# The 20th Winona Computer Science Undergraduate Research Symposium

May 1, 2019

10:00am to 12:30pm

Winona State University
Winona, MN

Sponsored by the Department of Computer Science

at Winona State University

WINONA
STATE UNIVERSITY
Computer Science Department
http://cs.winona.edu

# Table of Contents

# The Comparison of Speed and Memory Allocation Between Jython and Python

Garvey John
Department of Computer Science
Winona State University
Winona, MN 55987
(612) 910-5095

Gjohn17@winona.edu

## ABSTRACT

In the industry, there are many different programming languages that are used. Each one used for different reasons and accomplishes different task differently. Two popular programming languages are Python and Java. Each language has code that is broken down into a language that can be read by their own respected virtual machine to complete a task that is given. In this research, Jython was tested against its counter part Python. The tests that were conducted were to see if Jython could produce results as well or even better than Python could. In every trial, Python was able to complete the task faster and use less memory while doing this, which leads us to reject our original hypothesis.

## General Terms

Algorithms, Performance, Reliability, Experimentation, Languages.

## Keywords

Jython, Python, Java, Time, Memory

## 1. INTRODUCTION

One of the main purposes of a programming language is to give humans a way to communicate with computers. With this communication, humans are able to create advance algorithms for computer to compute. These computations can help humans solve advance problems, organize data, recognize patterns, and many more possibilities.

Programming languages have shaped our society in many ways. The same way that natural languages have shaped the way society thinks, programming languages have impacted the way that programmers think about a problem [6]. From first generation languages like FORTRAN, where the main objective of using these languages were to compute advance mathematical problems, to our current high level programming languages like Java that allow us to complete much more advance task than FORTRAN.

Like natural languages, there are many different programming languages. Two of the more popular languages are Java and Python. According to Tiobe Index, Java is the most popular programing language amongst all other programming languages and Python being the third most popular language [7].

Java revolutionized the programming language industry when it came out in 1995. The basis of Java was to allow any program to be “write once, run anywhere.” Java would turn code into bytecode which was interpreted on a its own virtual machine. The virtual machine would translate the code into code that could be understood by the host computer [4].

Python came out in 1991, and also changed the way people thought of programming. Some of the goals of Python were easy and intuitive language that’s able to compete with the other competitors in the industry, used for everyday task, and open source [9]. Like Java, Python interprets its code on a virtual machine.

With many powerful programming languages out there, Java’s virtual machine decided to add them. Java’s virtual machine has an implementation of many popular programming languages, but this research focuses on the implementation of Python within the virtual machine. This implementation is called Jython.

Jython, previously known as JPython, is a successful language that has been used by many successful businesses including IBM Websphere, Apache PIG, Robot Framwork, and many more [8]. One thing that sets Jython apart from Python is that it is able to run code in any environment as long as it supports a JVM [3].

The main goal of this research is to see if Jython is able to produce faster results than Python and use less memory within completing its task.

## 2. BACKGROUND KNOWLEDGE

As far as research for Jython compared to Python, there is limited research that has been conducted. Even though there’s a lot of research on Java and Python, Jython has limited research to its counterpart that it was created for. The goal of this research is to test if Jython is faster and uses less memory to run programs than Python.

This research was conducted on:

**Table 2.1**

| | |
|---|---|
| Model Name | MacBook Pro |
| Processor Name | Intel Core i5 |
| Processor Speed | 2.3 GHz |
| Number of Processors | 1 |
| Total Number of Cores | 2 |
| Memory | 8 GB |
| System Version | macOS 10.14.6 (18G103) |

| | |
|---|---|
| Jython Version | 2.7.1 |
| Python Version | 2.7.10 |

## 3. METHODOLOGY

To be able to test Jython and Python under the conditions of time efficiency and memory usage, three programs were created, and two programs ran under the same conditions while the general user interface program was tested under one less condition. These programs were made to search a file, organize a numerical matrix, and create a general user interface.

Each one of these programs were first written in Python code and then transferred directly over into Jython for all programs except for the general user interface program. This created three unique conditions for each program to be tested except for the general user interface which would only be tested under two conditions. The three conditions were to be ran in Python, Jython with Python code, and Jython using Java classes. The graphical user interface only tested Python code and Jython using Java classes.

### 3.1 Search

One important aspect of all programming languages is the ability to search a file. This is because files are used to hold important data and possibly organize it. It is possible to hold all information within the program itself, but once the program is done running, the data is lost. Because of this, we store our information within some kind of file which can be read and written to.

In this program, a file was searched to find if a city was within the file. This file had a list of famous cities within the United States [1]. This program would open a file and read in each value one by one. Opening a file allowed access to a program to a file. If the value was found it would close the file and report back that the city was found and at what line in the file. For example, if the city Minneapolis was put into the program, it would report back “Minneapolis was found at position 213.” If a value like Winona was put into the program, it would report back “Winona was not found in the file.”

In this program, for Python, nothing was imported from the Python library. While testing Jython using Python code, nothing was changed, and everything matched the same exact way as Python code did. While testing the Jython code using Java classes, the program implemented the Scanner class. The scanner class was used to open a file and return each line within the file. Python did the same thing, but just used something that was implemented into its own code respectively.

### 3.2 Matrix

Another important aspect of computing is being able to organize a mass amount of data and being able to manage it efficiently. In a programming language, we are able to create something called an array. These are a list of values that are similar. With these arrays, it is possible to create a two-dimensional array which is an array of arrays. Since the program is using numerical values in this two-dimensional array, this can be called a matrix and handled like one.

In this program, the objective is to organize a two-dimensional array and print them out from smallest to largest. A hundred by hundred array is created and reads in a hundred thousand values from a file.

Once read, the values are then organized from smallest to largest by using a modified selection sort to fit a two-dimensional array instead of the normal one-dimensional array. A selection sort goes into an array and starts at the first value. It then searches the whole array starting from the current position and finds the smallest value then switches that value with the starting point value. Once it switches the value, it goes to the next value and does this again until it gets to the end of the array and there are no more values to search for. This forces the smallest values to the beginning and the largest values to the end. This sort was modified to do the same thing except it would do it in a two-dimensional array that had one hundred thousand values.

Similar to the search program, Python code did not import anything from the Python library. While testing the Jython code using Python code, nothing was changed, and no Python libraries were imported. While running Jython with Java classes, we imported the Scanner class to read in the values into the matrix.

### 3.3 Graphical User Interface (GUI)

A graphical user interface is a way to convey information to the user while it can also have interaction with the user [2]. A good example of a graphical user interface is an internet browser or even using Microsoft Word. A GUI is used for many different businesses and are an important aspect of programming.

In this program, it creates a simple GUI that displays Jython and Python onto the display. The name of the GUI is called “GUI.” The size set to 250 x 250 and set to end program once the user exits the program. The Display is similar in both programs. The GUI created in Jython is Figure 3.1 and Python is Figure 3.2.

The GUI program is only tested under the conditions of Python code and Jython using Java code. Python’s GUI program used the Tkinter class. This class is a template for creating a GUI for a program to use and edit. Jython is not able to use the class TKinter and because of this, the study isn’t able to do a comparison of Jython using only Python code. Instead the program for Jython will have to implement the Javax.swing package. This does the same thing, but instead uses Java packages. These two programs were written similarly to make sure the results were not favored in one language compared to the others.

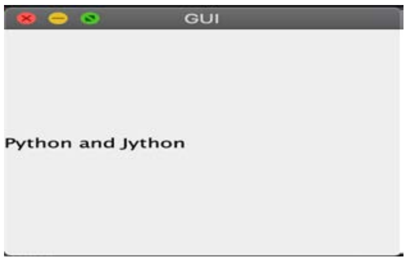


**Figure 3.1**

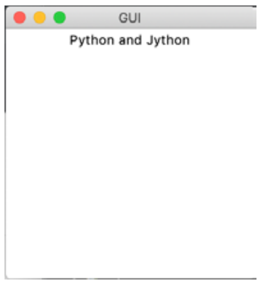


Figure 3.2

## 4. RESULTS AND ANALYSIS

Each program took the average of running ten times while only that program was running. The results were tested within each program so that there would be no human error while producing the results. In each program, the time and memory usage were measured by some kind of import within Python or Jython.

For all programs tested in either Python or Jython, the time class was used to measure how fast the program would finish. For memory, all Python code used the resources class to measure how much memory was consumed during the process. Within the resources class, it can measure the peak amount of memory that was taken up through the process. Jython was not able to use the resource class, so instead it imported the Runtime class. This is a Java class that has the capability of seeing how much memory is allocated in a Java program.

### 4.1 Time Trials

The time it took a program to complete was measured in seconds. In each program, Python out performed Jython programs by a large margin. Since the matrix program took so long to run with such a big array to sort, it caused the time to be a bit closer, but the Python code still ran significantly faster than any Jython program. One thing to note is that the next fastest program was Jython using the direct Python code. This could be because the Python code didn't import any classes like the Jython using Java classes did. A counter argument to this point would be that the Python GUI application imported a class and still out performed the Jython program.

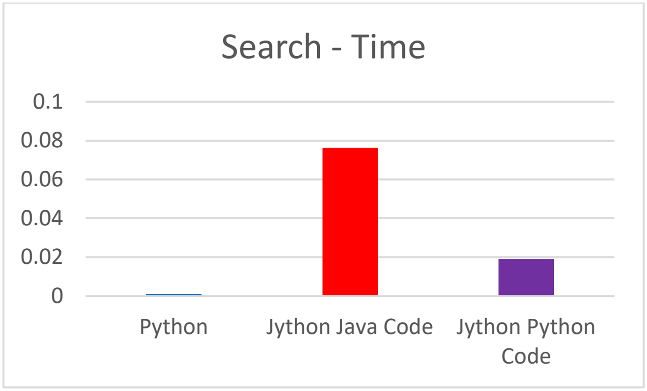


Figure 4.1

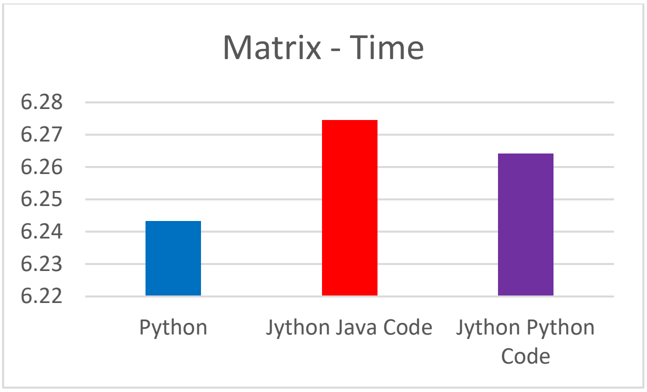


Figure 4.2

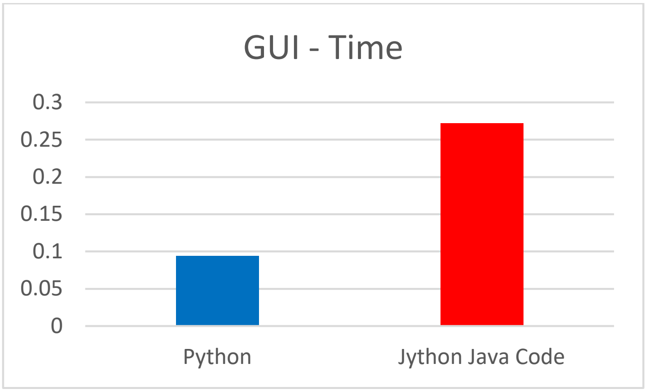


Figure 4.3

### 4.2 Memory Usage

The total amount of memory used was measured in Megabytes. Similar to the time trials, Python outperformed the Jython program by a large margin. Although the Python code seemed to outperform both Jython codes, Jython implementing Java classes was only a little bit larger than Jython running Python code. This is an improvement compared to the time trials. The closest test was comparing the GUI programs. Even though Python used less memory than Jython, it wasn't as big of a margin as the previous programs.

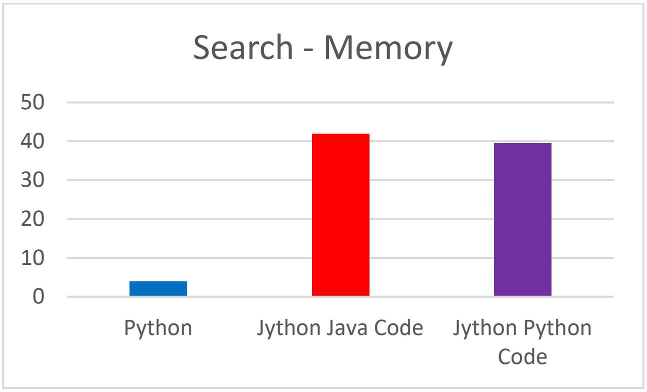


Figure 4.4

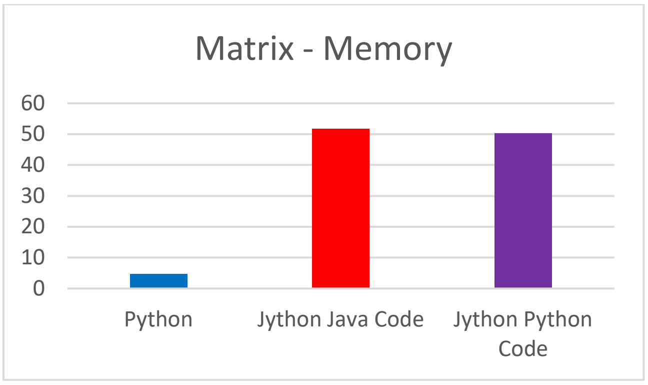


**Figure 4.5**

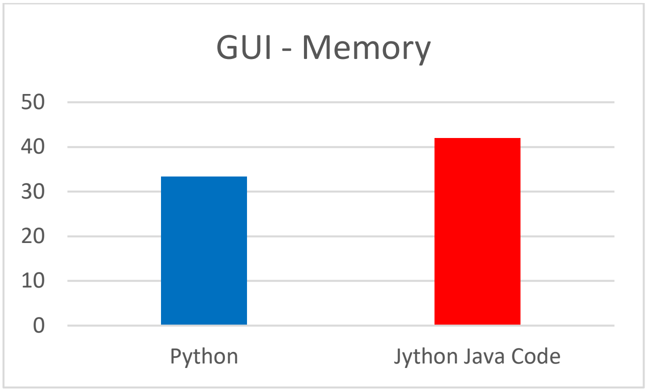


**Figure 4.6**

## 5. CONCLUSION

The main goal of this research was to see if Jython was able to produce results that were faster and used less memory than Python. From the results, we can see that the hypothesis is rejected. Python was able to beat every Jython program in speed and memory usage. In most cases, it was clear to see that Python could outperform Jython ten times faster and use a tenth of the memory to do so.

### 5.1 Continuing Research

For this research, smaller programs were tested instead of a large scale project that a business could use. One idea for testing a more realistic algorithm is to have a program that takes in a mass amount of data and have user input and output. In this program, it would be ideal for the program to be constantly running. With the program being more advanced and complex, it will allow observations on the algorithms that take more time and takes up more memory. With the program constantly running, the time and memory could be tested for different real life situations. This is one of many different ways to test the two languages in more experiments, but could produce different results than what was obrtained in this research.

## 6. REFERENCES

[1] "city_names.txt" Clint Bettiga https://gist.github.com/norcal82/4accc0d968444859b408 Accessed: March 1, 2020

[2] "GUI" Computer Hope https://www.computerhope.com/jargon/g/gui.htm Accessed: April 22, 2020

[3] "Intro to Jython, Part 1: Java programming made easier" Barry Feigenbaum https://www.ibm.com/developerworks/java/tutorials/j-jython1/j-jython1.html Accessed: April 26, 2020

[4] "Java" The Editors of Encyclopaedia Britannica https://www.britannica.com/technology/Java-computer-programming-language Accessed: April 22, 2020

[5] "Know your history — Java's rise to popularity" JAXenter Editorial Team https://jaxenter.com/java-know-your-history-149484.html Accessed: April 22, 2020

[6] Latifa BEN ARFA RABAI1, Barry COHEN2, Ali MILI "Programming Language Use in US Academia and Industry" Informatics in Education, 2015, Vol. 14, No. 2, 143–160

[7] "TIOBE Index for April 2020" Tiobe https://www.tiobe.com/tiobe-index/ Accessed: April 22, 2020

[8] "What is Jython?" Jython https://www.jython.org/ Accessed: April 21, 2020

[9] "What is Python?" Python Institute https://pythoninstitute.org/what-is-python/ Accessed: April 22, 2020

# Realtime Multi-Camera Virtual Reality Video Streaming

Ryan Rowe
Winona State University
Computer Science
rrowe222@gmail.com

## Abstract

Realtime Virtual Reality (VR) streaming has multiple limitations, most predominantly bandwidth of the communication method between a headset viewing device and the transmission device. Limiting the number of video streams with wider angle lenses and a higher resolution per camera is the best way to limit the amount unnecessary data streaming over the network connection. Selective video compression can also improve the performance of the stream, however on cheap consumer hardware, overuse of compression can overtax the transmission device.

## Categories and Subject descriptions

[Human Computer Interaction], [Networking]. Video compression standards, interaction with a virtualized environment. *– data compression, data streaming, virtual reality interaction, video editing*

## General Terms

Measurement, Experimentation, Human Factors, Performance,

## Keywords

VR, Virtual Reality, Streaming, Raspberry Pi, Index, Video Compression, Virtual Environment, Projection mapping

## 1. Introduction

These proceedings are an overview of history of VR and how using consumer level hardware, a space can be projected from a transmission device to a virtual reality headset for viewing.

The first Virtual Reality human computer interface was invented in 1990 by NASA [1] with the "Virtual Interface Environment Workstation" (VIEW) The goal of which was to enable the user to interact with a 3D computer generated environment with their own hands acting as the input method. NASA had planned on using this to control distant rovers, however, the technology of the mid 1980s and early 1990s was not advanced enough to support such a task.

Modern VR hardware and software has made significant advancements in the comfort of using such interfaces. Motion sickness, headaches, and eye strain were significant hurdles to overcome in developing VR technology. Accurately tracking the user's eyes and head significantly reduced the discomforts of using VR interfaces for most users.

## 2. Background

### 2.1

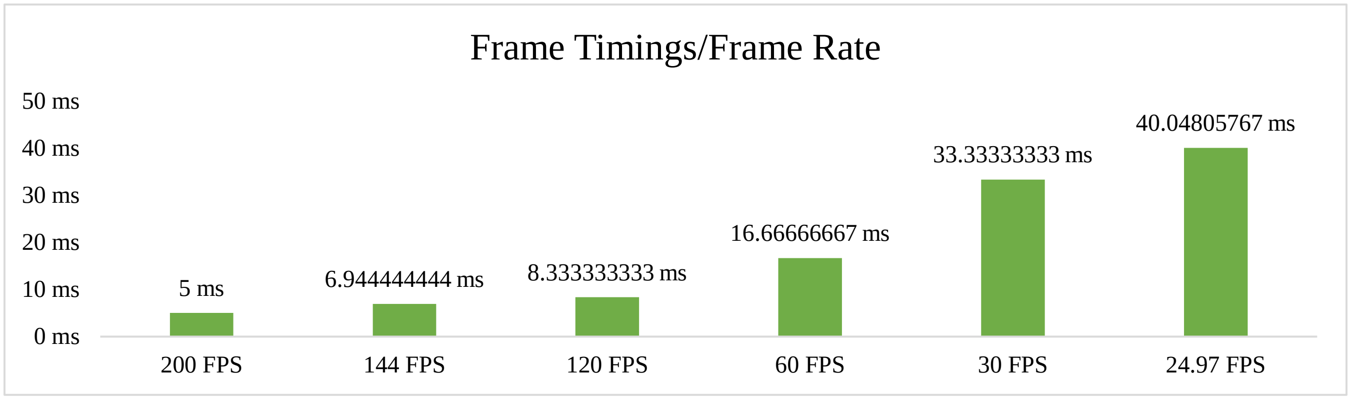
Viewing a completely virtual environment to a headset has certain advantages over viewing a real-world video stream. Since the developer of a VR application has total control over the environment, the developer can take steps to reduce physical discomforts and can make affordances to the user about the environment in which they are interacting. The virtual environment can be scaled depending on the user's height and viewing position. Virtual camera angles are controlled by the user, and the user will put themselves in the best position for viewing the content of the environment. Viewing a real-time real-world environment does not have this luxury. User is entirely at the mercy of the placement of the physical cameras and transmission equipment. The physical location of the headset is also a factor, since transmission over the internet requires a considerable amount of bandwidth that may be limited somewhere along the way.

### 2.2

Localized streaming of Virtual Reality video has the advantage of a high-speed Local Area Network (LAN) or a point to point network (Ad-Hoc). The position of the transmission device and the headset is limited in distance but provides a much higher bandwidth for a higher quality stream.

Low bandwidth connections are particularly unwanted, since video hitching and frame latency can cause significant disorientation and motion sickness in the user. Using a lower resolution video stream over the internet would create another problem in the form of visibility. VR headsets tend to have a higher resolution than standard monitors because the screens are so close to the users eyes, but even then, the so called "Screen Door Effect" [2] more formally known as "Fixed Noise Pattern" is a visual artifact caused by the relatively low pixel density of a screen. Although a VR headset screen is typically higher, the distance from the user's eye is so small, that significantly higher density screens are needed to reduce the eye strain.

**Table 1 (Frame Timing Data)**



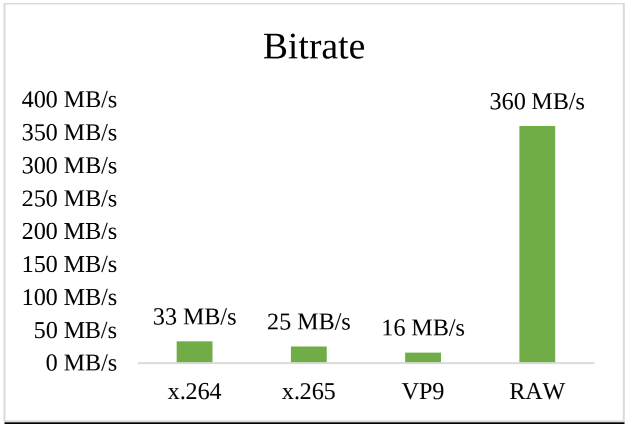
Because of this issue, low quality video streams will be stretched across the low-density screen and visual quality issues as well as compression artifacts will be significantly more distracting and even render the video stream incompressible due to how close the user's eye is to the screen.

## 3. Video Compression

### 3.1

Data compression is the primary way large chunks of data can be sent over the internet quickly. Lossy compression, or where the original data is compressed with a loss of information. Most video streamed over the internet from services like Twitch [3] and YouTube [4] use a lossy video coded called x2.64. x2.64 is an older codec and has since been improved upon with codecs such as x2.65, VP9/10, and AV1.

**Figure 1 (video bitrates based on 1080p 60fps)**



Lossless video compression doesn't technically exist. Instead different video algorithms are rated on their loss ratios. Which is just the RAW video data rate/ compressed data rate. This is measured in bits/second, or bitrate. The higher the bitrate, the more data is preserved in the form of color reproduction, edge preservation, and even framerate.

### 3.2

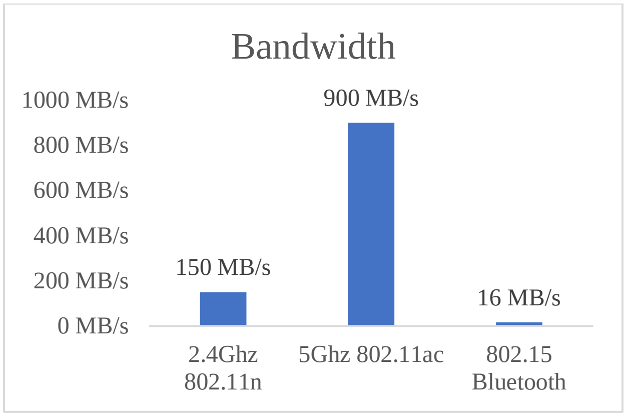
Video streamed over the internet is usually 30 or 60 frames per second. Meaning a new picture is being displayed once every 33.33ms or 16.67ms respectively. For virtual reality, the more frames being displayed to the user, the smoother and less fatiguing the experience is. As such the minimum video framerate for VR is 60, but preferably 120 or 144 frames per second. With new frames being displayed every 8.3ms and 6.9ms respectively. Figure 1 has a visual representation of different frame timings.

## 4. Data Connections

### 4.1

Local wireless connections for consumer hardware are some variant of the IEEE 802.11[i] standard. Currently the two most common 802.11 network types are 802.11n at 2.4Ghz and 802.11ac at 5Ghz. These will serve as the communications standards for the Ad-Hoc networks that will be tested. 802.15 Bluetooth will also be tested, as Bluetooth[ii] is often used as a local communication standard for low bandwidth requirement data transmission.

**Figure 2 (Ad-Hoc Bandwidth)**



|  | Required Bandwidth for 8 Camera (Mb/s) | Required Bandwidth for 2 cameras | Available bandwidth | Bandwidth Δ for 8 | Bandwidth Δ for 2 |
|---|---|---|---|---|---|
| RAW + 5GHz | 2880 MB /s | 720 MB /s | 900 MB /s | -1980 MB /s | 180 MB /s |
| RAW + 2.4GHz | 2880 MB /s | 720 MB /s | 150 MB /s | -2730 MB /s | -570 MB /s |
| RAW + Bluetooth | 2880 MB /s | 720 MB /s | 16 MB /s | -2864 MB /s | -704 MB /s |
| H.264 + 5GHz | 264 MB /s | 66 MB /s | 900 MB /s | 636 MB /s | 834 MB /s |
| H.264 + 2.4GHz | 264 MB /s | 66 MB /s | 150 MB /s | -114 MB /s | 84 MB /s |
| H.264 + Bluetooth | 264 MB /s | 66 MB /s | 16 MB /s | -248 MB /s | -50 MB /s |
| VP9 + 5GHz | 128 MB /s | 32 MB /s | 900 MB /s | 772 MB /s | 868 MB /s |
| VP9 + 2.4GHz | 128 MB /s | 32 MB /s | 150 MB /s | 22 MB /s | 118 MB /s |
| VP9 + Bluetooth | 128 MB /s | 32 MB /s | 16 MB /s | -112 MB /s | -16 MB /s |

### 4.2
802.11ac [5] is the fastest connection that will be testing, clocking in at nearly 1Gb/s. While 802.11n is an older standard it is still widely used as legacy devices do not support faster connections and some newer low-end consumer hardware still doesn't support the newer 802.11 standard. Bluetooth is a relatively low speed connection but is enough for the more compressed video standards.

## 5. Streaming hardware

### 5.1
The transmission hardware being tested is a Raspberry Pi 3 B+ [6]. With a 1.4GHz quad core processor, the Raspberry Pi 3 B+ has enough processing power to perform most compression tasks on the fly and is the most widely available small form factor computing device.

### 5.2
Two camera methods were used to achieve 360-degree video. An eight-camera setup, capturing 45-degrees of space a piece, and a two-camera setup with a spherical mirror used to capture 180-degrees of space. Each camera was capturing at 1080p 60 frames per second.

**Table 2 (Streaming Data)**

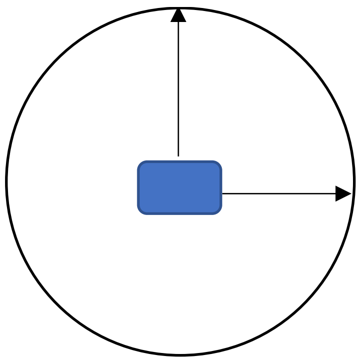
Every camera's footage was captured and then re-projected onto the inside of a three-dimensional sphere with the Virtual Reality headset at the center. The center of the sphere was always parented to the headset, so even if the user moved in space, they could not get closer to any surface of the sphere.

### 5.3
A Valve Index [7] was used as the Virtual Reality headset. The Valve Index was used over a Google Cardboard to eliminate any processing bottlenecks on the viewing end of the stream. The Valve Index also has a higher DPI and refresh rate, which helped reduce the screen door effect and motion issues.

## 6. Streaming Software

### 6.1
Video from the Raspberry Pi transmission device was fed into a custom spherical projection mapping [8] software to project the camera view onto the inside of a three-dimensional sphere. This was done so all video streams were equidistant from the virtual viewing device.

**Figure 3 (Spherical Projection Map)**



### 6.2
Dynamic resolution scaling can be used to regulate the workload of the compression algorithm. Passing the video into a game engine such as Unity or Unreal 4, the developer can set a frame time budget for the engine, so an individual frame is down sampled to achieve a better framerate and deliver a smoother viewer experience at the cost of frame quality and resolution.

## 7. Experiment

### 7.1

Nine total tests were performed, each test was done with one compression method and one network type. Each test was done over a 30 second interval. No motion was applied to the cameras or transmission device itself. The scene observed was a room with the window open. So, there was some motion from objects in the room moving.

## 8. Analysis

Bandwidth was the largest obstacle to overcome. Overtaxing the network connection with data caused frame drops and stuttering on the headset.

### 8.1

RAW video was used as a control with the expectation that while it would produce the highest quality video, it would not be feasible to consistently stream uninterrupted video across any of the tested network connections. Utilizing eight cameras produced the highest quality images, but the lowest quality streaming experience. Frequent frame drops and stuttering were common. Utilizing two cameras worked significantly better due to the decreased bandwidth requirements of two video streams over eight.

### 8.2

H.264 compression was used since it is the standard codec used by mainstream online streaming platforms. Performance with eight cameras and two cameras was generally stable on the 5GHz 802.11ac connection, with minimal frame drops and stuttering. 2.4GHz performance struggled with eight cameras, again producing noticeable frame drops and stuttering.

Had the video had more motion, H.264 might have struggled more, since H.264 strips away what it considers “unnecessary” data and utilizes previous frame data to construct a frame. This saves data but produces noticeable visual artifacting if there is significant motion in small details, even at high frame rates this is noticeable if the scene is moving enough.

### 8.3

VP9 is a Google developed standard mainly target at mobile platforms. And it is easy to see why. VP9 had the lowest bandwidth overhead with a measly 128MB/s for 8 cameras. VP9 performed well on both 2.4GHz and 5GHz connections.

However, the reason VP9 performs so well is that the video is heavily compressed, on mobile devices this is not noticeable, and since the compression workload is being performed on a server, the video stream to a mobile device is smooth. Using a Raspberry Pi to compress the video did have a noticeable impact on the transmission device. Although the video was visually smooth on both 2.4GHz and 5GHz, there was a noticeable delay in the stream. When monitoring the Raspberry Pi performance metrics, the CPU and memory utilization was often at 100%.

### 8.4

Out of all the network types, 802.11ac performed the best. 802.11n consistently fell in the middle and Bluetooth was unable to keep up the data being streamed, even with VP9 compression on two cameras, making it unviable for this type of video streaming.

While 802.11ac performed the best, because of the 5GHz signal, there was a limitation on the distance and number of obstructions in between the transmitter and the headset. This did not present as an issue in the experiment but is a potential limitation for this type of streaming in other conditions and locations.

## 9. Conclusion

Realtime Virtual Reality video streaming is possible locally with specific equipment and affordances, such as a high-speed point to point connection, a powerful transmission and compression device and minimal obstruction. However, practically, this type of streaming has many limitations. Using the two-camera setup had less bandwidth overhead but produced a stretched image that still had the cameras visible to the user. The surface of the mirror was also an issue, since any imperfections in the reflective coating would distort the light being captured. Defining “Down” was also a challenge, with the eight-camera setup, down was predefined with the arrangement of cameras, but if there was any movement of the cameras though space, “Down” might need to be shifted to reduce motion sickness issues for the user.

## 10. Further Work

To better this experimental setup, a more powerful transmission device could be used to more effectively compress the video streams. This would allow more aggressive compression methods to be used to then stream less data over the network connection and make slower connections more viable. More video compression algorithms could also be tested. In this experiment, H.264, and VP9 were used because they were the most popular, however, VP10 is a newer standard developed by Google, but not widely available. H.265 HEVC is a new codec continuing from H.264, but again, is not in general use for streaming.

## 11. References

[1] NASA, "The Virtual Interface Environment Workstation (VIEW), 1990," NASA, [Online]. Available: The Virtual Interface Environment Workstation (VIEW), 1990.

[2] M. Rouse, "screen door effect," whatis.com, 2017. [Online]. Available: https://whatis.techtarget.com/definition/screen-door-effect.

[3] "Twitch.tv encoding standards," Twitch.TV, [Online]. Available: https://stream.twitch.tv/encoding/.

[4] Google, "Youtube live streaming standards," Google, [Online]. Available: https://support.google.com/youtube/answer/2853702?hl=en.

[5] IEEE, "802.11-2016 - IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Sp," 07 12 2016. [Online]. Available: https://standards.ieee.org/standard/802_11-2016.html.

[6] "Raspberry Pi B+," [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/.

[7] Valve, "Valve Index," Valve, [Online]. Available: https://www.valvesoftware.com/en/index/headset.

[8] M. Calabretta, "Spherical Map Projections," 15 12 1992. [Online]. Available: https://library.nrao.edu/public/memos/aips/memos/AIPSIM_107.pdf. [Accessed 1 4 2020].

# Comparing Image Classification with Feature Extraction

Barett Jones
Winona State University
Winona, Minnesota
bgjones15@winona.edu

## ABSTRACT

Image classification in today's world is extremely important and advancements are constantly being made to improve performance and efficiency. There are many different mediums that are taking advantage of image classification. An example would be medical fields. Image classification is used to analyze X-RAY images to attempt to identify cancer spots. There are many different factors that come into play with classifying images. Training these models can be very time consuming and there can be techniques implemented to speed up this process. One technique would be reducing details within the picture and removing redundancies. Research has been done to test feature extraction within the image prior to training convolutional neural networks. Features considered were: canny edge, histogram of oriented gradients, and shape index. Results have shown that feature extraction can provide faster training times, but does not conclusively show an increase in accuracy. The control group provided the highest accuracy without the use of any feature extraction methods.

## General Terms

Measurement, Documentation, Performance, Reliability, Experimentation.

## Keywords

Machine Learning, CNN, Neural Network, Feature Extraction, Image Classification, Canny Edge, Shape Index, Histogram of Oriented Gradients.

## 1. INTRODUCTION

### 1.1 CNN

Convolutional Neural Networks (CNN) are a class of artificial neural networks (ANN) that typically are applied to deep image learning [2]. There are many different fields use image classification. Another field that uses image classification are hospitals. There are many advanced CNN that are used to help detect early signs of cancer [3]. Being able to detect these cancers early is essential to receive treatments that may potentially save the patients' life.

### 1.2 Feature Extraction

Something that will be considered in future work is feature selection and deriving features from different kinds of datasets. An example of this would be feature selection on an animals' dataset. Features that could be extracted would be fur and characteristics of legs or eyes. This research will focus on the Fruits 360 dataset. This dataset is provided by Kaggle and contains thousands of images of fruits with labels [4]. A subset of these fruits will be used for simplicity and getting a better understanding of the extractions affects. Feature extraction of an image is transforming the image, typically detail reductions and removal, to bring out specific traits of an image. This research considered three different kinds of features:

- Edges: Canny Edge
- Shape: Shape Index
- Direction: Histogram of Oriented Gradients

The features were extracted before training all on the same model. The model used was a simple convolutional neural network consisting of just dense layers [1]. These features were compared to the original image. The accuracy, training time, and loss metric at the end of training were all measured and recorded.

## 2. BACKGROUND RESEARCH

There is much research out there regarding feature extraction being used for large datasets to improve performance while also keeping an adequate accuracy. Most of these findings, however, do not include a comparison of feature extractions prior to the training. Also, a lot of the papers use complex feature extraction techniques that incorporate many different features [9].

For large datasets feature extraction has been shown to greatly improve performance time. It also has created a reduction in memory usage due to less feature space and processing needs.

## 3. METHODOLOGY

### 3.1 Data

#### *3.1.1 Collection*

Data was collected from the fruits 360 dataset provided on Kaggle [4]. This consists of 120 fruits and vegetables and a total collection of 82213 images. The images are 100x100 pixels, which was left the same for uniformity. The dataset was previously split into a testing and training set and labeled is its corresponding directory.

#### *3.1.2 Cleaning*

Most of the cleaning process was just sub-setting the data. From the 120 fruits, 10 fruits were selected based on individual characteristics that made them more unique than their counterparts. The fruits selected were: banana, strawberry, kiwi, pear, apple, raspberry, blueberry, tomato, and pineapple.

## 3.2 Feature Extraction

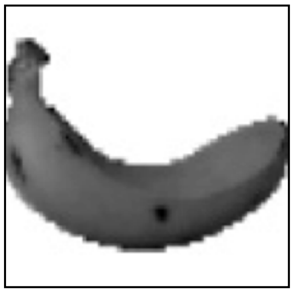


**Figure 1. Banana example without feature extraction**

### 3.2.1 *Edge*

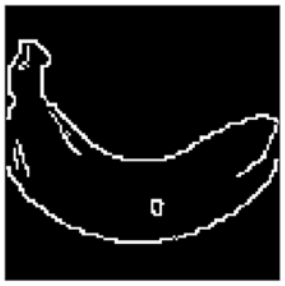


**Figure 2. Canny edge banana**

Canny edge detection was used to extract the edge features from the images. This is an algorithm designed to reduce noise from the image and only capture the edges of the main points of the object [6]. There are four main stages of this algorithm.

- Noise reduction
- Pinpointing gradient edges
- Edge finding
- Thresholding

Noise reduction is the first process. Due to canny edge being very susceptible to noise a gaussian filter is used to help reduce that. Gaussian filters blend together pixels so there is a blurriness feel to the image.

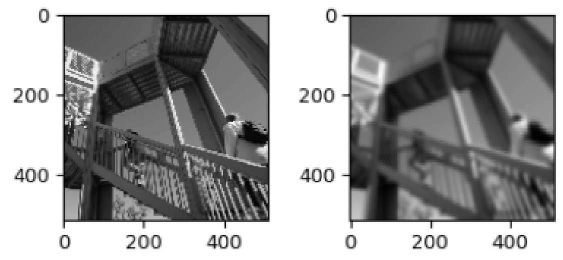


**Figure 3. Gaussian filter example**

After noise reduction edge gradients are calculated. This is calculated both horizontally and vertically. These edge gradients will be used to find the edges [7].

To find the edges, each pixel and its neighbors are compared to check the direction of their edge gradients. If these gradients are not the same direction it is considered an edge.

Finally, thresholding of the edges takes place. This will compare the intensities of each of the previous edges and create a threshold value to consider it an edge or not. If any edge is connected to another in range edge and is also above the threshold or within it, it will also be considered an edge.

### 3.2.2 *Shape*

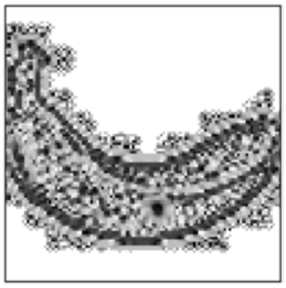


**Figure 4. Shape index banana**

Shape index is used for intensifying certain shapes within an image. The desired shape that was set for these images were circular. This shape was selected because most of the fruits were circular shape. Shape index may not completely be able to capture the shape of the fruit, but it is used for capturing circular aspects of the image and create a 3D visualization of the fruit. More experimenting could be done with different shapes.

### 3.2.3 *Direction*

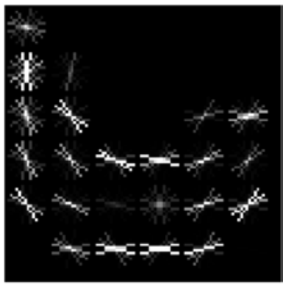


**Figure 5. Histogram of oriented Gradients banana**

Histogram of oriented gradients (HOG) will be used to calculate direction within the images. It is like the canny edge algorithm in calculating the edges, but it also calculates the direction of the edges and textures [6]. There are also many steps in the HOG algorithm.

- Calculating gradients
- Calculating gradients in cells
- Normalization
- Visualizing feature vector

Like in canny edge the gradients are calculated the same both vertically and horizontally. Each pixel will have a gradient value that will be compared to surrounding pixels.

Then the image is broken up into different cells where each individual cell will have a gradient calculated. For this research each 100x100 image was divided into a 6x6 grid. Within each of the cells a direction is formed by comparing local maxima gradient values then moving outward to minima values.

The image is then normalized to create a more accurate transition between the gradients. This will reduce the chance of a drastic change in direction throughout the image, which is what we want.

After each of the individual cells have calculated their gradients, visualization is used to see the direction of each of the cells. Visualization is aided by code by showing shapes with elongated shapes.

## 3.3 Training

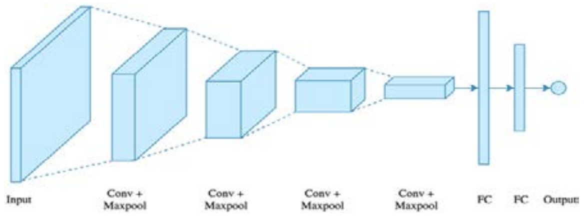


**Figure 6. Convolutional Neural Network**

A convolutional neural network was used to train the dataset after the features were extracted. To try and reduce any aid from the network only dense layers were used. Dense layers are layers that reduce the feature space and create a fully connected network. This would allow the features to be trained without a bias in the training process coming from max pooling layers. Max pooling layers would reduce the dimensions of the layer, while increasing the feature space [2, 10].

Tensorflow and Keras was used to help aid the training process [8]. These are both python libraries that are used for creating machine learning models and training. Tensorflow handled most of the training aspects, while Keras was used for building the sequential model [1, 10].

Each feature dataset was put through the same model with the same number of epochs. 20 epochs were chosen as that is the default value for Tensorflow models. They were in batches of 60 and the time was recorded by the software to start recording when the model began and ended its training process. Early stopping was implemented, but the only model that stopped early was the control group. Because this did not correctly measure the amount of time it would take to run through each epoch, early stopping was not considered when measuring time. The accuracy was also measured after the model was done training. It was evaluated using the testing set, so no images were used to validate the model that also went through training. Also, the validation loss metric was measured at the end of training. This metric measures how well the model is. Typically, this is the measurement that you try to optimize. The higher the model metric the worse the model did training. Most of the optimization for this metric would be towards changing the model rather than the dataset, so it is not that important for this research [5].

## 4. RESULTS ANALYSIS

### 4.1 Accuracy

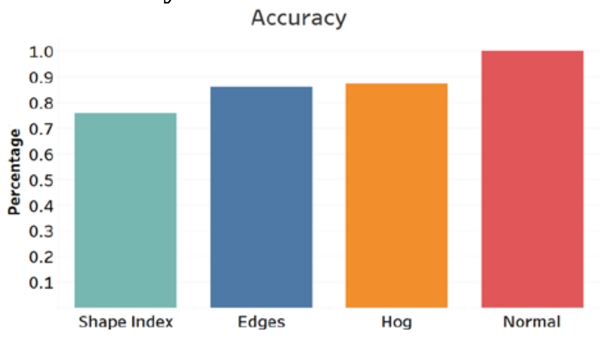


**Figure 7. Accuracy results**

For accuracy, the control group provided the highest accuracy, followed by edge and HOG. This was as expected as the control group kept all its features and had the most amount of pure details within the image.

HOG and canny edge produced a pretty similar image after feature extraction so it makes sense that they both provided close to the same accuracy. They still provided a very high accuracy too, so it might be useful to use these features because it contains a lot less details within the image and would not need as much memory if optimized.

Shape index came in at the lowest accuracy of the models tested. It is believed that the shape index brought a lot of extra noise within the image, that made it less accurate. More experimenting would need to be done to test whether the shape index parameters could be modified to create better results.

**Table 1. Accuracy Recordings**

| Trial | Validation Accuracy | Test Accuracy |
|---|---|---|
| Shape Index | 0.8479 | 0.7575 |
| Canny Edge | 0.9563 | 0.86 |
| HOG | 0.9375 | 0.8737 |
| Normal | 0.9979 | 1 |

### 4.2 Loss

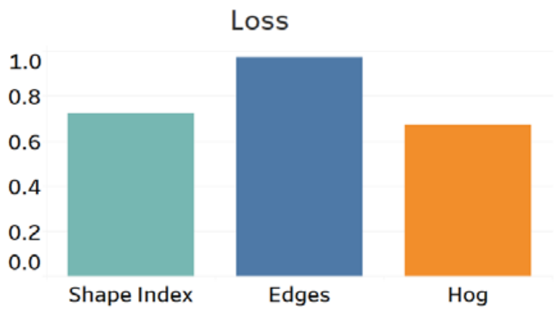


**Figure 8. Loss results**

As mentioned previously, loss is not the most important for this research. It was mostly included to show that with model optimization, feature selection. As shown, canny edge has the highest loss. This means that the model could be optimized more to potentially increasing the performance and accuracy of the images. The control group had a loss of < *0.01*. Because the loss is so low the model is already close to optimized for them.

**Table 2. Loss Recordings**

| Trial | Validation Loss | Test Loss |
|---|---|---|
| Shape Index | 0.4929 | 7.23E-01 |
| Canny Edge | 0.154 | 0.9716 |
| HOG | 0.4121 | 0.671 |
| Normal | 0.0067 | 1.02E-04 |

## 4.3 Time

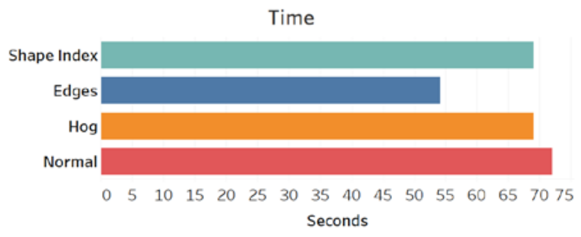


**Figure 9. Time results**

For the time, canny edge took the least amount of time to train. The other tests were very similar to the control group showing that they do not actually make that great of an impact for speeding up the training process. However, there was a significant reducing in training using the canny edge there is potential for dramatically increasing time complexity for larger datasets.

**Table 3. Time Recordings**

| Trial | Time |
|---|---|
| Shape Index | 1:09:24 |
| Canny Edge | 0:54:15 |
| HOG | 1:09:03 |
| Normal | 1:12:02 |

## 5. CONCLUSION

Overall, the feature extraction process did not make extreme changes during the training. However, there are some main points that should be made about these tests. The dataset used was small and the changes were not extremely different, but with a higher dataset it can be expected to see bigger gaps between these numbers. The results are very promising showing that there is potential to increase the time complexity using feature extraction. Something also not recorded was the amount of memory required. Using feature extraction would require less memory than a detailed picture due to a lot of removed pixels.

This research also shows that the accuracy is not far off not using feature extraction. This can be a form of accuracy tradeoff for increased performance with time and memory. If further tests were conducted a more concrete conclusion could be made, but for now canny edge and using the edge feature seems to show the biggest change in time while also keeping close to high accuracy compared to the other features.

## 6. FUTURE WORK

Much more work could be done to help improve this project. Something to consider would be trying different types of models with the convolutional neural network. It would be interesting to see how that would affect different times and if it would make a difference on if more hidden layers were used.

Something else that could be changed is using different data. For this research only, a subset of one dataset was used. Using different datasets could show much different results and objects that were not as simple could so that the feature extraction methods would yield different outcomes.

Lastly, many more feature extraction methods could be used and optimizing the current parameters for feature extracting would be helpful. Some other features that could be extracted might be texture and corners.

## 7. ACKNOWLEDGEMENTS

I would like to give a special thanks to Dr. Mingrui Zhang for the help with this research. He was my advisor and helped with planning and continuing work with this project. He also taught machine learning which started my interest in image classification. Without him, this research would not have been what it is. I would also like to thank Dr. Sudharasan Iyengar and Dr. Shimin Li for teaching this research practicum class. They provided immense help to all of us with preparation and questions. And lastly, thank you to all faculty and staff members who have taught me something and I would not be here if it were not for them.

## 8. REFERENCES

[1] Guide to the Sequential model - Keras Documentation. Retrieved April 22, 2020 from https://keras.io/getting-started/sequential-model-guide/

[2] Grogan, Michael. 2019. Image Recognition with Keras: Convolutional Neural Networks. Medium, Towards Data Science.

[3] Hong Wu, Hao Zhang, and Chao Li. 2011. Medical image classification with multiple kernel learning. In Proceedings of the Second International Conference on Internet Multimedia Computing and Service (ICIMCS '10), Association for Computing Machinery, Harbin, China, 189–192.

[4] Horea Muresan, Mihai Oltean, Fruit recognition from images using deep learning, Acta Univ. Sapientiae, Informatica Vol. 10, Issue 1, pp. 26-42, 2018.

[5] Mohammed, Ma'amari. 2018. How to Make A CNN Using Tensorflow and Keras. Medium.

[6] OpenCV-Python Tutorials — OpenCV-Python Tutorials 1 documentation. Retrieved April 22, 2020.

[7] skimage — skimage v0.17.dev0 docs. Retrieved April 14, 2020 from https://scikit-image.org/docs/dev/api/skimage.html

[8] TensorFlow. 2020. Convolutional Neural Network (CNN). Retrieved from https://www.tensorflow.org.

[9] Zhou Yawen, Dong Guangjun, and Xue Zhixiang. 2016. Hyperspectral image tensor feature extraction based on fusion

of multiple spectral-spatial features. In Proceedings of the 2016 International Conference on Intelligent Information Processing (ICIIP ’16), Association for Computing Machinery, Wuhan, China, 1–8.

[10] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu. Traffic-sign detection and classification in the wild. In CVPR, pages 2110–2118, 2016.

# Comparing self-extracted to third-party audio features for music genre classification

Bradley Erickson
Winona State University
BErickson15@winona.edu

## ABSTRACT

There is no definitive way to determine what genre music falls into. Conducting an analysis into the different audio features could prove useful in creating an automated process for determining genre. Using self-extracted audio features and third-party audio features, we create two artificial neural networks that will classify the song genre. Comparing our models with test data, we find self-extracted features do a better job at predicting genre.

## KEYWORDS

genre, classification, neural network

## 1 BACKGROUND

Song genres are primarily relative to the listener and there is no clear-cut way to classify which genre a song belongs to. With the power of machine learning, researchers have taken a crack at automating this process using artificial neural networks. Conducting this and other audio analysis can prove useful to music companies that wish to understand what customers enjoy the most. Other researchers have created convolutional neural networks to classify music by training on chunks of sound instead of various features [1–3]. For comparing data, two mirror artificial neural networks (ANN) will be created. One trained on the features extracted directly from the audio and the other trained using third-party song metrics. The two models will be compared to see which method has the better prediction accuracy.

## 2 DATA

We will focus on 10 musical genres, or categories, for classification:

(1) Blues
(2) Classical
(3) Country
(4) Disco
(5) Hip-hop
(6) Jazz
(7) Metal
(8) Pop
(9) Reggae
(10) Rock

*Proceedings of 20th Winona Computer Science Undergraduate Research Seminar, May 1, 2020, Winona, MN*

### 2.1 Self-extracted

Our first model will train using the GTZAN dataset which was created for extracting features and classifying music into genres [4]. This dataset contains 1,000 30 second song clips spanning across the 10 genres. We will use visual representation of frequencies, or spectrograms, to extract various features from the audio file. The following metrics will be extracted and used as input for our model:

(1) **Zero crossing rate**
**Description:** Rate of signal sign changes
**Usage:** Good for detecting percussive sounds
(2) **Chroma Shift**
**Description:** Map hertz to binned pitch class
**Usage:** What notes are being played
(3) **Spectral centroid**
**Description:** Brightness of tone
**Usage:** Distinguish between sounds
(4) **Spectral Bandwidth**
**Description:** Range of hertz
**Usage:** Specific frequency at interval
(5) **Spectral Rolloff**
**Description:** Central frequency
**Usage:** Approximates maximum or minimum frequency
(6) **Tempo**
**Description:** Estimated beats per minute
**Usage:** Determine speed of song
(7) **Root mean square**
**Description:** Perceived loudness of clip
**Usage:** Overall volume of track
(8) **Mel-frequency cepstral coefficients**
**Description:** Magnitude of shortened clip
**Usage:** Good for detecting level of voices

### 2.2 Third-party

For our second model, we will use data scraped from the Spotify API [5]. With this API, we will gather song metrics for inputs to our model and genres to for the output. Spotify does not include information about how they determine or detailed descriptions of the metrics they provide. The following metrics with brief descriptions will be used:

(1) **Key**
**Description:** Estimated key using standard pitch class
(2) **Mode**
**Description:** Major or minor
(3) **Time signature**
**Description:** Estimated time signature
(4) **Acousticness**
**Description:** Confidence of track being acoustic, non-electric

(5) **Danceability**
**Description:** How suitable a track is for dancing
(6) **Energy**
**Description:** Perceptual measure of intensity and activity
(7) **Instrumentalness**
**Description:** Confidence a song has no vocals
(8) **Liveness**
**Description:** Presence of audience
(9) **Loudness**
**Description:** Overall loudness in decibels
(10) **Valence**
**Description:** Musical positiveness
(11) **Tempo**
**Description:** Estimated tempo in beats per minute

## 3 METHODS

### 3.1 Data manipulation

The self-extracted data will be pulled from a spectrogram. An example spectrogram is shown in Figure 1. We will create a spectrogram for each song clip in the GTZAN dataset. Using the Librosa package in Python, we will extract our desired features and store them in a csv file. To obtain the third-party data, we query the top 100 songs for each of our genres using the Spotify API. For each song returned, we query our desired features. We store the features and genre label in a csv file. With both of our datasets, we transform the each feature to a normal distribution using Yeo-Johnson power transformations. Additionally, we will scale the distributions to use a [0, 1] range. Figure 2 shows a feature before and after the data manipulations.

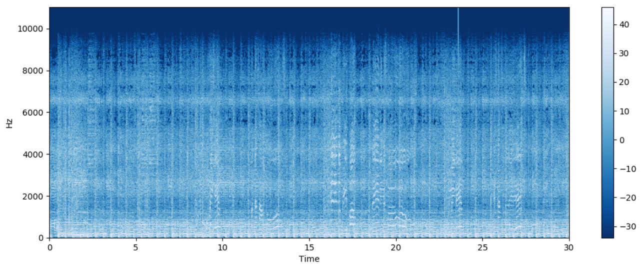


**Figure 1: Example spectrogram. Color corresponds to intensity of sound.**

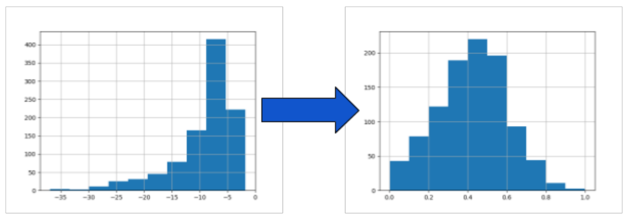


**Figure 2: Example feature before and after transformation.**

### 3.2 Model creation

With how complex music is, we want our classifier to have a deeper understanding of the inputs. This can be accomplished with an ANN, artificail neural network. Since, ANNs are a field of complex neurons and each input touches each neuron, this will allow our models to learn about relationships between our inputs. To compare the datasets, we will create two ANNs that are mirrored in structure. Our final model is shown in Figure 3 We start with a 64-node hidden layer with a ReLU output to provide a large initial layer on our model. We add a batch-normalization layer to keep the mean outputs of the first layer close to 0 with a standard deviation close to 1. This will help to not overfit the training data. We add a 32-node and a 16-node layer, both using ReLU as output. Lastly, we have a 10-node layer with a softmax output. One node for each genre.

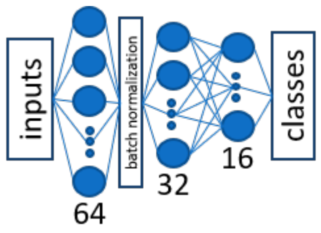


**Figure 3: Neural network model structure.**

### 3.3 Training

We split the data into training and test sets using an 80%-20% split, stratifying on genre. With each training dataset, we will conduct Monte-Carlo cross-validation to train 20 models. We will train using 20 epochs and a 64-unit batch size. These parameters yielded the best performance without overfitting. Finally, we will average the validation accuracy from each set of 20 models for comparison.

## 4 RESULTS

Figure 4 shows the self-extracted test data's actual versus predicted results for their model. This model did an outstanding job predicting data in the classical, metal, and pop genres. On the other hand, this model did a poor job predicting data in the rock genre, classifying 40% of the data-points as disco.

Similarly, Figure 5 shows the third-party test data's actual versus predicted results. This model only did well classifying data-points in the classical genre. This model did a poor job classifying data in the blues, pop, and rock genres. We notice that data belonging to pop and hip-hop are often misclassified as one another.

Table 1 shows the average validation accuracy and the test accuracy for each of our models. The self-extracted model correctly classified the validation data, on average, 6.32% better than the third-party model. For the test data, the self-extracted model correctly classified by 13% better than the third-party model. These results show the self-extracted data is better for predicting genre in a neural network than the third-party data.

## 5 FUTURE WORK

There are a few ways we can improve this work. First, we could use different classification methods, such as k-nearest neighbors or

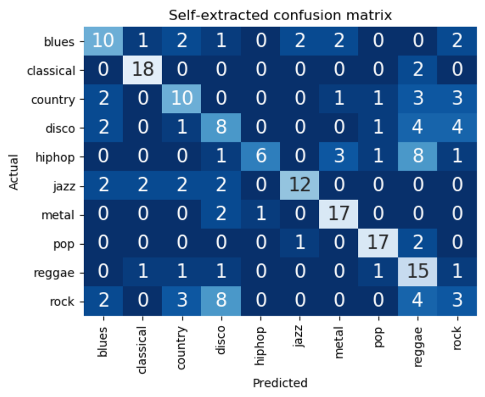


**Figure 4: Self-extracted model confusion matrix.**

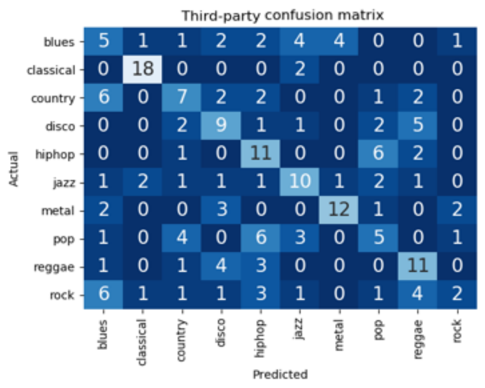


**Figure 5: Third-party model confusion matrix.**

**Table 1: Final accuracy percentages**

| | Avg. validation accuracy | Test accuracy |
|---|---|---|
| Self-extracted | 52.97% | 58% |
| Third-party | 46.65% | 45% |

a tree-based method. Additionally, we could expand the datasets to cover other genres and include more songs in our current genres. Lastly, the best option for comparison would be to use the same songs in each dataset.

## 6 REFERENCES

## REFERENCES

[1] Ahrendt, Peter, et al. *Decision Time Horizon for Music Genre Classification Using Short Time Features.* 6 Sept. 2004.
[2] Jeong, Il-Young, and Kyogu Lee. *Learning Temporal Features Using a Deep Neural Network and its Application to Music Genre Classification.* Ismir. 2016.
[3] Li, Tom LH, Antoni B. Chan, and Andy HW Chun. *Automatic musical pattern feature extraction using convolutional neural network.* Genre 10 (2010): 1x1.
[4] Tzanetakis, George, and Perry Cook. *Musical genre classification of audio signals.* IEEE Transactions on speech and audio processing 10.5 (2002): 293-302.
[5] Spotify for Developers API. https://developer.spotify.com/documentation/web-api/

# Performance Analysis of Heap, Merge, and Insertion Sort

Siddhant Grover
Department of Computer Science
Winona State University
Winona, Mn, 55987
sgrover16@winona.edu

## ABSTRACT

Numerical Data often has to be sorted for its application in different contexts and different fields. The data available is sorted through different algorithms based on different factors like time, efficiency, complexity, etc. This project focusses on taking large-size arrays and comparing the sorting through Heap, Merge and Insertion sort algorithms in two high-level languages: Java and Python

## General Terms

Algorithms, Performance, Languages, Theory.

## Keywords

Heap Sort, Merge Sort, Insertion Sort, Complexity, Space, Time, Java, Python.

## 1. INTRODUCTION

An algorithm focusses on several/few steps to provide a method to solve a problem. It can be defined as a well-formed procedure that takes input and provides output. Different algorithms can be used to solve one single problem[1]. The algorithm is chosen based on factors like efficiency, overhead, time, space, complexity, etc. Sorting is the method to organize/sort large numbers of items in a specific order. Sorting is an essential data structure operation, which performs easy searching, arranging, and locating the information. A simple sorting example would be sorting distances from short to long, for a pizza company to deliver food. Sorting algorithms usually take in a large amount of randomized data and return a sorted list of that data.

This research project focusses on a performance analysis of three such sorting algorithms: Heap Sort, Merge Sort, Insertion Sort; These algorithms are tested in two high-level programming languages: Java, which is a compiled language and Python which is an interpreted language.

1,6,8,2,9,11,
43,21,3,7,4
Sorting Algorithm
1,2,3,4,6,7,8
,9,11, 21,43

Figure 1:Sorting Example

## 2. SORTING ALGORITHMS

Sorting is a basic function that organizes a collection of randomized objects in a certain order. It is a trivial operation that is majorly used in data industries to organize a large collection of data. Sorting plays a huge role in searching data as well. Searching can be optimized in an environment where the data is already sorted. Searching would take longer to work on a collection of unsorted data as compared to sorted data. Some real-life examples of sorting would include File Directory systems, Words in a Dictionary, Inventor Management, Route locating systems, etc.[2].

### 2.1 Heap Sort

The understanding of the Heap data structure plays a huge role to truly understand the working of the Heap Sort Algorithm. Heap is a tree-based data structure. A tree data structure is a collection of nodes connected by directed (or undirected) edges. A tree can be empty with no nodes or a tree is a structure consisting of one node called the root and zero or more subtrees[2].

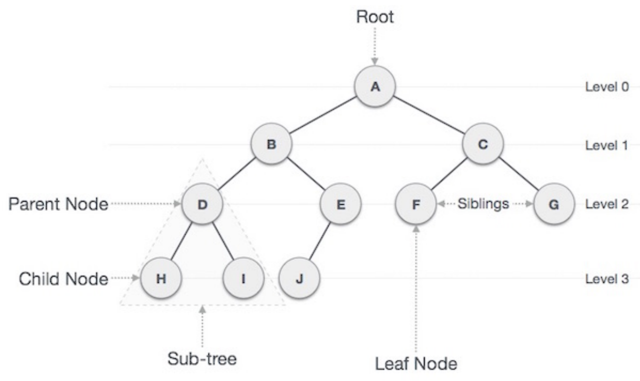


Figure 2: Tree data structure.

A heap is a tree data structure that satisfies the following properties.

1. Heap is always a complete binary tree. Every node part from a leaf should have two child nodes[1].

2. All nodes are either greater than or equal to or less than or equal to each of its children. This means if the parent node is greater than the child node it is called a max heap. Whereas if the parent node is lesser than the child node it is called a min heap[1].

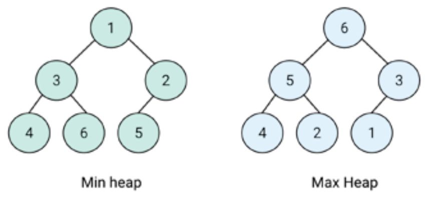


Figure 3: Min heap versus Max heap

Heap sort creates a max heap from the array and the element on the top of the heap is then separated and placed at the n-1th position of the array. The new heap has one lesser element and the next max element will be placed at the n-2th position. This keeps going on until the heap has only one element. The working of heap sort is shown below[1].

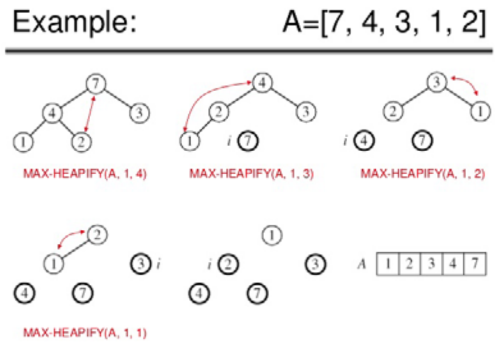


Figure 4: Working of Heap Sort.

## 2.2 Merge Sort

Merge Sort is a divide and conquer based algorithm that uses heavy recursion to sort an array. Divide and Conquer is a strategy used by algorithms to increase efficiency[2]. As the name suggests Divide and Conquer Algorithms consist of two parts: Divide-divide the problem into smaller sub-problems and the sub-problems are solved recursively, and Conquer-The solutions of the smaller sub-problems are merged together to find the solution of the original problem. Merge sort follows the same ideology. It divides the array into two halves recursively and then merges the sub-arrays. The working of heap sort is shown below[2].

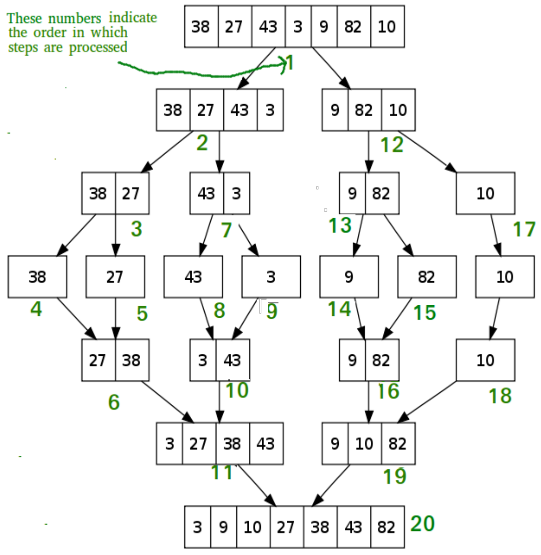

These numbers indicate the order in which steps are processed

Figure 5: Working of Merge Sort.

## 2.3 Insertion Sort

Insertion Sort is a simple sorting algorithm that follows the 'one at a time' methodology. In simple terms, it can be compared to sorting cards, where the 1st card is assumed to be sorted[2]. As cards are picked, they are compared with each of the already sorted cards in hand. Eventually, all the cards will be sorted as we go through each card in the pack[1]. At each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list and inserts it there. It repeats until no input elements remain[3]. Sorting is typically done in-place, by iterating up the array, growing the sorted list behind it. The working of insertion sort is shown below.

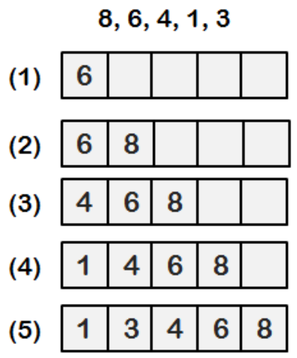


Figure 6: Working of Insertion Sort.

## 3. RESEARCH

The goal of this research project is to analyze these sorting algorithms on large-size numerical data. There has been previous work done to understand and compare different sorting algorithms. This research extends on previous understanding and research of sorting algorithms to two languages and shows a literal time comparison of the sorts in both languages. While time is a huge factor, the analysis further includes factors apart from time like simplicity and space constraints. The algorithms stated above are implemented in Java -compiled language and Python-interpreted language. Python being an interpreted language takes longer to run, however, the purpose of analyzing the algorithms in two different languages is to see the timing trend and pattern based on the size of the array in both languages[4].

### 3.1 Environment

The research is conducted on HP Elitebook x360 1032 G2 18 Microsoft Operating System, Windows 10, version: 1909, Processor: Intel® Core™ i5-72000U CPU @2.5 GHz, Installed Ram: 4 8GB. The Eclipse IDE, version: 2019-12 (4.14.0), Build id: 0191212-1212.Python3.7.6,PyCharm CommunityEdition 2019.3.3

### 3.2 Data

The data used for this project is purely Numerical. For Java, large size sorted and unsorted arrays consisting of 1-n distinct elements are used, where n varies from 50K to 10000K. In Python, large size sorted and unsorted arrays consisting of 1-n distinct elements are used, where n varies from 5K to 1000K. This is done to check the breaking point in terms of the number of elements at which the algorithms show visible time difference due to their complexity.

### 3.3 Methodology

Each Sorting Algorithm is implemented in both the programming languages as functions, and these functions are run over different sized arrays in Java and Python.

### 3.3.1 *Best Case Scenario*

The array created is in ascending order from 1-n elements, where n is the size of the array.

### 3.3.2 *Average Case Scenario*

Every individual array is shuffled randomly between 4 and 7 times to create a randomly distributed unsorted array. The array is not shuffled more than 7 times to prevent over shuffling the array and, in some way, leading to a more sorted array.

### 3.3.3 *Worst Case Scenario*

Heap Sort algorithms convert an array into a heap and thus there is no definite case of the worst-case scenario and thus a randomized array is used[2,3]. The worst case for a merge sort would be where there would be the greatest number of comparisons for the merge sort algorithm to make. This is done by creating two auxiliary arrays left and right and storing alternate array elements in them. This is done recursively on both left and right arrays, till only distinct elements are left[1]. These elements are put together to form the worst-case array. Insertions sort's worst-case array would simply be a sorted array in reverse order, i.e. the largest element first and the smallest element at the last position[3].

## 3.4 Obtaining Results

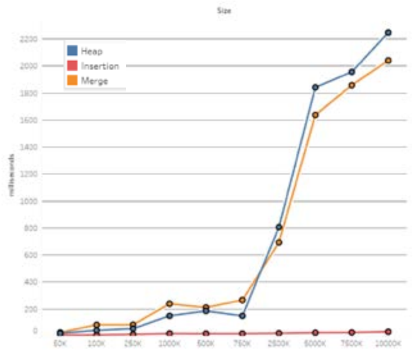
The Heap sort, Merge sort and Insertion sort functions are called on every individual array in both languages. The time taken to sort the array for each algorithm is noted. This whole process is repeated for 10 iterations on each different array size mentioned above. The average values obtained after performing the sorts are noted. The time values for the different size arrays are then put together and visualized through Tableau.

## 4. RESULTS

The results of this research consist of how long it took to perform the sorts while the analysis includes the reasoning and other factors of sorting.

## 4.1 Sorted Array

Sorting a sorted array presents a good idea of how well the algorithm will perform on a nearly sorted array[2]. The real-life application will be in context of nearly sorted set of data that needs to be sorted. The three algorithms were passed through a sorted array and the time taken was noted. Heap and Merge sort take a similar amount of time to sort the array, while Insertion sort takes the least time to sort a sorted array. As the number of elements increase, a much wider difference between the time taken by Insertion and Merge/Heap sort is seen.



Graph 1: Time taken to sort a sorted array in Java

Table 1: Time taken(milliseconds) to sort a sorted array by Insertion sort in Java.

| Size | Time (ms) |
|---|---|
| 50K | 7.8 |
| 100K | 9.2 |
| 250K | 13 |
| 500K | 17.8 |
| 750K | 18.3 |
| 1000K | **19** |
| 2500K | **21** |
| 5000K | **25** |
| 7500K | **27** |
| 10000K | 32 |

The data obtained for insertion sort is shown above.

The maximum time taken by insertion sort on a sorted array of 10000K element on an average is around 32 milliseconds, while it takes more time than that by the other sorting algorithms to sort 100K elements. This is due to the fact that the algorithm for Insertion sort uses comparison and then switching, and since it's a sorted array no element has to be displaced from its location in the array. In Python, Heap Sort and Insertion sort behave similarly but merge sort behaves slightly different. Merge sort behaves faster for larger size data in both the cases.

Graph 2:time taken to sort a sorted array in Python.

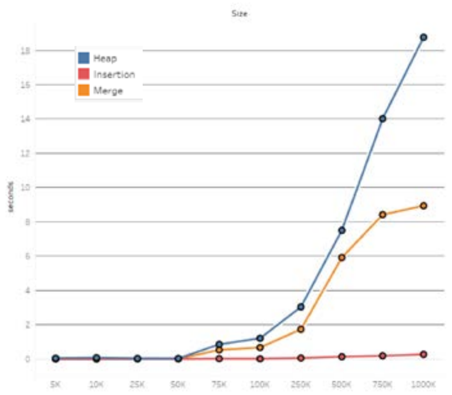


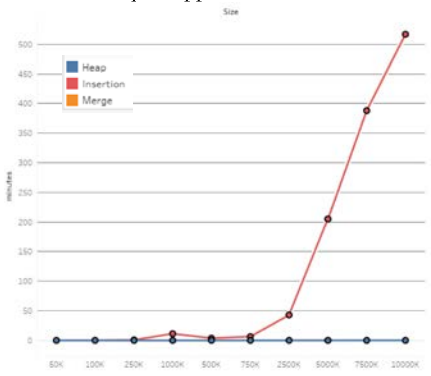
Table 2: Time taken(seconds) to sort a sorted array by Insertion sort in Python.

| Size | Time (s) |
|---|---|
| 5K | 0.00199866 |
| 10K | 0.00199819 |
| 25K | 0.00703454 |
| 50K | 0.01094365 |
| 75K | 0.02797771 |
| 100K | 0.02100396 |
| 250K | 0.06605411 |
| 500K | 0.14139819 |
| 750K | 0.19499683 |
| 1000K | 0.2761302 |

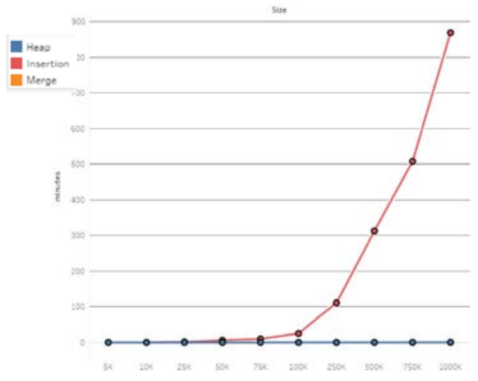
Python programs run comparatively slower than Java and thus the comparison for Python is done in seconds[4].

## 4.2 Unsorted Array

Insertion sort which took the least time to sort a sorted now takes the longest time for an unsorted array. The reason being that the sorted array had no switching after all the comparisons made. However, in an unsorted array, it has to compare every element to every other element in the list, and then the elements are displaced and put in their right position. Merge and Heap, however, have different methods that do not require comparing every element to the other. While Heap requires elements sort through a heap, Merge sort uses a divide and conquer approach with recursion.



Graph 3: Time taken to sort an unsorted array in Java(minutes)



Graph 4: Time taken to sort an unsorted array in Python(minutes)

As the number of elements increases, a large time difference is seen from the graph. The comparison on the graph shows that after 250K elements in java and 25K elements in Python wide time differences to sort the data are visible. An interesting comparison from graph 3 for 7500K elements is shown in the table below to see the time difference the different algorithms take.

Table 3:time taken to sort 7500k elements in java

| Sort | Heap | Merge | Insertion |
|---|---|---|---|
| Time Taken | 3.04 seconds | 1.42 seconds | 6.46 hours |

## 5. ANALYSIS

The analysis is done on a series of factors such as time complexity, space complexity, number of elements to be sorted, simplicity of the algorithm, etc. Every algorithm has a series of advantages as well as a series of shortcomings over the other.

The Big O notation plays a huge role in comparing algorithms for usage. It is defined as an upper bound of an algorithm, it bounds a function only from above. Mathematically f(n) = O(g(n)) if there exists a positive integer ~~n~~ and a positive constant c, such that $f(n) \leq c.g(n) \forall n \geq$ ~~n~~[2].

## 5.1 Heap Sort

Table 4: Heap Sort Complexity Analysis[3]

| | Complexity |
|---|---|
| Best Case | $O(n \log n)$* |
| Average Case | $O(n \log n)$ |
| Worst Case | $O(n \log n)$ |
| Space | $O(1)$ |

* $O(n \log n)$ for distinct keys or $O(n)$ for equal keys.

### *5.1.1 Advantages*

1. Time Complexity of the algorithm(in general) is$O(n \log n)$, therefore it takes an almost equal amount of time to sort in all cases for a given size.

2. The space required is constant. The algorithm does not require multiple arrays[3].

3. Non-recursive algorithm.

4. In-space and non-recursive nature make it suitable for large data sets[3].

### *5.1.2 Disadvantages*

1. Works slower than the Merge Sort with the same time complexity $O(n \log n)$.

2. Has to be converted to a heap data structure[1].

3.Unstable Sort: the relative ordering of elements is not preserved[3].

## 5.2 Merge Sort

Table 5: Merge Complexity Analysis[3]

| | Complexity |
|---|---|
| Best Case | $O(n \log n)$ |
| Average Case | $O(n \log n)$ |
| Worst Case | $O(n \log n)$ |
| Space | $O(n)$ |

### *5.2.1 Advantages*

1. Time Complexity of the algorithm(in general) is$O(n \log n)$, therefore it takes an almost equal amount of time to sort in all cases. It is comparatively a faster algorithm.

2. Works faster than Heap Sort algorithm,$O(n \log n)$ complexity, for larger datasets.

3. Merge sort breaks the input into chunks, each of which can be sorted at the same time in parallel[3].

4. It is a stable sort[3].

### *5.2.2 Disadvantages*

1. Recursion is required to sort data

2. Merge sort takes up O(n) extra space, O(log(n)) space for the recursive call stack[3].

## 5.3 Insertion Sort

Table 6:Insertion Sort Complexity Analysis[3]

| | Complexity |
|---|---|
| Best Case | $O(n)$ |
| Average Case | $O(n^2)$ |
| Worst Case | $O(n^2)$ |
| Space | $O(1)$ |

### *5.3.1 Advantages*

1. Simple Algorithm, easy to understand and implement.

2. It is very efficient (near O(n) complexity)for nearly sorted data[3].

3. The space required is constant. The algorithm does not require multiple arrays or recursion.

4. It is a stable sort.

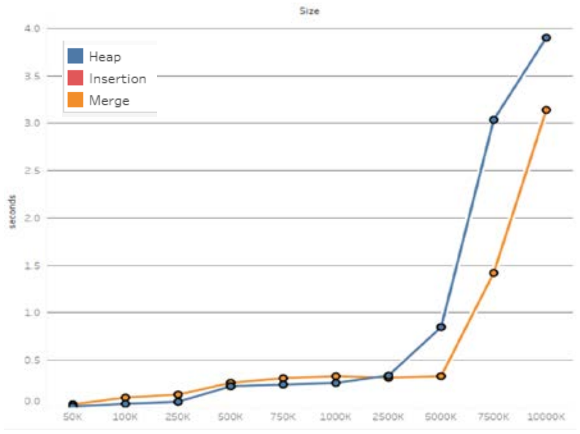
### *5.3.2 Disadvantages*

1 Has the worst time complexity for randomly sorted data.

2. Takes the longest time for unsorted large data sets.

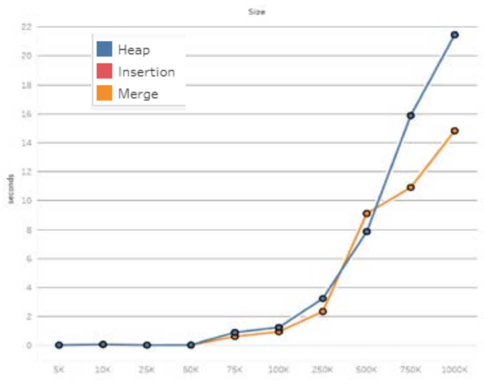
## 5.4 Which algorithm is the best?

The question that arises is, which algorithm works the best and should be used. The answer is simply “it depends”. The analysis above shows that all sorts have their own sets of disadvantages and advantages. Insertion sort is a simple algorithm to implement and works the fastest for nearly sorted data, while heap and merge sort algorithms work at a faster speed, however, they are more complex and require more space[1]. Heap sort requires constant space, but also is an unstable sort, while merge sort takes more space and requires recursion but takes the least amount of time to sort.



Graph 5:Merge vs Heap in Java

The graph shows that Merge sort starts working faster after 2500K elements in Java. In Python, the same change is seen however at 50K elements.



Graph 6: Merge vs Heap in Python

The graphs above show the time taken to sort an unsorted array in Java and Python. It is visible that while quite close to each other, Merge sort works faster than Heap sort. Insertion sort, on the other hand is a simple algorithm with the worst time complexity for a randomized dataset, however, it works the fastest for sorted and nearly sorted data. The advantages and disadvantages of every sorting algorithm lead the user to choose based on time, simplicity, and space constraints.

## 6. CONCLUSION

Sorting is an essential data structure operation, which performs easy searching, arranging, and locating the information. Sorting is a huge part of applications that are used on a daily basis. This project focused on analyzing three different sorting algorithms in two different languages. While the trends remain similar, the breakpoints at which the algorithms start behaving differently is surely noticeable and should be taken into consideration while choosing a programming language. It would be unfair to name one algorithm as the best. Factors like space, size, time play a huge role and an algorithm should be chosen based on these factors. While some algorithms may perform faster than the other, they may also require a larger overhead and thus might not be the best option. Thus, all factors should be taken into consideration along with the resources available by the user to choose a sorting algorithm for a program.

## 7. ACKNOWLEDGEMENTS

Special Thanks to the Computer Science Department of Winona State University, my academic advisor, Dr. Tim Gegg-Harrison, my professors and mentors- Dr. Sudharsan Iyengar, Dr. Shimin Li, Dr. Joan Francioni, Dr. Yogesh Grover, and my peers

## 8. REFERENCES

[1] Heineman, G. T., Selkow, S. T., & Pollice, G. T. (2009). Algorithms in a Nutshell. Retrieved from https://www.oreilly.com/library/view/algorithms-in-a/9780596516246/ch04s06.html.

[2] Goodrich, M., & Tamassia, R. (2015). Algorithm design and applications. John Wiley & Sons Inc.

[3] Pathak, D. K., Sharma, V. K., & Rastogi, M. (2017). Performance Analysis of Various Sorting Algorithms. International Journal of Advanced Research in Computer and Communication Engineering, 6(2).

[4] H, S. C. (2013). A Byte of Python . Minneapolis, MN: Open Textbook Library.

# Facebook Privacy Settings: Individuals belief on privacy versus their settings

Callie Kitoski
Winona State University Computer Science Department
175 West Mark Street
Winona, MN 55987
+1 (507) 457-5000
Ckitoski16@winona.edu

## ABSTRACT

This study looks at the difference between an individual's settings on Facebook versus their desired privacy. After the issues regarding Facebook's privacy issues people seemed to be more aware of their privacy online. The survey was administered, where participants shared their privacy settings so an analysis could be made. With the data in this study compared those to the desired level of privacy they want as well as the level of privacy they believe they currently have. The correlation between the two being drawn will result in an understanding of the difference between them in respect to the participating settings on the Facebook platform.

## KEYWORDS

Data privacy, human factors, Facebook, security, social media, protection, privacy settings

## 1. INTRODUCTION

The way privacy has evolved through the years with the extensive access to the internet worldwide has posed some concerns towards the privacy of users. Privacy itself can be defined as protection of an individual's information from another party and or individual. With privacy being important in many aspects of life being able to reassure the users that their information and data is only available to people they want is crucial to their feeling of privacy. Privacy has had its downfall into being considered as more of a want then a need. Privacy, in a sense, can be nonexistent but it does not have to be. The thing with privacy is it is perceived as being all or nothing when that does not have to be the case [1]. There are things that can be implemented to improve individuals' feelings of privacy online. Laws and legislation should be passed to protect user's information online. It is a simple way that gives users reassurance that someone has their back regarding their personal information online. According to an IEEE-USA position statement they state there are 4 main categories that public transparency, disclosure for users, control, and notification [2]. Privacy of individuals are not being treated at the level it should be. With the condensing into these four categories of the main issues that need to be changed and addressed. The access to people's information without protection is being accessed and being used in ways they the users do not know are. Legislation in the works and being introduced to congress in recent years.

One recent bill following Facebook's Privacy issue in 2018 is one that requires the user to opt-in, indicating they are aware of the data being collected from them on many social media platforms, as well as it requires then to be notified exactly when it's being gathered, shared, and sold to other parties [3]. This is the direction the legislation needs to be progressing as the ways we use resources online advance and are more available to a wider range of users.

Social media, in general, have developed their way and methods to help aid individual privacy on the platforms, but recently some of those have not been keeping user's data as private as they indicated. Facebook and Cambridge Analytica in 2018 received fines due to the exchange of Facebook user's personal information, Facebook is due to pay several fines for this incident, even one to U.K.'s Information Commissioner's Office [4]. This situation changed several things in privacy. Them agreeing to paying the fine was tremendous in the aspect that they did in fact do that and are going to be held accountable for their actions, but a fine only really goes so far. There was no concrete information on what happened to the user's data. The users were not provided reassurance or information on the data that was collected, and Facebook had no way for them to provide the users with this type of information. Facebook admitted to their role in the transfer of the user's data from there platform to Cambridge Analytica, which was done without any consent [5]. This issue with Facebook revealed a lot about the privacy individuals are receiving, by them not fully disclosing the information they are collecting and putting them up to be questioned on why this information needs to be collected and at what level of security is this information on these users being held at to protect them.

The prediction about the result I have in mind is that the Facebook privacy they want will not match what they believe they are receiving as well as settings will need to be more private (only me, friends, specific friends) rather than public.

## 2. BACKGROUND INFORMATION

Facebook's privacy settings are a separate tab within the settings labeled privacy. There are two sections in the privacy settings tab, your posts and how people find and contact you. For this study I choose to focus on how people find and contact your settings and use those for the survey. This category has five questions within it regarding those settings. Those regarding sending you friend requests, seeing your friend list, looking up via email, looking up via phone number, and can search engines outside Facebook link to your profile.

## 3. METHODS

Survey method is being selected as that is the most beneficial way to the individual population in retrieving the data pertaining to their settings and beliefs. The research proposal had to be approved by IRBNet due to the use of the human subjects in the research. Since participants had to be informed of their rights a consent form was attached to the survey to provide the participants with that information as well a contact for questions and concerns regarding the research that was being done. To keep participants as anonymous as can be, the only personal information about themselves collected was an age range that they fell into on a given set of options. This was to protect the individuals who participated in the survey as it was their personal Facebook settings being collected for the survey data. Survey questions included opinion based and specific setting questions. The data was collected and transferred to a spreadsheet. Once all the information was collected and transferred, evaluating the data began.
Ideally participants would be contacted randomly and asked to respond to the survey on their settings, but the resources were not as readily available to do that with the given state of the survey. Participants were selected using a few different ways. The first way was by using email groups that already existed and sent the survey to those on the mailing list explaining the research that was being done, and if the individuals have the requirements to participate in the survey (having a Facebook account and being 18 years or older).

## 4. RESULTS AND ANALYSIS

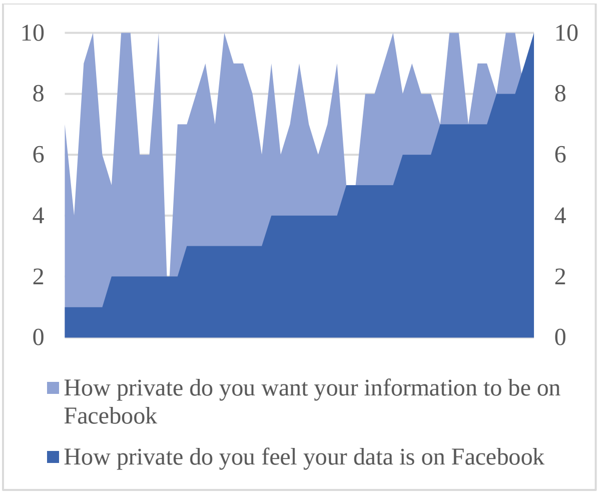


**Figure 1: contrast of how private they feel they are on Facebook then how private they want to be**

| | Mean | Median | Mode |
|---|---|---|---|
| **How private they feel** | 4.2941176470588 | 4 | 3 with 9 occurrences |
| **How private they want to be** | 7.843137254902 | 8 | 9 or 10 with 11 occurrences each |

**Table 1: statistics for the data in figure 1**

The graph in figure 1 visually depicts the contrast between the participants feeling of their privacy on Facebook and the level of privacy they want while table one shows the statistics of it. As you can see there is a clear difference in the mean, median, and mode trends between the statements of how they feel and the level of privacy they want. The difference in the mean values is 3.5490196078432 which when considering the scale was one to ten that is a significant difference in the values indicating that the gap between them is a testament to users feeling of privacy. The median reflects a similar concept with their feeling of privacy being four and their desired privacy being eight. Now the mode which depicts the number that appears most shows a similar trend with the level of privacy they feel they have most occurring at three and the level of privacy they want occurring most at nine and ten. This information indicates that users are feeling a loss on Facebook with their privacy and there is a clear desire for my privacy to the users.

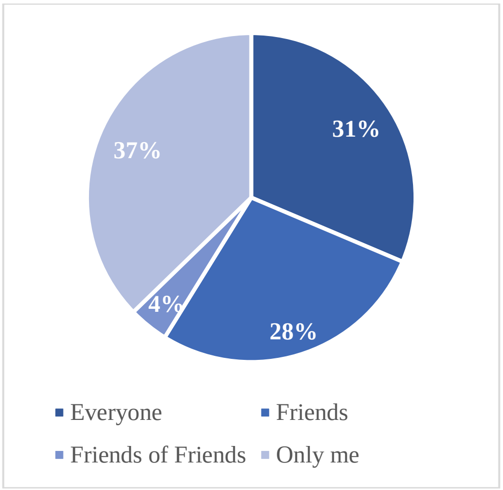


**Figure 2: Settings regard who can look them up from the number they provided**

In figure 2 it displays the participants settings regarding who can look up their profile by their phone number. 31% of the participants have no restrictions on individuals looking them up via their phone number. This means they have the lowest level of privacy on people finding their profile. Then 37% of the participants do not allow anyone to look them up via their phone number. Making that feature not being used to access their profile and providing the most privacy from people finding them via their phone number. Now the middle two settings of Friends of Friends, and Friends offer different populations. With friends of friends the population is still a large population, given that it is multiplying your friends by their friends. It is leaving a door open into people being able to view things and see your profile.

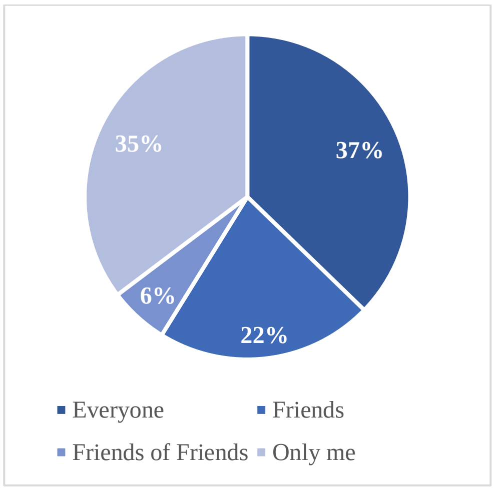


**Figure 3: Settings regarding who can look them up using the email they provided**

In figure 3, the setting itself resembles that of what would be done in figure 2 with individuals being looked up via an outside source, in this case it is their email. People are more willing to allow everyone to look them up via their email then with their phone number, as shown in figure 2. Phone numbers as shown in figure 2 are something that participants seemingly want more privacy, this may be because people see more of a separation between themselves and their email than with a phone number and would be more difficult for users to see and use that information to find them on the platform.

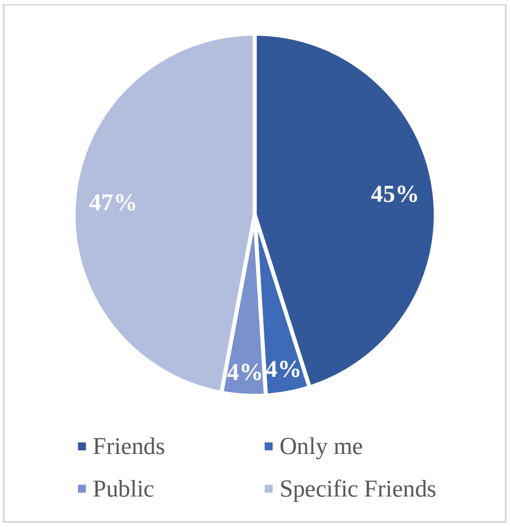


**Figure 4: Settings regarding who can see their friends list**

This data as depicted in figure 4 shows that there is a clear privacy desire. It indicates that over 90% of the participants want their friends list only shown to their own friends if not less users. This would show that there is a higher level of privacy regarding how people will know if you know another person and limit the access of the distance connections between people. Mutual friends on Facebook shows who you have in common with another person is a feature that displays those connections you have with the other users on the platform. This mutual friend connection will, in some encounters I have seen, only show you who you have in common. With that you can see those connections to the other individuals which then depicts a higher level of privacy on one's profile.

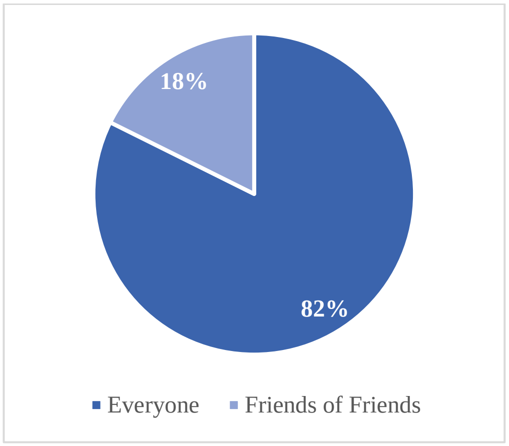


**Figure 5: Settings regarding who can send them friend requests**

Figure 5 shows that people are very open to everyone sending them friend requests. This in respect to visibility of their friends list as in figure 4 shows that this could open a door into a privacy risk if you accept a friend request of someone you do not necessarily know. It is interesting to see the population of individuals who only allow people with mutual friends send them a request, this is in line with the participants desire for more privacy on Facebook, as well it would limit the number of people you would need to go through on your request list because the requesting population would be smaller.

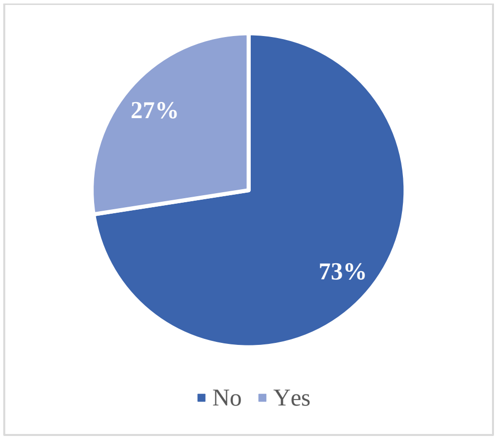


**Figure 6: Settings regarding if search engines outside of Facebook can link your profile**

In figure 6 it shows users do not prefer having their profile appear in search results on other sites such as Google. This protects them from being searched on Google and immediately found on Google. This protects their privacy on Facebook by keeping it only on Facebook. This follows the trend of wanting more privacy than they are receiving on Facebook.

## 5. CONCLUSION

Overall, the research shows that the level of privacy that they are receiving and the level of privacy they want have a significant difference. People want more privacy than they believe they are receiving. Facebook should alter and improve the level of privacy they can offer so that users feel more protected on social media and can market that to gain users. Privacy is something we feel like is going away but it does not have to be. Privacy can be altered and improved for the users benefit. The internet in general changes too frequently, it may be hard to keep up with it, but is still a benefit to the user to feel protected and safe while online.

## 6. FUTURE WORK

The idea of future work that could be done is lengthy. One being having a larger more random pool of individuals complete this same or similar survey and then compare their results as done in this study. Another direction is to take this but do it with several other social medias and compare their privacy settings to the level of privacy wanted to have. With a continuation of that you could then compare social media against one another to see which one's users feel most private on and as well as which one offers the most privacy on the social media.

## ACKNOWLEDGEMENTS

Thanks to Dr. Nicole Anderson for her advising with this research. Thanks to ACM SIGCHI for allowing us to modify this template for the research paper.

## REFERENCES

[1] Jennifer Granick. 2015. DATA AND PROTECTING THE RIGHT TO PRIVACY. The Center for Internet and Society at Stanford Law School (September 2015).

[2] Jim Isaak and Mina J. Hanna. 2018. User Data Privacy: Facebook, Cambridge Analytica, and Privacy Protection. Computer 51, 8 (August 2018), 56–59. DOI: http://dx.doi.org/10.1109/mc.2018.3191268

[3] IEEE-USA Board of Directors. 2018. Digital Personal Privacy, Awareness and Control. (June 2018). https://ieeeusa.org/wp-content/uploads/2018/08/DigitalPrivacy0618.pdf

[4] Paolo Zialcita. 2019. Facebook Pays $643,000 Fine For Role In Cambridge Analytica Scandal. (October 2019). Retrieved April 26, 2020 from https://www.npr.org/2019/10/30/774749376/facebook-pays-643-000-fine-for-role-in-cambridge-analytica-scandal

[5] Electronic Privacy Information Center. EPIC - In re Facebook - Cambridge Analytica. Retrieved April 26, 2020 from https://epic.org/privacy/facebook/cambridge-analytica/

# Machine Learning Algorithm Accuracy for Time Series Prediction

Isaac Plevak
Computer Science Department
Winona State University
yp5798hc@go.minnstate.edu

## ABSTRACT

Time series forecasting using machine learning is a common method of forecasting. Throughout time many algorithms have been used to complete this task. In this project an older algorithm, the multilayer perceptron (MLP), is compared to a newer algorithm, the long short-term memory (LSTM). Throughout testing the LSTM was able to give more accurate results than the MLP for the datasets used.

## KEYWORDS

Time series forecasting, Machine learning

## 1 INTRODUCTION

Time series data is data arranged in time order. In other words, time series data is data gathered multiple times throughout time. A business can view daily sales and get an idea of what time of year is the busiest in terms of sales. This is valuable information for many businesses and organizations. Being able to interpret this data is a useful tool and many algorithms have been designed to assist in forecasting this data. These range from simple algorithms to complex deep neural networks. These neural networks are becoming increasingly more common and accurate as time goes on.

The long shot-term memory (LSTM) algorithm is a machine learning algorithm built off of a recurrent neural network (RNN). RNN is designed to learn time series data more accurately than other algorithms. It achieves this by receiving an input from the previous time step as well as its features. This allows data from previous time steps to affect the current time step. The major problem of RNNs is the vanishing gradient problem. This problem occurs during the backpropagation algorithm of the training phase of RNN. The gradient used to update the weights of the RNN gradually decreases and thus affects how well the algorithm learns. LSTM includes an extra gate called the forget gate which affects the gradient and prevents the vanishing gradient problem.

MLP is a simpler and older algorithm. MLP takes in a set of inputs or features and applies a weight to them and gives an output. MLP can be used for many different types of datasets, from image recognition to time series. In this case MLP is our older algorithm to compare to our newer algorithm in the LSTM for time series datasets.

## 2 HYPOTHESIS

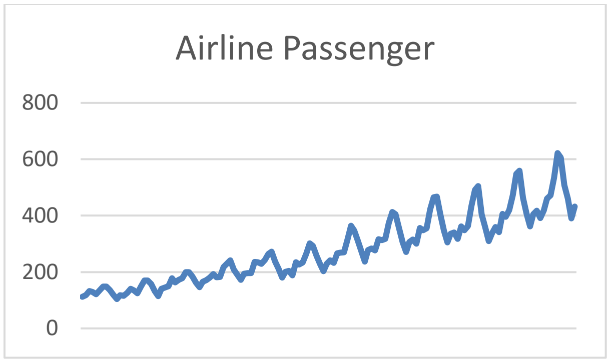
The long short-term memory algorithm will give more accurate results than a multilayer perceptron on certain time series datasets.

## 3 METHODS

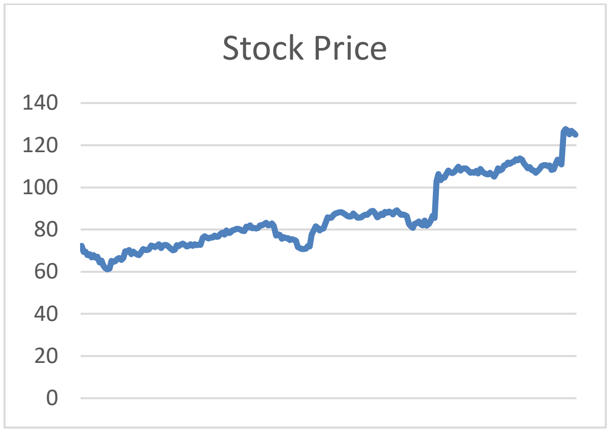
In this project two common machine learning algorithms, the multilayer perceptron and the long short-term memory recurrent neural network, are compared using time series datasets. Both algorithms were trained with the same two datasets and are compared both visually and comparing their mean squared error on the testing set. Python was the language used for this project and the algorithms were from the Keras library.

### 3.1 DATASETS

The datasets used are of airline passenger data and stock prices. These were chosen because of the real-world application of these datasets and how they varied. The airline passenger dataset has a distinct pattern to it throughout time, while the stock price dataset is far more unpredictable. This was done to test multiple scenarios for each of the machine learning algorithm. Some datasets will have a pattern to them while others will appear to be totally random. Each machine learning algorithm learns data differently, thus some will learn a pattern better than a more random dataset. Graph 1 is of the airline passenger dataset and graph 2 is of the stock price dataset. The training set was the first two thirds of the dataset while the remaining one third was used for testing.



**Graph 1: Graph of Airline Passenger dataset**



**Graph 2: Graph of stock prices dataset**

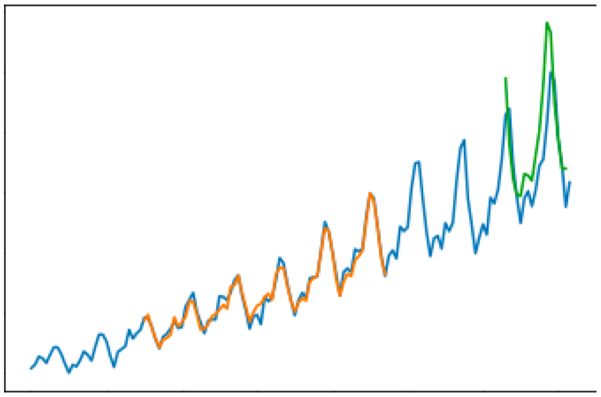
## 3.2 TRAINING

The MLP lookback was set to 30 for this project. The lookback is the number of datapoints it will use per step. Basically, it will use the last 30 datapoints to calculate the next output. In this case the lookback is the number of features the MLP takes in as input. The MLP was set to run 50 epochs for training (MLP ran through training set 50 times).

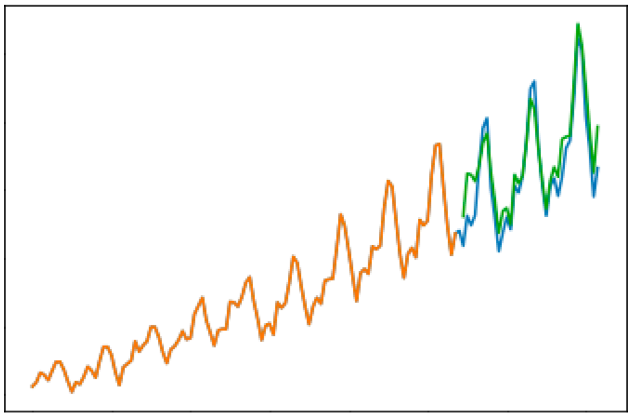
The LSTM had one feature, the current time step. LSTM ran 25 epochs rather than 50, this was done for time purposes. LSTM training takes longer than MLP training.

## 4 RESULTS

For the airline passenger dataset, the MLP was able to learn and get the basic curve of the dataset as seen in Graph 3. The LSTM learned the dataset better giving more accurate results. Graph 4 shows this.



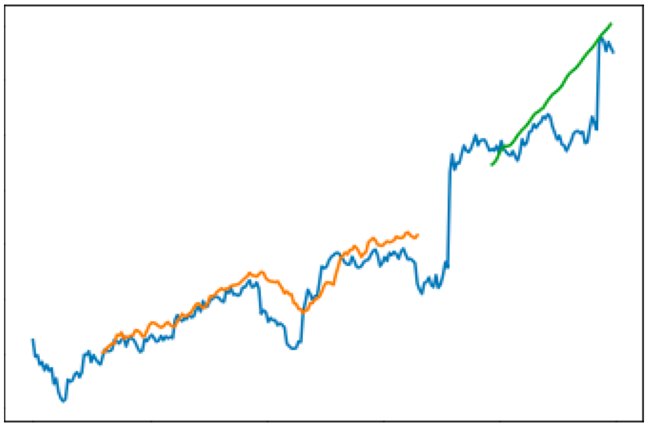
**Graph 3: Graph of MLP test for airline passenger dataset**



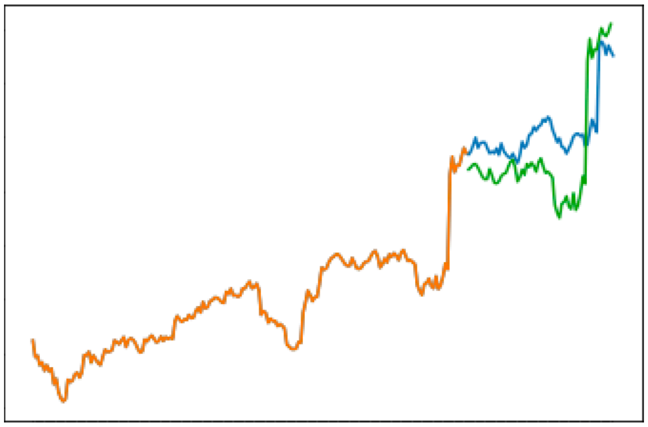
**Graph 4: Graph of LSTM test for airline passenger dataset**

The MLP learns the basic form of the dataset but didn't get it down exactly. The mean squared error (MSE) of the MLP is on average of 0.1029 for the test score while LSTM was roughly half that. The mean square error is used to calculate average of the square error of the algorithm. It is squared to make all errors positive. A smaller MSE is better.

For the stock prices dataset, the MLP didn't do nearly as well. The MLP couldn't learn the flow of the dataset at all. The LSTM wasn't precise but did learn the basic flow of the graph. Graph 5 shows the MLP for stock prices dataset and Graph 6 shows LSTM for stock prices dataset.



**Graph 5: Graph of MLP test for stock prices dataset**



**Graph 6: Graph of LSTM test for stock prices dataset**

For the stock prices dataset, the MLP struggled to predict the graph however the LSTM was able to predict the basic shape of the graph. The MLP outputs a linear result where the data curve is not. Even though the LSTM isn't deadly accurate it follows the same basic curve the dataset does, showing where the data spikes. On average the MSE of the MLP was 0.1931 which is not an ideal MSE. Once again, the LSTM achieved half the MSE with 0.0939 test score MSE.

## 5 ANALYSIS

Analyzing the results shows that the MLP was less affective at learning the datasets than the LSTM. The MLP was able to learn the basic trend of the airport passenger dataset but struggled to learn the stock market dataset. The MLP output for the airport passenger dataset followed the same curve as the dataset, even though the values were not always very accurate. As for the stock price data the MLP was unable to display the basic trend at all. This is partially to do with the randomness of the stock market dataset. Visually the airline passenger dataset has an obvious pattern to it, while the stock prices dataset has no such pattern. The LSTM was more accurate for both datasets. For the airline passenger dataset, LSTM was able to output values closer to the testing set values. The LSTM was also able to learn the basic trend of the stock price data.

## 6 CONCLUSION

The LSTM algorithm achieved higher accuracy than the MLP did for both datasets used in this project. Judging visually or by the mean squared error, the LSTM is the more accurate algorithm for time series forecasting.

The LSTM is a modern neural network developed to fix problems that older neural networks suffered from. It is no surprise that it is more accurate than the MLP, however it is still useful to know how much more accurate it is. MLP has its own advantages over the LSTM. One major advantage is speed. If a quick algorithm is more important than an accurate one, the MLP is a very good algorithm. However, if accuracy is important, then the LSTM is will give better results.

These algorithms are used to help with making accurate predictions but are rarely used alone. Stock prices rise and fall often with little notice. This is because there are many things that affect the prices. Far more things than what could be entered into any algorithm. Thus, algorithms used in forecasting can help get an idea of what the future holds if nothing dramatic happens. Because of this accuracy is not always the most important feature of a machine learning algorithm. There are a countless number of algorithms and all have their own strengths and weaknesses.

## ACKNOWLEDGMENTS

Thanks to Dr. Zhang, Dr. Iyengar, and Dr. Li for help with understanding the algorithms in use and the workflow of this research project.

## REFERENCES

[1] Brownlee, Jason. "Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras." Machine Learning Mastery, 5 Aug. 2019, machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/.

[2] Brownlee, Jason. "How to Develop Multilayer Perceptron Models for Time Series Forecasting." Machine Learning Mastery, 5 Aug. 2019, machinelearningmastery.com/how-to-develop-multilayer-perceptron-models-for-time-series-forecasting/.

[3] "Keras: The Python Deep Learning Library." Home - Keras Documentation, keras.io/.

[4] Brownlee, Jason. "When to Use MLP, CNN, and RNN Neural Networks." Machine Learning Mastery, 19 Aug. 2019, machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/.

[5] Mittal, Aditi. "Understanding RNN and LSTM." Medium, Towards Data Science, 12 Oct. 2019, towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e.