THE APPLICATION OF COMPUTER VISION, MACHINE AND DEEP LEARNING ALGORITHMS UTILIZING MATLAB

Andrea Linda Murphy
University of New Hampshire, Durham

THE APPLICATION OF COMPUTER VISION, MACHINE AND DEEP LEARNING ALGORITHMS UTILIZING MATLAB®

BY

ANDREA LINDA MURPHY

Bachelor of Science, Marketing, Virginia Commonwealth University, 2014

THESIS

Submitted to the University of New Hampshire

in Partial Fulfillment of

the Requirements for the Degree of

Master of Science

in

Information Technology

May 2020

This thesis was examined and approved in partial fulfillment of the requirements for the degree of Master of Science in Information Technology by:

Thesis Director, Dr. Mihaela Sabin, PROFESSOR,
Applied Engineering & Sciences

Karen Jin, ASSISTANT PROFESSOR,
Applied Engineering & Sciences

Jeremiah Johnson, ASSISTANT PROFESSOR,
Applied Engineering & Sciences

On April 29, 2020

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	NAME
AI	Artificial Intelligence
ANN	Artificial Neural Network
ASL	American Sign Language
BOVW	Bag-of-visual-words
BOW	Bag-of-Words
CAD	Computer-aided design
ConvNet/ CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
IDE	Integrated Development Environment
NN	Neural Network
ML	Machine Learning
SURF	Speeded Up Robust Features
SVM	Support-vector-machines
TL	Transfer learning

ABSTRACT

THE APPLICATION OF COMPUTER VISION, MACHINE AND DEEP LEARNING ALGORITHMS UTILIZING MATLAB®

by

Andrea L. Murphy

University of New Hampshire, Spring 2020

MATLAB® is a multi-paradigm proprietary programming language and numerical computing environment developed by MathWorks. Within MATLAB® Integrated Development Environment (IDE) you can perform Computer-aided design (CAD), different matrix manipulations, plotting of functions and data, implementation algorithms, creation of user interfaces, and has the ability to interface with programs written in other languages[1]. Since, its launch in 1984 MATLAB® software has not particularly been associated within the field of data science. In 2013, that changed with the launch of their new data science concentrated toolboxes that included Deep Learning™, Image Processing™, Computer Vision™, and then a year later Statistics and Machine Learning™.

The main objective of my thesis was to research and explore the field of data science. More specifically pertaining to the development of an object recognition application that could be built entirely using MATLAB® IDE and have a positive social impact on the deaf community. And in doing so, answering the question, could MATLAB® be utilized for development of this type of application? To simultaneously answer this question while addressing my main

objectives, I constructed two different object recognition protocols utilizing MATLAB_R2019 with the add-on data science tool packages. I named the protocols ASLtranslate (I) and (II). This allowed me to experiment with all of MATLAB® data science toolboxes while learning the differences, benefits, and disadvantages of using multiple approaches to the same problem.

The methods and approaches for the design of both versions was very similar. ASLtranslate takes in 2D image of American Sign Language (ASL) hand gestures as an input, classifies the image and then outputs its corresponding alphabet character. ASLtranslate (I) was an implementation of image category classification using machine learning methods. ASLtranslate (II) was implemented by using a deep learning method called transfer learning, done by fine-tuning a pre-trained convolutional neural network (CNN), AlexNet, to perform classification on a new collection of images.

I. CHAPTER: INTRODUCTION

Computer vision, machine learning and deep learning are closely related disciplines in the field in data science, especially when considering the application of object recognition. Computer vision is an interdisciplinary field that trains computers to interpret and understand the visual world. It seeks to automate tasks that a human visual system can do. Object recognition is a computer vision technique for identifying objects in images or videos and is the key output of deep learning and machine learning algorithms. Both techniques can be successful in solving object recognition problems, with similar approaches but differ in their execution. Machine learning algorithms require the features to be defined or extracted first before being classified. While using deep learning methods you do not need to specifically define the features in order to recognize objects, instead you rely on the use of convolutional neural networks (CNN).

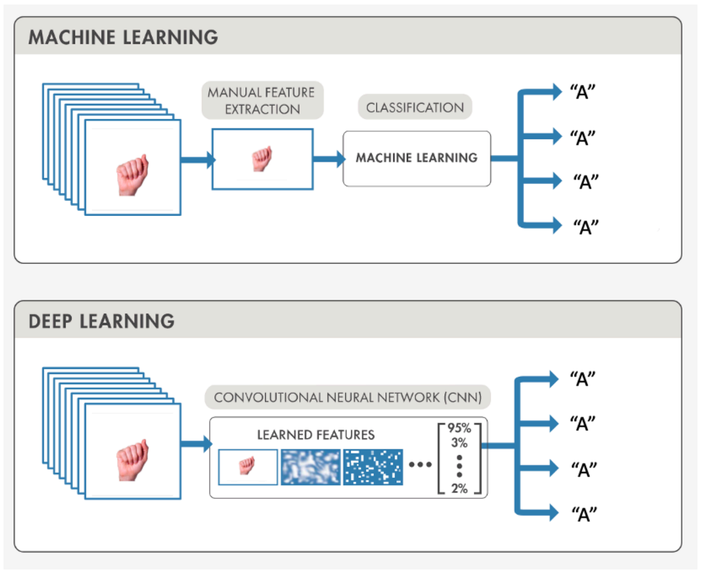


Figure 1: *Machine learning and deep learning techniques used for object recognition*

I.I Object Recognition

Object recognition is the process of identifying an object as a specific entity within an image or video. Object recognition is a crucial output of computer vision, machine learning, and deep learning algorithms. The technology that is behind many of the applications that we use within our everyday life. The basic goal of object recognition is being able to teach a computer to do what comes naturally to humans, to gain an understanding of what the image or video contains. Here are a few applications that utilize object recognition technology in their core functionality:

- **Self-driving cars:**
  - Tesla’s Autopilot, Google’s Waymo
- **Medical image processing and diagnostic tools:**
  - Google AI for early breast cancer detection
- **Robot-assisted surgery:**
- **Face recognition:**
  - Blippar
- **Biometric identification:**
  - FaceSDK
- **Google object detection applications:**
  - Google lens and translate
- **Surveillance and Security:**
  - Activity recognition

Object recognition involves identifying, recognizing, and locating objects within images with a degree of confidence or accuracy. To ensure the successfully identification the process can be broken down into the following tasks:

1. **Image Classification:** What is contained in the image? Image classification takes in an image as the input and outputs the classification of that image with some pre-defined metrics (confidence, accuracy, loss, probability, etc.).
2. **Object Localization:** Where is the object in the image? Object localization is the process of locating the main (or most prominent) object within an image.
3. **Image Tagging:** A visual tag on the image that identify what the object is.

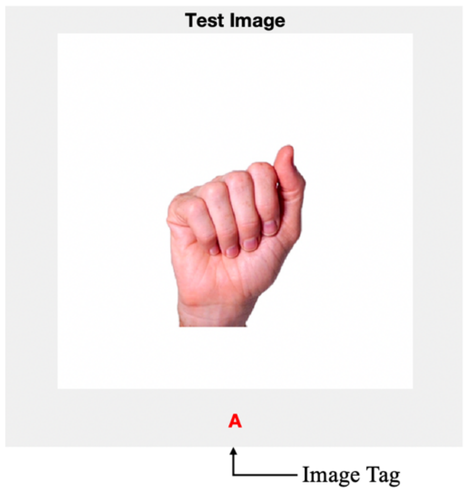


Figure 2: *Example of an image tag*

4. **Object Detection:** What is the object in the image, and where is the object in the image? Object detection algorithms act as a combination of both image classification and an object localization algorithm.

I.II Machine Learning

Machine Learning is the science (and art) of programming computers so they can learn from data

Here is a slightly more general definition:

[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed. —Arthur Samuel, 1959

And a more engineering-oriented one:

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

—Tom Mitchell, 1997

[Geron Aurélien 2017]

Machine learning (ML) refers to a systems ability to acquire and integrate knowledge through observation and improve on tasks learned from patterns and inference instead of having to be given explicit instructions. ML is a scientific study of algorithms and statistical models that computer systems use to perform a variety of tasks and solve a range of problems, including techniques used for object recognition.

I.II.I Machine Learning Classification Workflow

There are several machine learning best practices for solving classification problems. Along with server different overviews of the ML classifications workflows. I based my ML classification workflow overview on the methods that I acquired from MATLAB® Machine Learning Onramp course [1]. Which will be further discussed over the next few pages.

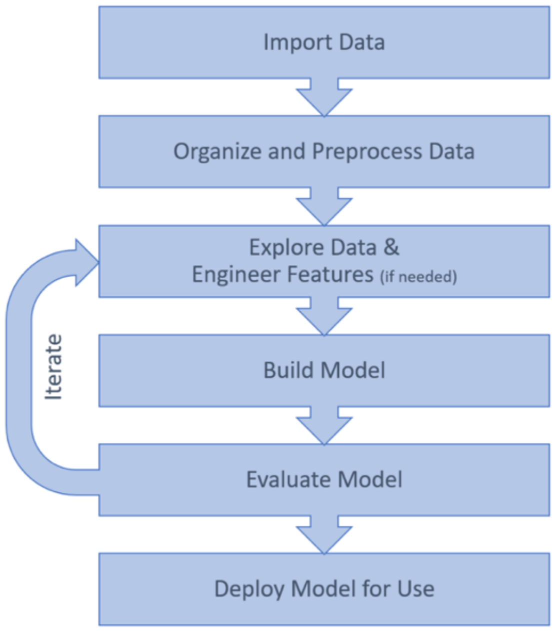

Import Data
Organize and Preprocess Data
Explore Data & Engineer Features (if needed)
Build Model
Evaluate Model
Deploy Model for Use
Iterate

Figure 3: *Overview of the ML Classification Workflow [1]*

Import Data, organize, pre-process data

Regardless of what type of model, machine learning system, simple or complex the first step in the classification workflow process will be to import data. This data is usually allocated into three different sets or datasets:

1. **Training Dataset:** A sample of data that is used to fit and train the model. This is the actual data that is used to train our model, this is what the model "learns from".
2. **Validation Dataset:** A sample of data used to provide an unbiased evaluation of a model that is fit on the training dataset while tuning model hyperparameters. The model will use this data indirectly to adjust the hyperparameters, but never directly to "learn from".
3. **Test Dataset:** A sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. Test data is the untouched dataset that is used to evaluate the model.

The dataset split ratio is the amount of data that is designated to each training, validation, and test datasets. The ratio depends on a few determining factors (total number of samples in your data, model you are using, etc.). A very commonly used ratio is an 80:20 split, where 80% is allocated for training/validation and 20% for testing.

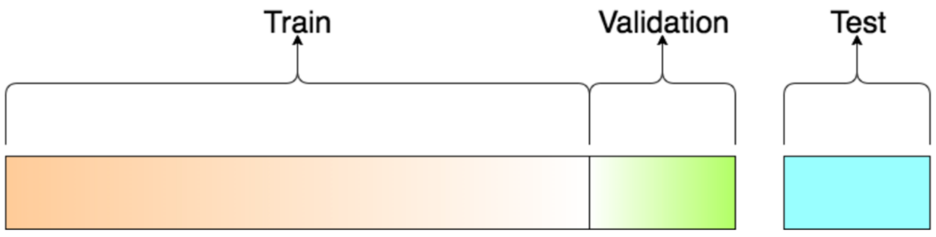


Figure 4: *Dataset Split Ratio 80:20 [2]*

The Model

Machine learning systems are generally split into four main categories: supervised, semi-supervised, unsupervised, and reinforcement learning. These categories are classified based on the amount of human interaction that the systems receive during training. Classification and regression are typical supervised learning tasks. Supervised learning provides a powerful tool to classify and process data.

Classification vs Regression models:

1. **Classification model:** Is image a dog? A classification model predicts discrete values. It partitions the space of predictor variables into regions. Each region is assigned one of the output classes.
2. **Regression model:** What is the probability a user will click on this add? A regression model predicts continuous values.

Training data that is fed into classification models that includes the desired solutions, are called labels. A label is the what we're predicting or the y-variable, and a feature is the input or the x-variable. The model is what defines the relationship between these features and labels. Having built a model from the data, you can use it to classify new observations. This just requires calculating the features of the new observations and determining which region of the predictor space they are in.

Building the Model:

There are serval types of supervised learning algorithms used in building classification models. Listed below are a few of the more popular supervised learning algorithms:

1. **K-Means Clustering:** Is a method of vector quantization, where its goal is to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. The goal to minimize the distance between each point in the scatter cloud and the assigned centroids. For each cluster centroid, there exists a group of points around it, known as the center.

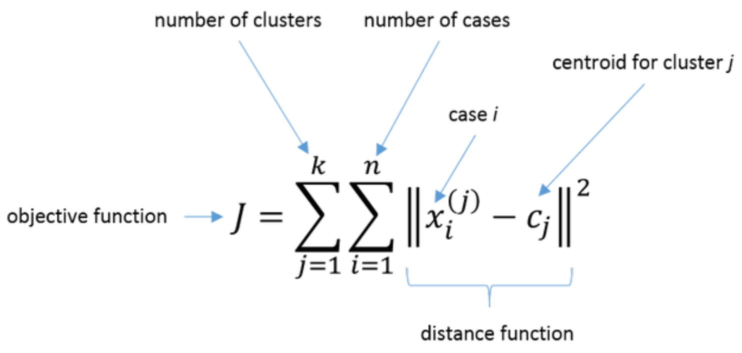


Figure 5*: K-means clustering equation [9]*

2. **Linear Regression:** A supervised machine learning algorithm where the predicted output is continuous and has a constant slope.

   Linear Regression equation: $y^1 = b + w_1x_1$

   where:

   `y1` is the predicted label (the desired output) `b` is the bias (the y-intercept), sometimes referred to as w0.

`w1` is the weight of feature 1

**Weight is the same concept as the "slope" m in the traditional equation of a line.*

`x1` is a feature (*a known input*)

3. **Logistic Regression:** A classification algorithm used to assign observations to a discrete set of classes. It transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.
4. **Support Vector Machines (SVMs)**
    a. Binary support vector machine (SVM)

Evaluating the Model:

To quantify the process on any model a performance measures needs to be defined and calculated. The model's predictions tell us how well the model performs on new data. Accuracy is one metric for evaluating classification models, this is achieved by calculating the proportion of correct predictions by dividing the number of correct predictions by the total number of predictions.

Another commonly used metric to evaluate a model is misclassification rate. The misclassification rate is a calculation of the proportion of incorrect predictions. Since classes are not always equally distributed in either the training or test data. Loss has been determined as less bias measure of misclassification because it incorporates the probability of each class in the calculation that is based on the distribution in the data.

Accuracy and misclassification metrics only give a single value for the overall performance of the model. To see a more detailed breakdown a confusion matrix can be used as well. A confusion matrix will show the number of observations for each combination of true and predicted class. All four MATLAB® functions can be visualized in the table below:

Accuracy Function	`accuracy = sum(iscorrect)/numel(predictions)`

Misclassification Rate function	`notcorrrect = (predictions ~= testdata.Character);`
	
	 `misclassrate = sum(notcorrrect)/numel(predictions);`

Confusion Matrix	`confusionchart(testdata.Character,predictions);`
Test Loss	`loss(model,testdata)`

Table 1: *Evaluating the Model*

I.III Deep Learning

Deep learning (DL) at its very basic level is a machine learning technique that teaches a computer to filter inputs through layers in order to learn by example in order to predict and classify information. Deep learning is inspired by the way that the human brain works and filters information. For a system to learn from or process data similar to a human, it needs to understand the data at an abstract level. A human mind can use abstract features of variation and other external factors such as viewing angle, smells etc. in order to classify the given information.

DL is a sub-branch of ML, meaning that it also has a set of learning algorithms that can train and learn on. More specifically DL is powered by neural networks. A neural network is computing system inspired by a mathematical model of a biological neuron. These networks "learn" to perform tasks by considering examples, generally without being programmed with task-specific rules.

Activation functions are mathematical equations that determine the output of a neural networks.

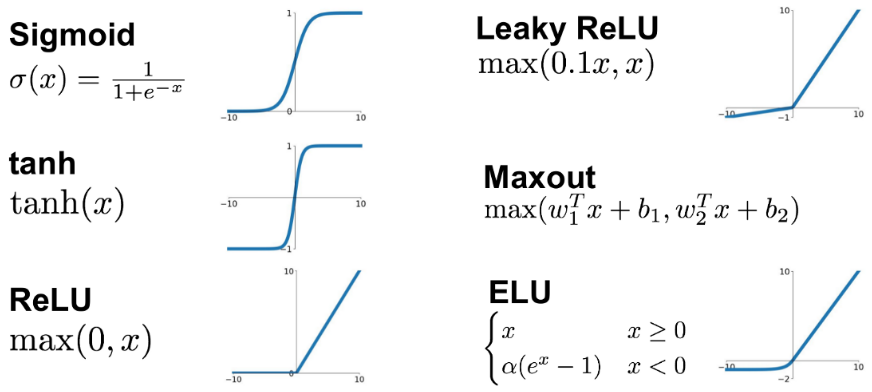


Figure 6: *Activation functions used in neural networks [10]*

There are many different types of neural networks implement today. Below there are some visual representation of important types of neural networks.

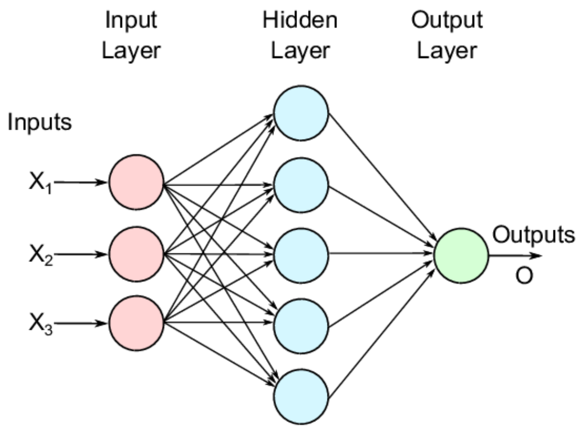


Figure 7: *A Feedforward Neural Network [14]*

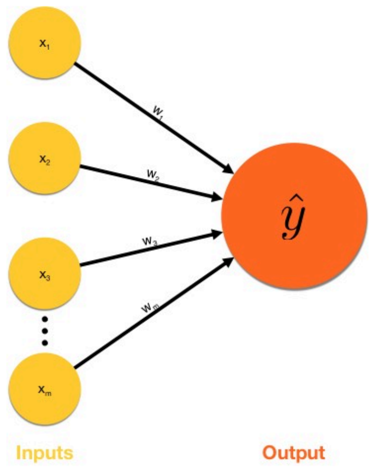


Figure 8: *Single-layer Perceptron [11]*

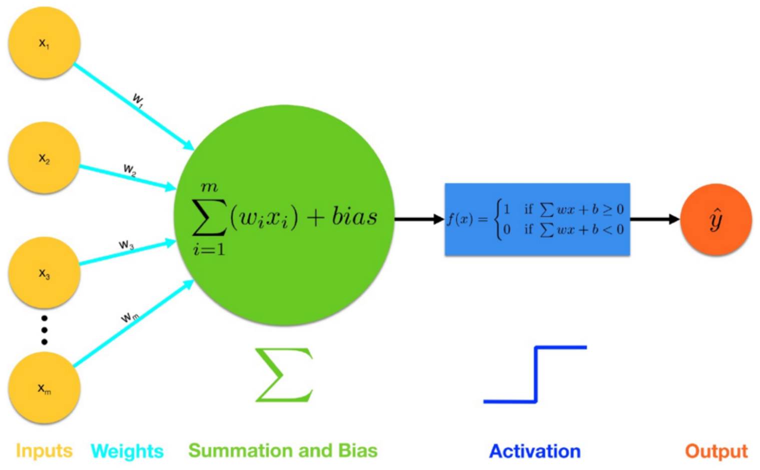

x_1
x_2
x_3
x_m
w_1
w_2
w_3
w_m
$\sum_{i=1}^{m}(w_i x_i) + bias$
$f(x) = \begin{cases} 1 & \text{if } \sum wx + b \geq 0 \\ 0 & \text{if } \sum wx + b < 0 \end{cases}$
$\hat{y}$
$\sum$
Inputs
Weights
Summation and Bias
Activation
Output

Figure 9: *Multi-layer Perceptron [11]*

1. Inputs are fed into the perceptron
2. Weights are multiplied to each input
3. Summation and then add bias
4. Activation function is applied
5. Output is either triggered

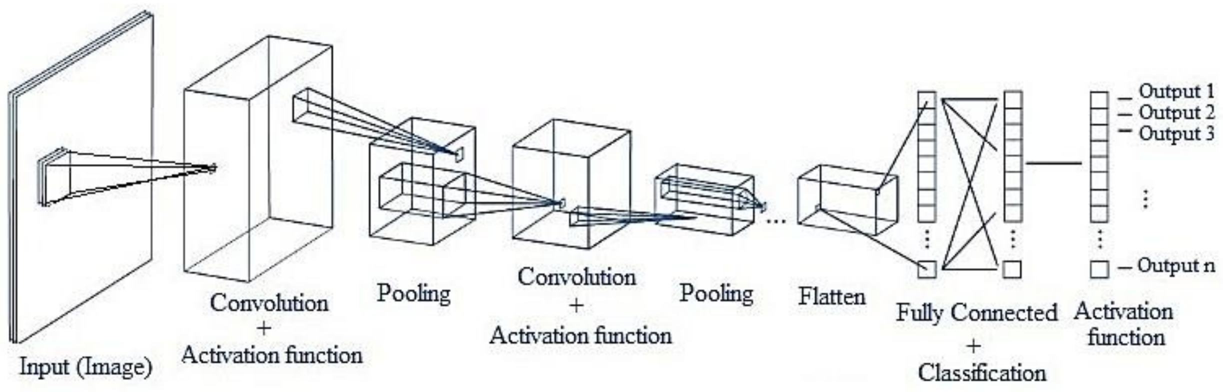

Input (Image)
Convolution + Activation function
Pooling
Convolution + Activation function
Pooling
Flatten
Fully Connected + Classification
Activation function
Output 1
Output 2
Output 3
Output n

Figure 10: *Convolutional Neural Network (CNN) [12]*

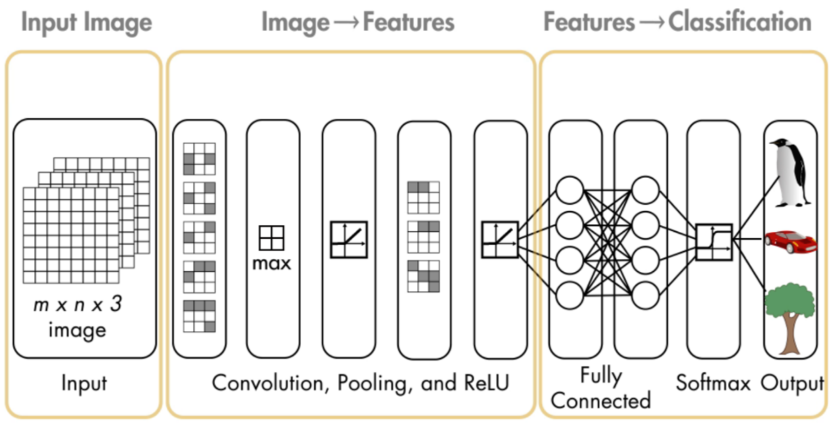


Figure 11: *A Pre-Trained CNN that can be used for Classification [13]*

II. CHAPTER: OBJECTIVES

American Sign Language (ASL) fundamentally facilitates communication in the deaf community. Over 5% of the world's population – or 466 million people – has disabling hearing loss. It is estimated that by 2050 over 900 million people – or one in every ten people – will have disabling hearing loss [2].

The main objective of my thesis was to research and discuss the field of data science, peculiarly pertaining to development of an object recognition application using MATLAB® IDE. Based on personal motivation to create an application that could have a positive social impact on the deaf community. Aiding in the ability of communication with the help of a simple portable application makes the research and development of ASLtranslate not only significant but also essential.

III. CHAPTER: CURRENT SYSTEMS

When I first started the development of ASLtranslate there were no other applications like ASLtranslate it on the market. Over the past few months more developers that have begun to emerge with similar applications. One being "deeplens-asl", an American Sign Language alphabet classifier. It was trained using transfer learning from squeezeNet with 18 layers. The data used for this training was collected using Amazon SageMaker. SageMaker took images one user at a time, using their input to capture and label the image. The data collected from this tool consisted of all alphabets but had to use special signs for letters 'j' and 'z' as the classifier could not train the model using their special characters. Due to this, the final model can also only detect these two letters if the user uses these special signs for these letters.

Deeplens-als's final model could classifies ASL alphabet gestures, but with only with 40% accuracy. The application uses amazon's device stream to get input. The input videos for this application needs to account for a couple of caveats. This includes having to go slower if there are words with repeating letters like "letter" for it to detect the two "tt" apart and also having to wait between words for the classifier to complete detecting the previous letter. If the pictures have an almost white background, with only the signs visible in the images, it can classify the gestures with higher level of accuracy.

IV. CHAPTER: HISTORY

IV.I Machine Learning, Computer Vision, Deep Learning

Major breakthroughs during the in the nineteenth century include the work of Thomas Bayes, which led Pierre-Simon Laplace to define Bayes' Theorem in 1812. Adrien-Marie Legendre also developed the Least Squares method for data fitting in 1805. Andrey Markov described analysis techniques later called Markov Chains in 1913. These techniques and many more are all fundamental to modern day machine learning.

In the late 1940s the development of stored-program computers began to advance. On June 21st, 1948, at Manchester University, shortly after 11 o'clock in the morning, the world's first stored-program electronic digital computer successfully executed its first program [3]. Stored-program computers hold their instructions or programs in the same memory used for data. Instructions could now be stored in memory and executed in sequence referencing the data values it needs on which to operate.

Alan Turing, was a British mathematician, widely known for the contribution he made to code-breaking efforts during the 2nd World War in 1947. Published a paper in 1950 entitled "Computing Machinery and Intelligence", in which he asks the still relevant question 'Can machines think?' [4]. The paper is based on a growing understanding of the overall power of computers. The paper represents one of the first attempts to describe how artificial intelligence (AI) could be developed. It famously discussed a test called the "Turing Test", an imitation game that could determine whether a computer was intelligent or not.

In 1951 Marvin Minsky and Dean Edmonds built a computer-based simulation based on the way organic brains works, and what would soon become known as the first artificial neural network SNARC. The Stochastic Neural Analog Reinforcement Computer (SNARC) learned from experience and was used to search a maze, like a rat in a psychology experiment. It was built along connectionist principles, representing a mind as a network of simple units within which intelligence may emerge.

Arthur Lee Samuel came up with the term Machine learning in 1952, and later popularized it. He developed a computer program for playing checkers in the 1950's. The program used a minimax strategy to compute the next move, which is the basis of minimax algorithm. To optimize the program by using ML techniques involving reward functions and move history, allowing the program to learn and become better.

In 1957, Dr. Frank Rosenblatt of Cornell Aeronautical Laboratory combined Hebbian theory of brain cell interactions and Arthur Samuel's Machine Learning research to create the first perceptron. The perceptron that was originally designed for the IMB 704, was installed in a custom-built machine that was called the "Mark 1 Perceptron" designed to process images. The perceptron seemed promising as the first successful neurocomputer, unfortunately it fell short and couldn't recognize many kinds of visual patterns. This caused a lot of frustration and many say stalling neural network and machine learning research.

In the 1960s, the breakthrough of multilayers networks that provided a new path for neural network research. It was discovered that providing and using two or more layers in a perceptron would significantly increase the processing power of the perceptron then just using one single layer. In 1965, Alexey Grigoryevich Ivakhnenko and Valentin Grigorʹevich Lapa, were amongst the earliest to start developing these types of Deep Learning algorithms. They used models with polynomial activation functions, that were analyzed statistically. From each layer, the best statistically chosen feature could be selected then provided as input to the next layer, all done manually.

During the 1970's the first AI winter kicked in, the result of promises that couldn't be kept. During the 1950's and 1960s there were enormous enthusiasm for AI research, but people became disillusioned when enormous breakthroughs didn't happen. By the 1970 the failure of machine translation and overselling AI's capabilities led to reduced funding. Along with the publishing of the 1973 Lighthill report, which stated that AI's most prominent algorithms of the time would not work against real world problems.

Even with the lack of funding, some researchers continued the work, which resulted in the development of CNN. In, 1979 Kunihiko Fukushima first developed Neocognitron, an artificial neural network (ANN) that inspired the development of CNN's. He designed the network using multiple pooling layers and convolutional layers. This design allowed the system to "learn" and recognize visual patterns. Neocognitron was the first NN that was specifically built for solving CV problems. Many of its concepts are still in use to this day, such as its selective attention model, which simulates what humans do while multitasking.

In 1995, Corinna Cortes and Vladimir Vapnik published a paper entitled "Support-Vector Networks", in which they introduced a type of learning machine named the support-vector-network or support-vector-machines (SVMs). SVMs quickly became a widely used algorithm because it could be used with simple models with features specific to a task, making it both cost effective and easier to understand comparatively.

In 2012, Alex Krizhevsky released AlexNet which was a version of a LeNet5 with a deeper and much wider architecture. AlexNet, a deep Convolutional Neural Network designed to use of rectified linear units (ReLU) as non-linearities, and a dropout technique to selectively ignore single neurons during training as a way to avoid overfitting of the model, and overlapping max pooling, avoiding the averaging effects of average pooling.

IV.II MATLAB®

The first version of MATLAB® was developed by Cleve Moler as a hobby project. This version of MATLAB® was a simple interactive matrix calculator based on research papers published between 1965 and 1970 by J.H. Wilkinson and his colleagues. Between 1979 and 1980, he was teaching a graduate course on Numerical Analysis at Stanford. In this course, he introduced his students to MATLAB®, which became an instant hit. One of his student's friends, Jack Little, adopted MATLAB® for his work and realized that MATLAB® could have commercial value.

In 1983, Jack Little proposed the creation of a commercial MATLAB®. Then, Jack Little and Steve Bangert worked on MATLAB® and added many things including functions, graphics and toolboxes. First named, PC-MATLAB, a commercial version of MATLAB® was debuted in December 1984 at an IEEE conference. Then a year later, Pro-MATLAB was debuted for Unix systems. They were also the core component of Simulink®, a MATLAB® companion for simulation and model-based design. MATLAB® introduced sparse matrices and cell arrays in 1992 and 1996. Then 2000, MATLAB® desktop was released.

Since then, MATLAB® started evolving steadily. MATLAB® originally had only one datatype, which was the IEEE Standard 754 double-precision floating point. In the early 2000s, MATLAB® started introducing more datatypes. By 2007, they had introduced single precision, three unsigned datatypes, three signed datatypes, and a logical datatype.

IV.II.I MATLAB® Data Science Toolbox Release Dates

As of 2020, MATLAB® has 63 different add-on toolboxes, that includes the data science focused packages, object-oriented programming capabilities, and parallel computing toolboxes. For the research and development of this thesis I used the following MATHLAB® toolboxes, Image Processing Toolbox™, Statistics and Machine Learning Toolbox™, Computer Vision Toolbox™, Deep Learning Toolbox™, and Parallel Computing Toolbox™. The first year MathWorks released those Toolboxes for MATLAB® can be viewed in the table below.

Toolbox	Release Date
Parallel Computing Toolbox™	2011
Image Processing Toolbox™	2013
Computer Vision Toolbox™	2013
Deep Learning Toolbox™	2013
Statistics and Machine Learning Toolbox™	2014

Table 2: *MATLAB® Data Science Toolbox Release Dates*

V. CHAPTER: DESIGN OF ASLtranslate

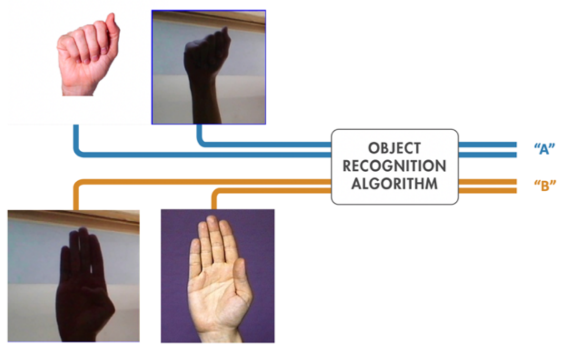


Figure 11: *Designing an object recognition algorithm to identify ASL hand gesture*

V.I Methodology and Approach

In computer science, there is a phrase that is frequently used “garbage in, garbage out”. This phrase describes a simple yet an important concept you must consider during the constructing of any ML model. If your input data is flawed, or garbage, chances are your output is going to be garbage too [5]. The quality of the data used plays a significant part on the accuracy of your model. For this reason, I spent a considerable amount of time selecting the right input image data to use. I based this decision on two key factors, quantity and quality.

After much consideration, the image dataset I used for the implementation of ASLtranslate was found on Kaggle. Originally, the dataset contained a collection of images of the ASL alphabets, that had been separated in 29 folders which represent various classes. The training dataset contained 87,000 8-bit digital 2D color images that are 200x200 pixels, with 29 classes, of which 26 are for the letters A-Z, and 3 classes for SPACE, DELETE and NOTHING. The test dataset contained only 29 images, to encourage the use of real-world test images. [6]

After downloading the dataset and going through the images, I determined that in order to successfully train my model I needed to modify some of the data first. I applied an essential ML tool called data cleaning. In the context of machine learning, data cleaning means filtering and modifying your data such that it is easier to explore, understand, and model. Filtering out the parts you don’t want or need so that you don’t need to look at or process them. Modifying the parts, you do need but aren’t in the format you need them to be in so that you can properly use them [7]. A visualization of the ASLtranslate dataset architecture I used for both versions can be view below.

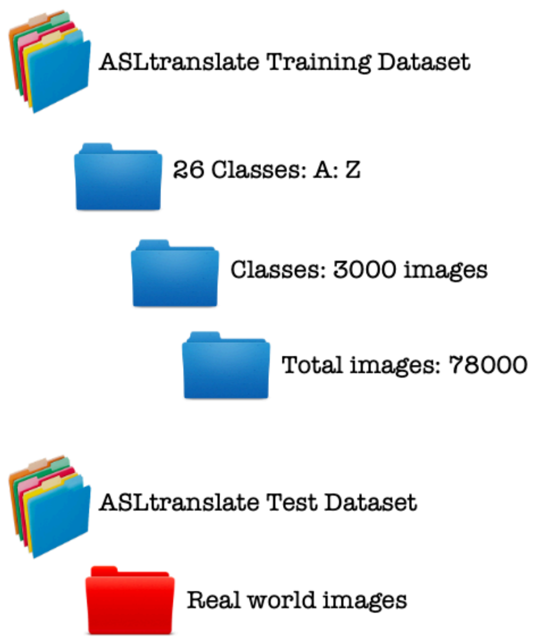


Figure 13: *ASLtranslate image datasets architecture*

V.I.I ASLtranslate (I)

The newly cleaned datasets were then uploaded into a MATLAB_R2019 project I named ASLtranslate (I). These images were used to create an ALS alphabet database or what is referred to as a gallery of images, and the data that will be used for classification. The first thing I needed to accomplish was to have the ability to easily view and manage the images contained within each file folder. In order to do this, I defined each imported folder as "collections of images" by applying MATLAB® `imageSet` method. This method not only returns an object for storing collections of images it also constructs the images into arrays. Additionally, this method automatically creates labels derived from the names of directory. These labels become a crucial part of what is needed for feature extraction and also the classification the images.

```
>> { imgSets.Description }

ans =

   1×26 cell array

   Columns 1 through 16

{'A'}    {'B'}    {'C'}    {'D'}    {'E'}    {'F'}    {'G'}

{'H'}    {'I'}    {'J'}    {'K'}    {'L'}    {'M'}    {'N'}

{'O'}    {'P'}

Columns 17 through 26

{'Q'}    {'R'}    {'S'}    {'T'}    {'U'}    {'V'}    {'W'}

{'X'}    {'Y'}    {'Z'}
```

Table 3: *Results after applying the imgSets method to the dataset*

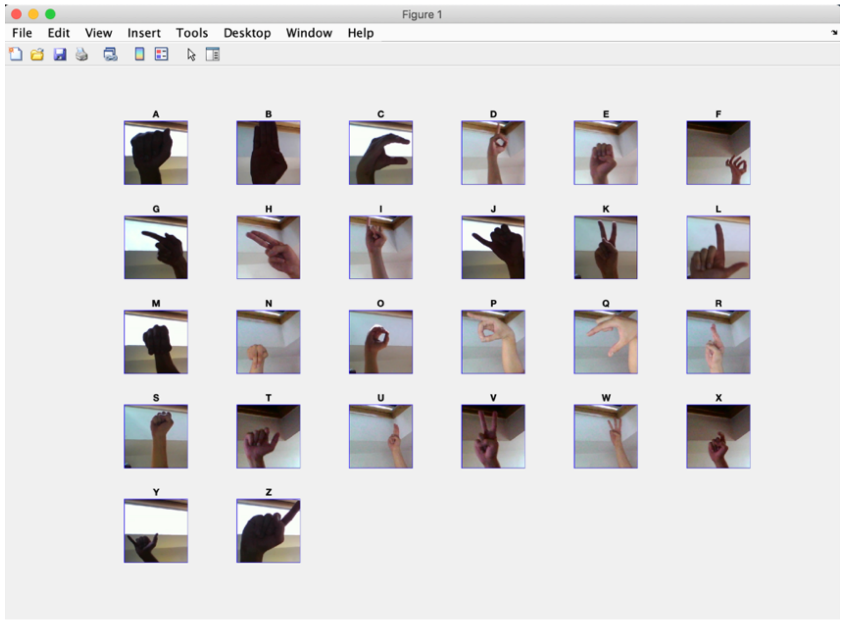


Figure 14: *One randomly selected image from each labeled image set*

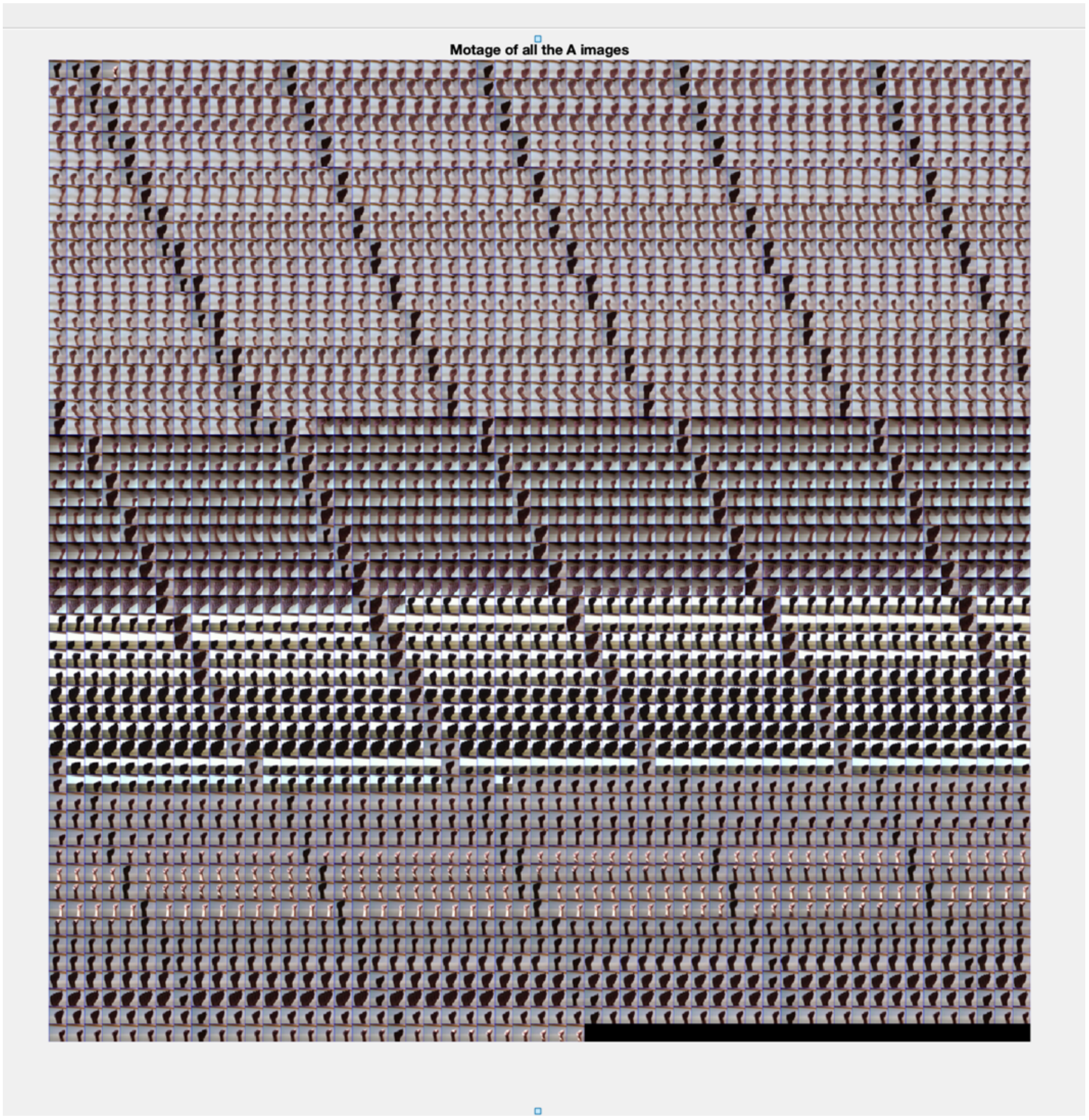


Figure 15: *Montage of all 3000 A's within the ASLtranslate database*

The next step was to partition the data into training and validation sets. I separated the sets using 30% of the images for training and the remanding 70% for validation. Then to further avoid biasing the results I also randomized the data split.

To be able to detect the input images and translate them from 2D images of hand gestures into character letters, I first transformed the images into a visual vocabulary. Then used the newly created visual words to train my image classifier to predict what each input images contained. This was accomplished by using a technique called bag-of-words (BOW) or rather bag-of-visual-words (BOVW). The BOW technique was adapted to computer vision from the world of natural language processing. During the process of BOW a count is made of how many times each word appears within a document, then the frequency of each word is used to create keywords of the document. To visualize the frequency distribution of these occurrences a bar-type graphical display is often used called a histogram. This same concept can be applied for image classification and object recognition problems and is referred to as BOVW. The same general methods are applied but instead of "words" being extracted from documents "features" are extracted from images. The BOVW process is commonly used for image classification and detection.

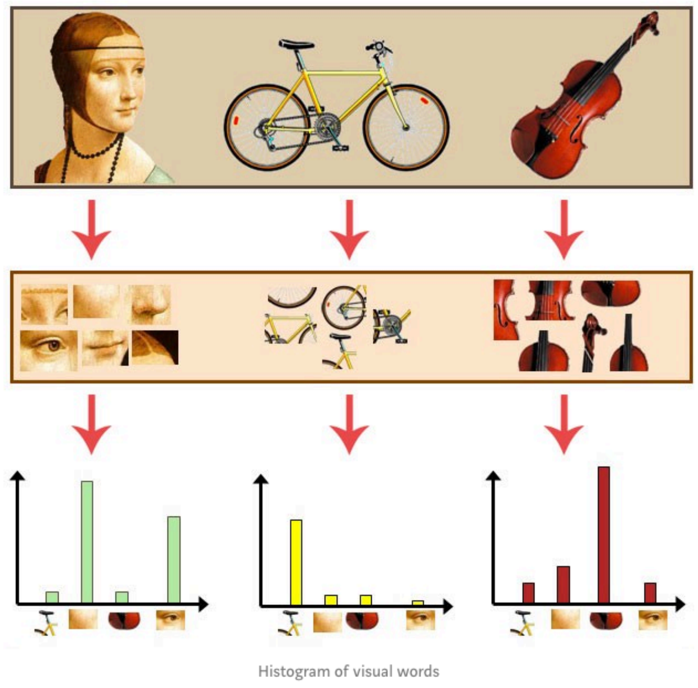


Figure 16: *A Histogram of visual words [8]*

Since images of hand do not actually contain any discrete words, you first have to construct a "vocabulary" of visual words. This is done by extracting feature descriptors from all representative images in each category. Feature extraction is a type of dimensionality reduction, a process of reducing the number of random variables under consideration by obtaining a set of only the principal variables, that efficiently represents the interesting parts of an image as a compact feature vector. To extract the features, I applied the feature detector method called Speeded Up Robust Features (SURF). In order to find principal points or keypoints within the defined images and then encode the information around these points as feature vectors. Then the visual vocabulary is constructed by reducing the number of feature vectors through quantization of feature space by utilizing K-means clustering.

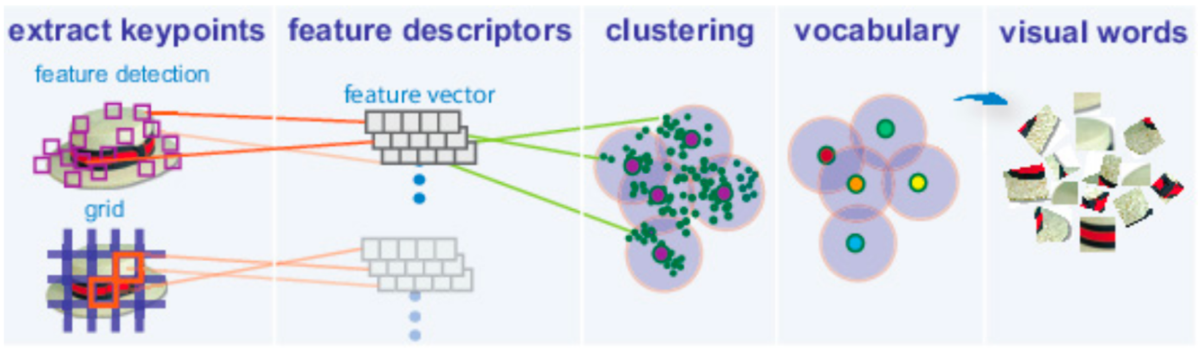


Figure 17: *Extracting visual words from training images* [9]

This process was accomplished with a single call to the `bagOfFeatures` function in MATLAB_R2019a.

1. Extract SURF features from all the images in the training sets
2. Construct a visual vocabulary by reducing the number of features through quantization of feature space using K-means clustering.
3. Provide an encoding method for counting visual words occurrences in images that can be visualized as a histogram

```
% Bag-of-Features
bagOfFeatures(trainingSets);
```

Table 4: *Single call to the bagOfFeatures method in MATLAB_R2019a*

The `bagOfFeatures` function also provides an encoded method for counting the visual word occurrences in the images, that is shown below. The histogram it produces becomes the new and reduced representation of the images. Encoded training images from each of the alphabet categories are then fed into a classifier training model that is invoked by a built-in image category classifier function, called `trainImageCategoryClassifier`.

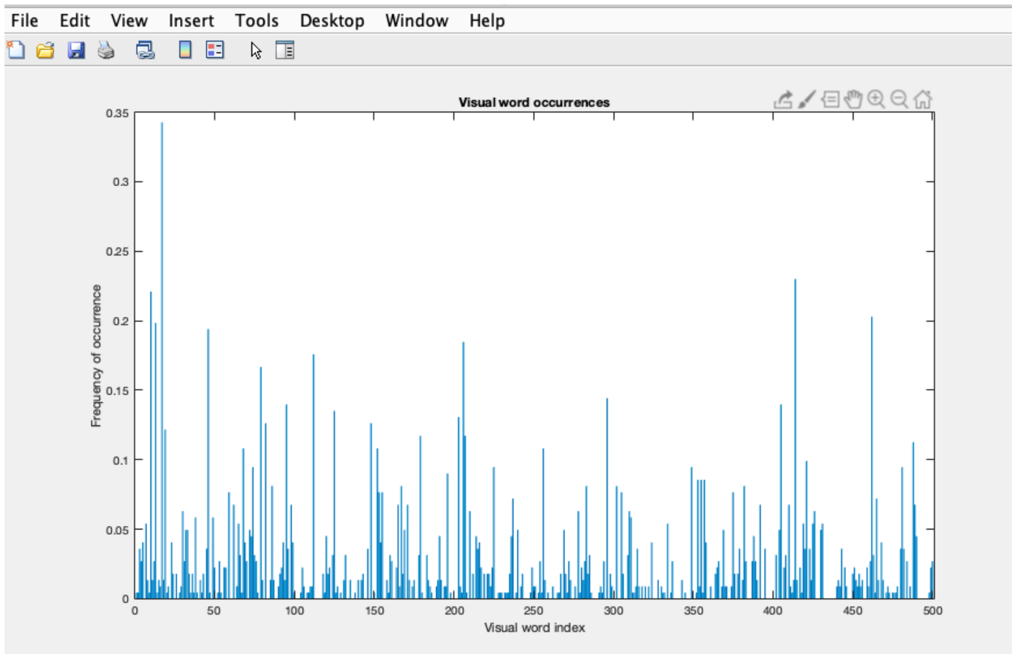


Figure 9: *Histogram of the occurrences of visual words in image "A"*

The `ImageCategoryClassifier` function returns an image classifier. The function trains a multiclass classifier using the error-correcting output codes (ECOC) framework with binary support vector machine (SVM) classifiers. The function utilizes the encoded method of the bag object to formulate feature vectors representing each image category. This formulate feature vectors representing each image category extracted from the `trainingSets` array of `imageSet` objects. The function is a built-in function from MATLAB® Statistics and Machine Learning Toolbox™.

Once the classifier finished training, I invoked the `ImageCategoryClassifier` again, this time to evaluate the performance of the classifier. Using a confusion matrix as a way to visualization of the performance the algorithm, and to obtain the average accuracy scores.

```
* Finished evaluating all the test sets.

* The confusion matrix for this test set is:

                  PREDICTED
  KNOWN    | A       B       C       D
--------------------------------------
  A       | 0.94    0.06    0.00    0.00
  B       | 0.01    0.99    0.00    0.01
  C       | 0.00    0.00    1.00    0.00
  D       | 0.00    0.00    0.00    1.00

* Average Accuracy is 0.98.
```

V.I.II ASLtranslate (II)

The initial methods and approaches to ASLtranslate (II) were the same as the original implementation. I used and uploaded the same cleaned datasets into a MATLAB_R2019 project that I named ASLtranslate_II. To manage and access the image datasets I created an image datastore object. The datastore, is repository for collections of the image files that are too large to fit in memory. This allows you to read and process data stored in multiple files as a single entity, the data isn't imported until it is needed. The function also creates the label names derived from the folder names. This method saved considerable about of time and effort in running the new application.

Transfer learning (TL) is a commonly used technique in deep learning applications. It is the method where a model or a network is developed for a task, then is reused as the starting point for a different model on a second task. Taking a pretrained network and use it as a starting point, fine-tuning the network with transfer learning is usually much faster and easier than training a network with randomly initialized weights from scratch.

Pre-trained learning models that you can use directly with MATLAB®:

- NASNet-Large
- ShuffleNet
- Places365GoogLeNet
- MobileNet-v2
- Xception
- DenseNet-201
- SqueezeNet
- ResNet-18
- ResNet-50
- ResNet-101
- Inception-v3
- Inception-ResNet-v2
- VGG-16
- VGG-19
- GoogLeNet
- AlexNet

AlexNet has been trained on over a million images and can classify images into 1000 object categories. The network has learned rich feature representations for a wide range of images. The network takes an image as input and outputs a label for the object in the image together with the probabilities for each of the object categories. AlexNet is one of the leading architectures for object-detection tasks, it has a comparatively simple architecture that makes it easy to modify and re-train, overall it was perfect choice for ASLtranslate.

TL with AlexNet was the main approach in constructing ASLtranslate (II). A depiction of the typical workflow I used while applying the transfer learning process in MATLAB® can be viewed below.

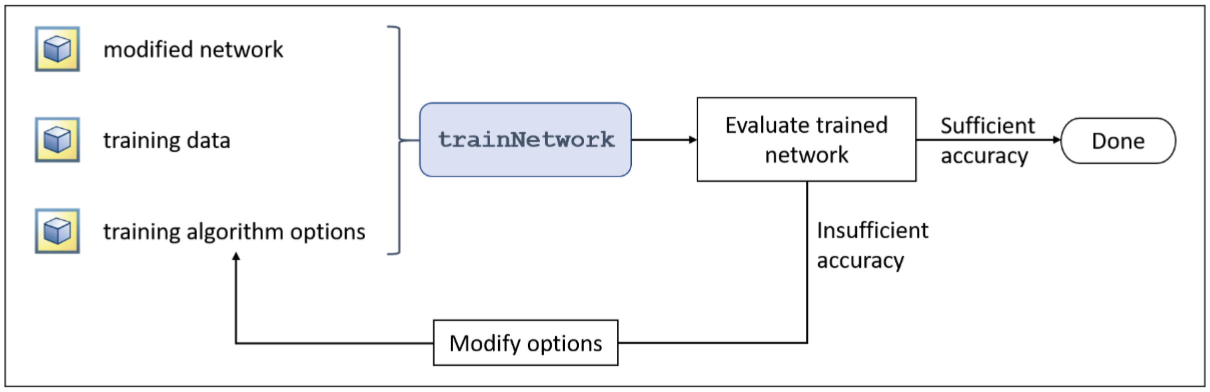


Figure 19: *Transfer Learning Workflow [13]*

After the datastore had been created I split the data into the training and validation sets. Use 70% of the images for training and 30% for validation. I used the split Each Label function that splits the images datastore into two new datastores. Then to further avoid biasing the results I also randomized the datastore split.

Next, I loaded the pre-trained network AlexNet to start the process of fine-tuning the network for classification. I used a analyze Network call in the command window to display an interactive visualization of the network architecture that shows detailed information about the network and layers called in order and in real-time. Then I opened the interactive drag-and-drop network designer by calling deep Network designer from the command line.

The last three layers of the pretrained network AlexNet are configured for 1000 classes, and ASLtranslate has 26 classes. To start classifying the new images I need to edit the pretrained network by replacing the final layers with new layers adapted to your new dataset. First, extract all layers except the last three, from the pretrained network. You want the number of classes to match your data, and the output size to match the number of class in your data.

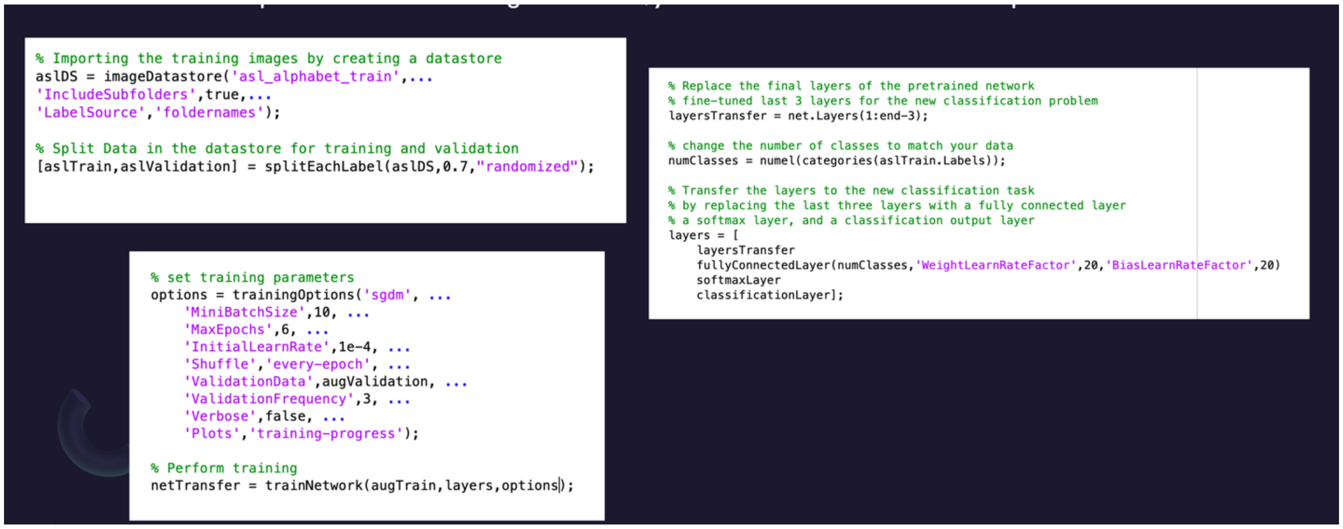
Then, I transferred the layers to the new classification task by replacing the last three layers with a fully connected layer, a SoftMax Layer, and a classification output layer. Specifying, the options of the new fully connected layer according to the new data. Set the fully connected layer to have the same size as the number of classes in the new data. I also increased both the weights and bias values to help the model learn faster.

AlexNet requires input image sizes to be 227-by-227-by-3, and the images in the ASL datastore are 200-by-200-by-3. So, before you can train the network you need to augment the image datastore first with an image augmenter function. This will automatically resize the images in the datastores, randomly flip the images along the vertical axis, and then randomly translates them up to 30 pixels horizontally and vertically. The image augmenter function also helps prevent the network from overfitting and memorizing the exact details of the training images.

To perform transfer learning in MATLAB®, you need to create three components:

1. An array of layers representing the network architecture. For transfer learning, this is created by modifying a pre-existing network.
2. Images with known labels to be used as training data. This is typically provided as a datastore.
3. A variable containing the options that control the behavior of the training algorithm.

These three components are provided as the inputs to the `trainNetwork` function which returns the trained network as output.



```
% Importing the training images by creating a datastore
aslDS = imageDatastore('asl_alphabet_train',...
'IncludeSubfolders',true,...
'LabelSource','foldernames');

% Split Data in the datastore for training and validation
[aslTrain,aslValidation] = splitEachLabel(aslDS,0.7,"randomized");
```

```
% set training parameters
options = trainingOptions('sgdm', ...
    'MiniBatchSize',10, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',1e-4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augValidation, ...
    'ValidationFrequency',3, ...
    'Verbose',false, ...
    'Plots','training-progress');

% Perform training
netTransfer = trainNetwork(augTrain,layers,options);
```

```
% Replace the final layers of the pretrained network
% fine-tuned last 3 layers for the new classification problem
layersTransfer = net.Layers(1:end-3);

% change the number of classes to match your data
numClasses = numel(categories(aslTrain.Labels));

% Transfer the layers to the new classification task
% by replacing the last three layers with a fully connected layer
% a softmax layer, and a classification output layer
layers = [
    layersTransfer
    fullyConnectedLayer(numClasses,'WeightLearnRateFactor',20,'BiasLearnRateFactor',20)
    softmaxLayer
    classificationLayer];
```

Figure 20: *Three components you need to perform transfer learning*

VI. CHAPTER: RESULTS

VI.I Obstacles

During the different stages of development, I repeatedly tried and failed to process all the data simultaneously on my local computer. One of the biggest problems that continually emerged was memory limitations and allocation. When initiating the bag-Of-Feature function on all of the 78,000 images in the data, the system attempts to extract 1,500,000 features from the images, then cluster 80% of those strongest features, to create a 500-word visual vocabulary.

Causing the following results:

1. Program runtime extending the span between 12- 15 hours before timing out and crashing MATLAB® then restarting my local computer
2. Program would time out and crash at initial runtime, closing out MATLAB® and restarting local computer

To try to rectify this problem I uploaded my entire ASL image dataset into MATLAB® cloud-based storage location MATLAB® Drive. In theory the cloud-based drive allows you to securely store and then access your files from anywhere. The drive runs outside of MATLAB® and is accessed from the notification area on your computer. With a current software license MATLAB® offers up to 5GB of free storage. This was not the solution to the memory allocation problems the application was experiencing.

MATLAB® Drive had the following results:

1. Maximum upload file-size allowed is only 256MB. Since, the ASL data went over the max at 1.27GB, data couldn't be directly uploaded into cloud storage.
2. To be able to download file sizes larger than 256MB into the cloud I had to install more software, called MATLAB® connector.
3. After downloading 2 new software packages I was still unable to access the images within MATLAB script or from a command line prompt.

In an attempt to improve the overall performance and runtimes while running my application I downloaded MATLAB® Parallel Computing Toolbox™. The parallel computing has the potential processing power of multicore desktops by executing applications on workers that run locally.

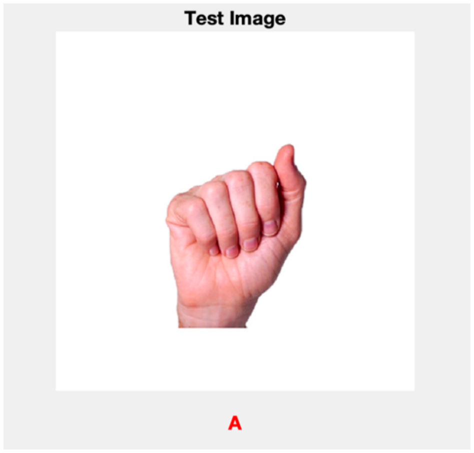
MATLAB® Parallel Computing Toolbox™ results:

1. Parallel pools can only be started if pools were available. Which are often not.
2. Once a pool was activated, you are given 30-minute increments of time, and have to keep re-activate the session. If a session expires during compiling the parallel feature becomes inaccessible.

The MATLAB® toolboxes and features I was attempting to use during development were so new I found it challenging to find information to help with troubleshooting. The machine learning and deep learning features that I wanted to use had only been released in 2019. Many programing languages and methods of doing things have been around for much longer, making community and collaborative learning more accessible. To overcome this obstacle, I enrolled in two of MATLAB® online certificate courses, “Deep Learning Onramp” and “Machine Learning Onramp”.

VI.I.II ASLtranslate (1) results

Since the process to iterate through all of the data did not seem obtainable at the time, I decided to scale the data and streamline the process. I reduced the data down to contain only the ASL images of [A, B, C, D], each letter class containing 3000 images, 12,000 images in total. The results below have been obtained using the scaled data only.



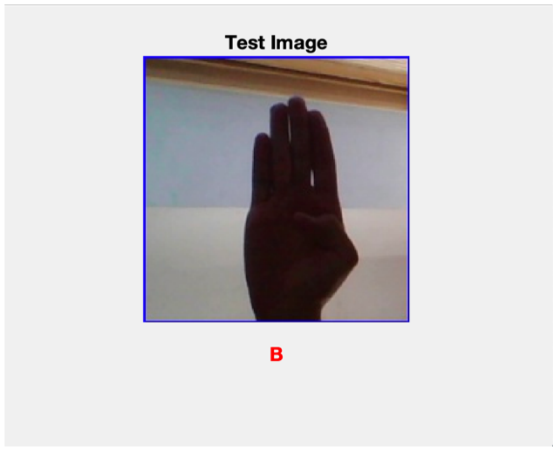
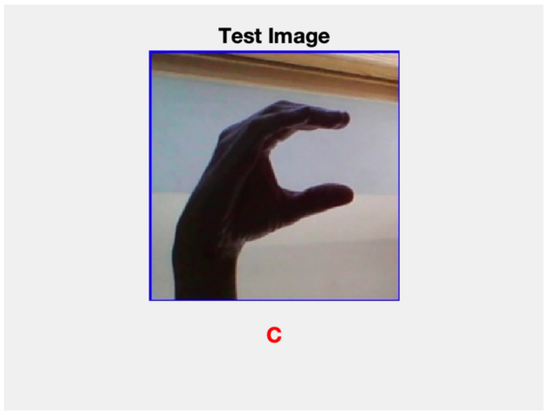
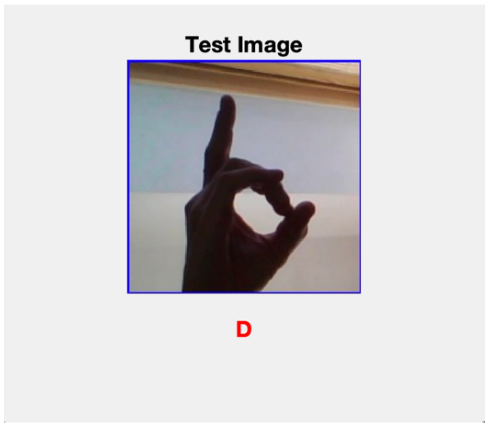


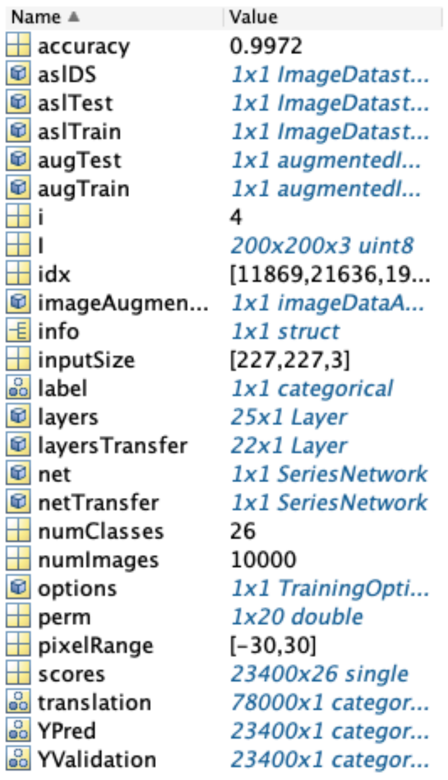
Figure 21: *Input image ASL gesture "A" Output alphabet character A*
ASL gesture "B" Output alphabet character B
ASL gesture "C" Output alphabet character C
ASL gesture "D" Output alphabet character D





VI.I.II ASLtranslate (II) Results

After discovering how to properly manage and allocate large amounts of data on my local computer through the use of a datastore. Finishing the second version of ASLtranslate was relatively simple by comparison to my fist attempt.



Name ▲	Value
accuracy	0.9972
aslDS	1x1 ImageDatast...
aslTest	1x1 ImageDatast...
aslTrain	1x1 ImageDatast...
augTest	1x1 augmentedI...
augTrain	1x1 augmentedI...
i	4
I	200x200x3 uint8
idx	[11869,21636,19...
imageAugmen...	1x1 imageDataA...
info	1x1 struct
inputSize	[227,227,3]
label	1x1 categorical
layers	25x1 Layer
layersTransfer	22x1 Layer
net	1x1 SeriesNetwork
netTransfer	1x1 SeriesNetwork
numClasses	26
numImages	10000
options	1x1 TrainingOpti...
perm	1x20 double
pixelRange	[-30,30]
scores	23400x26 single
translation	78000x1 categor...
YPred	23400x1 categor...
YValidation	23400x1 categor...

Figure 22: *ASLtranslate (II) Workspace after completion of Transfer learning*

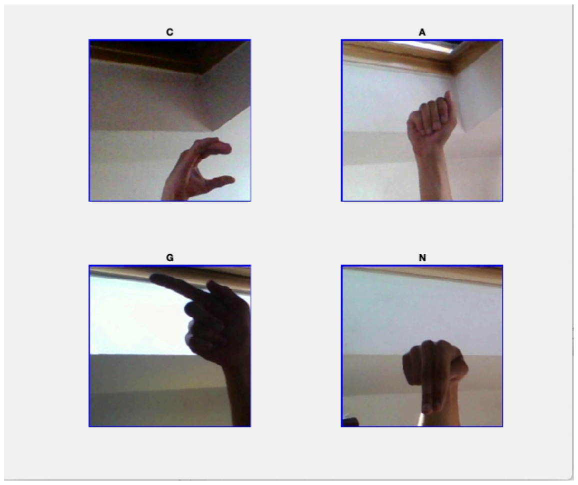
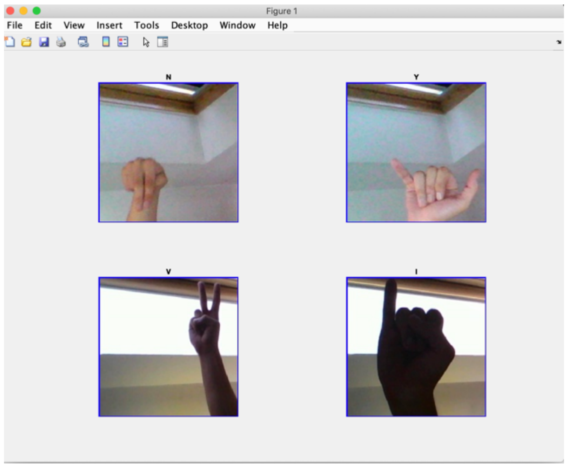
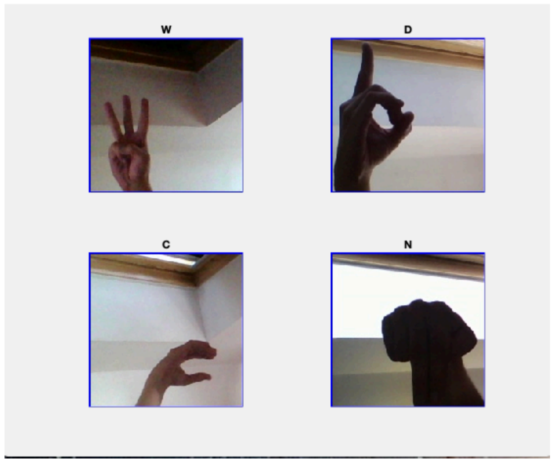
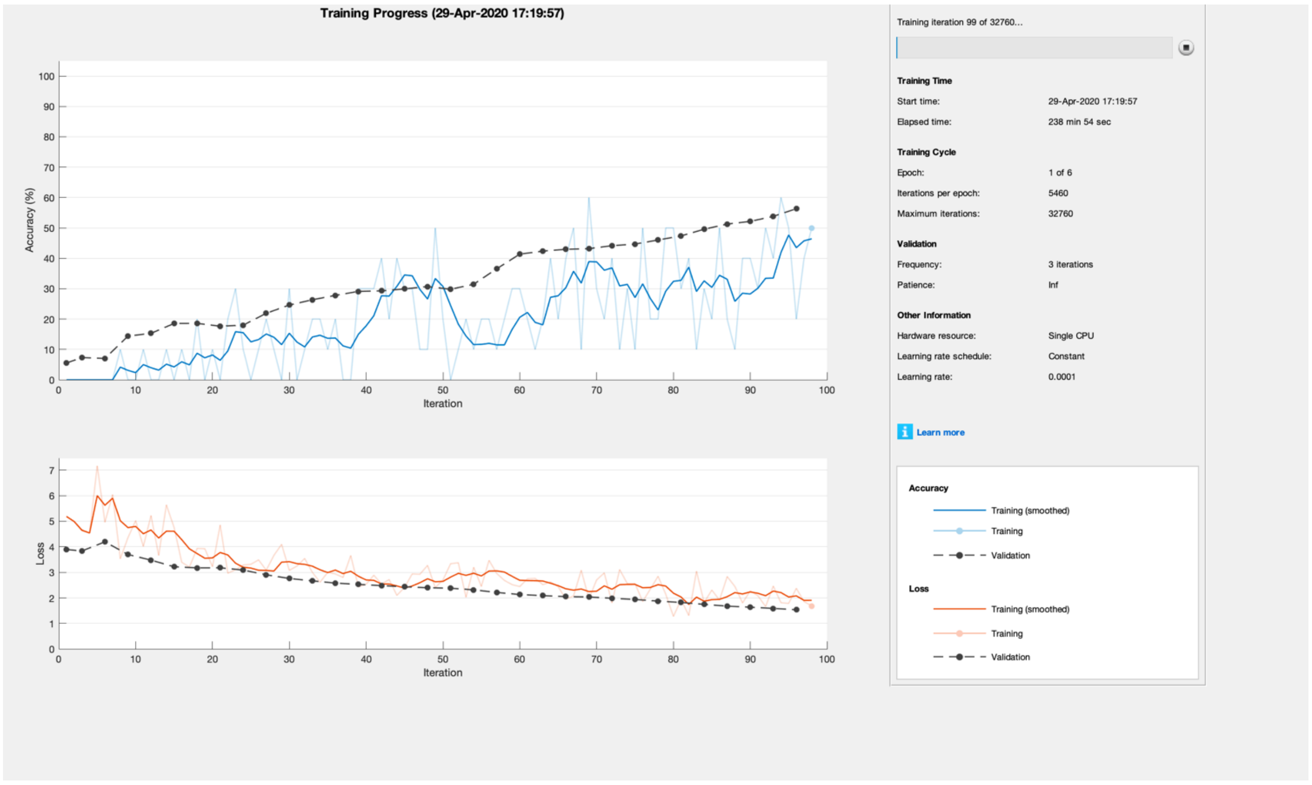


Figure 23: *Four sample validation images with their predicted labels*







Training iteration 99 of 32760...

Training Time

Start time:	29-Apr-2020 17:19:57
Elapsed time:	238 min 54 sec

Training Cycle

Epoch:	1 of 6
Iterations per epoch:	5460
Maximum iterations:	32760

Validation

Frequency:	3 iterations
Patience:	Inf

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.0001

Figure 24: *Training Progress*

Training accuracy: Classification accuracy on each individual mini batch

Smoothed training accuracy: Smoothed training accuracy, obtained by applying a smoothing algorithm to the training accuracy

Validation accuracy: Classification accuracy on the entire validation set

Training loss, smoothed training loss, and validation loss: The loss on each mini batch, its smoothed version, and the loss on the validation set

VII. EVALUATION AND DISCUSSION

In 2017, MathWorks started releasing MATLAB® new data science toolboxes included Statistics and Machine Learning™, Deep Learning™, Image Processing™, and Computer Vision™. Over the past few years and with the release of its newest updated version R2020a, I feel that MATLAB® is a viable option for data science and computer science engineers. MATLAB® can definitely be utilized for development of computer vision, machine learning, deep learning, and object recognition applications. More specifically, for the future and further development of the object recognition application ASLtranslate, I believe it would have a positive social impact on the deaf community.

VII.I Future Development of ASLtranslate

Currently ASLtranslate (I) and ASLtranslate (II) are both considered to be in the functional prototype phase of development. I constructed both versions as a proof of concept for the research and development of this thesis. In the future I am planning on further development of ASLtranslate and bringing the application to completion.

APPENDIX A: ASLtranslate Source Code

../handDB.m

```
% ASLtranslate
% Author: Andrea Murphy
% Date: Spring 2020
% DESC: American Sign Language translator
% Main function

% Store the output in a temporary folder
outputFolder = fullfile(tempdir, 'aslOutput');

% Load Images into a database
handDatabase = fullfile(outputFolder ,'asl_alphabet_train', 'recursive');

% All the “A” images in the gallery A
%figure;
%montage(handDatabase(1).ImageLocation);
%title('Motage of all the A within our database')

rootFolder = fullfile(outputFolder, 'ALS_ObjectCategories');

%Construct arrays of image sets
imgSets = [ imageSet('asl_alphabet_train/A'), ...
    imageSet('asl_alphabet_train/B'), ...
    imageSet('asl_alphabet_train/C'), ...
    imageSet('asl_alphabet_train/D'),...
    imageSet('asl_alphabet_train/E'), ...
    imageSet('asl_alphabet_train/F'), ...
    imageSet('asl_alphabet_train/G'), ...
    imageSet('asl_alphabet_train/H'), ...
    imageSet('asl_alphabet_train/I'), ...
    imageSet('asl_alphabet_train/J'), ...
    imageSet('asl_alphabet_train/K'), ...
    imageSet('asl_alphabet_train/L'), ...
    imageSet('asl_alphabet_train/M'), ...
    imageSet('asl_alphabet_train/N'), ...
    imageSet('asl_alphabet_train/O'), ...
    imageSet('asl_alphabet_train/P'), ...
    imageSet('asl_alphabet_train/Q'), ...
    imageSet('asl_alphabet_train/R'), ...
    imageSet('asl_alphabet_train/S'), ...
    imageSet('asl_alphabet_train/T'), ...
    imageSet('asl_alphabet_train/U'), ...
    imageSet('asl_alphabet_train/V'), ...
    imageSet('asl_alphabet_train/W'), ...
    imageSet('asl_alphabet_train/X'), ...
    imageSet('asl_alphabet_train/Y'), ...
    imageSet('asl_alphabet_train/Z')];

% Split the database into validation and training sets
% 30% for training and 70% for validation
[trainingSets, validationSets] = partition(imgSets, 0.3, 'randomize');

% Training set Validation
A = read(trainingSets(1),1);
B = read(trainingSets(2),1);
C = read(trainingSets(3),1);
D = read(trainingSets(4),1);
%E = read(trainingSets(5),1);
%F = read(trainingSets(6),1);
%G = read(trainingSets(7),1);
%H = read(trainingSets(8),1);

%figure
subplot(3,4,1);
imshow(A)
subplot(3,4,2);
imshow(B)
subplot(3,4,3);
imshow(C)
subplot(3,4,4);
```

../features.m

```
% ASLtranslate
% Author: Andrea Murphy
% Date: Spring 2020
%%% Using Bag-of-features to
%%% Extract SURF features from all images
%%% Constructs the visual vocabulary by reducing
%%% the number of features through quantization of feature
%%% space using K-means clustering

% Bag-of-Features
bag = bagOfFeatures(trainingSets);
```

../histVect.m

```
% ASLtranslate
% Author: Andrea Murphy
% Date: Spring 2020
% DESC: Histogram of the occurrences of visual words in an image

img = read(trainingSets(1),1);
featureVector = encode(bag, img);

% Plot the histogram of visual word occurrences
figure
bar(featureVector)
title('Visual Word Occurrences or Features')
xlabel('Visual Word Index')
ylabel('Frequency of occurrence')
```

../trainCLF.m

```
% ASLtranslate
% Author: Andrea Murphy
% Date: Spring 2020
% DESC: Training process invoked by the trainImageCategory function
        % That relies on a Support Vector Machine(SVM)
```

```
categoryClassifier = trainImageCategoryClassifier(trainingSets, bag);

% Evaluates the classifier performance
confMatrix = evaluate(categoryClassifier, trainingSets);
```

../valCLF.m

```
% ASLtranslate
% Author: Andrea Murphy
% Date: Spring 2020
% DESC: Evaluate the classifier on the validationSet

confMatrix = evaluate(categoryClassifier, validationSets);

% Compute average accuracy
mean(diag(confMatrix));
```

../predict.m

```
% ASLtranslate
% Author: Andrea Murphy
% Date: Spring 2020
% DESC: Using the trained classifier to categorize and predict
      % new images

% img = imread('asl_alphabet_test/a1_test.jpg');
% img = imread('asl_alphabet_test/B_test.jpg');
% img = imread('asl_alphabet_test/C_test.jpg');
 img = imread('asl_alphabet_test/D_test.jpg');

[labelIdx, scores] = predict(categoryClassifier, img);

% Display the string label
categoryClassifier.Labels(labelIdx)

figure;
imshow(img); hold on
xlabel((ans),'FontSize',20,'FontWeight','bold','Color','r');
title('Test Image', 'FontSize',20,'FontWeight','bold')
hold off;
```

APPENDIX B: ASLtranslate II Source Code

../aslData.m

```
% ASLtranslate II
% Author: Andrea Murphy
% Date: Spring 2020
% DESC: American Sign Language translator
% DESC: Creating a Datastore

% Importing the training images by creating a datastore
aslDS = imageDatastore('asl_alphabet_train',...
'IncludeSubfolders',true,...
'LabelSource','foldernames');

% Split Data in the datastore for training and validation
[aslTrain,aslValidation] = splitEachLabel(aslDS,0.7,"randomized");
```

../AlexNet.m

```
% ASLtranslate II
% Author: Andrea Murphy
% Date: Spring 2020
% DESC: Modifying AlexNet a pre-trained CNN to use a classification model
      % for ASLtranslate

net = alexnet;
inputSize = net.Layers(1).InputSize;

% Replace the final layers of the pretrained network
% fine-tuned last 3 layers for the new classification problem
layersTransfer = net.Layers(1:end-3);

% change the number of classes to match your data
numClasses = numel(categories(aslTrain.Labels));

% Transfer the layers to the new classification task
% by replacing the last three layers with a fully connected layer
% a softmax layer, and a classification output layer
layers = [
    layersTransfer

fullyConnectedLayer(numClasses,'WeightLearnRateFactor',20,'BiasLearnRateFacto
r',20)
    softmaxLayer
    classificationLayer];
```

../trainCNN.m

```
% ASLtranslate II
% Author: Andrea Murphy
% Date: Spring 2020
% DESC: Training the Network
% set training parameters
options = trainingOptions('sgdm', ...
    'MiniBatchSize',10, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',1e-4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augValidation, ...
    'ValidationFrequency',3, ...
    'Verbose',false, ...
    'Plots','training-progress');

% Perform training
netTransfer = trainNetwork(augTrain,layers,options);
```

../resize.m

```
% ASLtranslate II
% Author: Andrea Murphy
% Date: Spring 2020
% DESC: preprocessing all images to fit AlexNet input requirements of:
      % [227 227 3]

% Data augmentation helps prevent the network from overfitting
% Along with ensuring the input images match the requirements
pixelRange = [-30 30];
imageAugmenter = imageDataAugmenter( ...
    'RandXReflection',true, ...
    'RandXTranslation',pixelRange, ...
    'RandYTranslation',pixelRange);
augTrain = augmentedImageDatastore(inputSize(1:2),aslTrain, ...
    'DataAugmentation',imageAugmenter);

% Resize the validation images
augValidation = augmentedImageDatastore(inputSize(1:2),aslValidation);
```

../classCNN.m

```
% ASLtranslate II Using AlexNet
% Author: Andrea Murphy
% Date: Spring 2020
% DESC: Classify the test images using the fine-tuned network

[YPred,scores] = classify(netTransfer,augTest);

% Displays four random sample images with their predicted labels
idx = randperm(numel(aslTest.Files),4);
figure
for i = 1:4
    subplot(2,2,i)
    I = readimage(aslTest,idx(i));
    imshow(I)
    label = YPred(idx(i));
    title(string(label));
end
```

../evCNN.m

```
% ASLtranslate II
% Author: Andrea Murphy
% Date: Spring 2020
% DESC: Evaluate and test performance

plot(info.TrainingLoss)

YValidation = aslTest.Labels;
accuracy = mean(YPred == YValidation);

aslActual = aslTest.Labels;

% nnz==non-zero elements in an array
numCorrect = nnz(YPred == aslActual);
fracCorrect = numCorrect/numel(YPred);
```

../matrix.m

```
% ASLtranslate II
% Author: Andrea Murphy
% Date: Spring 2020
% DESC: Visualizations of the performance of the ASLtranslate algorithms

figure
cm = confusionchart(YPred, YValidation);
```

ENDNOTES

[1] MATHLAB Engine API enable execution of MATLAB® in another programming environments. Currently available for: C/C++, Fortran, Java, Python, and C#.

LIST OF REFERENCES

[1] Retrieved March 29, 2020 from https://matlabacademy.mathworks.com/R2019b/portal.html?course=machinelearning

[2] Tarang Shah. 2017. About Train, Validation and Test Sets in Machine Learning. (December 2017). Retrieved May 1, 2020 from https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

[2] Deafness and hearing loss.(March 1, 2020) Retrieved April 9, 2020 from https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss

[3] The Manchester Baby, the world's first stored program computer, ran its first program. Retrieved April 21, 2020, from http://www.computinghistory.org.uk/det/6013/The-Manchester-Baby-the-world-s-first-stored-program-computer-ran-its-first-program

[4] A. M. 1950. I.-COMPUTING MACHINERY AND INTELLIGENCE. (October 1950). Retrieved April 10, 2020 from https://academic.oup.com/mind/article/LIX/236/433/986238

[5] Garbage in, garbage out. (June 2019). Retrieved September 10, 2019 from https://en.wikipedia.org/wiki/Garbage_in,_garbage_out

[6] Akash. 2018. ASL Alphabet. (April 2018). Retrieved September 10, 2019 from https://www.kaggle.com/grassknoted/asl-alphabet

[7] George Seif. 2018. Retrieved April 7, 2020, from https://towardsdatascience.com/the-art-of-cleaning-your-data-b713dbd49726

[8] Bethea Davida. 2018. Retrieved April 2, 2020, from https://towardsdatascience.com/bag-of-visual-words-in-a-nutshell-9ceea97ce0fb

[9] DataFlair Team. 2019. Data Science K-means Clustering - In-depth Tutorial with Example. (May 2019). Retrieved May 2, 2020 from https://data-flair.training/blogs/k-means-clustering-tutorial/

[10] Pawan Jain. 2019. Complete Guide of Activation Functions. (June 2019). Retrieved May 4, 2020 from https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044

[11] Nahua Kang. 2019. Introducing Deep Learning and Neural Networks - Deep Learning for Rookies (1). (February 2019). Retrieved May 4, 2020 from https://towardsdatascience.com/introducing-deep-learning-and-neural-networks-deep-learning-for-rookies-1-bd68f9cf5883

[12] Devjyoti Saha, Diptangshu De, Pratick Ghosh, Sourish Sengupta, and Tripti Majumdar. 2020. Classification of Gender from Human Facial Images using Convolutional Neural Networks. (February 2020).

[13] Retrieved February 17, 2020 from https://matlabacademy.mathworks.com/R2019b/portal.html?course=deeplearning

[14] Vikas Gupta. 2017. Home. (October 2017). Retrieved May 4, 2020 from https://www.learnopencv.com/understanding-feedforward-neural-networks/

Retrieved March 10, 2020, from https://www.mathworks.com/vision/ug/image-classification-with-bag-of-visual-words.html

Michael T. Rosenstein, Zvika Marx, Leslie Pack Kaelbling. To Transfer or Not to Transfer. Computer Science and Artificial Intelligence Laboratory Massachusetts Institute of Technology Cambridge, MA 02139

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. Machine Learning 20, 3 (1995), 273–297. DOI:http://dx.doi.org/10.1007/bf00994018

Geron Aurélien. 2017. Hands-on Machine Learning with Scikit-Learn & TensorFlow: Concepts, Tools, and Techniques to build Intelligent Systems, Sebastopol: OReilly Media.

Abhishek Pandey. Pramod Rathore. Dr.S.Balamurugan. 2019. Machine Learning and Deep Learning Algorithms: BPB Publications.

A Gentle Introduction to Object Recognition with Deep Learning. July 2019. Retrieved September 9, 2019 from https://machinelearningmastery.com/object-recognition-with-deep-learning

Dang Ha The Hien. 2017. The Modern History of Object Recognition - Infographic. (November 2017). Retrieved September 9, 2019 from https://medium.com/@nikasa1889/the-modern-history-of-object-recognition-infographic-aea18517c318

Sumit Saha. 2018. A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way. December 2018. Retrieved September 9, 2019 from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

trainImageCategoryClassifier. Retrieved September 10, 2019 from https://mathworks.com/help/vision/ref/trainimagecategoryclassifier.html

Andreopoulos, Alexander & Tsotsos, John. 2013. 50 Years of object recognition: Directions forward. Computer Vision and Image Understanding.117. 827–891. 10.1016/j.cviu.2013.04.005.

Thesis and Dissertation Guide. Retrieved September 9, 2019 from https://gradschool.unc.edu/academics/thesis-diss/guide/

Framing: Key ML Terminology | Machine Learning Crash Course. Retrieved April 1, 2020 from https://developers.google.com/machine-learning/crash-course/framing/ml-terminology

2020. Object Detection vs Object Recognition vs Image Segmentation. (February 2020). Retrieved April 3, 2020 from https://www.geeksforgeeks.org/object-detection-vs-object-recognition-vs-image-segmentation/

Anon. Deep Learning. Retrieved April 4, 2020 from http://www.deeplearningbook.org/