td><strong>3m</strong></td>
<td>[0 - 243.07]</td>
<td>(243.07 - 255]</td>
</tr>
<tr>
<td><strong>4m</strong></td>
<td>[0 - 215.62]</td>
<td>(215.62 - 255]</td>
</tr>
<tr>
<td><strong>5m</strong></td>
<td>[0 - 221.15]</td>
<td>(221.15 - 255]</td>
</tr>
<tr>
<td><strong>6m</strong></td>
<td>[0 - 210.24]</td>
<td>(210.24 - 255]</td>
</tr>
<tr>
<td><strong>7m</strong></td>
<td>[0 - 213.27]</td>
<td>(213.27 - 255]</td>
</tr>
</tbody>
</table>
Table 7.13: 4 Level Full Color Cropped As Is Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th>Class 0s Threshold Ranges</th>
<th>Class 1s Threshold Ranges</th>
<th>Class 2s Threshold Ranges</th>
<th>Class 3s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1m</td>
<td>[0 - 181.05]</td>
<td>(181.05 - 215.69]</td>
<td>(215.69 - 226.92]</td>
</tr>
<tr>
<td>2m</td>
<td>[0 - 188.99]</td>
<td>(188.99 - 223.6]</td>
<td>(223.6 - 235.43]</td>
</tr>
<tr>
<td>3m</td>
<td>[0 - 201.34]</td>
<td>(201.34 - 223.33]</td>
<td>(223.33 - 236.34]</td>
</tr>
<tr>
<td>5m</td>
<td>[0 - 176.08]</td>
<td>(176.08 - 205.72]</td>
<td>(205.72 - 216.35]</td>
</tr>
<tr>
<td>6m</td>
<td>[0 - 170.03]</td>
<td>(170.03 - 197.4]</td>
<td>(197.4 - 206.66]</td>
</tr>
<tr>
<td>7m</td>
<td>[0 - 180.07]</td>
<td>(180.07 - 211.69]</td>
<td>(211.69 - 223.4]</td>
</tr>
</tbody>
</table>

Table 7.14: 4 Level Red Channel Cropped As Is Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th>Class 0s Threshold Ranges</th>
<th>Class 1s Threshold Ranges</th>
<th>Class 2s Threshold Ranges</th>
<th>Class 3s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1m</td>
<td>[0 - 194.63]</td>
<td>(194.63 - 244.85]</td>
<td>(244.85 - 250.25]</td>
</tr>
<tr>
<td>2m</td>
<td>[0 - 201.44]</td>
<td>(201.44 - 243.48]</td>
<td>(243.48 - 250.77]</td>
</tr>
<tr>
<td>3m</td>
<td>[0 - 191]</td>
<td>(191 - 239.35]</td>
<td>(239.35 - 249.66]</td>
</tr>
<tr>
<td>4m</td>
<td>[0 - 184.5]</td>
<td>(184.5 - 216.57]</td>
<td>(216.57 - 233.16]</td>
</tr>
<tr>
<td>6m</td>
<td>[0 - 171.57]</td>
<td>(171.57 - 204.76]</td>
<td>(204.76 - 220.44]</td>
</tr>
<tr>
<td>7m</td>
<td>[0 - 166.05]</td>
<td>(166.05 - 211.03]</td>
<td>(211.03 - 227.7]</td>
</tr>
</tbody>
</table>
Table 7.15: 3 Level Full Color Cropped As Is Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th>3 Level Full Color Cropped As Is Frames Threshold Ranges</th>
<th>Class 0s Threshold Ranges</th>
<th>Class 1s Threshold Ranges</th>
<th>Class 2s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1m</strong></td>
<td>[0 - 194.42]</td>
<td>[194.42 - 215.88]</td>
<td>[215.88 - 255]</td>
</tr>
<tr>
<td><strong>2m</strong></td>
<td>[0 - 191]</td>
<td>[191 - 224.3]</td>
<td>[224.3 - 255]</td>
</tr>
<tr>
<td><strong>3m</strong></td>
<td>[0 - 190.56]</td>
<td>[190.56 - 223]</td>
<td>[223 - 255]</td>
</tr>
<tr>
<td><strong>4m</strong></td>
<td>[0 - 182.36]</td>
<td>[182.36 - 203.35]</td>
<td>[203.35 - 255]</td>
</tr>
<tr>
<td><strong>5m</strong></td>
<td>[0 - 184.72]</td>
<td>[184.72 - 205.72]</td>
<td>[205.72 - 255]</td>
</tr>
<tr>
<td><strong>6m</strong></td>
<td>[0 - 170.17]</td>
<td>[170.17 - 196.72]</td>
<td>[196.72 - 255]</td>
</tr>
<tr>
<td><strong>7m</strong></td>
<td>[0 - 182.61]</td>
<td>[182.61 - 211.68]</td>
<td>[211.68 - 255]</td>
</tr>
</tbody>
</table>

Table 7.16: 3 Level Red Channel Cropped As Is Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th>3 Level Red Channel Cropped As Is Frames Threshold Ranges</th>
<th>Class 0s Threshold Ranges</th>
<th>Class 1s Threshold Ranges</th>
<th>Class 2s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1m</strong></td>
<td>[0 - 195.12]</td>
<td>[195.12 - 244.85]</td>
<td>[244.85 - 255]</td>
</tr>
<tr>
<td><strong>2m</strong></td>
<td>[0 - 219.38]</td>
<td>[219.38 - 243.01]</td>
<td>[243.01 - 255]</td>
</tr>
<tr>
<td><strong>3m</strong></td>
<td>[0 - 194.12]</td>
<td>[194.12 - 242.9]</td>
<td>[242.9 - 255]</td>
</tr>
<tr>
<td><strong>4m</strong></td>
<td>[0 - 184.21]</td>
<td>[184.21 - 216.11]</td>
<td>[216.11 - 255]</td>
</tr>
<tr>
<td><strong>5m</strong></td>
<td>[0 - 183.5]</td>
<td>[183.5 - 220.8]</td>
<td>[220.8 - 255]</td>
</tr>
<tr>
<td><strong>6m</strong></td>
<td>[0 - 172.01]</td>
<td>[172.01 - 213.03]</td>
<td>[212.03 - 255]</td>
</tr>
<tr>
<td><strong>7m</strong></td>
<td>[0 - 169.01]</td>
<td>[169.01 - 213.03]</td>
<td>[213.03 - 255]</td>
</tr>
</tbody>
</table>
Table 7.17: 2 Level Full Color Cropped As Is Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th>Class</th>
<th>0s Threshold Ranges</th>
<th>1s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1m</td>
<td>[0 - 216.88]</td>
<td>(216.88 - 255]</td>
</tr>
<tr>
<td>2m</td>
<td>[0 - 226.57]</td>
<td>(226.57 - 255]</td>
</tr>
<tr>
<td>3m</td>
<td>[0 - 226.25]</td>
<td>(226.25 - 255]</td>
</tr>
<tr>
<td>4m</td>
<td>[0 - 204.5]</td>
<td>(204.5 - 255]</td>
</tr>
<tr>
<td>5m</td>
<td>[0 - 206.72]</td>
<td>(206.72 - 255]</td>
</tr>
<tr>
<td>6m</td>
<td>[0 - 198.32]</td>
<td>(198.32 - 255]</td>
</tr>
<tr>
<td>7m</td>
<td>[0 - 214.07]</td>
<td>(214.07 - 255]</td>
</tr>
</tbody>
</table>

Table 7.18: 2 Level Red Channel Cropped As Is Frames Average Pixel Intensity Threshold Ranges

<table>
<thead>
<tr>
<th>Class</th>
<th>0s Threshold Ranges</th>
<th>1s Threshold Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1m</td>
<td>[0 - 245.85]</td>
<td>(245.85 - 255]</td>
</tr>
<tr>
<td>2m</td>
<td>[0 - 243]</td>
<td>(243 - 255]</td>
</tr>
<tr>
<td>3m</td>
<td>[0 - 243.01]</td>
<td>(243.01 - 255]</td>
</tr>
<tr>
<td>4m</td>
<td>[0 - 216.11]</td>
<td>(216.11 - 255]</td>
</tr>
<tr>
<td>5m</td>
<td>[0 - 220.58]</td>
<td>(220.58 - 255]</td>
</tr>
<tr>
<td>6m</td>
<td>[0 - 210.33]</td>
<td>(210.33 - 255]</td>
</tr>
<tr>
<td>7m</td>
<td>[0 - 213.27]</td>
<td>(213.27 - 255]</td>
</tr>
</tbody>
</table>
REFERENCES


Topical Meeting Series (SUM), pages 1–2, 2019.


