University of Utah
UNDERGRADUATE RESEARCH JOURNAL

NON-ANTHROPOMORPHIC DEEP LEARNING: DEVELOPING THE "EYES FOR THE INTERNET OF THINGS"

by

Stefan Kapetanovic,
and Faculty Mentor Rajesh Menon

In

Department of Electrical and Computer Engineering

A Senior Honors Thesis Submitted to the Faculty of
The University of Utah
In Partial Fulfillment of the Requirements for the

Honors Degree in Bachelor of Science

Approved:

Rajesh Menon, PhD Thesis Faculty Supervisor	Florian Solzbacher, PhD Chair, Department of Electrical and Computer Engineering
Neil Cotter, PhD Honors Faculty Advisor	Sylvia D. Torti, PhD Dean, Honors College

December 2017

ABSTRACT

Deep Learning (DL) and Convolutional Neural Networks (CNNs) have been established as a powerful class of models for image recognition problems. Encouraged by this, we implemented a system that can accurately predict handwritten numbers from a raw CMOS machine image. Current forms of deep learning do not target non-anthropomorphic camera images that our sensors will be producing. The concept of the project is that an implemented machine system can interpret our CMOS camera data thus having the ability to make out what it captures. To develop our system it took: creating a dataset, developing a trained convolutional neural network (CNN), and testing our system on live images. Our results included {0,1} at 99%, {0-4} at 80.6%, and {0-9} at 57.0% prediction accuracy respectively. In theory, from this we will have composed a system that can function on "non-human cameras as the eyes for the internet of things" for every machine system with an image capturing sensor embedded within it.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Background and Related Work

Deep learning (DL), a subfield of machine learning (ML), has been widely applied to image recognition/classification. Work on image recognition is commonly noticed services in Google and Apple Photos. A simple example of this image recognition is easily seen when a user types "dog" in the search bar and all the images with a dog are filtered at an astounding ~100% rate. To extend this application to data generated by a camera comprised of only a standard Complementary Metal Oxide Semiconductor
(CMOS) image sensor with no lens was our objective. The premise being that if machines were able to decipher their surroundings limitless advances in computing, robotics, medical imagery, autonomous driving, etc. could be based off these root principles. For example, imagine a machine being able to examine MRI scans without a doctor having to look through them and thus diagnosing a patient as a virtual assistant. It could give details that the doctor could then look into, assess, and ultimately help treat. If machine image recognition could advance to consistent and reliable accuracy, doctors could focus on patient interaction and improving practices with the security of having machines do thousands of image recognitions in an unprecedented amount of time. Cameras generally include lens to allow humans to distinguish what the sensor captures. When the lenses, stabilization, optimization and camera setting are all removed you're left with the bare bones of what the machine data really captures.

A primary motivation for these tests stems from the aspect that non-anthropomorphic images have not been widely targeted by DL and ML (a detailed overview of the differences between machine learning and deep learning are given further along). With a variety of image classification systems having already been implemented for anthropomorphic image

recognition/classification we theorized that the same general concept could be applied to machine data images. This concept could revolutionize the amount of physical space image recognition requires in devices while also stimulating machine computing power. With a strong understanding of the background and other related work we were confident that implementing a machine using only deep learning applications was entirely possible.

Machine data images are unreadable to the human eye; an example of a handwritten 7 taken by a machine CMOS sensor is given below:

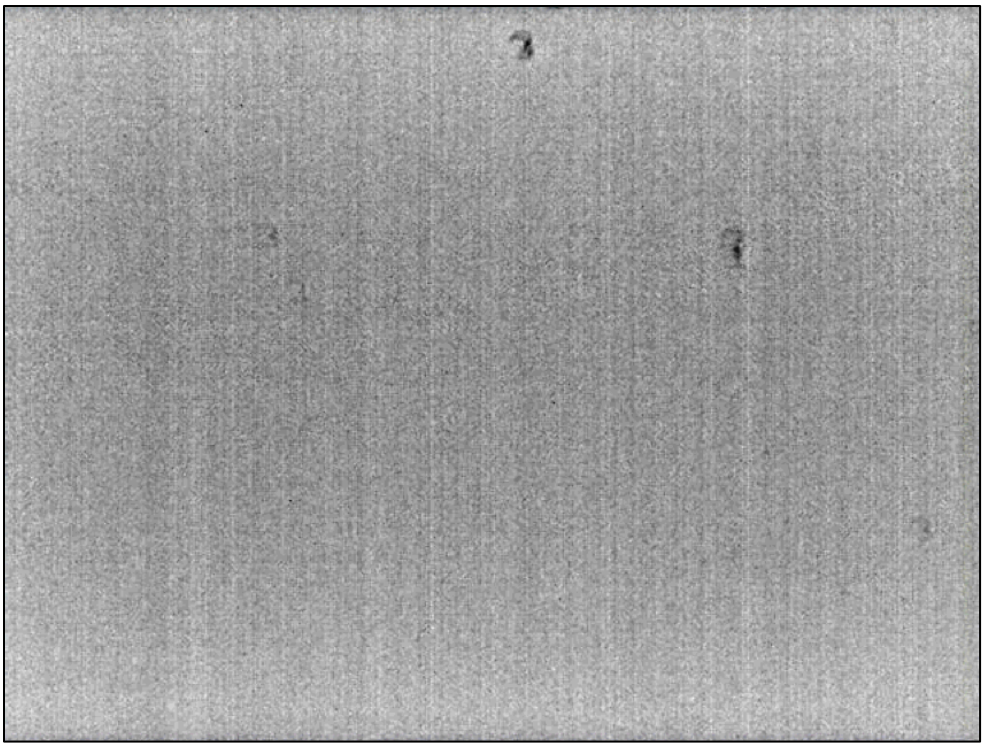


Figure 1: Depicts one image of our very own custom data set. The dark portions above are dust particles while the subtle streaks of white/grey are light.

To ensure that our concepts could transition into a project, experimentation had to be done before our deep dive into expensive time consuming big data computing. This process initially started with image reconstruction and feature extraction based off images of scattered CMOS readings. Image reconstruction techniques are used to create 2-D and 3-D images from sets of 1-D

projections. These reconstruction techniques form the basis for common imaging modalities such as CT, robotic imagery, MRI, autonomous sensors, and PET. [1] The mathematical foundations for these reconstruction methods are the Radon transform, the inverse Radon transform, and the projection slice theorem. Computational techniques include filtered back projection and a variety of iterative methods. We used pre-existing MATLAB radon transform and projection slice algorithms to reconstruct our images. Several projection geometries are commonly used, including parallel beam, fan beam, and cone beam. One of the principle reasons we started using MATLAB as our interface for this project was because of the sensational analysis and toolbox tools it contained. Typical deep learning projects are implemented using Python, which is an object-oriented, high-level programming language with dynamic semantics. Python gives an interesting approach with a lot of flexibility in designing a successful interface but lacks prebuilt analysis. Some of the analysis in MATLAB includes: capabilities for bagOfFeatures, Classification Learners, confusion matrixes, encoding plots, and powerful data processing capability/flexibility. The Shepp-Logan phantom image is often used to evaluate different reconstruction algorithms but for our purposes we simply used our judgment. To begin this experiment, a dark lab was used to gather data (images in the form of. png's) off LED screens. After our data samples were collected and results were captured we had a tremendous amount of capability.

Some examples of a Radon transform and splice theorem conversion performing on our scattered sensor image are given below:

The left reference is the image constructed on an LED board run by a simple Arduino (Open-source electronic prototyping platform) configuration. We used that as the reference image to compare our reconstructed image upon. The middle raw sensor image is what our CMOS image sensor captures without optic lenses. The right constructed image is the raw sensor image run through our MATLAB image reconstruction algorithms, Radon transform and splice theorem.

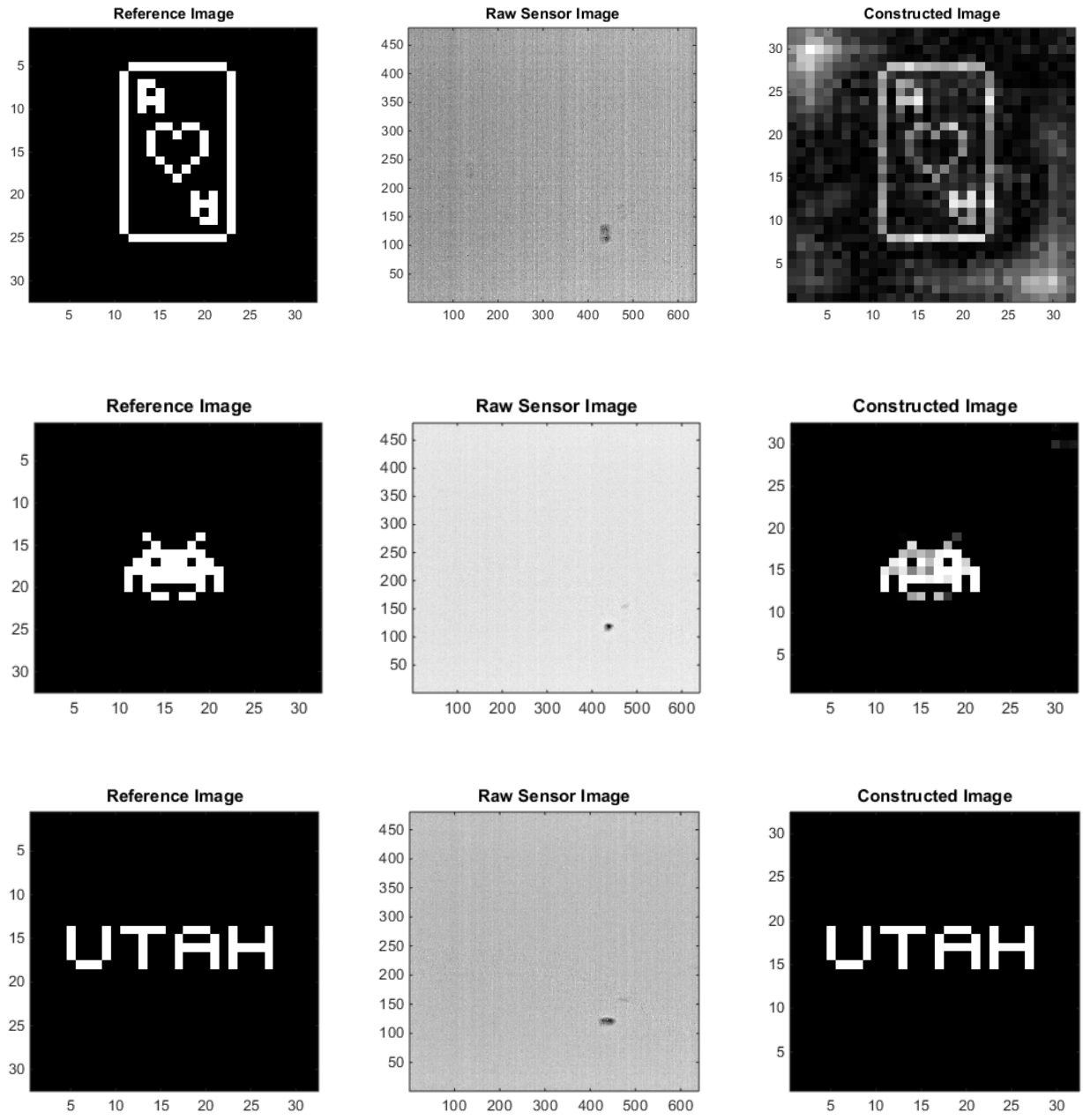


Figure 2: Positive image reconstruction progression is shown to test MATLAB analysis and to trust our sensors capturing capability of raw data images.

The next progression for the project was to capture raw sensor images of handwritten numbers using the MNIST online dataset. The reconstruction of the image was done, as well as initially programming for our system to understand what it was seeing. Although the reconstruction of the image can be done successfully, getting the system to understand what the machine is

reconstructing will become the largest challenge in our project's implementation.

1.2 Deep Learning

Deep learning is part of a broader family of machine learning methods based on learning representations of data. This makes machine learning and deep learning extremely similar terms but the subtle differences are vital when choosing to implement one or the other in a problem. With recent advancements in deep learning algorithms and GPU technology, we are able to solve problems once considered impossible in fields such as computer vision, natural language processing, medical imaging, and robotics. The key aspect stems from the data network the system runs upon. The system relies heavily on the data to be able to "learn" and generally the more data captured the better the results. The system learns from the data by extracting features and using those features as filters to progress through images, thus essentially storing details in the network that can be used in future iterations of image recognition. For example, if the data presented is a series of human heads the system will learn what a nose, eye, mouth, face shape, etc. mean and incorporate that as filters to distinguish between faces. Thus the system not only can learn features from the data, it can use that data to predict the next image it is given and presumably tell you not only that the image contains a human head but whose human head it is. Deep learning has also been known to be called deep structured learning, hierarchical learning or deep machine learning which applies a set of algorithms that attempt to model high-level abstractions in data. Common applications were targeted toward computer vision, automatic speech recognition, natural language processing, audio recognition and bioinformatics in the past. My work aims to incorporate deep learning in application with raw data imaging for the purpose of connecting machines to the world as well as other machines in a far smarter approach than ever before.

Research in this area attempts to make better representations and create models to learn these representations from large-scale unlabeled data. To further understand deep learning you must understand the principle of a neural network. Deep learning uses neural networks, which have been around for a few decades; what's changed in recent years is the availability of large labeled datasets and powerful GPUs. Neural networks are inherently parallel algorithms and GPUs with thousands of cores that combined we can take advantage of to dramatically reduce computation time needed for training deep learning networks. [2] There are several neural network techniques being used for once considered impossible feats in fields so it becomes crucial to understand what a neural network is and how to distinguish between them. Below is a simple representation of the most used neural networks in the field of image classification:

- Artificial Neural Network (ANN) – An artificial neural network consists of an input layer of neurons (or nodes, units), one or two (or even three) hidden layers of neurons, and a final layer of output neurons. Each connection is associated with a numeric number called weight. It contains an activation function. The purpose of the activation function is, besides introducing nonlinearity into the neural network, to bind the value of the neuron so that the neural network is not paralyzed by divergent neurons. A common example of the activation function is the sigmoid (or logistic) function. Other possible activation functions are arc tangent and hyperbolic tangent. They have similar response to the inputs as the sigmoid function, but differ in the output ranges. It has been shown that a neural network constructed the way above can approximate any computable function to an arbitrary precision. Numbers given to the input neurons are independent variables and those returned from the output neurons are dependent variables to the function being approximated by the neural network. Inputs to and outputs from a neural network can be

binary (such as yes or no) or even symbols (green, red, ...) when data are appropriately encoded.

- Multilayer Perception (MLP) – Multilayer perceptron network (MLP), FIR neural network and Elman neural network were compared in four different time series prediction tasks. Time series include load in an electric network series, fluctuations in a far-infrared laser series, numerically generated series and behavior of sunspots series. FIR neural network was trained with temporal back propagation learning algorithm. Results show that the efficiency of the learning algorithm is more important factor than the network model used. Elman network models load in an electric network series better than MLP network and in other prediction tasks it performs similar to MLP network. FIR network performs adequately but not as good as Elman network.
- Recursive Neural Network (RNN) – Unlike standard neural networks, recursive neural networks are able to process structured inputs by repeatedly applying the same neural network at each node of a directed acyclic graph (DAG). In the past they have only been used in settings where another (often symbolic) component was first used to create directed acyclic graphs. These DAGs were subsequently given as input to the RNN. In such a setting, each non-leaf node of the DAG is associated with the same neural network. A DAG (essentially) is a binary tree. In other words, all network replications share the same weights. The inputs to all these replicated feedforward networks are either given by using the children's labels to look up the associated representation or by their previously computed representation
- Recurrent Neural Network (RNN) – In recurrent networks, history is represented by neurons with recurrent connections - history length is unlimited. Also, recurrent networks can learn to compress whole history in low dimensional space, while feedforward

networks compress (project) just single word. In feedforward networks, history is represented by context of N − 1 words - it is limited in the same way as in N-gram backoff models. Recurrent networks have possibility to form short term memory, so they can better deal with position invariance; feedforward networks cannot do that. [27]

- Long term-short memory (LTSM) – Long Short-Term Memory (LSTM) is a specific recurrent neural network (RNN) architecture that was designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs. LTSM overcomes some modeling weaknesses of RNNs, is conceptually attractive for the task of acoustic modeling.
- Shallow Neural Network – In short, "shallow" neural networks is a term used to describe NN that usually have only one hidden layer as opposed to deep NN which have several hidden layers, often of various types. Multi-layered neural nets achieve better results than shallow ones that have the same computational power (e.g. number of neurons or connections). The main explanation is that the deep models are able to extract/build better features than shallow models and to achieve this they are using the intermediate hidden layers.
- Convolutional Neural Network (CNN) – The convolutional network extracts successively larger features in a hierarchical set of layers. Convolutional neural networks (CNN) incorporate constraints and achieve some degree of shift and deformation invariance using three ideas: local receptive fields, shared weights, and spatial subsampling. The use of shared weights also reduces the number of parameters in the system aiding generalization. Convolutional neural networks have been successfully applied to character recognition [3], [5], [22]–[24]. The network consists of a set of layers each of which contains one or more planes. Approximately centered and normalized images enter

at the input layer. Each unit in a plane receives input from a small neighborhood in the planes of the previous layer. Each plane can be considered as a feature map which has a fixed feature detector that is convolved with a local window which is scanned over the planes in the previous layer. Multiple planes are usually used in each layer so that multiple features can be detected. These layers are called convolutional layers. Once a feature has been detected, its exact location is less important. Hence, the convolutional layers are typically followed by another layer which does a local averaging and subsampling operation.

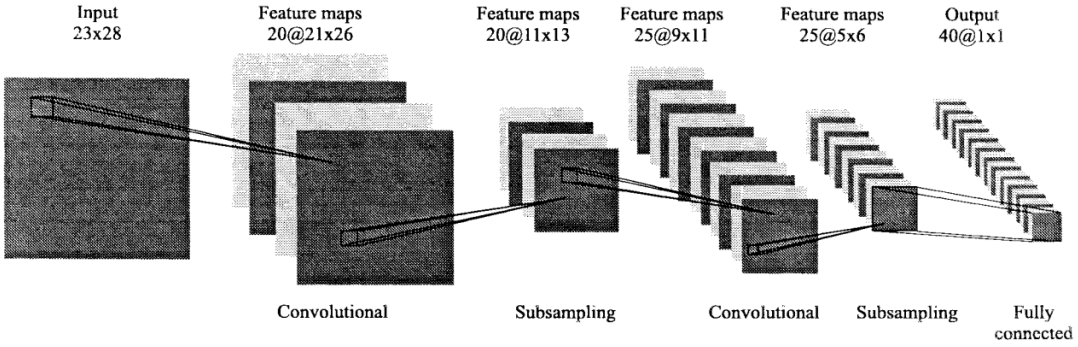


Figure 3: A typical convolutional neural network structure

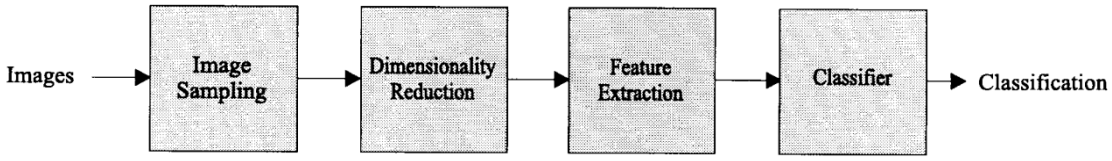


Figure 4: A high-level block diagram of the system we have used for the machine data handwritten numbers dataset we used.

In the system, our original data sample was large and composed entirely of images that did not require preprocessing. From this deduction it's confirmed that a convolutional neural net would be the best start to begin our deep learning. Convolutional neural networks are essential tools for deep learning, and are especially useful for image classification, object detection, and

recognition tasks, which helped our goal. CNNs are implemented as a series of interconnected layers. The layers are made up of repeated blocks of convolutional, rectified linear units (ReLU), and pooling layers. The convolutional layers convolve their input with a set of filters. These layers of images combined with the feature filters could theoretically determine which handwritten number was which. The filters were automatically learned during network training. The ReLU layer adds nonlinearity to the network, which enables the network to approximate the nonlinear mapping between image pixels and the semantic content of an image. The pooling layers down sample their inputs and helps consolidate local image features. With this solid basis we were ready to begin our projects essence.

1.3 Statement of Purpose

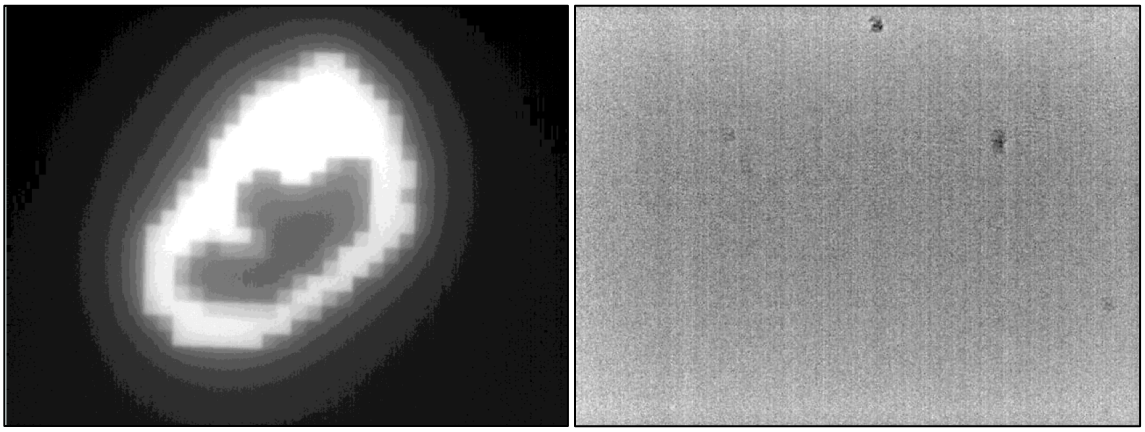
Our primary purpose was to connect machines as a system that could understand their surroundings. The ethics involved with machines learning and evolving as humans do were not taken into question. Learning and implementing a system to hopefully evolve every field imaginable was the only purpose of this project. With goals that came with limitless potential, the reality became of where to start. The background of anthropomorphic image classification weighed heavily on us to push ourselves. It was possible for a machine to reconstruct an image from a CMOS sensor data sample. And using those same algorithms with the help of a convolutional neural network and interconnected layers would allow for the machine not only to reconstruct an image internally but to also understand what it was reviewing.

2 METHODS

2.1 Project Overview

Recently, it has become common to train ML algorithms to recognize objects in images by exposing them to vast databases of labeled images [1-4]. Spectacular gains in classification accuracy have been obtained and ML algorithms are now able to make complex decisions based upon object recognition in image and video data. These algorithms are typically educated on

conventional (what we refer to as human-centric) images. Nonanthropomorphic images are images that are not discernable to the human eye. There have also been significant advances in lensless imaging, where a sensor that does not have a lens captures information from a scene or object [5-7]. Such lensless cameras offer advantages of simplicity, low cost, reduced weight and small form factors. With our initial testing for image reconstruction finished our primary project begins. Our first task was to create a database of lensless images of handwritten digits. Our lensless camera is simply a conventional bare CMOS sensor (The Imaging Source, DFM 22BUC03-ML). This sensor was used as our camera and provided the necessary machine data that a machine would produce when capturing its environment. We reduced the ambiguity in our case by making sure the camera focused on the handwritten number image. We use a liquid-crystal display (LCD) to show various images of handwritten digits from 0 to 9 [8]. Sample images of handwritten digits were obtained from the MNIST database. The MNIST database has been used for image classification test since the 1990's, and is widely used for assessing image classification accuracy. Examples of the original MNIST images and their corresponding lensless images are shown below:



(a) (b)

Figure 5: Figure (a) and figure (b) are the same handwritten number of "0" in this case, but performed on with a CMOS lens sensor and a CMOS lensless sensor.

The LCD is placed about 250mm away from and facing the CMOS sensor. The exposure time is 150ms and it averages 100 frames to reduce noise. Using this procedure, a database of exactly 70,000 images with appropriate labels was created. The digits 0-9 with approximately 7,000 numbers of each are organized in individual folders and tagged correspondingly. This process took quite a bit of time so as images were being captured, implementing the machine learning algorithm and the CNN occurred simultaneously. The variables considered when training are interesting to consider. To generalize, knowing that the more data the system can include for the CNN typically generates better end results motivated the need for large image quantities. With more data also comes a higher computing time model as a result. With datasets the size of 250,000 possible this process could take several days. And as I will explain in detail later encoding the images, training, and then finally testing on this relatively large data sample size is very time sensitive. Time was a fine line that had to be played with throughout our training. In our particular experiment, measuring if our training was being done properly, if the threshold for the amount of data was too high or too low, and what the prediction accuracy was once the measurements ({0,1}, {0-4}, and {0-9} lens and lensless) were all taken was the experimentation process. Throughout the process a trained ML algorithm was implemented on this dataset to ensure accuracies were within range bounds. To do this we had to generate our CNN in a fashion that could work relatively quickly on the data while being very precise. After completely finishing our data capturing and CNN/ML groundwork experimentation for accuracy began. Finally, our results demonstrated that the trained ML algorithm is able to classify the digits with accuracy as high as 99% for 2 digits on lensless CMOS sensor images. Our approach clearly demonstrates the potential for non-human cameras in machine-based decision- making scenarios. Previously, our project demonstrated that frames captured by a bare CMOS sensor could be used to reconstruct images that are recognizable by humans. Here, we show that such human-centric

reconstruction is actually not necessary for machine-based image recognition and classification. Specifically, the system trains a ML algorithm on a database of frames created by a lensless camera and demonstrates that the algorithm is capable of image classification using data directly from the lensless camera. A guideline of the procedure that was taken is shown below.

PROCEDURE

1. Capture images of the handwritten numbers using our CMOS image sensor in a controlled dark room environment for all three datasets (lenseless, lense, and original).

2. Once the image sets are all organized in folders of specified number I begin training.

- Training:

1. Extract features from the image sets of random numbers.

2. Properly train the CNN to behave on a variety of samples.

3. Encode the extracted features on a different image set.

- Testing:

1. After the training finished on the image sets, I test the system using different size networks.

2. After testing has occurred the plots of accuracy are made including a plot graph and confusion matrix.

3. I repeat training and testing with different random image sets for both the extracting features subsection and the encoding subsection.

4. Once the features and encoding sections have been determined for all three factions ([0-9], [0-4], and [0,1]) I move on to the next database of images.

2.2 Extract Training Features Using CNN

As raw images are fairly large (307,200 pixels in each image), it is not suitable for large-scale training. Hence, SURF (speeded up robust features) extraction [9] and subsequent K-mean clustering [10] are performed to reduce data dimension. Note that we used 2000 to 3000 images to perform feature extraction as they provide sufficient information to approximate image

features that are present throughout the whole set. Anything over plateaued and kept the accuracy within a threshold that wasn't worth the extra processing and computing time. By using bagOfFeatures, a MATLAB pre-built feature extraction algorithm our system the filtering for our CNN will come relatively easily. Determining if our features will effectively increase the prediction accuracy will be measured. One of the variables that is directly related to the accuracy of our system is how many features is used. This is vital to understand in order to be consistent. The best variation our system found in feature extraction was that more features did not serve better. Now, when extracting features we only looked for 100 features in all the images our project performed the extraction on. This proved to be more than enough and 2000-3000 images to pull from was definitely a large enough feature filtering mechanism. Considering that the system was only examining handwritten number images coupled with a controlled environment the number of features could be reduced. These extracted features are then piped to the MATLAB machine learning toolbox (also called classification learner) [11], after data is properly formatted for the tool, to perform the training. This training is crucial for the deep learning algorithm to work properly for future testing. In order to analyze the CNN our system will be also be forming confusion matrices and encoding plots. These will be showed more in the coming document sections. In brief, they provide information on the comparison results and when and if the encoding images are successfully compared for training. To ensure that our training was doing exactly what was expected/wanted with a variety of learning methods to train the model. Algorithms used to compare the training include:

- Support Vector Machine (SVM) – a pattern classification algorithm recently developed by V. Vapnik and his team at AT&T Bell Labs. , 24 can be seen as a new way to train polynomial, neural network, or Radial Basis Functions

classifiers, based on the idea of structural risk minimization rather than empirical risk minimization. [1, 18, 19, 20]

  - Linear – Linear kernel, meaning dot product
  - Cubic – Cubic kernel
  - Quadratic – Quadratic kernel

- Decision Tree – Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. Learned trees can also be re-represented as sets of if-then rules to improve human readability. These learning methods are among the most popular of inductive inference algorithms and have been successfully applied to a broad range of tasks from learning to diagnose medical cases to learning to assess credit risk of loan applicants. [12] and [18]
- K-Nearest Neighbors (KNN) – The nearest neighbor classifiers require no preprocessing of the labeled sample set prior to their use. The crisp nearest-neighbor classification rule assigns an input sample vector y, which is of unknown classification, to the class of its nearest neighbor [17], [23], [31]
- Ensemble Subspace Discriminant – A classification ensemble is a predictive model composed of a weighted combination of multiple classification models. In general, combining multiple classification models increases predictive performance. To explore classification ensembles interactively, use the Classification Learner app. [16]
- Logistic Regression – Logistic regression is named for the function used at the core of the method, the logistic function. The logistic function, also called the sigmoid function was developed by statisticians to describe properties of

> population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits. [15]

All of these algorithms were implemented to see which had the best effects on our data set. Overall, the analysis saw that whether or not the case was lens or lensless the training was relatively consistent in results. Training observation saw patterns with algorithms that reoccurred more often then not. By taking the highest accuracy percentage every time the CNN was trained. To further determine our CNN was working properly, the system needed to reflect that both a lens database and a lensless database accuracy model was showing similar accuracy trends. To see the difference between our lens and lensless data the project will include three different cases while fluctuating the training images. Our first case {0,1}: expected to have particular outcomes for both the lens and lensless classifications. Our team expected relatively high results for these and maybe a slight difference where the lensless faltered in comparison to the lens. Our second case {0-4}: expected that this dataset would also yield considerably high prediction accuracies hovering around 85-90%. For our last case, {0-10}: expected prediction accuracies around 80-85%. Now from our past experimentation the project had more understanding and ealistic views. The system goal wanted our average of the two datasets to hover around 65%. The results ensued that {0-9} came to prediction accuracies at lens 93% and lensless 57%. The general trend of our neural net showed us that the training images that were prime and not taking too much computing proved to be 19,000 images. Anything below this figure and the results could vary and anything above proved to be a threshold that cost too much time in valuable computing. Once trained, each model is measured for accuracy using a set number of test images as described below.

2.3 Test Filtering on New Image Sets

Testing always began with a picked model containing the highest prediction accuracy. In most cases, the SVM algorithm achieved the highest accuracy. The first area of training was to compare lens and lensless prediction comparisons. Testing our CNN and ML on new data sets of handwritten numbers will be conducted on a lens dataset. This is an intermediate step to test our program on both lens and lensless captures. The following data results contain the bagOfFeatures extraction features as well as the number of encoding images used.

LENS

- **10 digits - bagOfFeatures, encoding images: 10800 & 19050, 6800 & 14050, 3600 & 7800, 1600 & 3800, 500 & 600**

25% HOLDOUT: 14287, 10537, 5850, 2850, 450

HIGHEST 94.6 SVM Quadratic bagOfFeatures features [10 20 50 100 150 200 250 300] encoding images [10 20 50 75 100 150 200 250 300 350 400]

2nd HIGHEST 94.2 SVM Quadratic bagOfFeatures features [10 20 50 100 200 300] encoding images [10 20 50 75 100 200 250 300 400]

3rd HIGHEST 94.3 SVM Quadratic bagOfFeatures features [10 50 100 200] encoding images [10 20 50 100 200 400]

4th HIGHEST 94.8 SVM Cubic bagOfFeatures features [10 50 100] encoding images [10 20 50 100 200]

5th HIGHEST 82.4% Ensemble Subspace Discriminant bagOfFeatures features [50] encoding images [10 50]

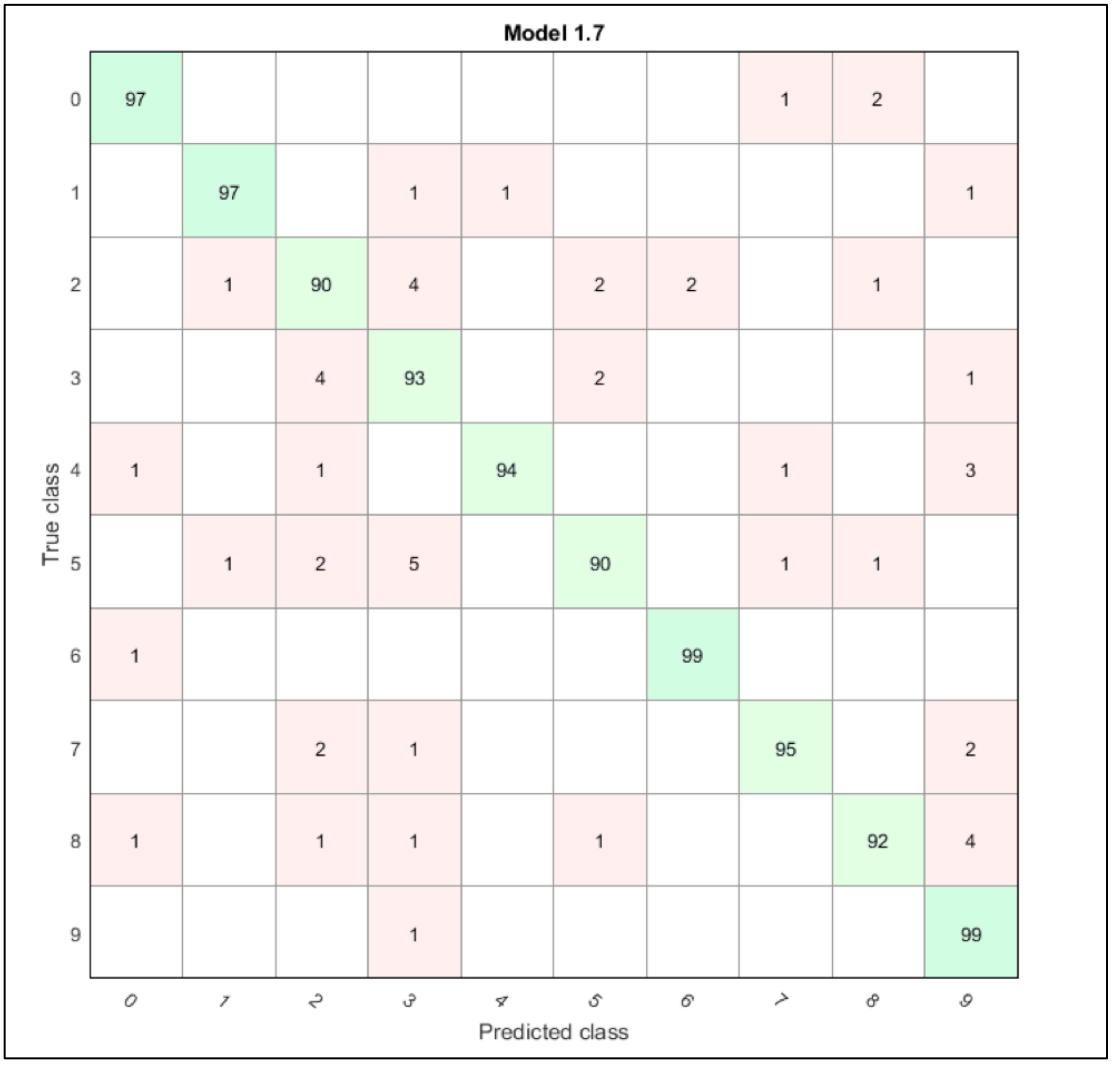


Figure 6: A confusion matrix showing the training system and how it works in comparison of numbers.

A confusion matrix is a direct analysis tool provided by the MATLAB Machine Learning Toolbox. It allows us to compare the CNN and ML to see if the training of images and features were correlated correctly. For a direct example, in Figure 6 we see that the blank spots are contained when numbers like 0 and 9, 1 and 8, or 4 and 6, etc. That implies that those numbers were not mistaken to one another during the training system. The high diagonal coordination shows when the system came to two numbers that are the same. The percentages in that diagonal stretch are averaging around 95% of correct matches. The minimal numbers across the board where numbers like 2 and 3, 3 and 8, and 5 and 9, etc. intersect are percentages where the system

confused the two numbers 1-5% of the time. The lens data set was extremely accurate but does not portray machine images.

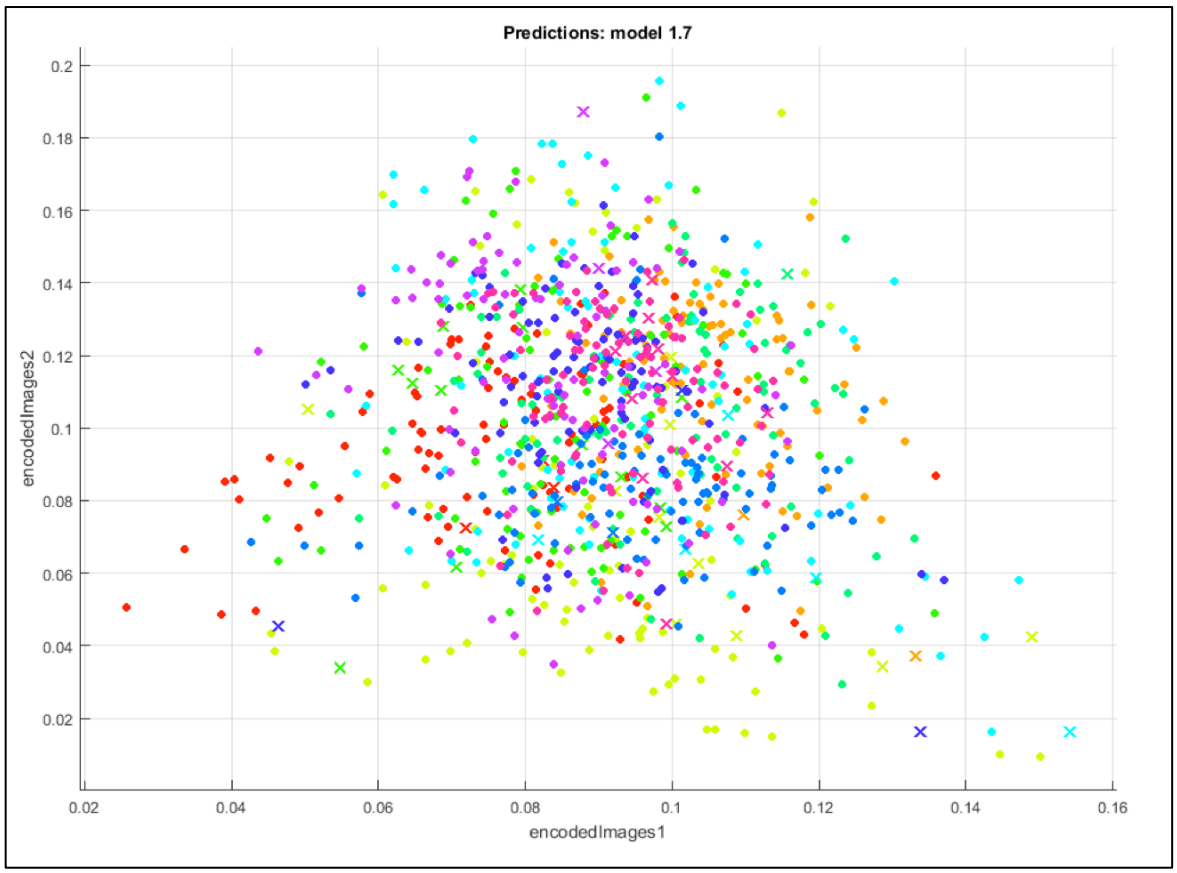


Figure 7: The encoding plot also shows the highest prediction accuracy on the lens dataset.

An encoding plot is a direct analysis tool provided by the MATLAB Machine Learning Toolbox as well. It is a bit more complex to read. There are a few key takeaways in this plot that are vital to consider. The correlation of the points in the same clustered area, the dot and "X" visual tool, as well as the axes are crucial in understanding what the plot actually means. The encoding plot correlates the correct fits of the number matching that the system makes. The "X" points mean misses in the data where the numbers that were matched did not match to the system. Now that is a comparative point because the numbers may be wrong but the system may think it's correct and vice versa. The more dots and the better proximity of dots demonstrate a better system of matching. In general, it allows us to compare the CNN and ML to see if the training of images

and features were correlated correctly.

The following results go on to provide interesting information on the principles implemented. When it comes to the lens data set we successfully captured the principle that getting a system to learn on the machine raw data is much more difficult than on a general anthropomorphic image. In most cases, the {0-9} case is most valuable since the data is most crucial then. Creating a system that can distinguish between a 0 and a 1 like in the {0,1} case scenario is not impressive. Being able to correlate through several features and then train a CNN to test on a new larger set of data sets becomes the whole concept. In regards to the middle case scenario of {0-4} is also important to consider when trying to formulate the best testing strategy for training and testing samples. The {0-9} case and the {0-4} case share the same amount of feature extraction since this was implemented to be best in the training of the CNN and ML.

- **5 digits –**

HIGHEST 97.6% SVM Quadratic bagOfFeatures features [10 20 50 100 150 200 250 300] encoding images [10 20 50 75 100 150 200 250 300 350 400]

2nd HIGHEST 97.2 SVM Quadratic bagOfFeatures features [10 20 50 100 200 300] encoding images [10 20 50 75 100 200 250 300 400]

3rd HIGHEST 97.8 SVM Cubic bagOfFeatures features [10 50 100 200] encoding images [10 20 50 100 200 400]

4th HIGHEST 96% SVM Quadratic bagOfFeatures features [10 50 100] encoding images [10 20 50 100 200]

5th HIGHEST 96.8% Ensemble Subspace Discriminant bagOfFeatures features [50] encoding images [10 50]

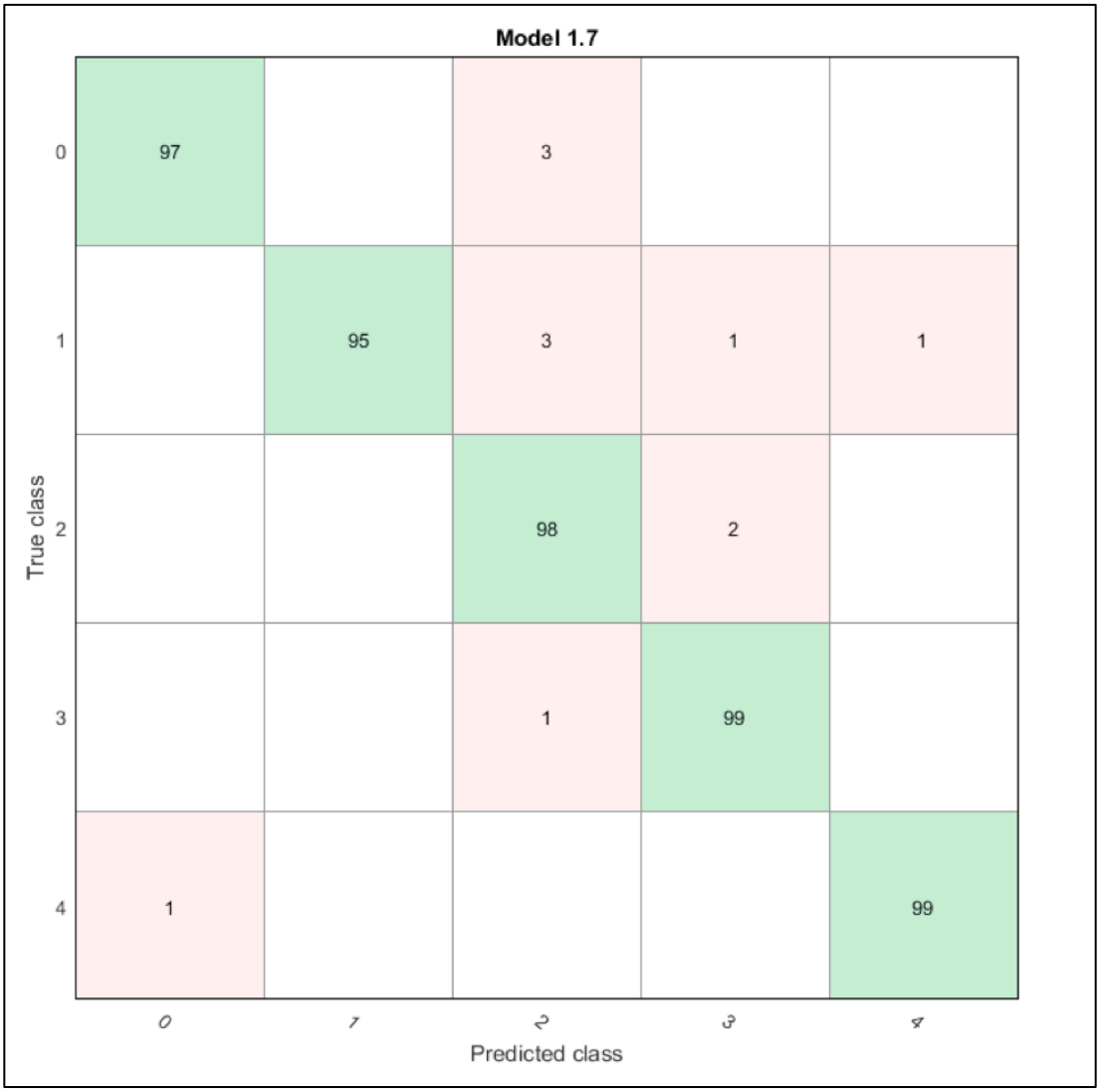


Figure 8: This confusion matrix shows the highest prediction accuracy on the lens dataset for {0-4} indicating the feature extraction is working successfully.

The lensed {0-4} case scenario confusion matrix shows a pattern developing. Lens solutions clearly are not faltering to the levels of lensless models. This non-regression implies that the future of raw image recognition in the case scenario {0-9} will also be high. Anthropomorphic training is much easier with direct features to extract and less room for error amongst widely differing images.

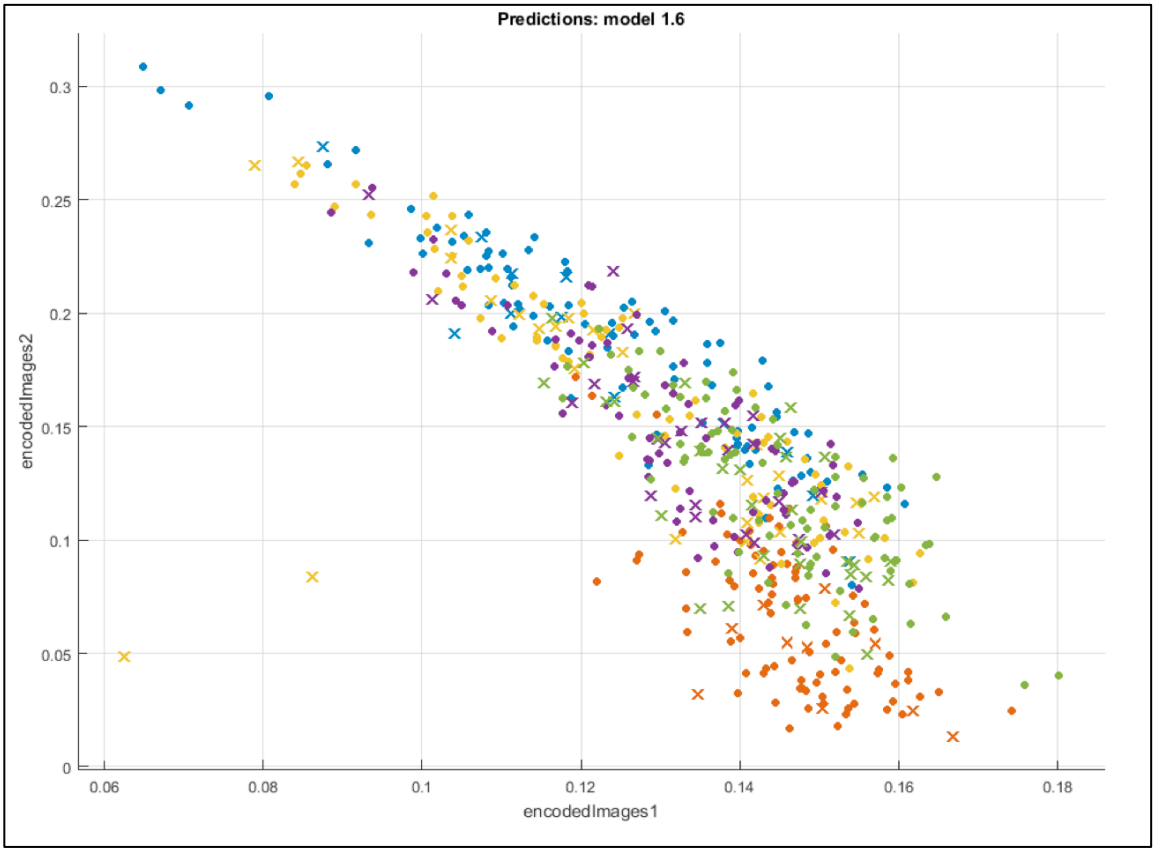


Figure 9: This encoding plot shows the highest prediction accuracy on the lens dataset for {0-4} showing the correlation of successful feature extraction.

The correlation of lensed case scenario {0-4} shows the ease at which the success is generated from this system. The linear progression of encoding images as well as the very few "X" marks on the plot indicates ease in training. The rare outliers are far off the correlation pattern and that indicates that the layers within the training are very often confused with previous data collection. Encoding was a crucial factor in determining the outcome of training. If our encoding sizes varied, results were extremely volatile. If you examine the encoding images in each case you'll notice the trend of this matrix being the best option [10 20 50 75 100 150 200 250 300 350 400].

- **2 digits -**

HIGHEST 98.5% Logistic Regression bagOfFeatures features [10 20 50 100 150 200 250 300] encoding images [10 20 50 75 100 150 200 250 300 350 400]

2nd HIGHEST 100% Quadratic Discriminant bagOfFeatures features [10 20 50 100 200 300]

encoding images [10 20 50 75 100 200 250 300 400]

3rd HIGHEST 100% KNN Fine KNN bagOfFeatures features [10 50 100 200] encoding images [10 20 50 100 200 400]

4th HIGHEST 100% SVM Linear bagOfFeatures features [10 50 100] encoding images [10 20 50 100 200]

5th HIGHEST 100% SVM Quadratic bagOfFeatures features [50] encoding images [10 50]

Up until this point the training and limited testing was going extremely well. The differences in the lens and lensless application results were very different. Using the same principles in the lensed variation the system had problems with training and testing in this environment. As you can image the differences in machine images are extremely subtle which was a direct cause of poor feature extraction. Lensless accuracy will be conveyed below in much more detail.

LENSELESS

- **10 digits - bagOfFeatures, encoding images: 10800 & 19050, 6800 & 14050, 3600 & 7800, 1600 & 3800, 500 & 600**

HIGHEST 57.0% SVM Quadratic bagOfFeatures features [10 20 50 100 150 200 250 300] encoding images [10 20 50 75 100 150 200 250 300 350 400]

2nd HIGHEST 51.1% SVM Linear bagOfFeatures features [10 20 50 100 200 300] encoding images [10 20 50 75 100 200 250 300 400]

3rd HIGHEST 55.5% SVM Linear bagOfFeatures features [10 50 100 200] encoding images [10 20 50 100 200 400]

4th HIGHEST 53.6 Ensemble Subspace Discriminant bagOfFeatures features [10 50 100] encoding images [10 20 50 100 200]

5th HIGHEST 36.0% Tree Complex Tree bagOfFeatures features [50] encoding images [10 50]

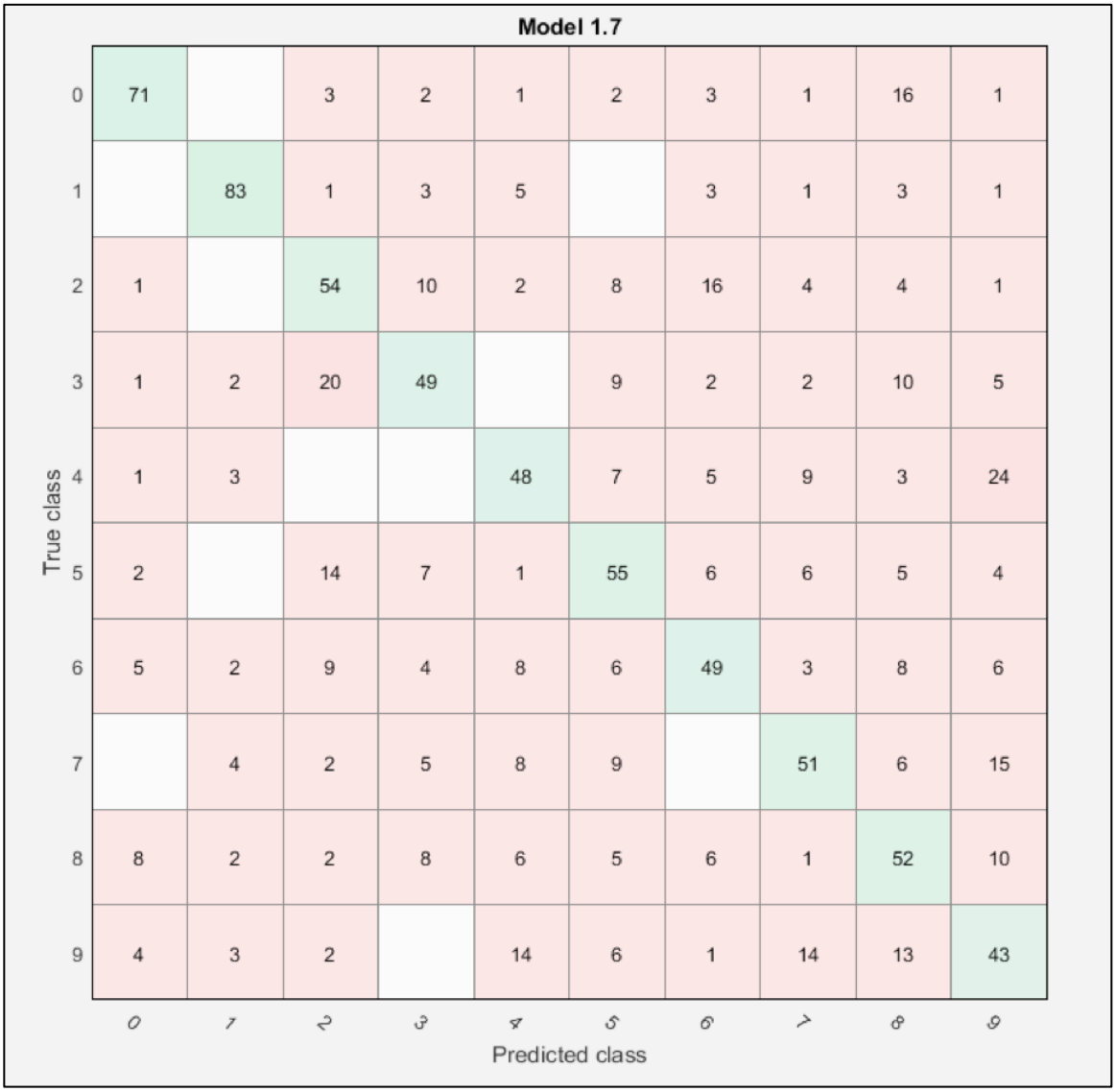


Figure 8: This confusion matrix shows the {0-9} highest prediction accuracy for the lensless dataset.

This confusion matrix shows a lot more error. The relative decrease in empty intersections shows the percentage of error increasing in about every case. Another trend observed in the lensless case was that the amount of structures giving highest accuracies varied almost every run through. From the expected SVM Linear to Ensemble Subspace Discriminant and Tree Complex Tree the structures from the CNN were unique in most cases. The accuracy obtained from these measurements was 57%.

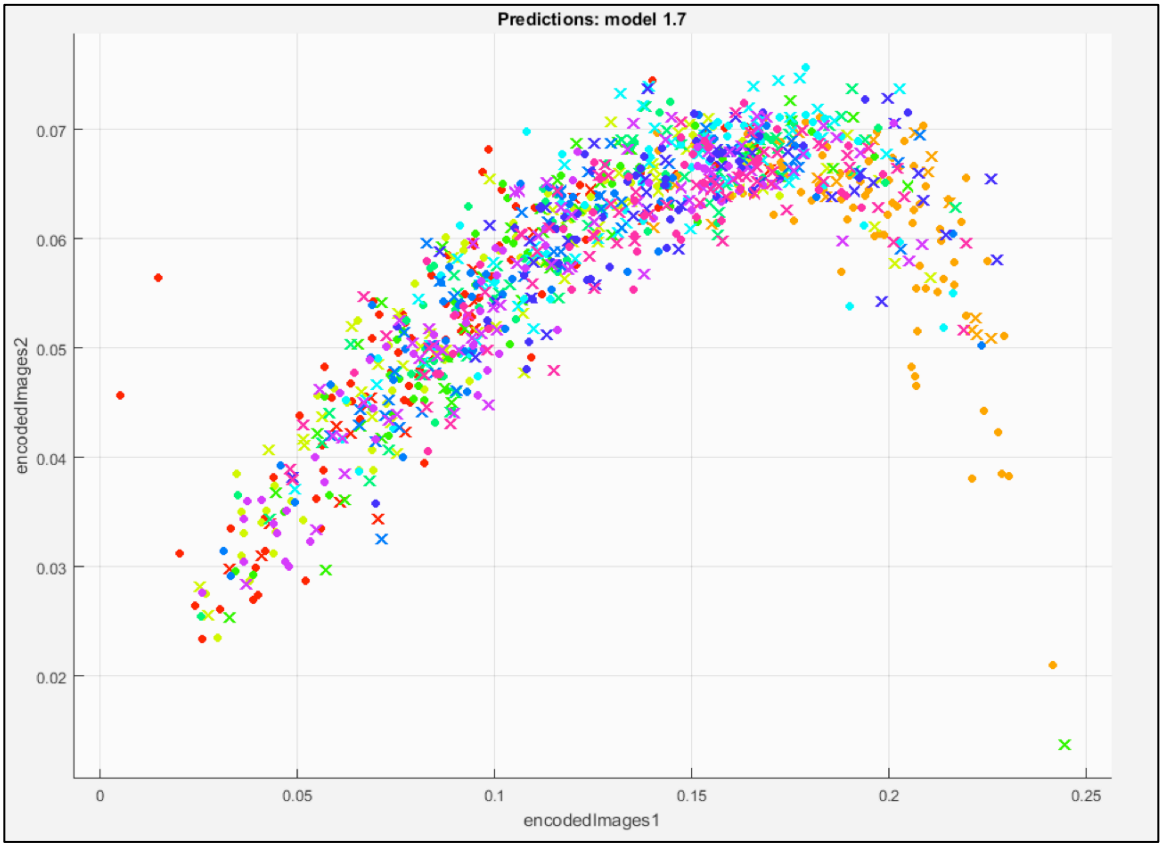


Figure 9: This encoding plot shows the highest prediction accuracy for the lensless dataset.

This encoding plot showed a very interesting pattern. The density in areas makes the error hard to see but plots of individual data showed that almost half of the training features failed. This drastically reduced the number of features in the neural network and subsequently reduced the amount of layer classifiers. With few outliers and few accurate training spots the system is relatively poor at feature extraction. One solution to consider is implementing a better platform for image capturing to ensure the images are of a certain quality particularly needed for precise feature extraction.

- **5 digits –**

HIGHEST 80.6% SVM Linear bagOfFeatures features [10 20 50 100 150 200 250 300] encoding images [10 20 50 75 100 150 200 250 300 350 400]

2nd HIGHEST 78.0% SVM Quadratic bagOfFeatures features [10 20 50 100 200 300] encoding images [10 20 50 75 100 200 250 300 400]

3rd HIGHEST 74.8 % SVM Linear bagOfFeatures features [10 50 100 200] encoding images [10 20 50 100 200 400]

4th HIGHEST 54.4% SVM Linear bagOfFeatures features [10 20 50 100 200 300] encoding images [10 20 50 75 100 200 250 300 400]

5th HIGHEST 41.9% Quadratic Discriminant bagOfFeatures features [50] encoding images [10 50]

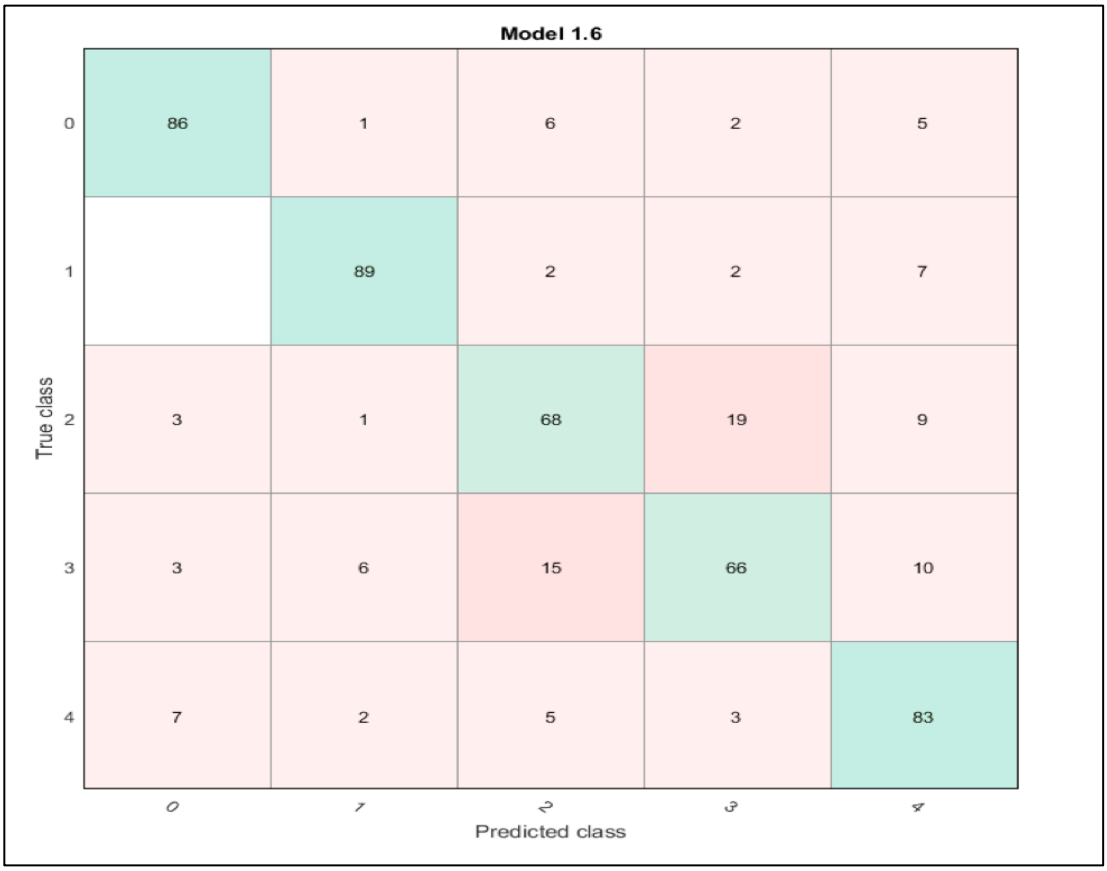


Figure 10: This confusion matrix shows the highest prediction accuracy for the {0-4} lensless dataset.

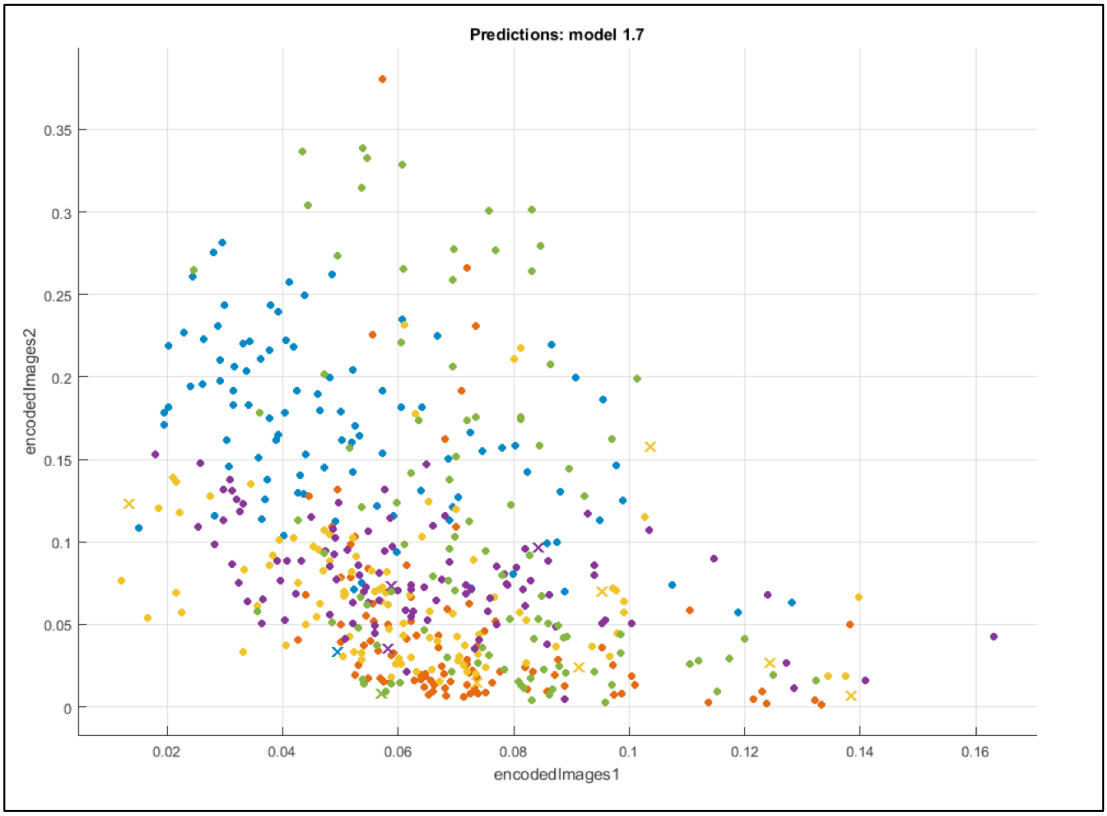


Figure 11: This encoding plot shows the highest prediction accuracy for the {0-4}lensless dataset

This lensless confusion matrix and encoding results show that even this dataset was affected. The pattern of feature extraction failing is not as visible in this set when compared to {0-9} but the accuracy decrease is narrowed only to features. No other scenario could cause such a drastic drop off in accuracy considering all the successive experimentation performed throughout.

- **2 digits -**

HIGHEST 96% SVM Linear bagOfFeatures features [10 20 50 100 150 200 250 300] encoding images [10 20 50 75 100 150 200 250 300 350 400]

2nd HIGHEST 98.5% SVM Cubic bagOfFeatures features [10 20 50 100 200 300] encoding images [10 20 50 75 100 200 250 300 400]

3rd HIGHEST 98.5% SVM Cubic bagOfFeatures features [10 50 100 200] encoding images [10 20 50 100 200 400]

4th HIGHEST 97% Tree Simple bagOfFeatures features [10 50 100] encoding images [10 20 50 100 200]

5th HIGHEST 56% Logistic Regression bagOfFeatures features[50] encoding images [10 50]

3 RESULTS

3.1 Key Accomplishments

Template matching methods such as [24] operate by performing direct correlation of image segments. Template matching is only effective when the query images have the same scale, orientation, and illumination as the training images [24]. This type of testing is what our sytem implemented and our margins of error in terms of training virus testing images were conclusive to be extremely minimal. Our controlled environment never changed and allowed for our captured query images to reflect identical properties. The results for the {0,1} dataset were extremely similar in both cases. The lens database, which we assumed, was going to be slightly better than the lensless determined our programming was functioning correctly. The lens system was much better than our lensless, even more so than anticipated. The prediction accuracies were approximately 20-30% higher for lens cases when compared to lensless cases. We found that regardless, when only the dataset included handwritten numbers {0,1} the predication accuracy of our system was a highly optimistic sign of approximately 99%. Only a few outliers deterred our system from correctly understanding the difference between a 0 and a 1. The lens {0-4} case ended up giving us a prediction accuracy of 96% that was an incredible result not expected to occur. The {0-4} case almost measured up exactly to the {0,1} case. The lensless case was relatively lower but came with the average of ~80% prediction accuracy. This type of drop was concerning because accuracy was stumbling extremely low with an additional three numbers. This result was even lower than expected. It did not bode well for the {0-9} case scenario, but our system had hope that we could still accomplish results higher than 60-70% accuracy. This was an interesting note and the large difference truly began to occur one we started incorporating all the digits {0-9}. The ending prediction accuracy our system gave for the {0-9} case was 57%.

With this basis our project was submitted a paper to the OSA Imaging and Applied Optics Congress. Our paper passed editorial review but not the peer review stage. Unfortunately our paper was also adds a new and impactful result to the field of optics that requires rapid publication citing, “You show that lensless imaging can be combined with machine learning. You apply the known machine-learning method to classify the images captured by a camera without optics and no new method was developed. The results are only partially successful when more than a couple of characters are used, but you demonstrate that the technique is viable. However, there are questions regarding whether the evaluation employed images that were contained within the set used for training and whether performance is affected by changes in the setup (modifying the distance, tilting the screen, etc.).” With this feedback information our sytem had to go farther in our exploration. And the key was to incorporate deep learning with modifying the image’s distance and also incorporating tilting the screen. These are understandable conclusions that directly assume this technology could be further applied to a real world application where images are not always perfectly coordinated. Our machine learning results with feature and encoding experimentation are given below:

LENS

- **10 digits - bagOfFeatures, encoding images: 10800 & 19050, 6800 & 14050, 3600 & 7800, 1600 & 3800, 500 & 600**

HIGHEST 94.6 SVM Quadratic bagOfFeatures features [10 20 50 100 150 200 250 300] encoding images [10 20 50 75 100 150 200 250 300 350 400]

- **5 digits –**

HIGHEST 97.6% SVM Quadratic bagOfFeatures features [10 20 50 100 150 200 250 300] encoding images [10 20 50 75 100 150 200 250 300 350 400]

- **2 digits -**

HIGHEST 98.5% Logistic Regression bagOfFeatures features [10 20 50 100 150 200 250 300] encoding images [10 20 50 75 100 150 200 250 300 350 400]

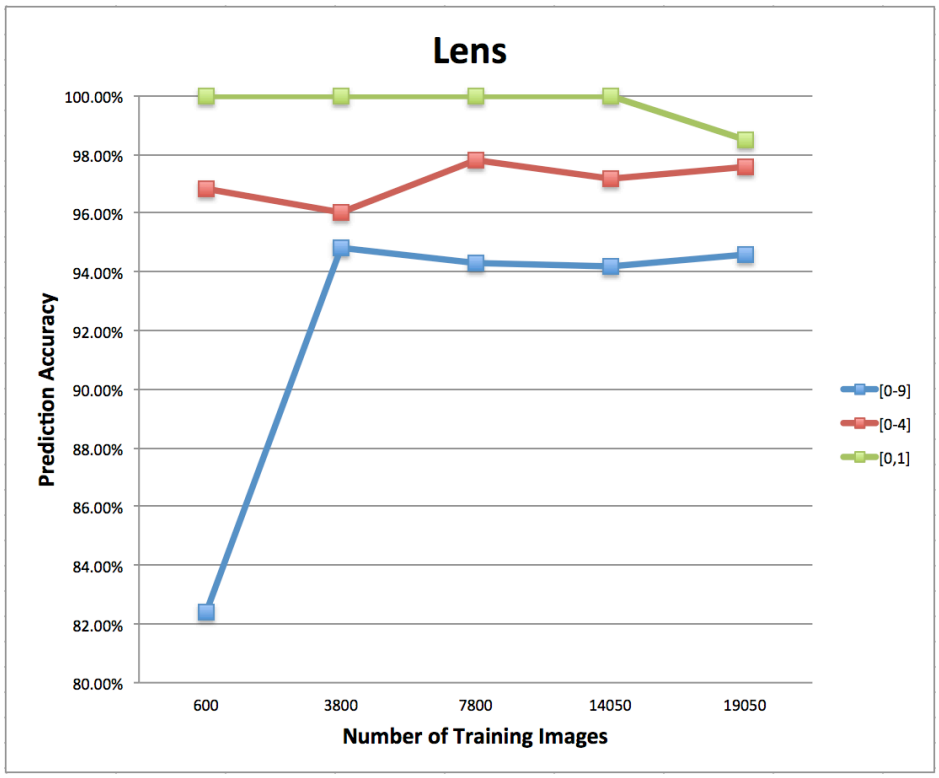


Figure 12: This plot shows the overall success of a lens cases {0,1}, {0-4}, and {0-9} dataset .

This final lens graph shows the three cases we experimented on, their individual accuracies along the variance of # of training images, with a constant feature extraction number. Lensed camera image recognition is not the primary component of this project but used as a guideline for testing the project's progression.

LENSELESS

- **10 digits - bagOfFeatures, encoding images: 10800 & 19050, 6800 & 14050, 3600 & 7800, 1600 & 3800, 500 & 600**

HIGHEST 57.0% SVM Quadratic bagOfFeatures features [10 20 50 100 150 200 250 300] encoding images [10 20 50 75 100 150 200 250 300 350 400]

- **5 digits -**

HIGHEST 80.6% SVM Linear bagOfFeatures features [10 20 50 100 150 200 250 300] encoding images [10 20 50 75 100 150 200 250 300 350 400]

- **2 digits -**

HIGHEST 96% SVM Linear bagOfFeatures features [10 20 50 100 150 200 250 300] encoding images [10 20 50 75 100 150 200 250 300 350 400]

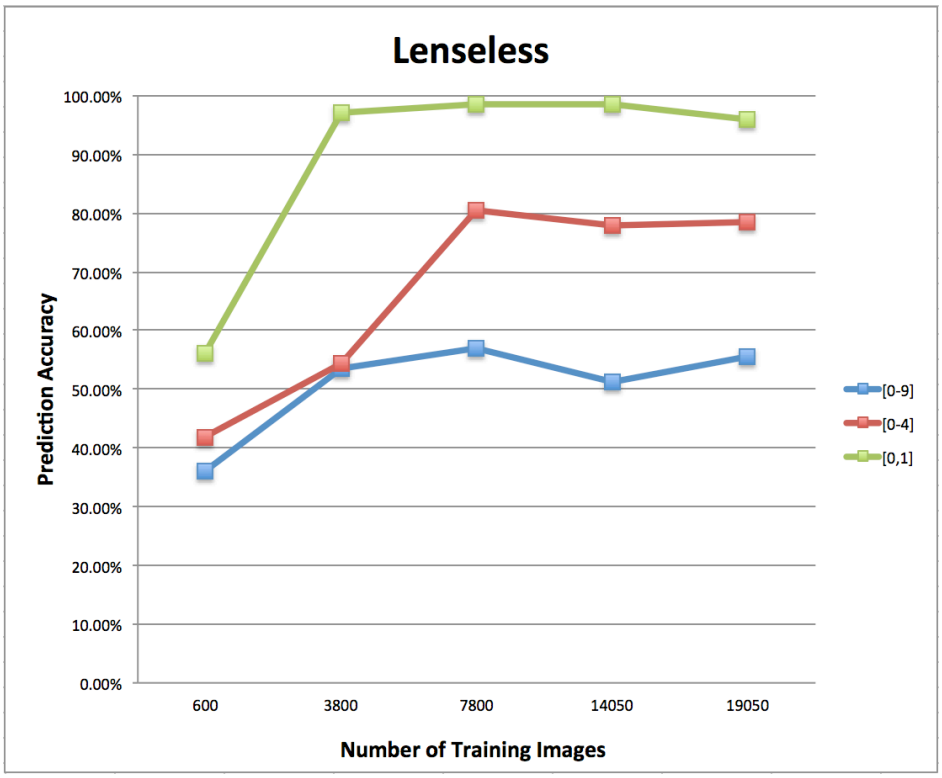


Figure 13: This plot shows the overall prediction accuracy of a lensless dataset in all three cases {0,1}, {0-4}, and {0-9}.

This final lensless graph shows the three cases we experimented on, their individual accuracies along the variance of # of training images, with a constant feature extraction number. This was our final ML application. To generate higher prediction accuracies and appeal to the Optica journal the project stemmed into deep learning, modifying the distance, and tilting the screen. These experiments have been established and are being worked on currently. Applying the ML model established for the handwritten numbers can be relayed for images scattered through a angle tilt or of a glass projection. These results depict that the ML system created can depict images correctly 57% of the time if the images. These are impressive readings for a process that has never been implemented previously.

3.2 Key Frustrations

Convolutional networks have traditionally been used on raw images without any preprocessing. Without the deep learning preprocessing, the resulting convolutional networks are larger, more

computationally intensive, and have not performed as well in our experiments. The frustration comes from a lack of time. The concept of deep memorization comes into play when images are too large. The images our system captured from our experiment caused the implementation to memorize the handwritten numbers and did no learning. The implementation seemingly ran perfectly and gave results in the high 90's. Upon speaking to experts in the field and analyzing the timing it was concluded that the system was indeed only memorizing. No learning was taking place in the handwritten dataset networks. This was seen in the drastic reduction in computing time to train and analyze images. Originally training and testing data took several hours to complete for the largest case {0-9}. In memorization the system only took minutes. What does it mean to 'memorize' a training set? In the context of learning, memorization means a failure to generalize. This brings us to another definition of memorization: not learning patterns from data. Phrases like "brute-force memorization" [34] connote fitting a dataset without capitalizing on any patterns in the data. In contrast, we believe that DNNs first learn and then refine simple patterns, which are shared across examples, in order to quickly drive down training loss.

4 DISCUSSION

4.1 Future Direction

With my last semester at the University of Utah coming to a close the duration of this project must be handed off. The direction of this project has limitless possibilities and its application beyond handwritten numbers is yet to be seen. To generate higher prediction accuracies and appeal to the Optica journal the project stemmed into deep learning, modifying the distance, and tilting the screen. I predict that accuracies could improve to about {0,1} at ~99%, {0-4} at ~90%, and {0-9} at ~85%% with a successful deep learning implementation. This would require recapturing the data images with a different CMOS sensor in order to produce smaller images. This would negate the possible affects of deep memorization. Deep memorization has its purpose but considering that our system would never encounter the same image twice the memorization

occurring would not help our system at all. When testing the distance and screen tilt I believe that the project will have difficulty calibrating images while continuing to extract features at a high success rate. This screen tilt experimentation is partially occurring with scattered handwritten images being projected through glass. The image is scattered and tilted at a 90-degree angle. Simply switching the database image sets in the programming would allow the machine system already implemented to run on the new data. Varying the distances variable was not attempted at all in our iteration of our experiment. Our approach was based off the best resolution and calibration practice for our images. Once the lens data image is as anthropomorphic as possible we took the CMOS sensor machine image. This phase of the project has not ended. Its completion is ongoing and Dr. Rajesh Menon will continue to help work on the implementation of deep learning and tilting image ratios atop the basis of machine learning algorithms, CNN training, and testing processes already established.

4.2 Summary and Conclusion

In conclusion, we demonstrate that machine learning can be effectively applied to classify lensless images. Such non-human image-based decision-making could lead to significant improvements in the ability of autonomous agents to navigate and make sense of the external world. Further testing and experimentation is definitely required to pus the boundaries of this project. With future work being done in modifying the CMOS distances and/or tilting the LCD screen interesting results will come about. The process is no longer the experimentation. The theory has been applied and proved in real life cases. Only furthering these experiments with CMOS distance changes and screen tilt will allow for extremely real life scenarios. With these enhancements made I'm confident that the Optica journal will reconsider our application for peer review. This document will serve as Stefan Kapetanovic's Honors Thesis at the University of Utah and will reflect upon the process and result of his project guided by Dr. Rajesh Menon. Our general process is shown in Figure 14.

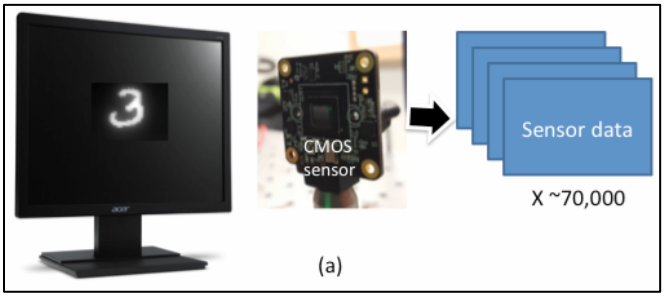


Figure 14: Overview schematic of our experiment. Handwritten digits are displayed on an LCD, and a bare CMOS sensor captures lensless images. Approximately 70,000 training images were created.

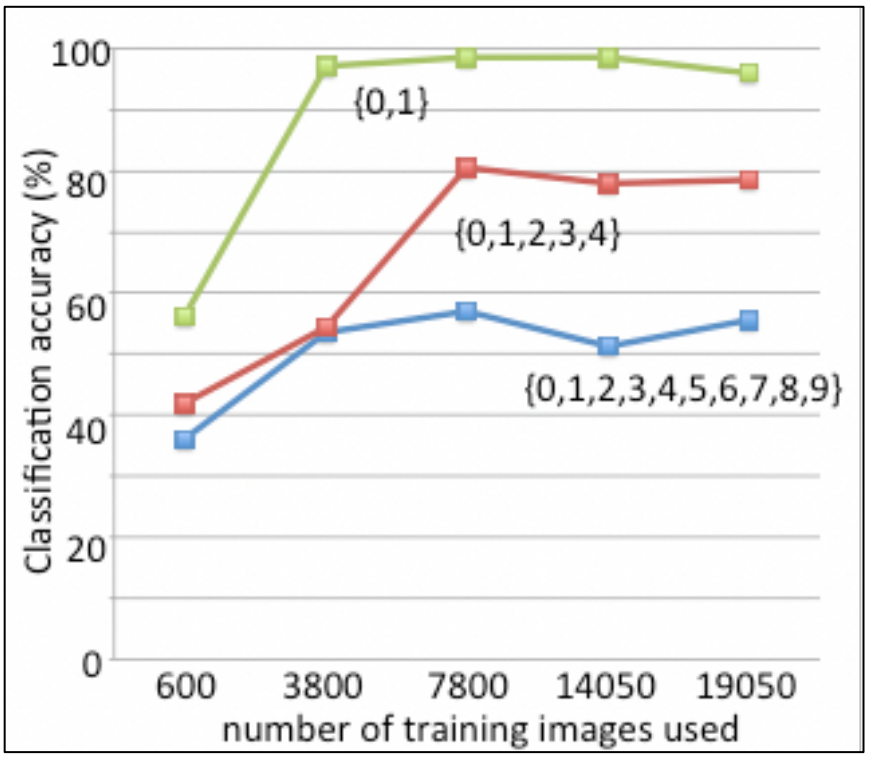


Figure 15: Best classification accuracy as a function of number of training images used.

We implemented a system that can accurately predict handwritten numbers from a raw CMOS machine image. Current forms of deep learning do not target non-anthropomorphic camera images that our sensors will be producing. Our image sizes held back the project when the deep learning stage occurred. The numerous layers within our application would use the extensive data to deep memorize rather than deep learn. The concept of the project is that an implemented machine system can interpret our CMOS camera data thus having the ability to make out what it

captures. To develop the system creating a dataset, generate a trained convolutional neural network (CNN), and test our system on live images becomes the experimentation process. Our results included {0,1} at 99% accuracy, {0-4} at 80.6% accuracy, and {0-9} at 57.0% prediction accuracies. In theory, from this we will have composed a system that can function on "non-human cameras as the eyes for the internet of things" for every machine system with an image-capturing sensor. Our major findings and system results for the lensless case are given in the graph below:

5 APPENDICES

5.1 MATLAB Configurations

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

Deep Learning Classifcation: https://www.mathworks.com/help/vision/examples/image-category-classification-using-deep-learning.html?requestedDomain=www.mathworks.com

Deep Learning: https://www.mathworks.com/help/vision/deep-learning.html

Pattern Recognition: https://www.mathworks.com/help/nnet/pattern-recognition-and-classification.html

BagofFeatures algorithm: https://www.mathworks.com/help/vision/ref/bagoffeatures-class.html

Classification Ensembles: https://www.mathworks.com/help/stats/classification-ensembles.html

5.2. MNIST Dataset

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. It is a good database for

people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

Main MNIST Webpage: http://yann.lecun.com/exdb/mnist/

Four files are available on this site:

- train-images-idx3-ubyte.gz: training set images (9912422 bytes)
- train-labels-idx1-ubyte.gz: training set labels (28881 bytes)
- t10k-images-idx3-ubyte.gz: test set images (1648877 bytes)
- t10k-labels-idx1-ubyte.gz: test set labels (4542 bytes)

6 ACKNOWLEDMENTS

Many thanks go to Rajesh Menon for his support, help, advice, and effort in helping me throughout the process of this Honors thesis.

My thanks also go to my Rachael Palmer who was always willing to make time for me in her busy schedule and work on capturing me more data if need be and Ganghun Kim who worked patiently with me to program the machine learning skeleton and MATLAB analysis tools.

Thanks to Dr. Neil Cotter, Tolga Taszdien, and Mehran Javanmardi for checking my thesis in my times of need with deep learning.

Thanks to Kyle Isaacson for all his previous work in the lab as well.

Funding. National Science Foundation (Grant # 10037833). Utah Science Technology and Research Initiative. University of Utah's Undergraduate Research Opportunities Program (UROP).

Thanks to the University of Utah Electrical and Computer Engineering Department

7 REFERENCES

[1] O. Chapelle, P. Haffner and V. N. Vapnik, "Support vector machines for histogram-based image classification," IEEE Trans. Neural Netw. 10, 1055- 1064 (1999).

[2] D. Ciregan, U. Meier and J. Schmidhuber, "Multi-column deep neural networks for image classification," in 2012 IEEE Conference on Computer Vision and Pattern Recognition (IEEE, 2012), pp. 3642-3649.

[3] O. Boiman, E. Shechtman and M. Irani, "In defense of Nearest-Neighbor based image classification," in 2008 IEEE Conference on Computer Vision and Pattern Recognition (IEEE, 2008), pp. 1-8.

[4] I. Laptev, M. Marszalek, C. Schmid and B. Rozenfeld, "Learning realistic human actions from movies," in 2008 IEEE Conference on Computer Vision and Pattern Recognition (IEEE, 2008), pp. 1-8.

[5] G. Kim, K. Isaacson, R. Palmer, and R. Menon, "Lensless photography with only an image sensor," Appl. Opt. 56, 6450-6456 (2017)

[6] M. S. Asif, A. Ayremlou, A. Sankaranarayanan, A. Veeraraghavan, R. Baraniuk, "FlatCam: Thin, Bare-Sensor Cameras using Coded Aperture and Computation", https://arxiv.org/abs/1509.00116

[7] P. R. Gill and D. G. Stork, "Lensless ultra miniature imagers using odd- symmetry phase gratings," in Proceedings of Computational Optical Sensing and Imaging, (Optical Society of America, 2013), paper CW4C.3

[8] Y. LeCun, C. Cortes, C. Burges, "The MNIST Database of Handwritten Digits", http://yann.lecun.com/exdb/mnist/

[9] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded Up Robust Features," in Computer vision-ECCV. 2006, 404-417 (2006).

[10] C. W. Chen, J. Luo and K. J. Parker, "Image segmentation via adaptive K- mean clustering and knowledge-based morphological operations with biomedical applications," IEEE Trans. Image Process. 7(12), 1673-1683 (1998).

[11] Mathwork, "Statistics and Machine Learning Toosbox", https://www.mathworks.com/products/statistics.html

[12] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," IEEE Trans. Syst., Man, and Cybern. 21(3), 660-674 (1991).

[13] Y. Bengio, Y. Le Cun, and D. Henderson, "Globally trained handwritten word recognizer using spatial representation, space displacement neural networks, and hidden Markov models," in Advances in Neutral Information Processing Systems 6. San Mateo, CA: Morgan Kaufmann, 1994.

[14] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. Jackel, Y. Le Cun, U. Muller, E. Sackinger, P. Simard, and V. N. Vapnik, "Comparison of classifier methods: A case study in handwritten digit recognition," in Proc. Int. Conf. Pattern Recognition. Los Alamitos, CA: IEEE Comput. Soc. Press, 1994.

[15] Y. Le Cun, “Generalization and network design strategies,” Dep. Comput. Sci., Univ. Toronto, Canada, Tech. Rep. CRG-TR-89-4, 1989.

[16] Y. Le Cun and Y. Bengio, “Convolutional networks for images, speech, and time series,” in The Handbook of Brain Theory and Neural Networks, M. A. Arbib, Ed. Cambridge, MA: MIT Press, 1995, pp. 255–258.

[17] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, “Handwritten digit recognition with a back propagation neural network,” in Advances in Neural Information Processing Systems 2, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990, pp. 396–404.

[18] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifier. InProc. 5th ACM Workshop on Computational Learning Theory, pages 144{152, Pittsburgh, PA, July 1992.

[19] C. Cortes and V. Vapnik. Support vector networks. Machine Learning, 20:1{25, 1995.

[20] V. Vapnik.The Nature of Statistical Learning Theory. Springer, New York, 1995.

[21] Mathwork, “Train support vector machine classifier”, https://www.mathworks.com/help/stats/svmtrain.html

[22] Friedman, Jerome, et al. “Additive Logistic Regression: a Statistical View of Boosting (With Discussion and a Rejoinder by the Authors).” The Annals of Statistics, vol. 28, no. 2, 2000, pp. 337–407., doi:10.1214/aos/1016120463.

[23] Keller, James M., et al. “A Fuzzy K-Nearest Neighbor Algorithm.” IEEE Transactions on Systems, Man, and Cybernetics, SMC-15, no. 4, 1985, pp. 580–585., doi:10.1109/tsmc.1985.6313426.

[24] Osuna, E., et al. “An Improved Training Algorithm for Support Vector Machines.” Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop, doi:10.1109/nnsp.1997.622408.

[25] Lawrence, S., et al. “Face Recognition: a Convolutional Neural-Network Approach.” IEEE Transactions on Neural Networks, vol. 8, no. 1, 1997, pp. 98–113., doi:10.1109/72.554195.

[26] Zen, Heiga, and Hasim Sak. “Unidirectional Long Short-Term Memory Recurrent Neural Network with Recurrent Output Layer for Low-Latency Speech Synthesis.” 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, doi:10.1109/icassp.2015.7178816.

[27] Mikolov, Tomas, and Geoffrey Zweig. “Context Dependent Recurrent Neural Network Language Model.” 2012 IEEE Spoken Language Technology Workshop (SLT), 2012, doi:10.1109/slt.2012.6424228.

[28] Bowman, Samuel R., et al. “Recursive Neural Networks Can Learn Logical Semantics.”Proceedings of the 3rd Workshop on Continuous Vector Space Models and Their Compositionality, 2015, doi:10.18653/v1/w15-4002.

[29] Lehtokangas, Mikko, et al. “Neural Network Modeling and Prediction of Multivariate Time Series Using Predictive MDL Principle.” Icann â□™93, 1993, pp. 826–829., doi:10.1007/978-1-4471-2063-6_233.

[30] Wang, Sun-Chong. Interdisciplinary Computing in Java Programming Language. Kluwer Academic, 2003, pg. 81-100.

[31] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," IEEE Trans. Inform. Theory, vol. IT-13, pp. 21-27, Jan. 1967.

[32] R. Brunelli and T. Poggio, “Face recognition: Features versus templates,” IEEE Trans. Pattern Anal. Machine Intell., vol. 15, pp. 1042–1052, Oct. 1993.

[33] I. J. Cox, J. Ghosn, and P. N. Yianilos, “Feature-based face recognition using mixture-distance,” in Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE Press, 1996.

[34] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. International Conference on Learning Representations (ICLR), 2017.

Name of Candidate:	Stefan Kapetanovic
Birth date:	May 13, 1995
Birthplace:	Sarajevo, Bosnia & Herzegovina
Address:	56 E Resaca Dr. #A5, Sandy, Utah, 84070