2017

Designing 2D Interfaces For 3D Gesture Retrieval Utilizing Deep Learning

Spencer Southard
University of North Florida, n00910750@ospreys.unf.edu

DESIGNING 2D INTERFACES FOR 3D GESTURE RETRIEVAL UTILIZING DEEP LEARNING

by

Spencer Southard

A thesis submitted to the
School of Computing
in partial fulfillment of the requirements for the degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA
SCHOOL OF COMPUTING

December, 2017

The thesis "Designing 2D Interfaces for 3D Gesture Retrieval Utilizing Deep Learning" submitted by Spencer Southard in partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences has been

Approved by the thesis committee: Date

_____ _____

Dr. Ching-Hua Chuan
Thesis Advisor and Committee Chairperson

_____ _____

Dr. Kenneth Martin

_____ _____

Dr. Sherif Elfayoumy

Accepted for the School of Computing:

_____ _____

Dr. Sherif Elfayoumy
Director of the School of Computing

Accepted for the College of Computing, Engineering, and Construction:

_____ _____

Dr. Mark Tumeo
Dean of the College

Accepted for the University:

_____ _____

Dr. John Kanter
Dean of the Graduate School

ACKNOWLEDGEMENT

I would like to give a special thank you to Dr. Chuan, my thesis advisor, for all her support and guidance when I needed help, reassurance when I wasn't sure I would succeed, and the push I needed to get back on track to success. Thank you to my thesis committee, Dr. Elfayoumy and Dr. Martin, for their support and flexibility. And finally, to all my friends and family who had to bear my endless discussions about research and artificial intelligence, I thank you for your immense patience and never-ending support. I am successful in this endeavor because of all of you.

Table of Contents

FIGURES

TABLES

ABSTRACT

Gesture retrieval can be defined as the process of retrieving the correct meaning of the hand movement from a pre-assembled gesture dataset. The purpose of the research discussed here is to design and implement a gesture interface system that facilitates retrieval for an American Sign Language gesture set using a mobile device. The principal challenge discussed here will be the normalization of 2D gestures generated from the mobile device interface and the 3D gestures captured from video samples into a common data structure that can be utilized by deep learning networks. This thesis covers convolutional neural networks and auto encoders which are used to transform 2D gestures into the correct form, before being classified by a convolutional neural network. The architecture and implementation of the front-end and back-end systems and each of their respective responsibilities are discussed. Lastly, this thesis covers the results of the experiment and breakdown the final classification accuracy of 83% and how this work could be further improved by using depth based videos for the 3D data.

Chapter 1

INTRODUCTION

Gesture recognition, the means of receiving natural motion input and classifying it as a meaningful action, is a popular topic of interest in artificial intelligence related research. Researchers often seek to investigate or improve upon gesture recognition in countless articles and journals with wide ranges of interests in this focus. However, gesture retrieval, very nearly related to gesture recognition, is somewhat less frequently explored and is a natural progression after well-defined gesture recognition. Gesture retrieval can be defined as the problem of storing gestures and utilizing users queries to retrieve the correct meaning of the hand movement from a preassembled dataset. The research conducted in this thesis seeks to incorporate gesture recognition and retrieval in a real-time system, designed for American Sign Language (ASL) recognition and gesture retrieval.

In everyday human computer interactions, the usual method of initiating an information retrieval search is often keyword based, either through typing or voice diction input. However, both methods find themselves ill-suited when extending to the task of 2D and 3D gesture retrieval. Typed words simply do not have the expression power to represent complex action motions produced by gestures, which are the essential elements in ASL. As an individual observes a sign in ASL for the first time, it may be challenging to verbally describe the sign. Instead, it would be far easier and more natural to recreate the

gesture using your hands or even your finger tips to draw the trajectory of the motion on a mobile touchscreen. The purpose of this research is to craft a system that bridges a more appropriate search mechanism, such as using a tablet with a 2D touchscreen for gesture query input to match it with more complex 3D gestures as recorded in videos for retrieval.

This research will contribute to the field of computer science by extending the field of human computer interaction with a novel system that allows feasible gesture related searches for users who are not familiar with complicated gestures in ASL. Gesture retrieval with the purpose of identifying a gesture without first knowing the verbal mapping to which an associated gesture first belongs presents a very difficult task when limited to a keyword search. The goal is to use a simple gestures that imitate a more complicated gesture in the hope of making a feasible gesture retrieval search possible.

The goal of this research is to create effective mappings between ASL gestures captured in videos and their 2D interpretations (i.e., the trajectory of the hand movement in a 2D drawing), and then build a retrieval system that returns the name of the ASL gesture that is relevant to a given 2D query input. The performance of the retrieval system is evaluated quantitatively using confusion matrixes and overall accuracy. The research in this thesis focuses on ASL gestures in which hands mostly move left, right up, and down (x and y axis). Signs with a z-axis component, hands that move either towards or away from the camera, may prove substantially more difficult to map to a 2D input as the z-axis complement needs to be accounted for in some manner. Other challenges will

include implementing a proper database index that allows us a fast and efficient gesture retrieval system that not only returns the correct result, but also other gestures similar to the correct one.

Over the course of the research this thesis will answer how to best normalize the 2D interpretation of the gestures and 3D gestures in videos, how to retrieve relevant gestures, and what algorithms are most suitable for retrieving the most relevant gestures from the database for a given 2D input query.

Convolutional neural networks and auto encoders are utilized to aid in the gesture retrieval process. A single convolutional neural network is responsible for classifying the 3D representation of the gesture data. An auto encoder is used to transform the 2D input query into a form that can be consumed by the convolutional neural network that is responsible for 3D classification.

The hardware requirements for this thesis require powerful computational hardware. The 2D to 3D gesture mapping is accomplished using convolutional neural networks, which requires a capable CUDA processing unit to build and train in a reasonable about of time. Because of the intensive resource requirement, it is infeasible to build the system on any mobile device. Mobile devices simply do not have the computational power necessary to support computationally intensive neural networks and performance would be poor for clients to hold all the gesture data needed to build these models. Instead, a simple remote

cloud computer with a dedicated database and CUDA processing unit, capable of servicing many mobile devices simultaneously, was used.

Chapter 2

BACKGROUND

This chapter describes the background of gesture search, gesture retrieval, and gesture recognition as it relates to this study's primary research. Gesture search in this context refers to the action of using gestures to search for other content, most likely non-gesture based content. Gesture retrieval on the other hand describes a search to retrieve the name of the gesture, initiated with a query that imitates a gesture partially or completely. The goal of this chapter is to establish a firm understanding of the mechanics that contributes as subcomponents of the gesture retrieval system, and to investigate previously proposed techniques, accomplishments, and shortcomings of prior research.

2.1 Gesture Search

Gesture retrieval begins with gesture search, the need to search for some relevant information or data, given a query input. As stated in the introduction, the focus of this research is using a tablet with a 2D touchscreen as an input medium for gesture retrieval. In work by Li [Li10], the author explored touch screen devices like Apple's iPhone and Google Android phones as a new search interfaces that offer new advantages, and some disadvantages, to the traditional desktop based searching. Here, the author used gesture searches as shortcuts to other pieces of information or areas within the app. Most relevant to the focus of this study, the author indicated a "… mismatch between mobile

user tasks and existing mobile user interfaces" [Li10], referring to the menu and keyword based search systems of traditional desktop computers, which mobile systems have inherited. Compared with the existing menu/keyword based interface, gesture shortcuts in this instance are more natural and easier to perform on touchscreen devices. In addition, gesture shortcuts are capable of being associated with any targeted feature that a user might want to search for or jump to. This approach is very similar to defining shortcuts by tying some action or search to a keyword input, but instead using motion data as the query input. The benefit of this approach is that it allows users to avoid the cumbersome menu-driven interface that are not well suited for touchscreen related input.

In a later publication by Li [Li12], the author describes the success of the Google Gesture Search application, a featured used to create functionality shortcuts by using 2D gestures as the search input, with these insights in mind. This novel search idea was very well received by users with a 4.5 out of 5-star rating over several thousand user reviews. Furthermore, users could successfully complete their gesture query with a single gesture 61% of the time [Li12]. This sets the precedence that users do feel the need and find gesture search as an appropriate search mechanism even though the application was only capable of returning the desired search 61% of the time. When performing a query in the Google Gesture Search app, handwriting detection is first applied to generate a probabilistic model of the search queries character sequence. Afterwards this sequence is compared to the database of mapped gestures, and the gestures with the highest probability of being the queried gesture are returned.

In research by Stefan [Stefan09], the authors identify a similar problem with text-based gesture search mentioned in the introduction of this thesis – it is challenging to lookup the meaning of American Sign Language gestures when users do not know the name. As such their paper proposed an automated system for gesture dictionary lookup, but the approach used in their paper relies on computer vision techniques rather than taking 2D drawing as the interpretation of gestures. In their system, a user was first captured mimicking a sign in front of a camera. With the use of various algorithms, the authors attempted to retrieve the most similar signs from their database of signs. The two algorithms that the authors used to map gesture to gesture are Dynamic Time Warping [DTW] and Dynamic Space-Time Warping [DSTW]. The strength of the DTW algorithm in this use case is that it aligns sequences of data and computes a matching score [Stefan09]. This allows the authors to easily compute the similarity between each gesture in the dataset and the query gesture. DSTW is similar to DTW, but enhanced to "work with multiple candidate hand locations per frame" [Stefan09]. The challenging part of their research was utilizing computer vision to detect the position of the subject's hand in each frame. To compensate for this, the authors manually annotated the hand location when using DTW and performed the detection automatically using DSTW. The results of the research concluded that the manual-DTW approach was more accurate than the automated DSTW approach, however their method returned the correct result in the top ten returned results about 23% of the time.

2.2 Gesture Recognition

There exist many different approaches for implementing a gesture recognition system. In additional work by Li [Li12B], the author creates a hand gesture recognition system using a Kinect depth sensor to feed data to a layered classification system of the author's own design. The proposed system consists of three main classifiers: A finger counting classifier which first classifies gestures based on the number of visible digits in the image, a finger name classifier that examines which ones of a hand's digits are extended, and the third classifier performs finger vector matching. Using these three classifiers the author can classify a handful of simple gestures such as start, thumbs up, and thumbs down accurately between 83% and 99% of the time. This thesis improves upon Li's result, and expands the list of retrieved gestures to include signs from American Sign Language.

While Li largely utilized a collection of algorithms in conjunction with the Kinect sensor data to perform classification, including a modified Moore-Neighbor Tracing algorithm, K-means clustering, and the Graham Scan algorithm [Li12B]; another popular trend in gesture recognition has been the move to deep learning for classification. Deep Neural Networks are capable of their own feature extraction when given raw data, without necessarily requiring the use of individual and complex feature extraction algorithms. As stated by the authors in [Tang15], both deep belief network and convolutional neural networks are both capable of performing gesture recognition with a high degree of accuracy. In their experiment a small amount of pre-processing was necessary on the

dataset included segmenting, resizing, and normalizing the Kinect data. Following their minimal pre-processing steps on the data, the pre-processed data is forwarded to either a deep belief network, a variant of multi-layer perceptron with undirected communications in the top layers of the network [Tang15], or a convolutional network for classification.

In the experiment, the authors gathered thirty-six American Sign Language gestures from eight different individuals. Their hand gesture data was fed into a seven-layer convolutional network and resulted in a classification accuracy of 94.17%. The same experiment was conducted on a deep belief network and had an accuracy rate of 98.12%. Both algorithms have very long training times taking between 490 and 740 minutes to build and train. After training is completed though, each can output a classification result in around 1ms, showing that the approach is suitable for real time systems.

2.3 Gesture Retrieval

As described in the beginning of the chapter, gesture retrieval is the effort of storing gathered gestures in a database system and effectively retrieving relevant gestures from the database later given an input query. There are two main challenges associated with this. The first is in formatting the query for the database and the second is in indexing the database so that queries can be performed efficiently. In research by Athitsos [Athitsos04], the authors attempted to solve both problems.

Using Chamfer distance and Lipschitz embedding the authors in [Athitosos04] showed

that the authors can retrieve images that closely resemble the query image from their database. While using Chamfer distance to get images that are similar to the query is effective, it is very time consuming. To complete that retrieval, the algorithm involves iterating through the entire database of 107K images and takes approximately fifteen minutes to complete; a completion time unsuitable for interactive uses. Using Lipschitz embedding's allows the authors to estimate Chamfer distance, an integer value representing how similar two images are, by comparing the images against a set of references images. The underlying premise is that if two images are similar, then the Chamfer distance to the reference images will also be small, if two images are not similar then the distance will be large. Utilizing these algorithms provide a form of primitive index in that only the gestures that are approximate to the query image will be iterated through and compared with the exact Chamfer distance. The result of this combined approach is that the accuracy is only moderately decreased, but the execution time needed to return a result drops down from fifteen minutes to fifteen seconds.

The work conducted by these authors is of immense value to this thesis. While their result can be improved on in terms of accuracy, the existing work provides us with a baseline measurement to work with and a base method to build upon.

Chapter 3

DESIGNING AND IMPLEMENTING THE GESTURE RETRIEVAL SYSTEM

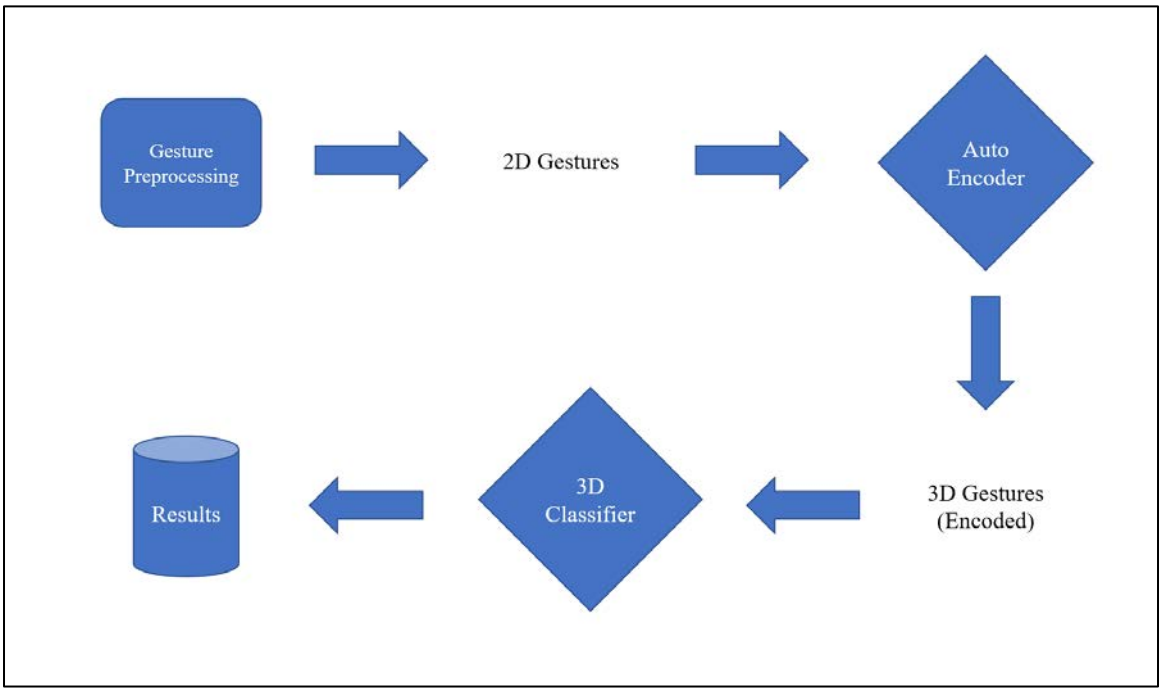


Figure 1: Gesture Retrieval System Overview

The proposed retrieval system is illustrated in Figure 1, which shows the list of components in the gesture retrieval system and describes how the data should flow through the completed system. The 2D gestures are gathered by watching a video of an American Sign Language gesture performed, then the gesture is drawn on a 2D touchscreen. After 2D gestures are gathered, the gestures are passed into the auto encoder and then are converted to 3D gestures, referred to as 3D encoded gestures. The auto encoder is a neural network responsible for converting data to another form, more details will follow in a later section. The 3D encoded gestures are then fed into the 3D

classifier, a convolutional neural network, to be identified. The 3D classifier has been trained to identify all the 3D gestures in the dataset and uses the prior knowledge of its training to determine what is the most likely result. The results returned from the classifier are an array of all possible gestures that match the 2D input query, and each of these possible gestures is reported with a confidence percentage indicating how likely a particular gesture is the correct one.

The following sections describe the components of the system in detail. Section 3.1 explains how 2D (trajectory of hand movement) and 3D (video) gesture data are preprocessed before being fed into the auto encoder. Section 3.2 describes the deep learning solutions used in this thesis, including auto encoder for transforming processed 2D gesture data to match processed 3D data and convolutional neural networks as the 3D gesture classifier. Section 3.3 shows how the gesture retrieval interface is implemented using a service-oriented architecture. At last, section 3.4 discusses the hardware and software components used in the implementation of the gesture retrieval system.

3.1 Data Pre-processing

The pre-processing stage of the system involves the normalization of all 2D and 3D gestures into a common form that be fed to the convolutional classifier and the auto encoder. The normalization consists of several steps. First, the 2D and 3D gestures are resized so that both share the same dimension. Second, one summary image is created to show how the hand movement for a gesture changes over time. Details regarding the first

step can be found in section 3.1.1 shape transformation, while section 3.1.2 discusses summary image generation for the second step.

3.1.1 Shape Transformation

The format that the 2D and 3D gestures are initially collected in are very different. The 2D gesture data is simply a set of (x, y) coordinates that represent pixels that were drawn on the touchscreen by the user. The 3D gesture data is collected as an array of (640 x 480) frames that together comprise a video. To best use both sets of data in the gesture retrieval system it is important to normalize both into a common data form; this is referred to this as shape transformation.

Before describing the process of shape transformation, it is important to understand the meaning of shape in relation to the dimension of arrays. The shape of the data describes the dimension of all of the arrays that comprise each set of data. For example, an array of [0, 1, 2, 3] would have a shape of [4]. That is because there is only one array present here with a length of four. Similarly, an array of [[0, 1, 2], [3, 4, 5]] would have a shape of [2, 3]. The first array in this unit of data has a length of two and array in each index has a length of three.

This is more challenging for the 2D data which consists of any array of (x, y) coordinates; a shape of [m, 2] which is very different from the shape the gesture data needs to move to. The variable m refers to the number of (x, y) points collected for the

gesture. The collection of gestures is converted into a shape of [n, 32, 32, 3] by dividing the coordinates in the array into sets of n, $n \leq m$. The shape of [n, 32, 32, 3] was decided on because it would reduce the resolution of each video frame to 32 by 32 and still preserve the 3 RGB values. This size is small enough to allow for fast processing and big enough to still contain all the needed data. Each set of n represents a frame of data. The (x, y) coordinates in each set are then placed into a spot in the 32 by 32 frame that is directly proportional to their current values. Each value in an active spot of the frame receives pixel values of [255, 255, 255], the color white, to indicate activity.

Transformation for the set of 3D gestures is significantly easier. Each 3D gesture is collected as a video, or sequence of frames, and is already in the shape of [n, 640, 480, 3], where n is the number of frames in the length of the video. Placing the gestures in the right shape simply involves resizing the set of frames to 32 by 32 pixels, delivering the desired shape of [n, 32, 32, 3].

3.1.2 Summary Image Generation

At this point the 3D gestures, while being of the same shape as the 2D gestures, still contain an overabundance of data. As only the motion of the gesture performed in each video is necessary for the purposes of this project, it is important that non-essential background noise be removed from the collected samples. Background noise includes, but is not limited to: attributes of the individual performing the gesture that do not include the hand gesture, the general background behind the individual, and any personal characteristics of the individual.

This is accomplished by applying a moving average to over the course of each set of frames. A small number of frames are averaged together as the reference frame and then the difference between this reference average and the next frame is computed. The resulting frame is referred to as a diff frame. The diff is then balanced and scaled so that each pixel will present with a normal range of values. Balancing consists of taking the pixels with negative values and rounding each up to zero. The rest of the pixel data is then scaled proportionally with the largest pixel value becoming 255 and the smallest becoming zero. This process recursively continues until the final average yields a singular frame. This is referred to as the summary image for the gesture.

Figure 2: Summary Image for 3D Gesture "Now"

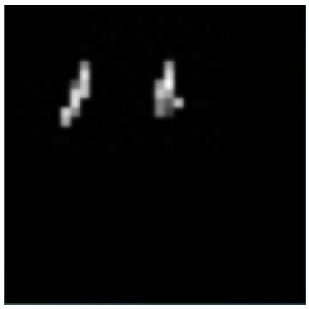


Figure 3: Summary Image for 2D Gesture "Now"

Figures 2 and 3 show summary images for 2D and 3D "now" gestures, respectively. The 3D gesture is no longer recognizable to humans as most attributes related to the original video have largely been removed. What remains is essentially a heat map, highlighting areas of concentrated movements across the frames over time in the video.

Since the 2D data used to generate the summary image only consists of active points, the resulting 2D summary image looks rather different than the 3D summary image. The black color in Figure 3 represents areas of no activity, while the areas of white represent areas of high activity. While it appears that there may be little to no correlation between these two images, the auto encoder, described in the next section, empirically shows that it is capable of converting the 2D summary into a 3D summary image for the 3D gesture

classifier to recognize the gesture.

3.2 Deep Learning Solution

Discussed in the following subsections are the deep learning solutions used to perform gesture recognition and retrieval in the proposed system.

3.2.1 Auto Encoder

Auto encoders are a type of unsupervised neural network model that take an input dataset and output a similar dataset with an identical shape (as discussed in section 3.1.1 Shape Transformation). The purpose of the auto encoder is to learn to encode data into another usable form. For example, in image recognition training auto encoders can be deployed against a training set of data to generate new, but similar, images to the training set [Geron17]. Doing so allows the researches to generate hundreds of new data samples used to increase their training set size. The variation in each image produced by the auto encoder allows the classifier being trained to learn to identify attributes of the image instead of overfitting each image. In work by Geron [Geron17], a study is described where an auto encoder was used against an image set consisting of handwritten digits. To expand their dataset, the authors used an auto encoder to reproduce each image, with slight variations, so the neural network in the study would have many similar images of each class of number to train with. In this way, the network can learn a variety of ways to write a particular digit instead of requiring a digit to be written exactly as in a specific

example.

Another common task of this type of neural network is for denoising images. In work by Kim [Kim17], the authors use a convolutional auto encoder to remove noise out of sonar images taken by an automatous under-water vehicle [Kim17]. As conditions under water can be very cloudy, the use of an auto encoder on the image data is to allow both humans and other machine learning algorithms recognize what is present in each image. The noisy sonar image would be fed into the auto encoder and the resulting image produced by the auto encoder would be the same image but with less visual distortions. The authors concluded that by using an auto encoder the noise in each image was reduced to acceptable levels.

The purpose of the auto encoder in this study is to take a 2D summary image and convert it into a 3D form. To train the auto encoder, a set of 2D summary images is provided as the input. The 3D summary images that represent the same gesture are also provided to the auto encoder but as the target output for the 2D input. Using gradient decent optimizations [Geron17], the auto encoder gradually changes the weights of the network so that the output of the network for the 2D input becomes more and more similar to the target 3D output.

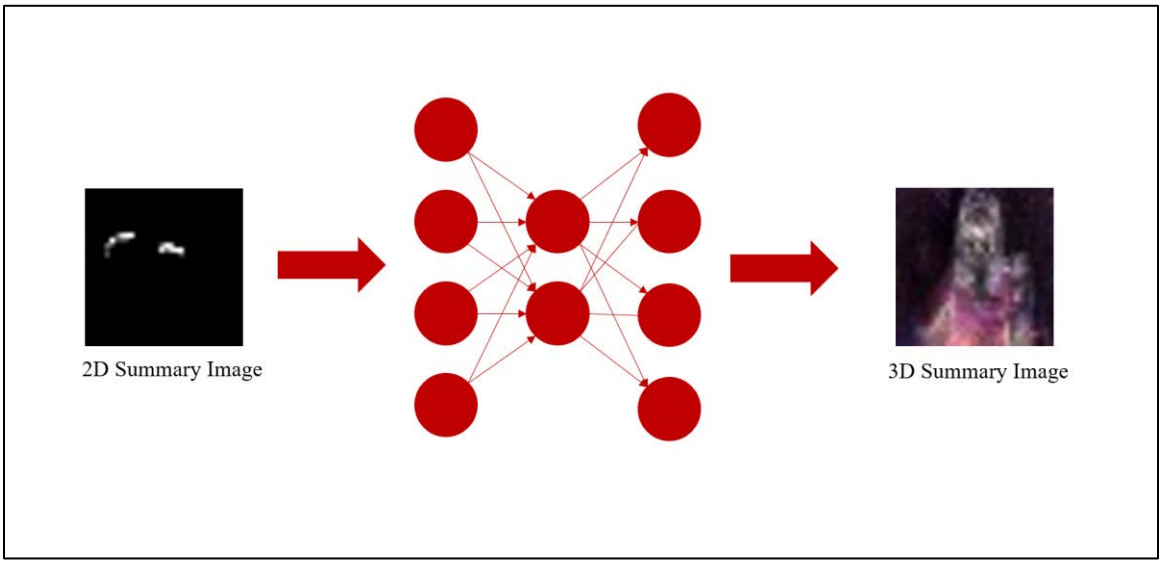


Figure 4: Auto Encoder Layers

As shown in Figure 4 the auto encoder used in the system is comprised simply of three fully connected layers. The 2D summary image is passed into the first fully connected layer, being reduced and resized in the hidden layer, and then restored to its original size in the last fully connected layer.

The data that this auto encoder is trained with consists of the cross product of the 2D summary images with all 3D summary images of the same gesture. For example, for each 2D summary image for the "now" gesture as an input for the auto encoder, it is paired with every 3D summary image of the "now" gesture as the target output. If both the 2D and 3D "now" gestures had 20 samples respectively, this would produce 400 training samples after being cross applied. The rationale behind this is that each 2D summary image, when properly converted by the auto encoder, should produce a 3D summary image of the same gesture. There is a many to many mapping between 2D and

3D gestures. It is important that the auto encoder learns that it is converting a class of object to a different representation of the same class, instead of learning that every specific "now" gesture has a very specific 3D gesture it should be converted to. This approach is also applied to the 3D gestures that are submitted to the auto encoder for training. Each 3D gesture is crossed applied to all other 3D gestures of the same type. After all the input data is paired with an appropriate output this auto encoder was trained against around 20,000 sample conversions.

3.2.2 Convolutional Neural Networks

A convolutional neural network, or CNN, is a form of feed-forward artificial neural network like that of the multi-layer perceptron [Lecun99]. As described in Lecun's research [Lecun99], the development of CNNs originate in the 1960's and have been expanded on in the following decades; inspired by research into the visual system of cats, which contain locally-sensitive, orientation-selective neurons [Lecun99]. This is modeled in CNNs with several layers of convolutional layers each composed of many neurons. The neurons in each layer of the CNN are connected only to a small region of the layer above it, as observed in Figure 5. A benefit of this is that each layer of the CNN can focus on smaller features in the image it is being used on and as the image passes through the network those smaller features can be viewed as components of larger features. Convolutional neural networks can contain pooling layers between their convolutional layers to help enhance the neural network; discussed later in this chapter.

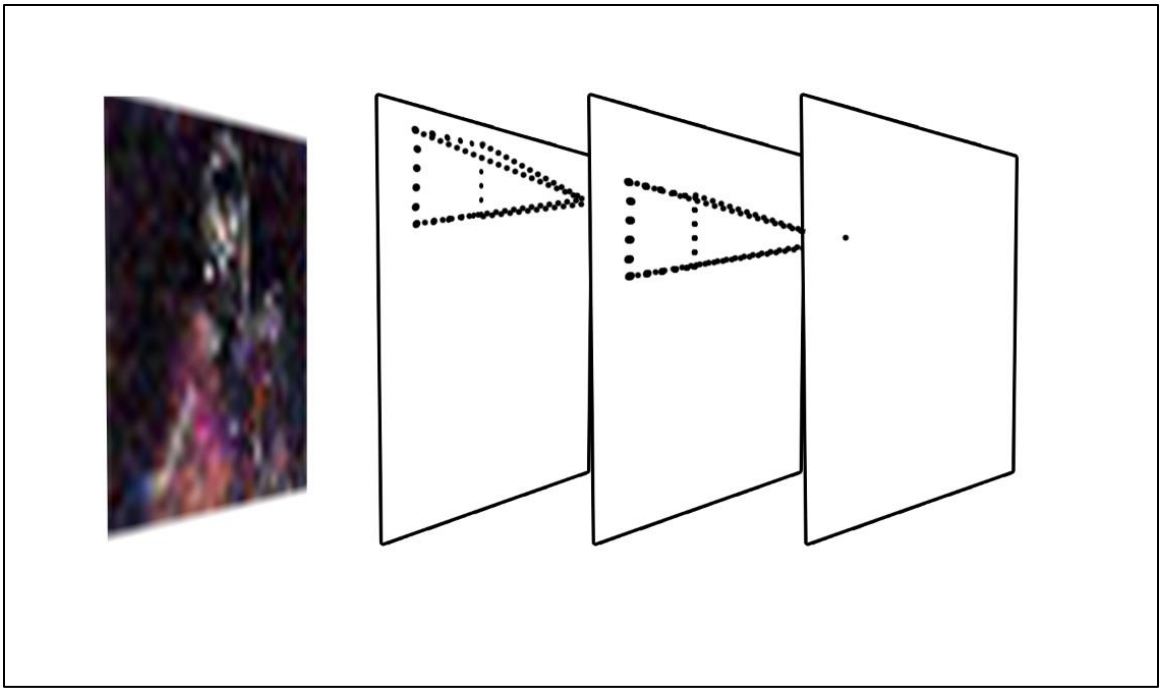


Figure 5: Convolutional Neural Network

Convolutional Neural Networks are widely used for their efficiency at image classification [Krizhevsky12]. Convolutional Neural Networks are commonly used in popular machine learning benchmark datasets such as LabelMe [Krizhevsky12], ImageNet [Krizhevsky12], and MNIST [LeCun]. MNIST is a popular collection of over 60,000 handwritten digits commonly used for benchmarking and testing neural networks [LeCun]. LabelMe and ImageNet are two image sets with hundreds of thousands of images over thousands of categories [Krizhevsky12]. Using a traditional feed-forward neural network on extremely large datasets such as these would be a very time-consuming process. However, due to the construction of CNNs, which have far fewer connections between layers, CNNs are able to be trained considerably faster at minimal accuracy loss [Krizhevsky12]. The example in Figure 5 shows neurons of a convolutional neural network being mapped to regions of the convolution layer above

each one. This is an example of how the connections of a convolutional neural network differ from a multi-layer perceptron which is fully connected at each layer.

The last step in the proposed gesture retrieval system, 3D summary images extracted from videos are taken as the input for the convolutional neural network to extract features useful for recognizing different gestures.

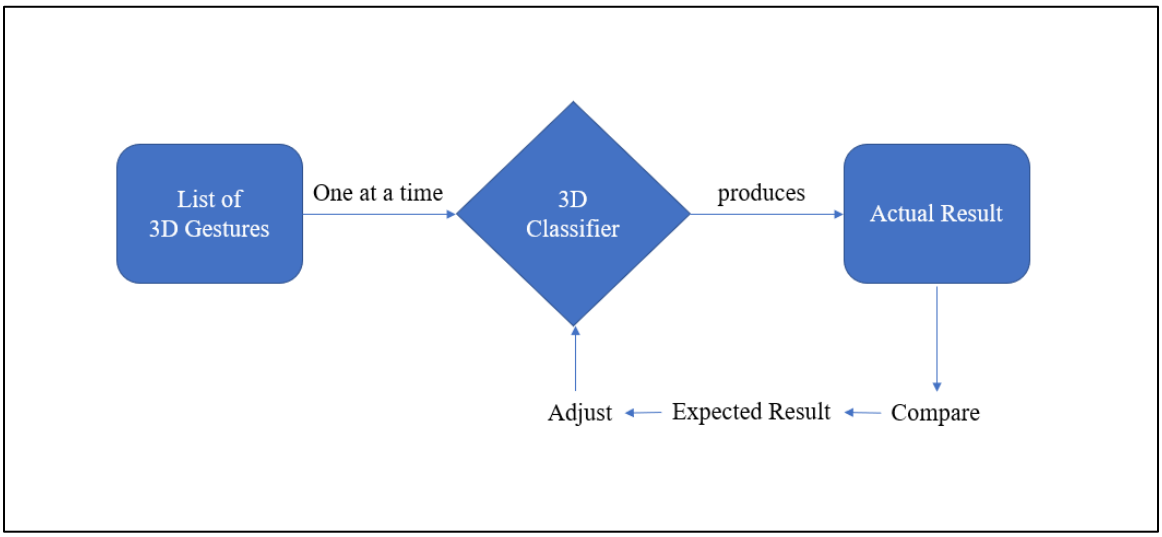


Figure 6: Convolutional Neural Network Training

To examine the validity of the pre-process stage two convolutional classifiers are trained and benchmarked; one for 2D summary images and one for 3D summary images. As observed in Figure 6, training is done by continuously feeding data into the classifier. The output of the classifier is then compared to the expected result. The classifier then adjusts its internal logic based on the result and continues training with the next gesture. The purpose of this step is to verify that each set of data can be recognized by a neural

network in its original form before being transformed by the auto encoder. If the classifier cannot differentiate different gestures in the 2D summary images nor different gestures in the 3D summary images, it would be highly unlikely that auto encoder would be successful.

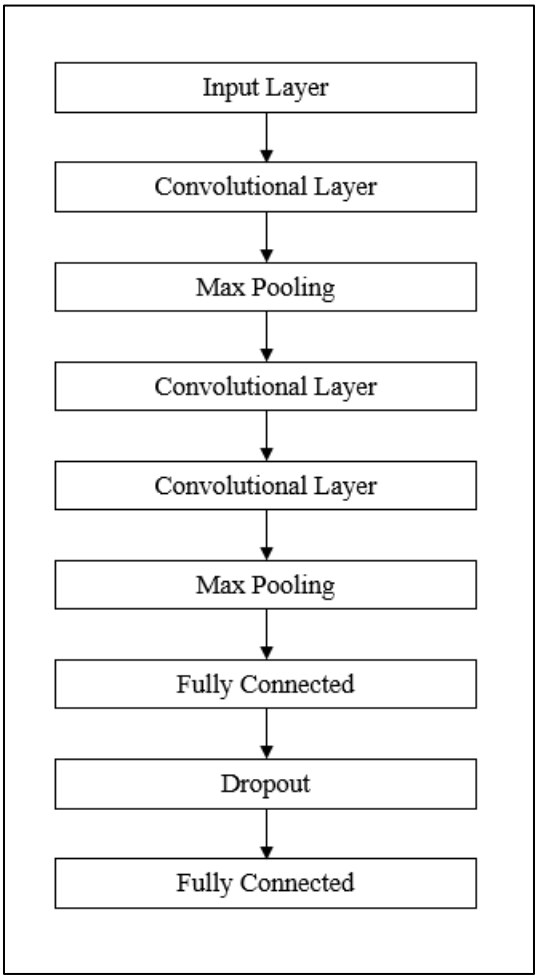


Figure 7: Layers for 3D Model Classifier

Figure 7 shows the various layers that comprise the convolutional neural network used for the classification of the 3D data. The input data layer simply takes in the training data

according to the contracted shape of [*n*, depth, 32, 32, 3] where depth is the number of summary images generated from each sample.

The convolutional layers form a hierarchical structure in which neurons, explained in the beginning of this subsection, from each proceeding layer of the network are connected to a small region of the layer above it. The power of this approach is that it allows fewer connections between layers to represent an entire image. Each layer does not need to have a full set of neurons to take in all the data at once, which would result in a prohibitive amount of connections to layers further on.

The pooling layers of the neural network in Figure 7 are responsible for subsampling and reducing the resolution of the feature maps to aid the neural networking in achieving spatial invariance [Scherer10]. The most commonly used pooling strategy is max pooling, which uses the maximum value to represent the region in the convolution kernel [Geron17]. The final fully connected layer reduces the output of the convolutional neural network into twenty outputs, which correspond to the twenty possible gestures. These twenty outputs are used to determine the ranking of the gestures for retrieval. The gesture that has the highest value among all the outputs is considered as the most relevant gesture.

3.3 Gesture Interface and Service Overview

The proposed retrieval system is implemented via a mobile interface as the client with server-oriented architecture as the backend server as show in Figure 8. The client consists of an Android application responsible for managing touch events, API calls, and displaying information to the end user. The backend system features several service layers that facilitate data transformation and storage from the Android application. The API maintains a direct IP connection to the Mongo database, and calls into the neural service, which contains the convolutional neural network, using HTTP GET calls. Together these service layers act as a seamless functional unit, while internally distributing responsibilities to designated service layers. The service layer acts as the retrieval node, as discussed at the beginning of Chapter 3, to answer a call to the API and return the relevant gesture results for a given query.

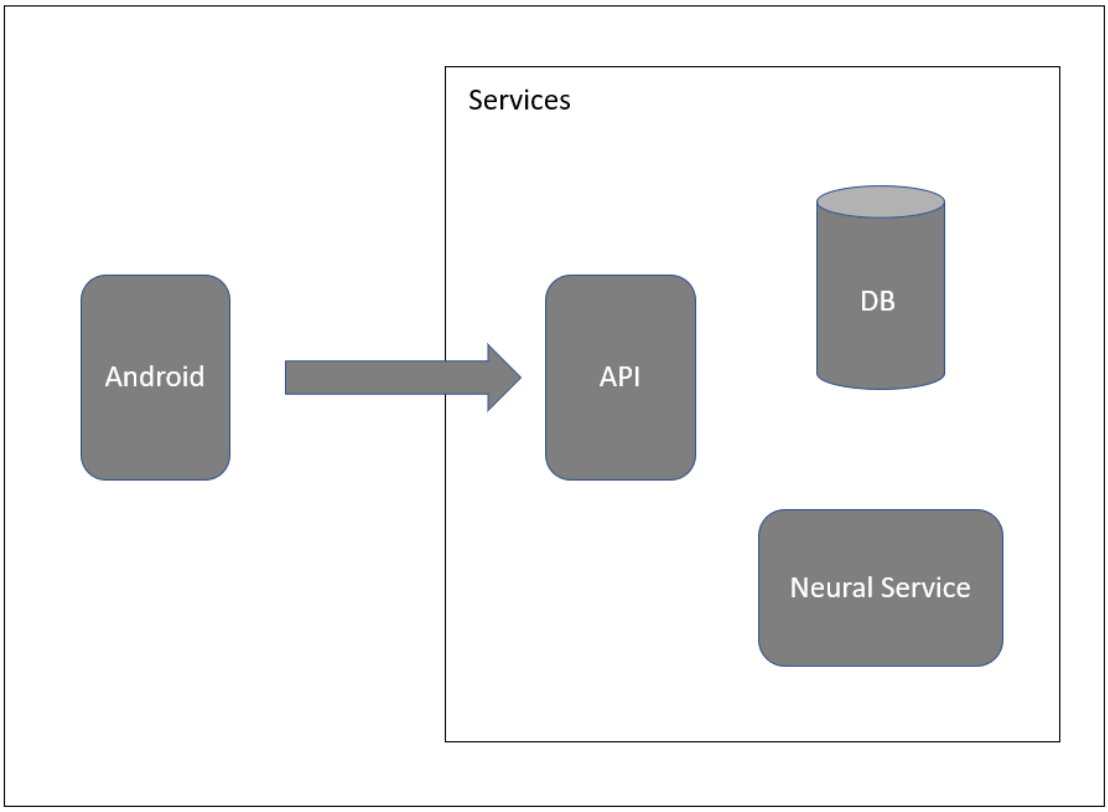


Figure 8: Client and Services Overview

The mobile device interface in Figure 9, shows the user interface used to describe 2D gestures as queries. Users simply select the play button when ready to record their gesture. Users move the right and left hand in the general motion of the gesture to query, and press play when finished. The icons representing left and right hands are mirrored to represent how the user will see the gesture perform. After completing the gesture, users are prompted to name the gesture before it is uploaded for processing.

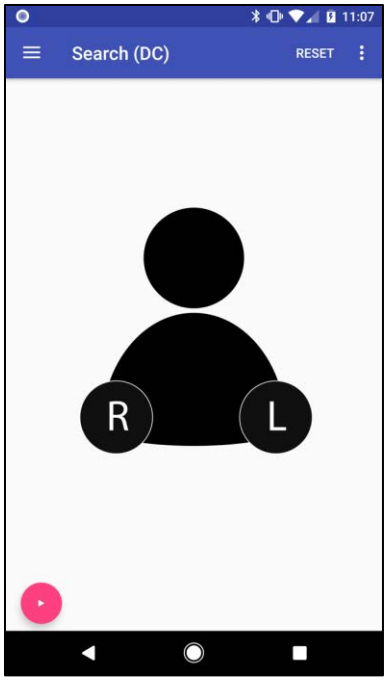


Figure 9: Gesture Interface

3.4 Hardware and Software Requirements

Discussed in the subsections below are the hardware and software components used to facilitate both the client application and the backend services. Some of these hardware components may be substituted with similar or comparable hardware at the expense of added complexity and development time.

3.4.1 The Mobile Application

A mobile touchscreen device capable of running Android applications is used in this study. An Android device is not specifically required to recreate this study, any mobile

platform with a touchscreen could do. The Android client is responsible for capturing all touch events in real-time, storing them temporarily in memory, and querying an external API for classification. The client will also serve as the full implementation of the 2D interface required by this study.

3.4.2 The Backend Server

An external server is required to host the various services which comprise the core services of the study, including the API, neural service, and Mongo database. The Android client communicates to the external server to store and process data on the fly. For the purposes of this study, a dedicated Linux machine was used to host all the services simultaneous, although each service could be separated and distributed across several different machines in future applications of the experiment.

3.4.3 NVidia Graphics Card

An Nvidia GPU (graphics processing unit) with a minimum of computing capability 3.0 and 3GBs of GDDR memory is not required, but highly recommended, for the training and execution of the convolutional neural networks used in this study due to the complexity of the network being constructed and trained.

Using a graphics card for heavy computing tasks is widely known as general purpose graphics processing or GPGPU processing. A single graphics card usually consists of

thousands of smaller, less powerful, execution units compared to modern day CPUs, which commonly run four execution units. Tasks that are capable of being run in parallel benefit greatly and experience a rapid acceleration in processing speed while running against graphics cards such as these. The performance boost granted by even mid-tier graphics cards can reduce the time needed to build and train a convolutional neural network from several hours to just a few minutes.

Included in the use of this study was an Nvidia GeForce GTX 1080TI. Commonly recognized as a gaming graphics card, it shares many similarities with its sibling the GTX Titan X, including core count, memory type, speed, and size. The Titan X is often used as a workstation card for general-purpose (GP) GPU tasks because of its high performance and attractive price point. Over the course of this study GPGPU acceleration has saved countless hours that would have gone unused while networks were building in the background.

3.4.4 Video Capture Device

A video capture device, such as a Kinect sensor or laptop camera, is required to gather the needed gestures to train the neural networks against. A minimum resolution of 640 by 480 pixels is required to gather an acceptable level of detail. It is important the video capture device remain stationary and the setting that the device is used in contains minimal distractions in the background to reduce the noise of the data samples.

3.4.5 Tensorflow

Tensorflow is an open source machine learning library developed by Google and the developers on the Google Brain project. Developed in part to handle massive datasets and distributed workloads. Tensorflow also affords developers fine grained controller over execution and memory utilization of their applications [Abadi16].

Tensorflow can run on multiple general-purpose GPUs, distributing its workload across several thousand cores and two or more graphics cards. The ability to utilize GPU acceleration and distribute tasks across multiple GPUs allow far more complex neural networks with far more training data to be built and trained in a reasonable time frame [Abadi16].

In this study, Tensorflow was used to construct the convolutional neural network that serves as the primary 3D classifier and to construct the auto encoder which is responsible

for transforming 2D summary images into 3D summary images. Tensorflow is available on a wide range of operating systems and is most commonly used as a Python3 library.

Chapter 4

EXPERIMENTAL RESULTS

This chapter describes the experiments conducted for evaluating the proposed retrieval system. First, it is described how data is prepared and collected for the experiments in section 4.1. In section 4.2, the results are shown for the experiment used to validate the effectiveness of summary images as 2D and 3D gesture classification tasks. Section 4.3 describes the experiment results for the gesture retrieval system. The experimental results are observed as the accuracy in retrieving the correct 3D gesture when given any 2D gesture. As this is an experiment of a gesture retrieval system, the system produces a list of ranked gestures where the top one on the list is considered the most relevant result for the query. In this study, the accuracy of the results are examined as the top-n retrieval: a successful retrieval is considered by having the correct gesture in the top n gestures on the list.

4.1 Data Collection

The data collected for this study consisted of twenty American Sign Language gestures performed on a low-resolution, less than 720p, web cam. Twenty samples of each sign were collected, some against different backgrounds, and some gestures being very similar in appearance. Appendix A contains a complete list of signs used in this experiment. IRB approval was gathered to conducted data from participants to be used in this study

under case number 967107-3.

The 2D data as the drawn trajectory of hand movement was generated using the Android client. All twenty signs were replicated on a 2D interface using an Android mobile device. Between twenty and thirty samples of each 2D sign was collected for a total of 515 gesture samples.

4.2 Data Verification Using CNN Classifiers

After generating the 2D and 3D summary images it is important to verify the data that was normalized is still identifiable from gesture to gesture. To accomplish this, two classifiers are employed using convolution neural networks for each set of data: a 2D classifier and a 3D classifier. By running each set of data through their respective classifiers it can be confirmed that both sets of data can still be identified.

CNN	Accuracy
2D Classifier	87%
3D Classifier	97%

Table 1: 2D and 3D Classifier Accuracy

The results in Table 1 show that both sets of data are capable of being classified when trained against only 2D or 3D data. It was also observed that the 2D classifier is around 10% less accurate at classifications than the 3D classifier. This is likely due to the

limitations of gathering the gesture data on a 2D interface. The 3D classifier in Table 1 is very accurate so this learned model is saved for the final step in the retrieval system to classify the data that is outputted from the auto encoder.

4.3 Experimental Results for the Gesture Retrieval System

The results of the experiment are gathered by taking all 2D gestures collected and performing pre-processing steps as shown in Figure 1. Once all 2D gestures have been converted to 3D encoded gestures by the auto encoder, the encoded data can then be tested by applying each gesture individually through the saved 3D classifier from the data verification step as described in the previous subsection.

On average, 83% of the 3D encoded gestures can be retrieved as the top gesture. The gesture that is the least accurate to retrieve was the gesture for “thank you”. This gesture was only retrieved successfully for around 41% of gestures. Nearly half of the signs sampled, 9 out of 20, could be retrieved 100% successfully, as shown in Figure 11.

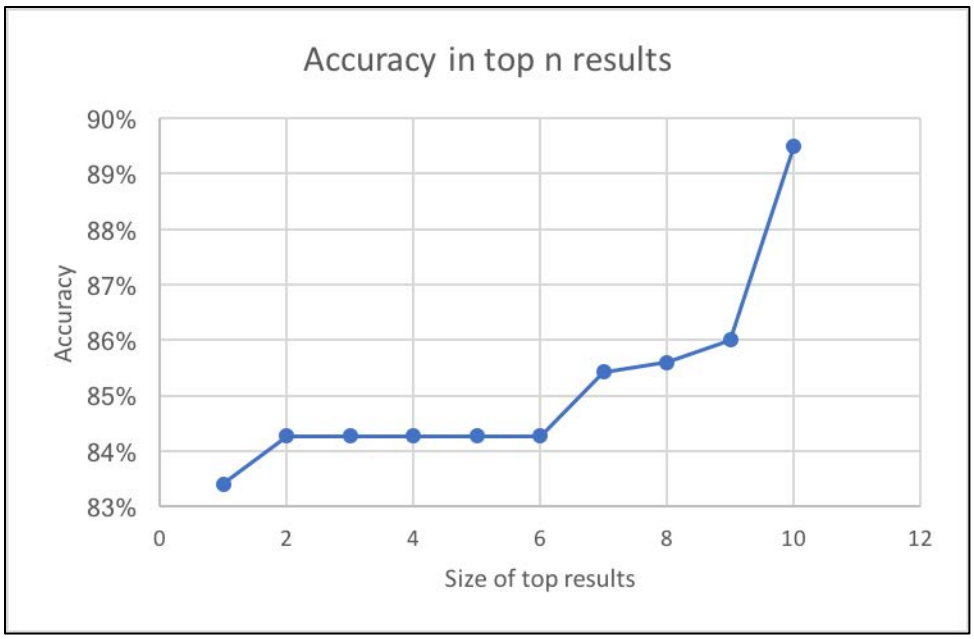


Figure 10: Top *n* results

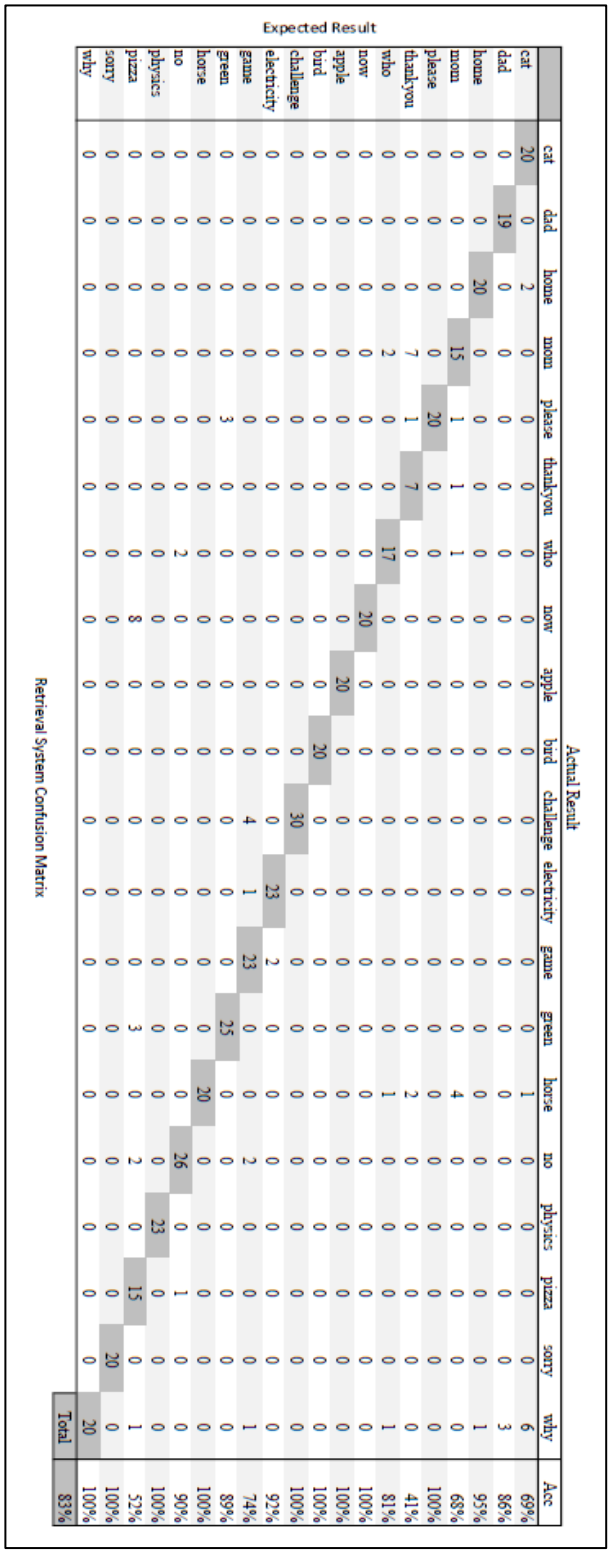
Figure 10 shows the top-*n* retrieval results. Compared with the top 1 retrieval result, the top three gestures retrieved increases the success rate between 0% and 2%. It was expected that measuring success by numbers in the top three results would increase the success rate more significantly, however observations of the results show that the gains are minimal at best. Looking at Figure 10 it can be observed that it isn't until the top 10 results are measured that are results increase substantially, but at that point the top results includes half of the dataset for a 6% gain in retrieval accuracy.

Figure 11 shows the confusion matrix generated from the result set. A confusion matrix is an excellent tool for showing how the classifier performed in its classification tasks. The left labels represent the actual gesture that was the target to be classified and the top result returned by the retrieval system. Using the confusion matrix, it can be observed that the gesture for "thank you" was commonly misclassified as the gesture for "mom"

and “please”. Out of seventeen attempts, it was misclassified as “mom” seven times and misclassified for “please” just once. This misclassification can be easily understood by comparing the hand movements for these ASL signs as both “thank you” and “mom” have a hand moving towards or away from the camera in the center of the body.

Some of the observed errors can be attributed in the confusion matrix to inadequacies of the auto encoder. Figure 12 illustrates a confusion matrix for a classifier that is trained on 3D data only. In this confusion matrix, it is clear to observe that the convolutional neural network has little to no difficulty in distinguishing most 3D gestures from each other. The lowest classification accuracy observed in this matrix is still as high as 85%.

To understand how the auto encoder performs the transformation in detail, Figure 13 shows examples of summary images to allow for visual observation of any similarities between the images.



	cat	dad	home	mom	please	thankyou	who	now	apple	bird	challenge	electricity	game	green	horse	no	physics	pizza	sorry	why	Acc
cat	20	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	6	69%
dad	0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	86%
home	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	95%
mom	0	0	0	15	1	1	1	0	0	0	0	0	0	0	4	0	0	0	0	0	68%
please	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100%
thankyou	0	0	0	7	1	7	0	0	0	0	0	0	0	0	2	0	0	0	0	0	41%
who	0	0	0	2	0	0	17	0	0	0	0	0	0	0	1	0	0	0	0	1	81%
now	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	100%
apple	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	100%
bird	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	100%
challenge	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	100%
electricity	0	0	0	0	0	0	0	0	0	0	0	23	2	0	0	0	0	0	0	0	92%
game	0	0	0	0	0	0	0	0	0	0	4	1	23	0	0	2	0	0	0	1	74%
green	0	0	0	0	3	0	0	0	0	0	0	0	0	25	0	0	0	0	0	0	89%
horse	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	100%
no	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	26	0	1	0	0	90%
physics	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	100%
pizza	0	0	0	0	0	0	0	8	0	0	0	0	0	3	0	2	0	15	0	1	52%
sorry	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	100%
why	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	100%
Total																					83%

Figure 11: Retrieval System Confusion Matrix

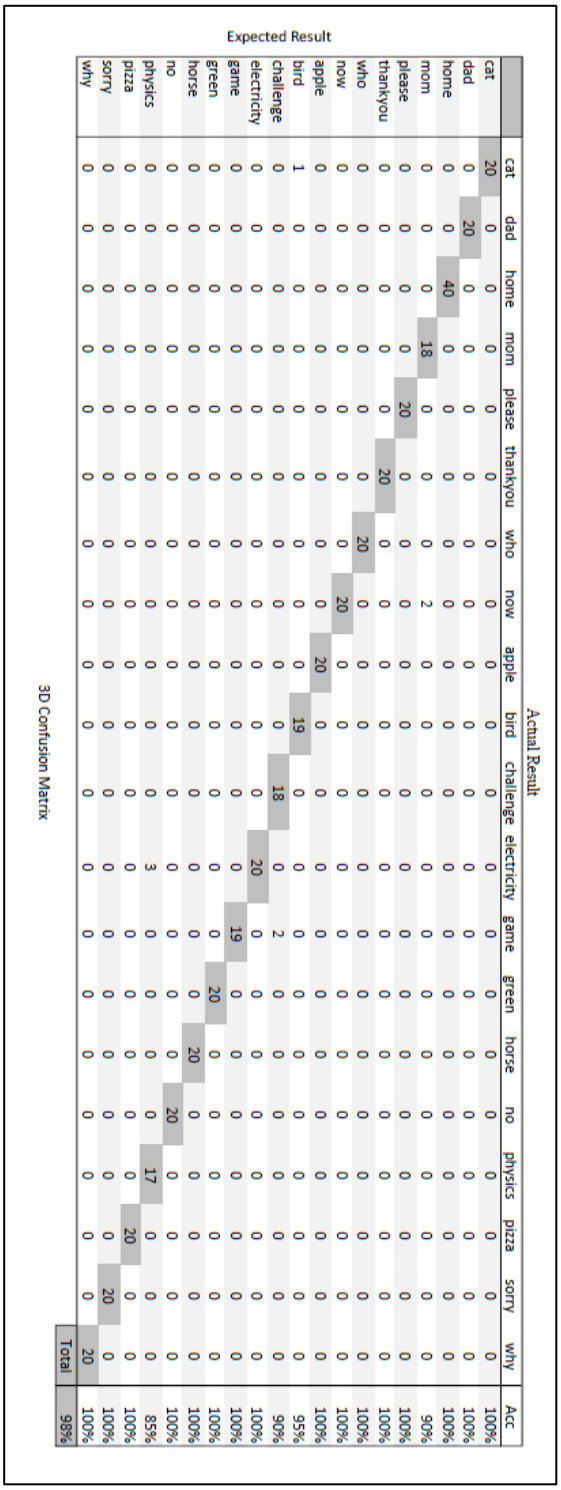


Figure 12: 3D Confusion Matrix

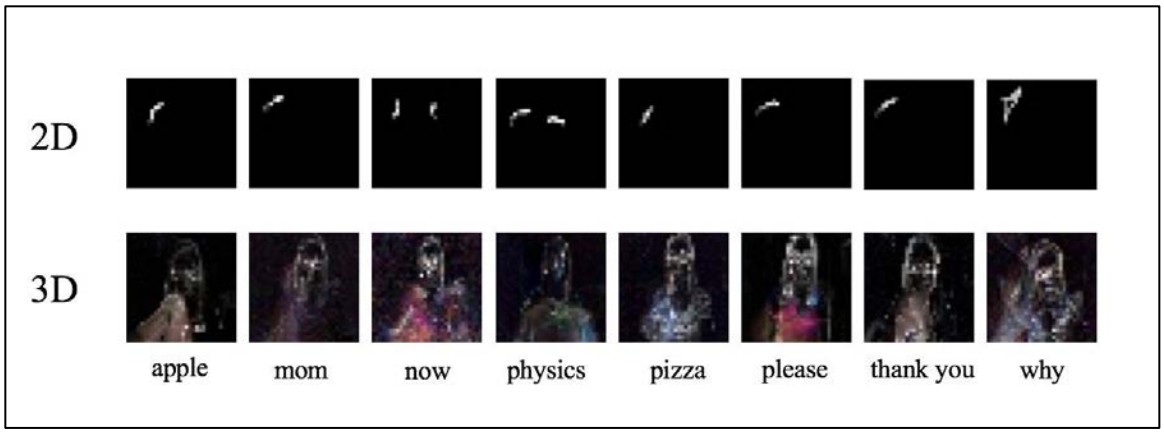


Figure 13: 2D and 3D Summary Images

As shown in the confusion matrix in Figure 12, the gesture for “pizza” was another gesture that was commonly misclassified, however this sign was misclassified as four different gestures. It was misclassified as “apple” eight times, “green” three times, “no” twice, and “why” once. The difference between this kind of misclassification and the previous miscalculation with “thank you” is that the classifier was much less precise with its error. Looking at Figure 13, there is a clear similarity between the 2D summary images for “pizza” and “apple”. Even though these signs were commonly misclassified, it does not mean that the 2D samples are necessarily bad. While at a glance both gesture summary images appear to be very alike to one another, humans are at a disadvantage for comparing these two signs manually compared to the convolutional neural network. The neural network has access to the raw pixel data and can easily observe subtle changes in the pixel data that may present a clear pattern for each sign that an ordinary human would not be able to observe.

The final accuracy average across all signs was 83% for the retrieval system. While

image classification tasks generally achieve a much higher percentage of accuracy than this, this can be considered a great success, because the results illustrate that it is possible for an image that was constructed with one data source to be classified as an image from a different data source after passing through the auto encoder. The results clearly show that the auto encoder can extract the necessary features from the 2D summary images and reconstruct it into a 3D summary image that can be correctly classified by a reliable percentage of the time.

Chapter 5

CONCLUSION

Gesture retrieval and the process of normalization and transforming both 2D and 3D data have proven to be a challenging task to accomplish. The availability of machine learning frameworks such as Tensorflow has greatly aided developers and data scientists by providing a tool capable of leveraging the power of modern day graphics cards.

Chapters 1 and 2 discussed the background into the gesture retrieval problem and provided information of other approaches and techniques used in this line of work prior to this research. Chapter 3 delved into the implementation of the proposed gesture retrieval system and set the expectation of what the result would look like. This chapter covers some of the core concepts that were applied to the approach and the pre-processing techniques used to normalize both sets of gesture data.

In Chapter 4 the observed result was discussed. While there were many signs the system was highly efficient at retrieving, there were also several that were difficult to be retrieved reliably. The premise of a successful retrieval being defined as the correct output being listed in the top n of the returned results proved unnecessary given the accuracy of the system at large. Combining the auto encoder and gesture classifier, the system showed that these deep learning solutions were able to correctly classify a 2D gesture without the need of looking further down the results list. In fact, if the auto

encoder and classifier were not able to get the correct answer in the top result, then the correct result would be unlikely to be returned in the next top three results.

The results of this work, as it relates to the previous research covered in Chapter 2, stands on its own. None of the previous research attempted to use 2D gestures as a means of querying for gestures gathered in a 3D space. The results of the 3D gesture classification is on par with what has been observed from other studies such as one conducted by Tang [Tang15]. However, there currently exists no studies that use 2D gestures as a query mechanism for 3D gestures that the results of this study can be directly compared against.

While the results of this experiment were very promising, there is still more research to be done. Future iterations of this work include expanding the data to observe how the accuracy of the system would scale across a large volume of sign gestures. As for the gestures that were classified incorrectly, more research needs to be done to help those gestures return in the top three gestures to be retrieved. As depth cameras become more prominent, a study to observe how the results of this experiment might change if the 3D data included depth information might yield results on how this research could further improve.

Lastly, when given a large enough dataset it would be interesting to see if the system would be able to classify 2D gestures that the auto encoder has never trained against. It is possible that when trained on enough initial data, the auto encoder could successfully

convert 2D gestures to 3D encoded gestures without ever having seen the 2D example before.

REFERENCES

Print Publications:

[Abadi16]
Abadi, Martin, et al. "TensorFlow: A System for Large-Scale Machine Learning." Operating Systems Design and Implementation. Vol. 16. 2016, pp. 265-283

[Athitsos04]
Athitsos, Vassilis, and Stan Sclaroff. "Database Indexing Methods for 3D Hand Pose Estimation." Gesture-Based Communication in Human-Computer Interaction Lecture Notes in Computer Science, 2004, pp. 288-99.

[Geron17]
Geron, Aurelien. “Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems”, O'Reilly Media, Sebastopol, California, 2017, pp. 353-376, 413-435

[Ji13]
Ji, Shuiwang, Wei Xu, Ming Yang, and Kai Yu. "3D Convolutional Neural Networks for Human Action Recognition." IEEE Transactions on Pattern Analysis and Machine Intelligence IEEE Trans. Pattern Anal. Mach. Intell.35.1, 2013, pp. 221-31.

[Kim17]
Kim, Juhwan, Seokyong Song, and Son-Cheol Yu. "Denoising auto-encoder based image enhancement for high resolution sonar image." Underwater Technology (UT), IEEE, 2017, pp. 1-5

[Krizhevsky12]
Krizhevsky, A., Sutskever, I., & Hinton, G. E. “Imagenet classification with deep convolutional neural networks”. In Advances in neural information processing systems, 2012, pp. 1097-1105

[LeCun99]
LeCun, Yann, et al. "Object recognition with gradient-based learning." Shape, contour and grouping in computer vision, 1999, pp. 823-823.

[Li10]
Li, Yang. "Gesture Search: A Tool for Fast Mobile Data Access." Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology, 2010, pp. 87-96

[Li12]
Li, Yang. "Gesture Search: Random Access to Smartphone Content." IEEE Pervasive Comput. IEEE Pervasive Computing 11.1, 2012, pp. 10-13.

[Li12B]
Yi L. “Hand gesture recognition using Kinect”. 2012 IEEE International Conference On Computer Science & Automation Engineering, 2012, pp. 196-199

[Mehndiratta17]
Mehndiratta, P., Sachdeva, S., & Soni, D. “Detection of Sarcasm in Text Data using Deep Convolutional Neural Networks”. Scalable Computing: Practice and Experience, 2017, pp. 219-228.

[Scherer10]
Dominik Scherer, Andreas Muller, and Sven Behnke. “Evaluation of pooling operations in convolutional architectures for object recognition”. In Proceedings of the 20th International Conference on Artificial Neural Networks: Part III (ICANN'10), 2017, pp. 92-101.

[Stefan09]
Stefan, Alexandra, Haijing Wang, and Vassilis Athitsos. "Towards Automated Large Vocabulary Gesture Search." Proceedings of the 2nd International Conference on Pervasive Technologies Related to Assistive Environments - PETRA '09, 2009, pp. 16

[Tang15]
Tang, Ao, Ke Lu, Yufei Wang, Jie Huang, and Houqiang Li. "A Real-Time Hand Posture Recognition System Using Deep Neural Networks." TIST ACM Transactions on Intelligent Systems and Technology ACM Trans. Intell. Syst. Technol. 6.2, 2015, pp. 1-23.

Electronic Sources:

[LeCun]
LeCun, Yann, et al. “The MNIST Database of Handwritten Digits.” MNIST Handwritten Digit Database, http://yann.lecun.com/exdb/mnist/, last accessed December 11, 2017.

APPENDIX A:

GESTURE LIST

1. Cat
2. Dad
3. Home
4. Mom
5. Please
6. Thank you
7. Who
8. Now
9. Apple
10. Bird
11. Challenge
12. Electricity
13. Game
14. Green
15. Horse
16. No
17. Physics
18. Pizza
19. Sorry
20. Why

APPENDIX B:

IRB APPROVAL

UNF
UNIVERSITY of NORTH FLORIDA.
Office of Research and Sponsored Programs
1 UNF Drive
Jacksonville, FL 32224-2665
904-620-2455 FAX 904-620-2457
Equal Opportunity/Equal Access/Affirmative Action Institution

MEMORANDUM

DATE: September 22, 2017

TO: Ching-Hua Chuan, Ph.D.
Computing

FROM: Dr. Jennifer Wesely, Chairperson
On behalf of the UNF Institutional Review Board

UNF IRB Number: 967107-3
Approval Date: 09-22-2017
Expiration Date: 09-22-2018
Processed on behalf of UNF's IRB

RE: Review of Revisions for New Project by the UNF Institutional Review Board
IRB#967107-3: "American Sign Language Gesture Research"

This is to advise you that your project titled "American Sign Language Gesture Research" underwent "Expedited" Categories 6 & 7 review on behalf of the UNF Institutional Review Board. Your reviewer recommended approval without further modifications. As of today, your proposal which required modifications was approved.

This approval applies to your project in the form and content as submitted to the IRB for review. All participants must receive a stamped and dated copy of the approved informed consent document when possible. Any variations or modifications to the approved procedures or documents must be cleared with the IRB prior to implementing such changes. *For example*, if you plan to make changes to your stamped and dated informed consent form, it will be necessary to submit a copy of the revised form via an amendment so that it can be reviewed and approved prior to use. Once approved, a new stamp and date will be included on the revised consent form so that it can be used. To submit an amendment, please complete an Amendment Request Document and submit it along with any updated documents affected by the changes via a new package in IRBNet. Any unanticipated problems involving risk and any occurrence of serious harm to subjects and others shall be reported by completing this Event Report Form and sending it promptly to the IRB within 3 business days.

Your study has been approved for a period of 12 months as of 09/22/2017. If you would like your project to continue for more than one year, you will be required to provide a completed Status Report and other continuing review documentation to the UNF IRB prior to **08/22/2018.** An extension will be necessary if your study will be continuing past the 1-year anniversary of the approval date. *We ask that you submit your status report and other continuing review information 30 days before the expiration date as noted above to allow time for review and processing.* When you are ready to close your project, please complete a Closing Report Form. Please note that it will be necessary to create a new package in IRBNet in order to submit amendments, status

reports, or closing reports in the future. All applicable records relating to this research shall be retained for at least 3 years after completion of the research.

CITI Training for this Project:

Name	CITI Expiration Date
Dr. Chuan	03/28/2019
Mr. Blair	01/16/2020
Mr. Southard	01/04/2020

CITI Course Completion Reports are valid for 3 years. The CITI training for renewal will become available 90 days before the current CITI training expires. Please renew your CITI training when necessary and ensure that all key personnel maintain current CITI training. Individuals can access CITI by following this link: http://www.citiprogram.org/. Should you have questions regarding your project or any other IRB issues, please contact the research integrity unit of the Office of Research and Sponsored Programs by emailing IRB@unf.edu or calling (904) 620-2455.

This letter has been electronically signed in accordance with all applicable regulations, and a copy is retained within UNF's records. All records shall be accessible for inspection and copying by authorized representatives of the department or agency at reasonable times and in a reasonable manner. A copy of this approval may also be sent to the dean and/or chair of your department.

UNF IRB Number: 967107-3
Approval Date: 09-22-2017
Expiration Date: 09-22-2018
Processed on behalf of UNF's IRB

VITA

Spence Southard is employed as a Software Engineer in Jacksonville, FL, and is a graduate student at the University of North Florida. Outside of work he is an avid application developer simultaneously working on many different projects, with a core focus around Android applications and RESTful APIs. In doing so, he works extensively with Python, Java, and JavaScript. His academic interests lay in artificial intelligence and machine learning.