Winter 2022

Automated License Plate Recognition Systems

Nicholas Noboa
University of New Hampshire, Durham

Automated License Plate Recognition Systems

Nicholas Noboa

Thesis
Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Master of Science
in
Information Technology
December 2022

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

Thesis Director, Timothy Chadwick, Lecturer (Applied Engineering & Sciences)

Mihaela Sabin Ph. D., Professor (Applied Engineering & Sciences)

Michael Jonas Ph.D., Associate Professor (Applied Engineering & Sciences)

On November 28, 2022

Table of Contents

Abstract

Automated license plate recognition systems make use of machines learning coupled with traditional algorithmic programming to create software capable of identifying and transcribing vehicles' license plates. From this point, automated license plate recognition systems can be capable of performing a variety of functions, including billing an account or querying the plate number against a database to identify vehicles of concern. These capabilities allow for an efficient method of autonomous vehicle identification, although the unmanned nature of these systems raises concerns over the possibility of their use for surveillance, be it against an individual or group. This thesis will explore the fundamentals behind automated license plate recognition systems, the state of their current employment, currently existing limitations, and concerns raised over the use of such systems and relevant legal examples. Furthermore, this thesis will demonstrate the training of a machine learning model capable of identifying license plates, followed by a brief examination of performance limitations encountered.

SECTION 1 – THESIS INTRODUCTION

1.1 Topic Introduction

Machine learning and artificial intelligence are fields that have seen considerable growth in the past decade. In part powered by advances in computational power and data accumulation, artificial intelligence is being seen employed in a greater number of tasks perhaps once seen as improbable for a machine to assume. Powered by artificial intelligence, computer vision tasks rely on a model's ability to classify images or detect and identify objects within images. One such task is license plate identification and transcription, which is conducted by automated license plate recognition systems.

A computer vision application, automated license plate recognition systems take in image data and outputs predictions in regards to the license plate's number. The uses of these systems vary, from police utilization to faster identify vehicles in question, to the next evolution in allocating tolls and enforcing traffic violation fines. The automated nature of these systems makes license plate number identification faster than manual review, as well as being able to identify every vehicle that passes by, and it's for this reason that the technology has seen opposition. Far from achieving perfect detection and transcription performance, automated license plate recognition systems face difficulty under adverse conditions, a fact that needs to be taken into consideration when employing such a system.

1.2 Thesis Objective

This thesis aims to explore the components that run automated license plate recognition systems, image data and machine learning, as well as the systems themselves, including uses, concerns, and technical limitations. Furthermore, this thesis will document the creation of a simple license plate recognition model and make use of said model to demonstrate the limitations such a model faces in real-world use. Through examining these facets of automated license plate recognition systems, this thesis contributes to the understanding of how these systems work, how and why they are employed, and the legal and ethical concerns that arise from their usage.

1.3 Thesis Overview

- **Section 2** deals with the background information that concerns automated license plate recognition systems, including image data and machine learning. General overviews of both topics with be covered, with the intent to inform the reader as to the technology behind such systems.
- **Section 3** covers automated license plate systems, including their usages, concerns over privacy and surveillance, current legal limitations, and known limitations they face such as adverse weather and low light environments, such as nighttime.
- **Section 4** details the training of a license plate detection model using the open-source object detection library Detectron2. Data set performance results are detailed, as well as testing in known limitations and a discussion as to potential workarounds, if any exist.
- **Section 5** makes recommendations as to further research or exploration to be carried out as a result of the evaluations in **Section 4**.
- **Section 6** concludes the thesis.

SECTION 2 – BACKGROUND

Before diving into automated license plate recognition systems, there exist two topics that should first be covered to best ensure a proper understanding of the technology. The first topic being image data, the medium from which automated license plate recognition systems are able to identify and transcribe license plates, and the second being machine learning, the driving force behind automated license plate recognition systems.

2.1 Image Data

Digital cameras, video games, websites, and more take advantage of the advances in the storage and representation of image data to better express and represent their respective mediums, but the complexities of such data is much greater than simply representing a mixture of red, green, and blue lights. Examining image data as it's represented on a machine as well as how image data is displayed on screens creates a greater understanding of how the aforementioned mediums make use of image data.

2.1.1 Introduction to Image Data

Image data and digital images are used in nearly every aspect of modern technology, including graphical user interfaces, video games, and even movies. Despite its prevalence, a solid understanding of image data is surprisingly uncommon, and while color theory such as additive colors may be understood, the more complex facets of image data often go unrecognized, as modern compression algorithms rely on advanced mathematics and an understanding of a human’s perception of color to achieve optimal performance. True, the fundamental concept of image data revolves around the mixing of the colors red, green, and blue, however, to simply end one’s knowledge there is a great disservice to the complexities and accomplishments of modern methods of storing and representing image data. With that aside, to begin discussing image data, the ideal starting point would be that of general color theory, and in that how computers can create over sixteen million different colors by combining 24-bits representing red, green, and blue. With the understanding that the human eye can only differentiate ten million different colors, 24-bits allows computers to be able to store and display more color than the human eye can recognize **[1]**.

Computers utilize monitors to display graphics to the user, and from the user's point-of-view, graphics are made up of thousands, even millions of pixels, and often seem to flow naturally. While monitors are capable of displaying millions of different colors, the driving force behind this capability is simple. Behind each pixel on a screen are individual red, green, and blue lights, and the colors we perceive on a screen are actually blurred mixtures of red, green, and blue lights at different levels of intensity. Black presents itself with all three lights being turned off, while white is visible by having all three colors omit maximum brightness. Similarly, yellow can be made with just the green and red lights, and this seemingly optical-illusion can be seen when a pixel's individual lights become discernible from one-another **[2]**.

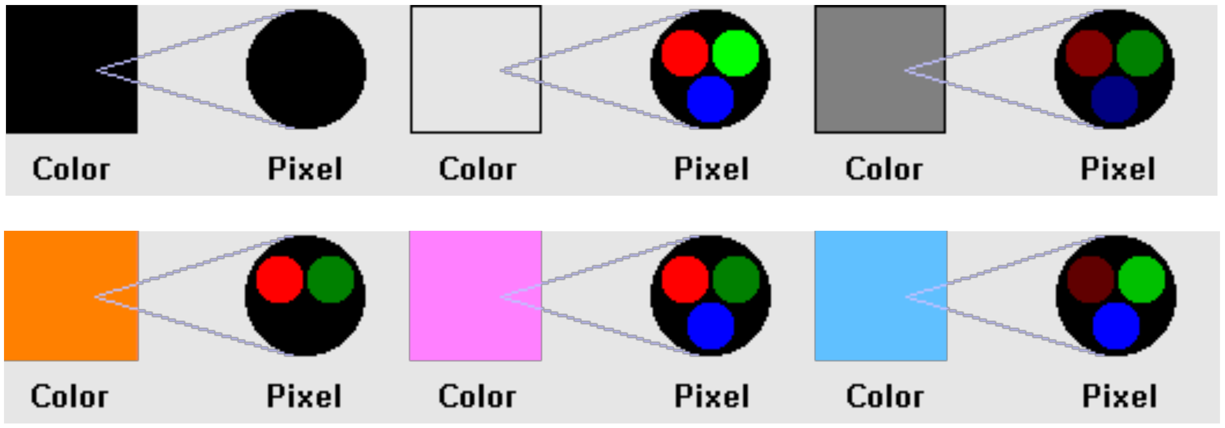


Figure 1. Example colors as a result of mixing red, green, and blue lights **[3]**.

It should be noted that computer monitors use RGB as the basis for all colors, while printers rely on cyan, magenta, yellow, and black. This is due to RGB being additive colors, while CMYK are subtractive colors, with the difference being additive colors will become whiter as the colors intensify, while subtractive colors become darker as the colors intensify **[4]**.

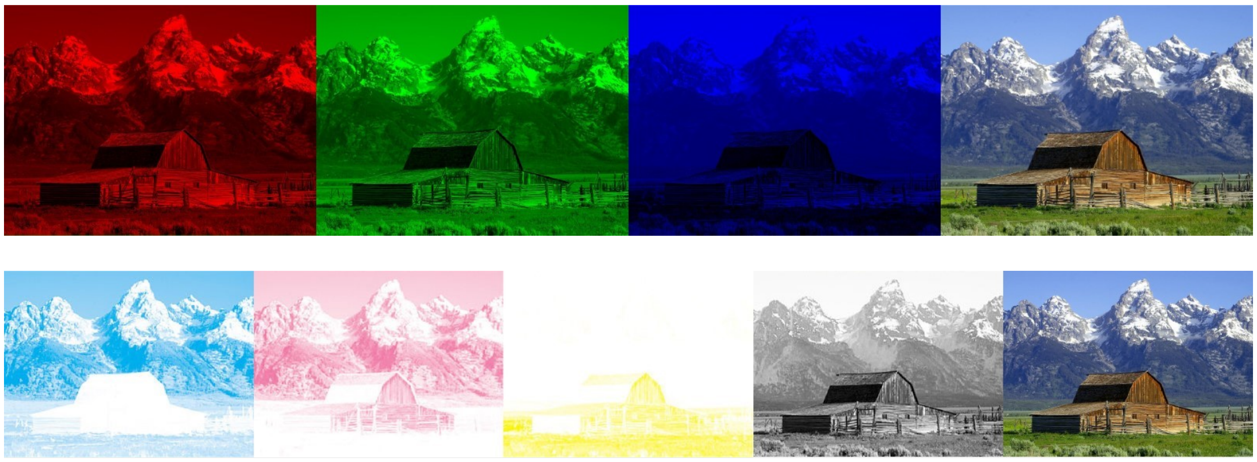


Figure 2. A picture displayed used RGB and CMYK **[4]**.

Several different formats can be used to denote a pixel's color, however computers will resolve the binary to the same values. At the highest level, colors can be written as 'red,' 'orange,' 'yellow,' 'green,' 'blue,' 'purple,' and so forth. This, naturally, becomes infeasible at a certain point, as naming sixteen million different colors is a feat near-impossible, and remembering all of them even more so. However, in limited applications, such as mock-up CSS, names can be used to note colors. A more common and popular format used to denote color is with hexadecimal as it more closely relates to the underlying binary. While the format used to save the image will determine the range of colors available, 24-bit color is a frequently used format that allocates one byte (8-bits) for red, green, and blue, respectively. Rather than writing out twenty-four zeros and ones, hexadecimal shortens this to six characters, ranging from 0-9, A-F, with every two characters representing a different underlying color. Hexadecimal notation is frequently prefixed with the '#' symbol, although the prefix ‘0x’ can also be used to denote hexadecimal. #FF0000 represents pure red, #00FF00 pure green, and #0000FF pure blue. The maximum of each color, FF, resolves to the decimal 255, and while decimal representation can and is used, the two are often interchangeable. The number 255 is used as this is the decimal equivalent to 1-byte being all 1's, or 11111111, which in binary equates to 255 (2^8 - 1), although 0 is reserved for no color, which allows for the range of 0-255.

2.1.2 Image Data Formats and Storage

Two of the most well-known and popular image data formats are the portable network graphics format, PNG, and the joint photographic experts group format, JPG, although the bitmap format, BMP, can also commonly be found. In general, PNG and JPG differentiated by being either lossless or lossy. In other words, a PNG retains the original value of an image, while a JPG loses the original image over time, and can be seen in the artifacting of images on the internet. A PNG, when downloaded, uploaded, downloaded, and so forth, will look the same as when it was first uploaded, while a JPG may begin to look 'blocky,' and lose the sharpness of edges.

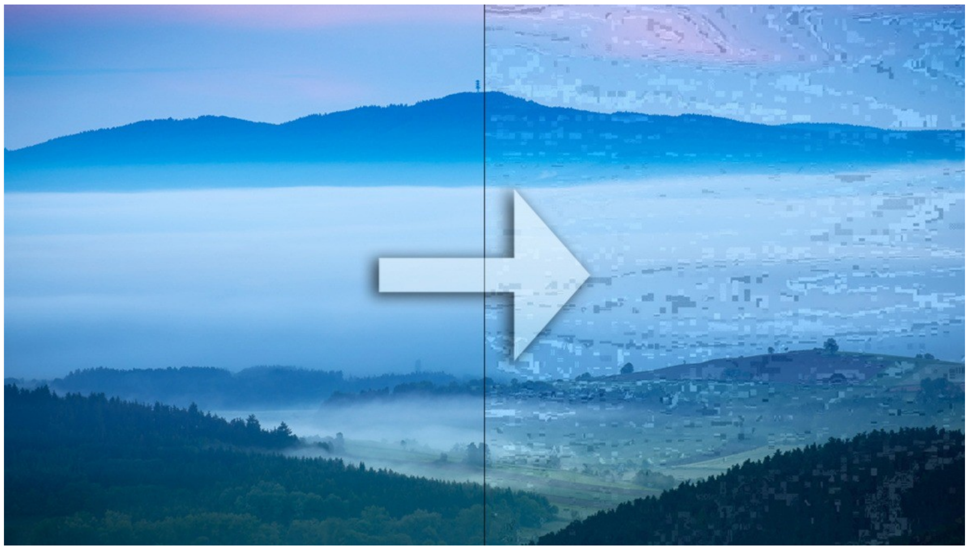
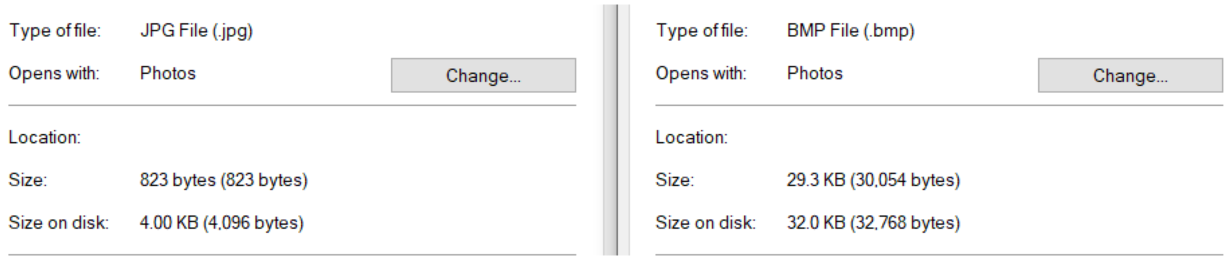


Figure 3. Portable Network Graphics versus Joint Photographic Experts Group formats **[5]**.

While this comparison is not entirely inaccurate, it fails to capture the complete picture. A PNG, while lossless, utilizes complicated algorithms to compress the image to a more manageable size. A true lossless format, without any compression, exists in the format bitmap, or BMP. A bitmap applies no compression algorithm, meaning a 24-bit bitmap will reserve 3-bytes for every pixel in the image, and due to this, a bitmap can swell in size quickly. A 100x100 bitmap, regardless of the colors used, will cost at a minimum 30kb, while a 100x100 JPG with all black (#000000) pixels costs just 823 bytes, or a little under 1kb. Another key difference to be noted is the support for transparency, which can be found in PNG, and not BMP **[6]**.



Type of file:	JPG File (.jpg)
Opens with:	Photos
Location:	
Size:	823 bytes (823 bytes)
Size on disk:	4.00 KB (4,096 bytes)

Type of file:	BMP File (.bmp)
Opens with:	Photos
Location:	
Size:	29.3 KB (30,054 bytes)
Size on disk:	32.0 KB (32,768 bytes)

Figure 4. JPG vs Bitmap file sizes, as sourced from a local Windows 10 machine.

An alternative to these three formats is a vector file, which rather than being based on storing pixels, instead stores mathematical equations that create the image when drawn, often by way of shapes. While uncommon, vector images find some use in web design, as the mathematical principles allow the images to scale with the page, often helpful with providing support for a range of screens. One such file format that takes advantage of vector properties is the scalable vector graphics file, or SVG **[7]**.

With all this said, more goes into a file format than a mere file extension, and as such, renaming image.png to image.jpg will not convert the image from lossless to lossy. Similar to many other file formats, image data file formats are prefixed with unique codes that tell an application what file they are. While these file signatures can be seen in binary, for the sake of readability, they will be presented in hexadecimal. A JPG file could see one of several signatures, some including FF D8 FF DB, FF D8 FF E0, and FF D8 FF EE. A PNG contains the file signature 89 50 4E 47 or 0D 0A 1A 0A, while bitmaps contain a shorter file signature of 42 4D. Some applications can convert a file from one format to another, however the mere act of renaming a file extension does not change the underlying format.

PNG and JPG image file formats rely on compression algorithms to keep an image's size smaller than a bitmap, and for this reason are differentiated from raw bitmaps. JPG obtains its compression in part due to how the human eye perceives color, however this same reasoning also leads to the lossy nature of a JPG image. Using cyan, magenta, yellow, and black (CMYK) as the base colors, JPG decreases file size by reducing cyan and magenta values. This is based on the understanding that the brain is more sensitive to changes in yellow (or more precisely, luminance, in short, the brightness of an image) values than in cyan or magenta. While the actual, raw color may change, the human brain is likely to not notice the difference. JPG will take a block of 8x8 pixels, compress the cyan and magenta filters, and move on to the next block, until the entire image has been iterated upon. This method of taking 8x8 pixels at a time is ultimately what gives JPG the blocky artifacts that can be seen in heavily compressed images **[8]**.

PNG compresses images based on pixels' relations to each other. While scanning image data, PNG looks for mathematical relations between the pixels, specifically the RGB values, and will apply a filter to the entire line, however doing so to each value independently. A great oversimplification of a complex algorithm, the following can be taken as an example. A line of pixels contains the decimal values 100, 90, 80, 70, and 60, and contain the same values for red, green, and blue. Rather than store these larger numbers, PNG identifies a descent of -10 in the value of each pixel and will instead save the line as 100 -10 -10 -10 -10. When that PNG file is opened, the original values are restored by adding each subsequent value. The actual PNG compression algorithm is far more complex than this example, which can lead to two alike images having two wildly different file sizes, however at a basic level, this relationship between pixels and their individual color values is what drives the algorithm **[9]**.

2.1.3 Bitmaps

In comparison to the complex algorithms that go into the compression of JPG and PNG images, bitmaps rely on a simpler one-to-one ratio between pixels and bits stored. Bitmaps can be written by hand using a hex editor, requiring only a file header and the pixel data itself. The most drastic of differences between bitmap formats can be attributed to color availability, ranging from monochrome and grey-scale to 'true' color, or more colors than the human eye can differentiate. While monochrome and grey-scale can be accomplished using true color bitmaps, there exist options that enforce these characteristics, and in fact greatly reduce the size of the file. With the computational power and storage capacity of modern computers, saving bytes isn't as big as issue, so often bitmaps will be saved as 24-bit bitmaps, or the aforementioned true color. A monochrome bitmap's image is stored, much the like color, as either 'color' or 'no color,' which maps to either 1 or 0, and as such, are known as 1-bit bitmaps, as one bit represents one pixel. Had a 1-bit bitmap been converted to a 24-bit bitmap, then 24-bits would be required to represent each pixel, or #FFFFFF and #000000, ballooning the file size by a factor of twenty-four. Some other options for bitmap color depth include 4-bit, 8-bit, 16-bit, and 32-bit. While the only real difference between 4-bit, 8-bit, 16-bit, and 24-bit is the number of bits reserved for color, 32-bit bitmaps introduce an extra byte for the alpha channel, more commonly known as transparency. 32-bit bitmaps are fairly uncommon, as support for the added alpha channel is not universally supported, instead favoring PNG's transparency **[6, 10]**.

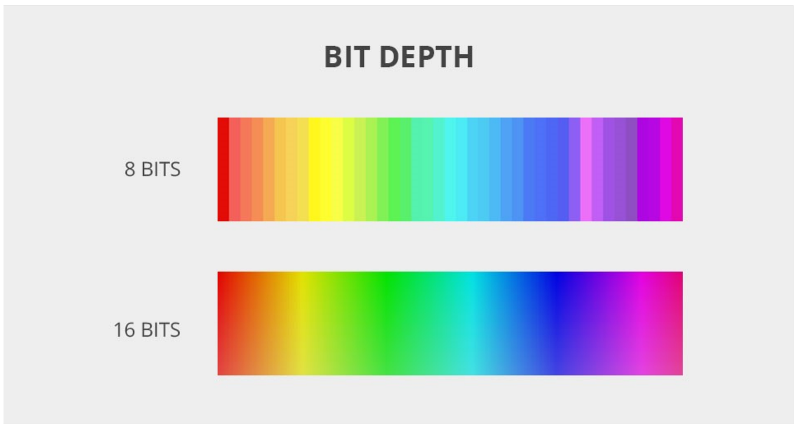


Figure 5. 8-bit versus 16-bit depth comparison **[11]**.

In order to properly set the color depth of a bitmap, the file header must be written out so as to tell the computer exactly what it's looking at. A bitmap header is approximately 40-bytes in length, although this can vary slightly depending on the settings written out. The file header will be written out automatically by editing applications, however if a hex editor is used, a bitmap can be written by hand. The following binary will be represented in hexadecimal, but the same can be accomplished if written as the binary equivalent. All bitmaps will begin with 42 4D, which specifies that the following data belongs to a bitmap file. Several bytes must then be reserved for the file size, with subsequent bytes being used by the image editor as needed, although this has no effect on the bitmap itself, and can be left as 00 00 00 00 if created in a hex editor. In order to differentiate the header from the image data itself, the length of the header in bytes must then be specified. Next is the width and height of the bitmap in pixels, with two bytes being reserved for each, allowing for a maximum width and height of 65,536 x 65,536 pixels, which in 24-bit color, requires over a hundred billion bits, or a staggering 12.8 GB. In those circumstances where large dimensions are required, formats supporting compression should be preferred. Input relating to colors is then required, with the most important piece being the number of bits reserved for each pixel, 00x18 for 24-bit color. Total colors and important colors can also be specified, although these are often left blank, or 00 00 00 00 for both. While compression algorithms can be set at this point, some bit-depths, notably 16 and 32-bit, do not support compression. The length of the header should now match what was previously specified, so image data can now be written out. By default, bitmaps read in image data from bottom-up, left-to-right, meaning the first line of pixels in the file correlates to the bottom line of pixels in the actual image.

A 24-bit bitmap with a width and height of 4x4 will have the following header:

```
42 4D 4C 00 00 00 00 00 00 00 1A 00 00 00 0C 00 00 00 04 00 04 00 01 00 18 00
```

With the header completed, finishing the bitmap is as simple as writing out the colors of each pixel while abiding by the color depth, width, and height specified in the header **[6, 10, 12]**.

2.2 Machine Learning

Machine learning, despite its recent surge in usage, has existed for decades, but has only within the last decade or so become viable thanks to the rapid increases in computational power and data availability. As the driving force behind many modern-day artificial intelligence systems, a minimum understanding of machine learning is required to understand how an AI system functions and why certain limitations may exist **[13]**.

2.2.1 Introduction to Machine Learning

At its core, machine learning is about minimizing loss for a given problem. There are various algorithms and techniques encompassed by the overarching topic of machine learning, from the explainable decision tree to the opaque convolutional neural networks, but without fail these algorithms focus on minimizing a loss of some sort to create a model that can reliably create an accurate output for a given input. The actual inputs, outputs, and loss metrics depend on the problem at hand, as predicting housing prices requires a vastly different loss metric from large language models **[14]**.

Creating a machine learning model is called training, and itself can be supervised or unsupervised. In a supervised train, the data set is labelled so as to tell the algorithm of choice the input features and the output feature(s) to optimize the model for. Continuing with the example of predicting a house's price, the input features could consist of the number of bedrooms, number of bathrooms, location, and so on, while the desired output would be the house's price. As this is a regression problem, a metric for loss could be the mean squared error between the predicted price and the actual price. In machine learning, two common tasks are classification and regression. While classification is concerned with correctly labeling the input as one or more classes, regression is focused on creating an equation that fits the train data to a certain degree. Given that the output, the house's price, is expected to be correlated with the input, the house's rooms and amenities, the machine learning model would create an equation with these features weighted appropriately, with each weight being determined through a series of iterations that seek to minimize the loss. The end result of the training is a model with weights trained for predicting the prices of houses that don't fall strictly within the data set. **[14, 15]** An issue arises, however, if the input data is too far removed from the data in which the model was trained on, for example a west coast city versus an east coast city.

While there exist a number of machine learning algorithms, for illustrative purposes a simple algorithm, K nearest neighbors, will be used as an example of how machine learning works. The K nearest neighbors algorithm is perhaps the easiest to understand – given a set of data and an input point, return the majority output found within the K nearest neighbors, where K is an arbitrary number set by the developer. For example, if K=3, the majority consensus from the three nearest data points will be returned as the predicted class output. As visualized as a scatter plot in **Figure 6**, K nearest neighbors scopes out values that fall within certain classes and predicts output classes for new inputs falling outside of the train data set. Although not as simple to visualize given a greater number of features, K nearest neighbors allows for predicting simpler tasks with close correlation to one another, such as adverse weather and public transportation delays **[16]**.

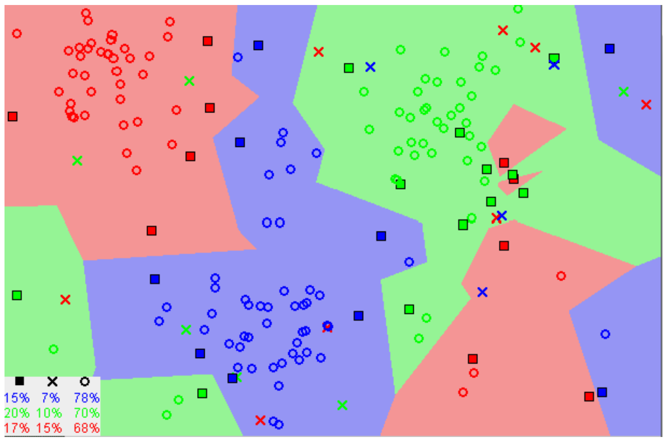


Figure 6. A visualization of K nearest neighbors **[16]**.

2.2.2 Neural Networks

Neural networks, sometimes referred to as deep learning, are made up of artificial neurons spanning a given number of layers. Each individual neuron takes in some input, combines it with its known weight or bias, and creates an output. With larger or fully connected layers, individual neurons can take in tens, hundreds, or even thousands or more inputs and must determine the output accordingly. Neural networks make use of a chosen activation function to decide a neuron's output – such examples include the rectified linear unit, sigmoid, and hyperbolic tangent activation functions. The end result of the neural network is the model's prediction, which varies depending on the task at hand. **[17, 18]** For image classification, the output is the predicted image class, for object detection, the output typically consists of bounding boxes for any predicted targets. Naturally, more complex tasks require more neurons to interpret the data at hand, resulting in neural networks with fifty or more layers, with total parameter counts reaching into the millions **[19]**.

A neural network can learn to make accurate predictions on train data by way of back propagation, which is itself controlled by gradient descent. Gradient descent is an optimization algorithm that allows a neural network to iteratively adjust its weights as it sees more data, with the goal of minimizing loss, or incorrect predictions. The steepness of the gradient descent is managed with the learning rate parameter, with larger learning rates correlating to a steeper gradient descent. The benefit of using a larger learning rate is the decreased training time, however the tradeoff is that the model has a higher chance of missing the best possible loss minimization. Alternatively, a smaller learning rate may take longer to train, but will likely have less training loss. **[20, 21]** After each iteration, the model adjusts its weights in accordance with the steepness of the gradient descent, with back propagation handling these changes. When the data is fed into the neural network, each neuron calculates its output from the input and passes it forward. With the error calculated upon prediction, the neural network adjusts its weights from the back layers first so as to attempt to generate more accurate outputs for the next iteration. This is also why overfitting occurs – with a limited amount of data, or when trained on the same data points too many times, the neural network essentially creates an extremely precise set of weights that perform excellent on the train data but is too specific to perform adequately on new data **[22]**.

2.2.3 Neural Networks and Image Data

Tasks relating to image data typically require deeper networks due to the complexity of extracting features from an image, which to the model is nothing more than an n-dimensional array. The input layer breaks down the image into smaller pieces that get passed through the network, with the simplest features being identified first before building up more complex ones through the network's neurons putting together what it believes the image to be of **[23]**. As seen below in **Figure 7**, the example neural network begins by breaking down the data into a set of features, looking for orange and blue points on the left and right, top and bottom, and so on. As the neurons receive the outputs from previous layers, more complex features begin to arise until finally the output layer contains the most complex features that are put into the predicted output. While oversimplified, the blue and orange dots can represent objects within an image that the neural network is trying to learn to identify **[24]**. In some implementations of object detection, such as Detectron2, the library used for the experiments in **Section 4**, the features extracted by the neural network are fed into a secondary region proposal network that attempts to piece together the features into bounding boxes and instance segmentations, if applicable.

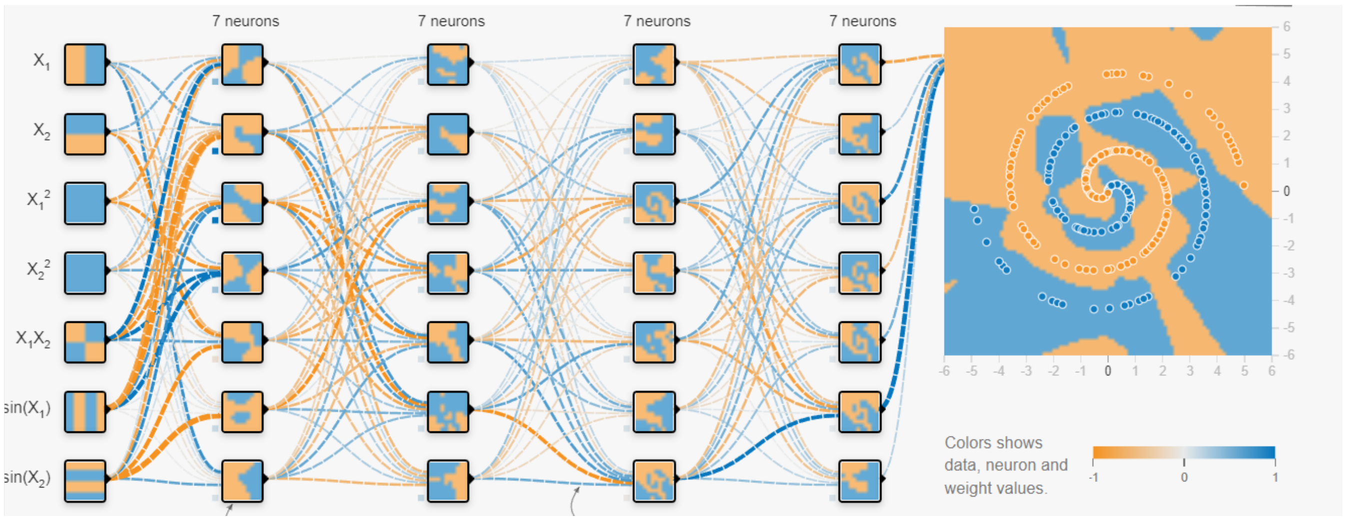


Figure 7. A neural network identifying features in data **[24]**.

SECTION 3 – AUTOMATED LICENSE PLATE RECOGNITION SYSTEMS

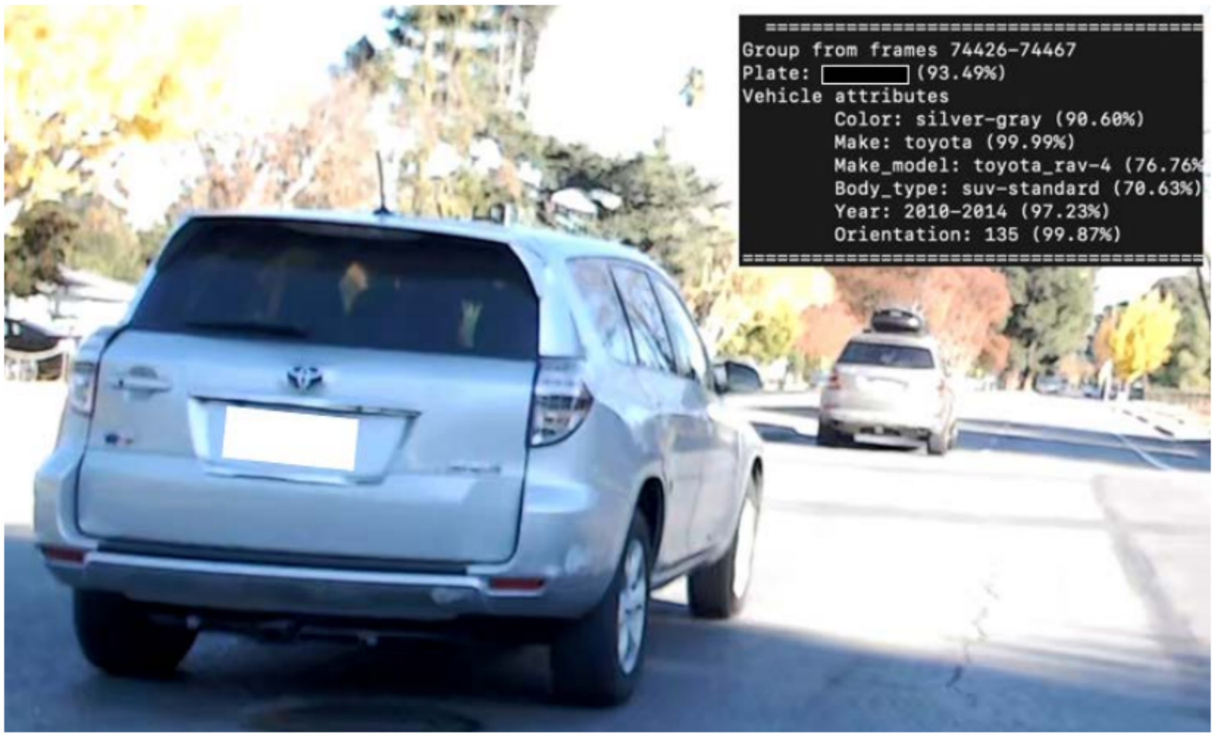
Automated license plate recognition systems make effective use of machine learning and large quantities of image data to identify and transcribe license plates within images. Although the actual implementations may vary, in general these systems are used to reduce the number of manhours required to sift through images of license plates, be it for looking for certain plate numbers or just transcribing the plate number for another use. Despite the seeming benefits of automated license plate recognition systems, they aren't without their own set of limitations and concerns – primarily when it comes to individuals' privacy and the increased ability for operators to use automated license plate systems as a form of surveillance.

3.1 How it Works

Automated license plate recognition (henceforth ALPR) systems can be fielded on a number of hardware platforms and can be found as both stationary cameras or mobile systems such as mountable cameras and even smartphones. This versatility allows ALPR to be made use of in most locations where needed. Both private and public sectors field ALPR, with a notable public sector use being in that of policing. While not every ALPR system is identical in capability, a general commonness between them is the ability to detect license and transcribe license plates in images and return that text transcription in some form or another. Many systems include a database, be it localized to the computer it runs on or larger, regional databases of license plates **[25]**.

The U.S. Department of Homeland Security's Science and Technology Directorate is working with the private sector company Synthetik Applied Technologies to produce

DeepVIEW, an ALPR system aimed at providing law enforcement with a powerful tool for not just identifying license plates but also identifying the characteristics of the vehicle as well. As ALPR systems in use can vary from each other in terms of features, DeepVIEW will be used as an example of what current ALPR is capable of. The goal is to have DeepVIEW low-cost to law enforcement by making use of existing hardware in-use and even the smartphones carried by officers on a daily basis. In addition to transcribing license plates, DeepVIEW is capable of identifying a vehicle's color, make, model, body type, approximate year made, and even the orientation of the vehicle in the image. The intent behind providing these additional characteristics is to allow for law enforcement to more rapidly identify particular vehicles beyond just the license plate number. DeepVIEW will also be able to connect to external databases for even faster vehicle identification **[26]**.



```
=======================================
Group from frames 74426-74467
Plate: [redacted] (93.49%)
Vehicle attributes
        Color: silver-gray (90.60%)
        Make: toyota (99.99%)
        Make_model: toyota_rav-4 (76.76
        Body_type: suv-standard (70.63%)
        Year: 2010-2014 (97.23%)
        Orientation: 135 (99.87%)
=======================================
```

Figure 8. Example scan from a prototype build of DeepVIEW **[26]**.

3.2 Applications

As ALPR systems provide for rapid identification of vehicles by license plate, and in some cases even make and model, it stands to reason that such technology has seen a respectable amount of use. As demonstrated by the U.S. Department of Homeland Security's push to field a common ALPR to law enforcement, the value of using ALPR in policing has been noted, however another use case exists by way of traffic enforcement.

3.2.1 Toll Booths and Traffic Enforcement

Toll booths across the nation make use of RFID tags to maintain and properly bill accounts as vehicles cross by. First fielded in the 1990s, E-ZPass and similar RFID systems revolutionized the toll system, and in some cases now support entirely cashless tolling, a feat once thought of as a futuristic concept **[27]**. With the growth of deep learning in the past decade, ALPR has started to see more use in tolling, although not quite to the level of RFID systems. Typically, ALPR is used alongside RFID tags, and is used for the purpose of quickly identifying and enforcing toll payment violations, although the option of using ALPR for billing has become increasingly available across the United States. Part of the hesitance in adopting ALPR as the sole executioner of tolling is the need for perfect accuracy, and as of 2021 commercial ALPR systems maintain an accuracy of around 98% under good conditions. While far from poor performance, the 2% of cases where an ALPR is unable to transcribe a license plate requires a human to manually review and audit the images, resulting in higher costs. It's for this reason that in areas where ALPR is offered as an alternative to RFID that the option generally boasts a higher fee to offset the inevitable need for manual reviewers **[28]**.

RFID systems continue to remain the dominant system for tolling, however, ALPR can offer benefits beyond identifying a license plate number for billing purposes. Commerical ALPR systems work to integrate with existing RFID systems, allowing for the cross-checking of license plates with the identified RFID tag, aiding in the prevention of toll fraud. Some ALPR systems go beyond the minimum requirement of transcribing a license plate and can even identify various markings on vehicles such as those which denote the carrying of hazardous materials, among others **[29]**. Outside of tolling applications, stationary ALPR cameras have been used to enforce red-lights, speed limits, and assorted traffic violations. Being simple to set up and with high enough accuracy for the infrequency of such violations, ALPR can be utilized as an effective and relatively low-cost method of enforcing minor traffic violations and recouping their costs in fines **[30]**.

3.2.2 Policing

Most police departments make use of ALPR systems in one way or another, and a majority plan to expand their usage of ALPR in the future **[31]**. For police and law enforcement, ALPR systems aid in finding stolen vehicles as well as identifying vehicles that may be delinquent on traffic violations or linked to criminal activity. Those ALPR systems used by law enforcement can include both mobile and stationary cameras, and many times can be operated from a patrol car. In other cases, ALPR is run from a temporary setup, such as on top of traffic lights, road signs, and the like. This diversity of deployment methods makes ALPR effective in finding vehicles of concern, in fact, one study noted that ALPR usage led to tripling the number of stolen vehicles found and doubling the number of vehicles returned **[31]**, all the while requiring less police manhours to operate.

As many ALPR systems collect data on the geographic location, date, and time for each license plate transcribed, police are able to pick up on patterns of when suspect vehicles drive by certain locations, which can then be used to track down more serious crime, such as trafficking or even terrorist activity. This is in part enabled by the long-term storage of when and where license plate numbers were captured. Local police departments can then collaborate by way of their databases of license plate scans to further enable tracking of suspect vehicles, and this concept extends to the regional and even national level as well, ensuring that a vehicle of concern won't slip out of jurisdiction of the local police department **[31]**.

3.3 Concerns

Although ALPR has greatly aided police and law enforcement in their investigations into stolen vehicles and ongoing criminal activity, it's for these same reasons that have benefited law enforcement that have also raised concerns about the technology. For as seemingly small a piece of information a license plate, location, date and time may appear, quite a detailed recount of an individual or even a group can be built, be it by police or malicious actors.

3.3.1 Surveillance

As previously mentioned, police and law enforcement are able to take advantage of ALPR to track vehicles of concern and even survey the comings and goings of areas with suspected criminal activity. By the nature of ALPR, all license plates detected are transcribed and recorded, for at the least if nothing more than to check against a database of suspect license plate numbers, sometimes referred to as a 'hot list.' However, many ALPR systems maintain records of all scans, resulting in databases building up large amounts of information on traffic in a relatively short period of time. For example, a review of ALPR systems in the state of New York by the New York Civil Liberties Union reported that a particular police department had recorded 164,043 license plates between April and June of 2011, a mere three-month period. It should be noted that the town surveyed contained a population of just around eight thousand. In another instance, the New York Civil Liberties Union found that on one street, over the course of a week, one license plate appeared twenty-four times, and was enough to deduce when the vehicle, and likely its owner, had stayed overnight **[25]**.

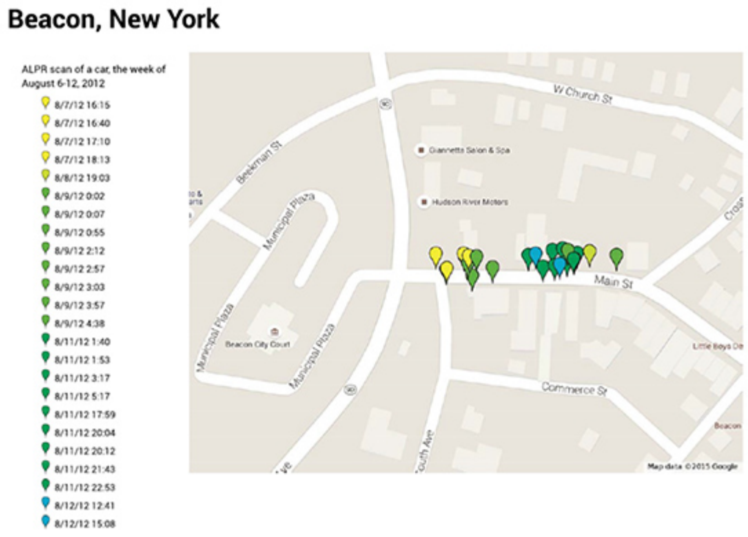


Figure 9. A particular vehicle was detected 24 times in a week on a single street **[25]**.

The argument against ALPR is furthered when it's to be considered that such systems will likely capture and record what would otherwise be private activities for individuals. Individuals that attend pseudo-anonymous rehabilitation meetings such as Alcoholics Anonymous could be consequently identified by ALPR systems, and other medical or hospital-related visits couldn't be guaranteed privacy to a certain degree. Other consequences of widespread mobile ALPR usage would be the potential for surveillance of a given group, as those attending religious gatherings or political protests could be identified with a quick scan of a parking lot, and following future movements would be as simple as registering those license plate numbers as targets of concern that will ping when detected by other ALPR systems. Proponents of ALPR systems argue that while it may be true that data on vehicles' whereabouts are collected, a license plate number (and the vehicle itself) isn't necessarily private information, as anybody crossing by can take note of the license plate number, and information about the individual isn't stored by ALPR systems and databases. Furthermore, ALPR has been proven to be effective in aiding police and law enforcement in their efforts, making the technology worthwhile to adopt. Those in favor of ALPR suggest that with proper oversight and laws regulating the use of ALPR and the information gathered from such systems, widespread implementation can lead to safer communities **[25, 31]**.

3.3.2 Privacy

Alongside concerns about widespread general surveillance, there exist further concerns about privacy given the storage of so much data about license plate scans, locations, dates, and times. As previously mentioned, police departments share data collected from ALPR systems, and the U.S. Department of Homeland Security's DeepVIEW would further this capability **[26]**. This suggests that an individual whose scans appear on a local ALPR system could be visible to law enforcement agencies and departments across the country, despite that individual having done nothing suspect or criminal in nature. This concern is furthered by the existence of private sector data brokers that sell access to their databases amassed from their ALPR systems employed by both private and public sector entities. One such company, Vigilant Solutions, features (as of 2015) a database of 2.2 billion entries and sees an additional eighty million entries per month **[25]**. The New York Police Department has contracted with Vigilant Solutions, allowing them to effectively track a vehicle's travel history or even see where it was last seen in real time **[25]**. With ALPR systems becoming more sophisticated, as evidenced by DeepVIEW, the amount of information recorded on the vehicles scanned will likely increase as well.

In some instances, data collected by ALPR systems can be publicly accessible, as was the case in the New York Civil Liberties Union's Freedom of Information Law (FOIL) request. Although the data received was for informational purposes, given the example in F**igure 9**, the NYCLU was able to recreate part of an individual's whereabouts during over a week, which begs the question what a malicious actor could do with access to the same information **[25]**. Restricting access to these databases could alleviate some of these concerns, but at the same time this gives way to the possibility of making the auditing of official usage more difficult **[31]**.

3.3.3 Legal Challenges

As is so often the case, the technology powering ALPR has evolved faster than the laws surrounding it, and as a result, in many cases there are little to no restrictions in regards to the usage of ALPR and the storage of the data collected. Without laws or regulations, as was mentioned in the previous section, there are cases where any data collected by an ALPR system can be publicly accessible **[31]**. On the flip side, there are also cases where the exact opposite occurs – law enforcement agencies refuse to disclose any ALPR data, as similarly there is no established precedent for releasing such information in the areas they operate. The Electronic Frontier Foundation, a digital privacy advocate group, alongside the American Civil Liberties Union of Southern California, sued the Los Angeles Sheriff's Department and Police Department for doing just this, as the LAPD argued that ALPR data couldn't be released as they were investigative records. The counterargument was made by the Electronic Frontier Foundation and ACLU that this effectively implied that anyone who had had their license plate captured was under investigation, which further meant anyone living in the Los Angeles area could be considered under investigation. The California Supreme Court ultimately sided with the Electronic Frontier Foundation and ACLU and in 2017 sent the case back to a lower court **[30]**.

California is one of the few states, as of writing, with legal protections for ALPR data. The state requires that police departments and law enforcement agencies maintain access logs for their ALPR data as well as requiring usage and privacy policies. Furthermore, California prevents police departments and law enforcement agencies from sharing or selling data collected from ALPR systems to private companies, although the data can still be exchanged among public agencies. Another point of importance is that California requires police departments to hold public meetings to discuss the use of ALPR before the start of any implementation of any kind of ALPR system. Such meetings are intended to keep the public informed about the use of ALPR as well as the effects it may bring about, be it positive or negative **[30]**.

When it comes to law enforcement, an ALPR scan matching a number on a hot list is cause for action, sometimes leading to an arrest. Unfortunately, while ALPR transcription is correct most of the time, there have been cases where the ALPR system incorrectly transcribed a license plate number, inaccurately reporting that a certain vehicle was on the hot list. Those against the usage of ALPR argue that these misidentifications can and will lead to the unfair treatment of innocent individuals. One such event took place in San Francisco in 2009, in which an ALPR scan informed police that a passing vehicle was stolen, when in reality the ALPR had misread the license plate. The driver was handcuffed at gunpoint and had their vehicle searched, before later realizing that there had been an error with the ALPR system used. In response to this event, the U.S. Ninth Circuit Court of Appeals would eventually rule that technology, and thus ALPR, cannot be the sole basis for a traffic stop **[30]**. While this was a major win for opponents of ALPR and similar systems, this regulation only applied to those falling under the Ninth Circuit Court of Appeals.

As of 2015, only ten states have laws in place that either restrict or prohibit the use of ALPR and ALPR data by police and law enforcement, with six restricting law enforcement use and eight limiting the length of time ALPR data can be stored. Three states, Arkansas, Maine, and New Hampshire, outright prohibit the private use of ALPR systems **[31]**. With the growing legal challenges against ALPR and ALPR data, the likelihood of more states adopting some degree of formal regulation has increased considerably since the inception of the technology. Opponents argue that ALPR can be used to either intentionally or indirectly curb 1st amendment rights through the deployment of such systems at political protests, rallies, and religious gatherings. Those in favor of ALPR in response argue that an appropriate middle ground can be found through a degree of regulation, be it regulating the amount of time data can be stored, who can access it, or even outright discarding scans that don't match a number on a police hot list **[30]**.

3.4 Limitations

Although ALPR systems function at a high level of accuracy under good conditions, there are still a substantial number of causes that can degrade performance. Like many other machine learning object detection models, ALPR performance tends to suffer under poor visibility, adverse weather conditions, and overall poor image quality. Perhaps even more concerning than those conditions which cause a drop in overall performance is the emergence of adversarial AI attacks and the resulting damage that is possible.

3.4.1 Lighting

Low light conditions such as nighttime present a predictable challenge for ALPR systems but affect mobile systems more so than stationary ALPR cameras. Stationary ALPR cameras such as those used for traffic enforcement or tolling can incorporate stationary lighting at their locations, a luxury not always afforded to ALPR systems fielded for mobile use. A sufficient lighting setup can help to improve visibility in low light conditions by illuminating vehicles as they pass by, however this fails to serve as a panacea for the issue. Certain license plates and their holders can be highly reflective at certain angles, resulting in an ALPR system failing to accurately capture a license plate number, if detected at all. In these cases, a manual audit can sometimes resolve the issue, however it should be noted that there are occurrences when a human transcriber is equally unable to accurately determine the license plate number **[28]**.

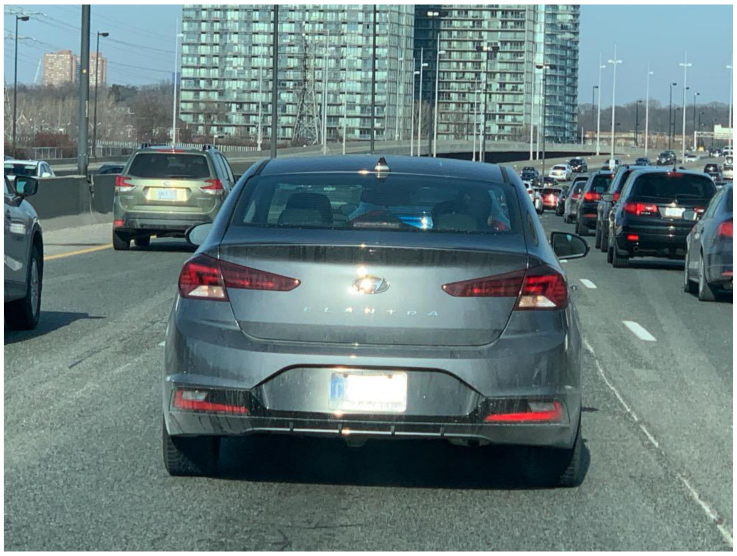


Figure 10. A highly reflective license plate **[32]**.

Mobile deployments of ALPR systems are unable to take advantage of a consistent source of stationary lighting, and as such, have a more difficult time performing at night or under low light conditions. The Department of Homeland Security aims to make this a problem of the past with the development of their DeepVIEW ALPR system, and early testing suggests that the new system will be successful at nighttime detection and transcription **[26]**. One technique currently used to boost performance and accuracy under low light conditions is to create a binary mask based on edge detection, which has been shown to improve an ALPR model's ability to differentiate the license plate number from its surroundings **[33]**.

3.4.2 Adverse Weather

Adverse weather creates a number of issues for ALPR systems and the cameras that power them. Rain, snow, and even fog often create lower than ideal light conditions by way of overhead clouds, but these conditions have the additional effect of creating partial obscuration. Heavy rain or snow can, to an extent, be considered environmental noise, and in the case of snow, accumulation can outright obscure license plates or stationary external ALPR cameras. Fog, on the other hand, can limit visibility, with thicker fog having the potential to prevent ALPR cameras from having a line of sight to vehicles as they pass by.

With rain and snow functionally acting as environmental noise, image processing techniques that are effective in reducing or eliminating noise's negatives effects in object detection find some use in these scenarios. Edge detection has difficulty in identifying license plate numbers due to the increased noise, however unsharp mask filtering an image is effective in improving transcription performance when first provided with a successful license plate detection **[34]**. Additionally, high framerate cameras capture more images of a passing vehicle in a given time frame when compared to a standard camera, increasing the likelihood for a successful detection. Multi resolution analysis of the multi frames captured, combined with contrast enhancement, even further improves the rate of license plate detection **[34]**. In the event that a license plate can be identified, but not transcribed, a manual audit is required.

3.4.3 Image Quality Factors

A machine learning model that involves image data in some capacity will inevitably have its performance intertwined with the quality of the input imagery. While ideally the train data should be similar to the anticipated operational data in terms of signal-to-noise ratio, relative edge response, and spatial resolution, the mobility of ALPR systems frequently results in disparities between the train data and real-world input data. The degree to which changes in the signal-to-noise ratio, relative edge response, and spatial resolution of input imagery affect a machine learning model's performance in context of object detection is a subject of research and debate, however, there exists a consensus that these three image quality metrics remain impactful to a machine learning model's performance **[35]**.

As previously noted, adverse weather such as rain or snow can function as environmental noise in an image. With greater quantities of noise in an image, the more difficult it becomes for both humans and machine learning models to discern one object from another. False positives may also arise as noise distorts features within an image, incorrectly leading a model to believe it has identified a license plate where it may otherwise would not have. As such, when able, ALPR systems should seek to implement cameras with high signal-to-noise ratios, so as to obtain more interpretable imagery and as a defense against environmental noise **[35]**.

Relative edge response, in plain terms, is how 'sharp' or 'crisp' an image appears, as opposed to an image being blurry. With license plates being rectangular in design, a slightly decreased relative edge response may not pose much of a threat to the license plate detection step of ALPR, however the numbers inscribed will rapidly become blurry and difficult for an ALPR system to accurately transcribe. In some cases, a low relative edge response may cause an ALPR system to misclassify various numbers and letters, for example identifying an '8' as a 'B,' and vice versa. Such a problem, gone undetected, can result in erroneous traffic ticketing or falsely reporting a valid vehicle as being one on a police or law enforcement hot list **[35, 36]**.

The spatial resolution of an image effectively explains the degree of detail in an image. In context of overhead imagery, ground sampling distance is used as the metric for spatial resolution, with a one-meter ground sampling distance meaning that one pixel on the ground corresponds to a square 1x1 meters in the real-world. In general imagery, spatial resolution can be correlated with the pixel count of a given image, with the overall same idea as ground sampling distance, although in a less uniform context. A 256x256 picture of a toll booth is said to have a lower spatial resolution than a 512x512 picture of the same toll booth, as finer details can be discerned with the higher resolution image. When capturing imagery for ALPR purposes, the logical goal is to maximize spatial resolution to better support license plate number transcription, although a company needs to keep in mind any potential technical limitations imposed by the neural network used **[35, 36]**.

3.4.4 Adversarial Attacks

Adversarial artificial intelligence or adversarial machine learning is a relatively new field that attempts to find weaknesses and exploits in machine learning models. Due to the statistical opacity of a trained neural network, unlike traditional software, it can be extremely difficult to determine a model's weak points, or if any exist. One known adversarial attack that affects image data-focused machine learning models is the use of adversarial patterns, such as that seen in **Figure 23**. For unknown reasons, certain patterns or patches cause valid objects to become seemingly invisible to the model. Research has been conducted in regards to protecting deployed models against such patterns, including the use of a defensive model trained to identify known adversarial patterns, however, previously unknown patterns evade these defensive models entirely. These patterns are difficult to handle as they can be physically applied to the target as well as introduced into input imagery in the event of a security breach. In addition to universal adversarial patterns that successfully confuse most models, some patterns are more effective given certain targets, most notably stop signs **[49]**. Successful attacks via adversarial patches can allow license plates to go undetected by ALPR systems.

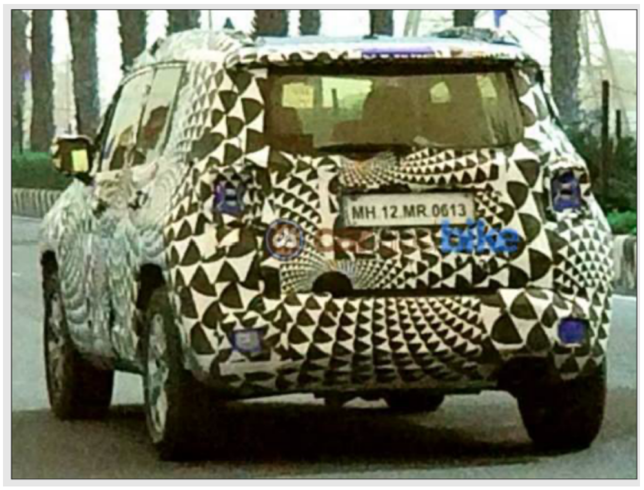


Figure 23. A potential adversarial pattern **[38]**.

Machine learning models can additionally be attacked by poisoning the training data set(s), which serves to insert a trojan undetected into the model. Poisoning a data set consists of applying some degree of distortion, a filter, or mark that appears innocuous to a human viewer but gets picked up on by a neural network, and then labeling that poisoned base class as the redirect target class. Data poisoning is estimated to be successful with 1% of the data poisoned during fully supervised learning, 0.1% for semi-supervised learning, and a stark 0.0001% for unsupervised or self-supervised learning **[50]**. This small requirement would require a mere five images to be poisoned in the data set described in **Section 4.1.2**. Although a single class model would likely raise alarms when a class other than 'license plate' was detected, a transcription model could very easily be poisoned to misidentify characters. Furthermore, an intelligent attacker could attack the model in such a way as to hide the origin of the attack even in the event of the discovery of misclassification by redirecting similar characters to each other. One such protective measure against poisoned images is to apply denoising, although it should be noted that this has been found to decrease model performance **[51]**. Comparing a trojan model's output layer weights to those of a non-trojan model's have been found to be statistically significantly different, however compromised non-trojan models can be studied so as to hide this statistical difference **[52]**. Due to the rapidly emergent nature of adversarial machine learning, new vectors of attack are emerging faster than defenses to protect against them are.

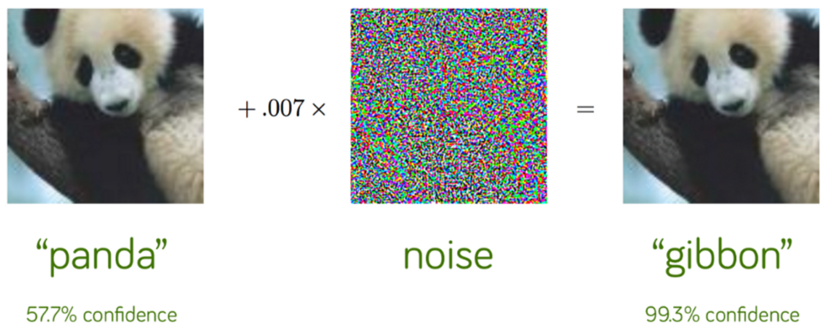


Figure 24. Famous example of a panda class being redirected to a gibbon class **[53]**.

SECTION 4 – EXPERIMENTS

This section is concerned with the training of license plate detection models and their performance across both ideal conditions and those that automated license plate recognition systems typically perform poorly in.

4.1 Model Training

As the powerhouse behind automated license plate recognition systems, machine learning is employed to train a model to be able to detect license plates in images. As with many other image recognition or classification problems, a convolutional neural network is used as the machine learning algorithm for the models created for these experiments. Another important aspect of machine learning is the environment and tools used to create the models, which will also be covered in this section.

4.1.1 Detectron2

Detectron2 is Meta (formerly Facebook) AI Research's premier object detection library, and as the name suggests, is the successor the Meta AI Research's original Detectron. The Detectron2 library is an open-source project, with the source code available at Meta AI Research's GitHub repository **[37]**. The open-source nature of this library makes it ideal for researchers, students, and machine learning enthusiasts, as the library is both free and easily accessible. Additionally, Detectron2 supports both CUDA GPU acceleration as well as CPU-only usage. Written primarily in Python, Detectron2 can be modified to best fit the user's needs without needing to recompile any code. As an object detection library, Detectron2 supports a multitude of capabilities, including but not limited to identifying objects with both bounding boxes and segmentation masks **[37]**.

Installing Detectron2 is fairly straightforward, with the largest hurdle being the requirement of a UNIX-like operating system. Installation of Detectron2 onto a Windows system is possible, however not officially supported by Meta AI, and can introduce complications. For the experiments conducted within this thesis, an Oracle VM VirtualBox running a fresh installation of Ubuntu 20.04.4 LTS was used for preliminary training and evaluation and an Amazon AWS EC2 Instance running Ubuntu 22.04.1 LTS was used for finalized training. Detectron2 is built upon PyTorch and Torchvision, both of which are also Meta AI Research projects, and requires the two to be installed in the operating environment. Although Detectron2 can be built using the system's default environment, it is highly recommended to use a virtual environment to isolate installation versions and avoid dependency issues. A Conda virtual environment was used on the virtual machine, while the Python native module venv was used on the EC2 instance. Although Detectron2 is written primarily in Python, as noted previously, a C compiler is still required to build some portions of the source code. The GNU Compiler Collection, more commonly referred to as 'GCC', was used in both cases. With these prerequisites in place, Detectron2 can then be cloned from the GitHub repository and built using Python's installation manager, pip **[37]**.

4.1.2 Data

The data set used for the training and evaluation of the models within this section is the "Cars License Plate Detection" data set, which is publicly available at the website Kaggle **[38]**. The data set has been curated by Andrew Larxel (username Larxel), a senior data scientist at the Hospital Israelita Albert Einstein, located in São Paulo, Brazil. Larxel is a frequent user of Kaggle, and as of writing has a hundred data sets to his name accumulating over ten thousand upvotes and is the number one data set uploader on the site **[39]**.

The data set consists of four hundred and thirty-three images and an associated extensible markup language (XML) format annotation file for each image. Most images face the license plates directly and are taken close to the plate itself. There exist several images that contain multiple visible license plates, however the annotations occasionally omit bounding boxes for some of license plates. Although this does not cause an issue for the purposes of training the model, it does present an issue when it comes to evaluation. If properly trained, a model should be able to detect the multiple license plates in the image, however automated scoring will erroneously mark the detections as false positives due to the annotations lacking information on these license plates. This negatively affects both the precision and recall scores, and thus the F1 score, as false positives drop the precision, and false negatives may not be accurately tracked, boosting the recall **[38]**.

The XML annotations are denoted in the Pascal VOC format and include notation on whether or not the license plate is truncated, occulated, or difficult to identify. However, this metadata is left as '0', rendering it useless. Usable information within the annotations includes the x and y min and max coordinates for a license plate's bounding box and the dimensions of the image in terms of width, height, and channels **[38]**.

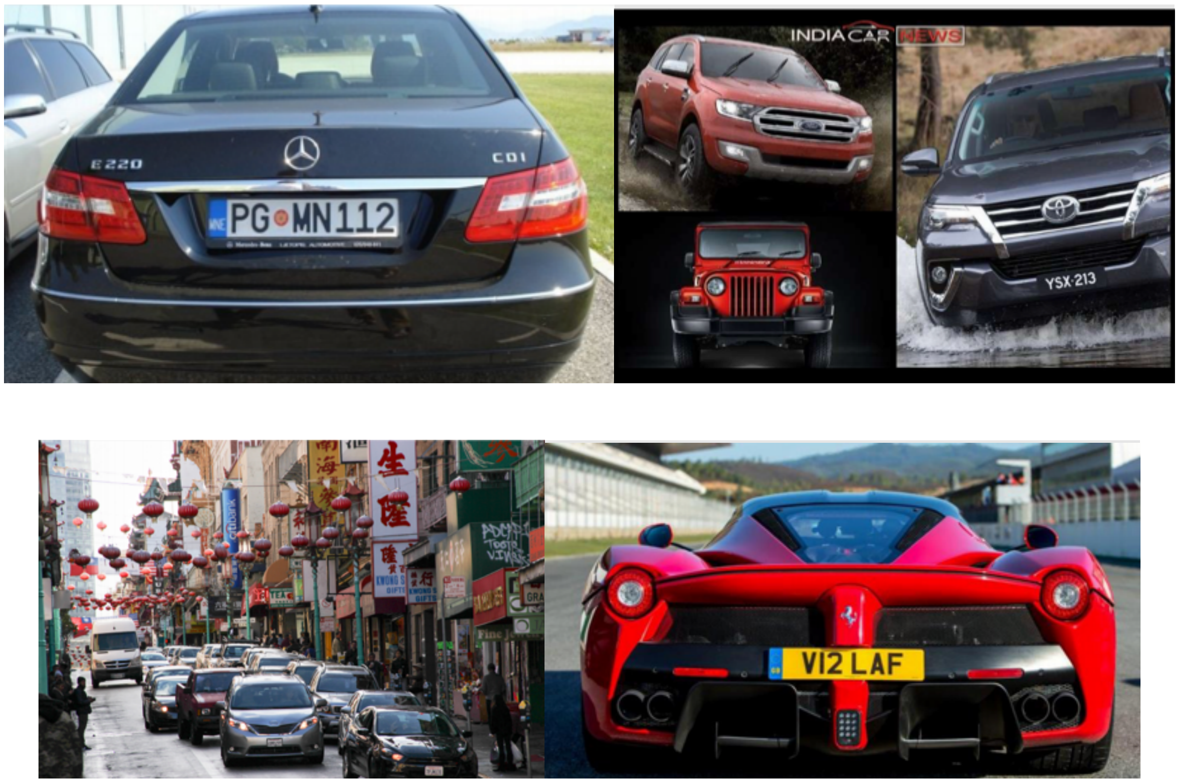
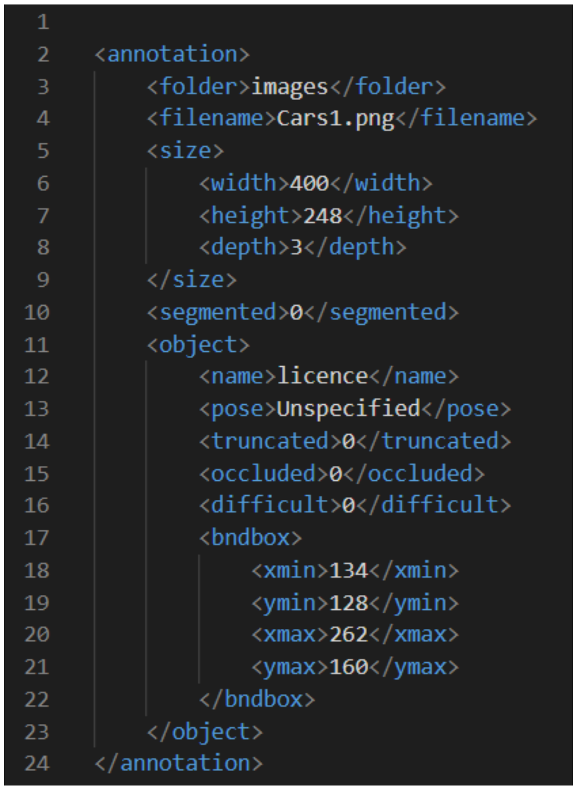


Figure 11. Select images from the "Cars License Plate Detection" data set **[38]**.

The lack of a great deal of metadata makes it difficult to build models that can identify license plates based on a variety of factors and in practice would make for a poor choice of data set if not supplemented with additional data. As for the imagery itself, almost every image features license plates in daylight or some other direct lighting. For purely educational purposes, this data set is sufficient, however the lack of low light or adverse weather conditions makes it unlikely that any model trained solely on the imagery contained within would be robust enough to field in real-world usage. License plates also appear to originate from around the world, which would be unlikely to have any benefit for a model deployed on a U.S. roadway, although in limited quantities an argument could be made that this variety in fact builds a model's resilience towards cars that sport unconventional license plates. These individual factors would need to be assessed on a case-by-case basis, as automated license plate recognition can be employed in a multitude of environments. With the limitations and potential shortfalls of a data set noted, a model's performance can be better understood and can help to guide further changes **[38]**.



```xml

<annotation>
    <folder>images</folder>
    <filename>Cars1.png</filename>
    <size>
        <width>400</width>
        <height>248</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>licence</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>134</xmin>
            <ymin>128</ymin>
            <xmax>262</xmax>
            <ymax>160</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 12. XML format annotations **[38]**.

4.1.3 Converting Annotations

While it is possible to utilize XML format annotations in Detectron2, by default the library requires COCO format annotations. **[37]** COCO, or Common Objects in Context, is a large data set that features, as the name suggests, well-known and common objects in easily understandable context, such as everyday life. The data set annotates the data based on polygonal segmentations, rather than the more commonly found bounding boxes. The popularity of the data set has led to COCO format annotations being widely used for polygonal (otherwise known as instance) segmentation, as both bounding boxes and more complex polygons can easily be represented **[40]**.

In order to convert the XML annotations to COCO, the default annotations are first converted to JSON. This intermediate conversion is taken as it allows for a common format with labels created by the annotation tool ‘labelme’, a popular Python-based technology capable of producing annotations for instance segmentations. Once converted to JSON, the Python script ‘labelme2coco’ then converts the labelme JSONs into a singular COCO format annotation, which is then capable of being loaded into Detectron2 with minimal effort **[41]**.

4.1.4 Model Backbone

Detectron2 offers a variety of backbones to choose from and provides functions for downloading said model checkpoints and their associated configuration files from Meta’s servers. The models produced for these experiments make use of the residual network architecture, more commonly known as ResNet, by way of both the ResNet-50 and ResNet-101 models. Mask R-CNN ResNet-50 FPN and Mask R-CNN ResNet-101 FPN backbones pre-trained for COCO Instance Segmentation and their respective configuration files were used as the starting point for the ensuing training.

The residual network architecture was first introduced in 2015 by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in their paper “Deep Residual Learning for Image Recognition” **[42]**. ResNet was a massive breakthrough for neural networks at the time of its introduction, as at the time networks could only reach so deep before higher error rates started to appear in both the train and test validations **[43]**. ResNet makes use of a new layer known as a ‘residual block,’ which itself is made up of ‘skip connections.’ These skip connections pass gradients deeper into the network by passing over certain layers, which ultimately helps to avoid vanishing gradients, a problem that had previously plagued deeper networks. These residual blocks were proven to be so effective that even a thousand-layer network still functioned properly, **[42]** a massive improvement over the nineteen-layer VGG-19 network. On top of this, ResNet has a lower computational complexity than the older VGG-16 or VGG-19, with ResNet-152 requiring 11.3 billion FLOPs as opposed to the formers’ 15.3 and 19.6 billion FLOPs, respectively **[44]**.

As denoted in their names, the difference between ResNet-50 and ResNet-101 is the number of layers within the network. ResNet-101 adds more 3-layer bottleneck blocks to ResNet-50's architecture, allowing for the network to learn more complex and detailed features of an image or object **[44]**. Detectron2 can implement either network as the backbone of the larger Meta architecture, GeneralizedRCNN, by feeding the extracted features into the region proposal network, frequently abbreviated to RPN. The region proposal network uses the features extracted from the ResNet backbone to identify regions with potential objects, while the ROI Heads (Box Head) takes the RPN’s proposed boxes and fine tunes the ultimate prediction output boxes (or instance segmentations). A simplified diagram of Detectron2’s GeneralizedRCNN architecture can be seen below, **Figure 13 [45]**.

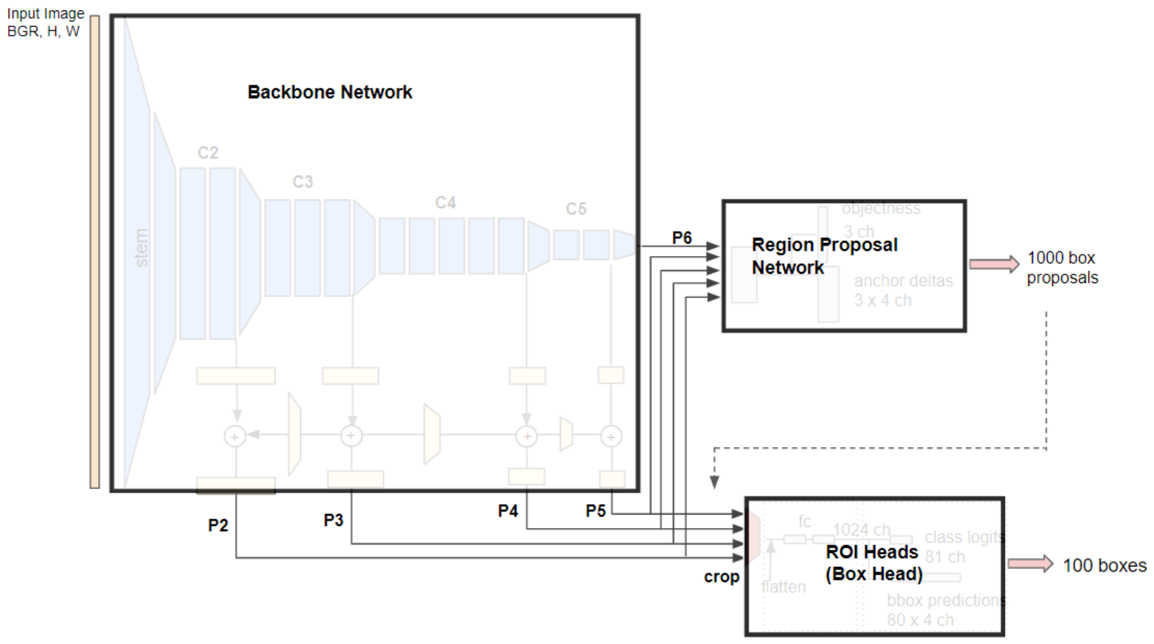


Figure 13. Detectron2 implementation of the ResNet architecture **[45]**.

4.1.5 AWS EC2 Instance

The cloud computing business model has made it possible to leverage the vast resources of giant technology companies for one's personal use. The process to get started using cloud computing is fast and easy, and typically requires only a billing account and minimal setup. While software as a service, or SaaS, is perhaps the most well-known and utilized by consumers, a similar business model exists by way of infrastructure as a service, or IaaS. Amazon Web Services, commonly referred to as AWS, provides a number of cloud computing services, including the aforementioned IaaS. AWS's Elastic Compute Cloud (EC2) service provides users with the ability to a run an 'instance,' which operates as a fully functioning computing environment **[46]**.

In order to create an AWS EC2 instance, a user must first be registered for an AWS account. Once logged in, an EC2 instance can be created by moving to the EC2 dashboard within AWS and clicking on 'launch instance.' The benefit of using Amazon's extensive computing power is the ability to choose from a large number of different hardware specifications alongside many popular operating systems, including but not limited to Amazon Linux, Ubuntu, Windows, macOS, and Debian. With a free account, users get access to a t2.micro instance for a limited number of hours. While not very powerful with only a singular vCPU and one gigabyte of memory, the t2.micro is a great way for users to familiarize themselves with AWS EC2 before running their own instances **[47]**. Once an instance's hardware and operating system has been decided upon, the user can assign or create a key for secure login, establish network settings and firewall rules, and configure storage for the instance. Once this has been completed, the user clicks 'launch instance' and within a few minutes the EC2 instance will be up and running on Amazon's servers **[46]**.

The instance chosen for training the models is the P3 High Performance GPU Double Extra Large EC2 (p3.2xlarge) with an Ubuntu 22.04.1 LTS operating system. The p3.2xlarge was chosen primarily for its inclusion of a NVIDIA Tesla V100 GPU, which is an ideal GPU for machine learning thanks to its power. Along with the Tesla V100 GPU's 16 GB of video memory is an Intel Xeon E5-2686 v4, eight vCPUs, 61 GB of memory, and 70 GB of storage. Although more powerful instances exist by way of the p3.8xlarge and p3.16xlarge, the p3.2xlarge was sufficient for this use-case. The cost to run a p3.2xlarge runs at about $3.00 per hour at time of writing **[48]**. EC2 allows for both SSH and HTTP/S traffic to be permitted for an instance, however only SSH with a personal key was allowed for the p3.2xlarge instance that was created **[46]**.

The time required to set up and train both models was around seven hours, with most of that time being used to properly set up and configure the environment, as well as create prototype models. The final models took a combined total of just under two hours to train, with the ResNet-101 model taking slightly longer of the two. The cost for running the instance for a total of seven hours is estimated to be no more than $25.00. Had an existing setup pre-configured to train Detectron2 models been used, the actual training cost would come in at around $6.00.

The primary benefit to making use of an EC2 instance is the rapid availability of powerful computational resources. A first-time user can have an instance running in around thirty minutes, whereas provisioning on-site resources can take a considerable amount of time longer. Additionally, for those needing an instance with powerful hardware for limited use, the cost savings alone can make EC2 preferable, as many state-of-the-art GPUs for machine learning can run up thousands of dollars. Cloud computing is not without any downsides, however. In the context of machine learning, the data set used can constrict a user’s ability to train on an EC2 instance if it contains proprietary, sensitive, or even classified information. Although an open-source data set was used to train the license plate detection models noted within this paper, had the images of license plates been gathered from, for example, cars that passed through a local highway, the legality and ethics of moving said data onto Amazon’s servers makes EC2 a hesitant choice. Although considered secure, a company or user needs to take into consideration the implications of moving data onto another company’s servers.

4.1.6 Training Parameters

Detectron2 offers a SimpleTrainer class, which is designed to allow users to run trains will minimum setup. The models trained in this section make use of DefaultTrainer, a class that extends SimpleTrainer and accepts a config file as a parameter. A default configuration file for a given backbone can be downloaded from Meta using a built-in function, although users can also load config files from their local computer as well **[37]**. The changes made to the default config file involve modifying Detectron2's data loaders to take advantage of the EC2 instance's power as well as setting the base learning rate to 0.01 and maximum iterations to 10,000. With an 80:20 train/test split, the maximum iteration count equates to just under sixty epochs. The learning rate and maximum iteration count can greatly affect a model's performance, however for the purposes of these experiments the aforementioned configuration will be used.

4.1.7 Scoring Methodology

The methodology used to score the effectiveness of an object detection model can vary depending on the task at hand, and oftentimes there is no one-size-fits-all way of going about it even within the same domain. Comparing the center points of the ground truth bounding box and the model's predicted bounding box is a simple way of determining whether or not the model has made an accurate prediction, however in some cases it can be more beneficial to compare the overlap of the segmentation masks (bounding box or polygon if provided). By measuring the amount of overlap between the segmentation masks, insights as to what the model is or isn't detecting can be gathered, whereas a center point comparison will only return a true or false.

In context of automated license plate recognition, detecting half of the license plate may return a positive detection when comparing center points but fail the transcription itself. With that noted, license plates tend to be rectangular in most instances, which lends itself well to comparing the minimum and maximum x and y points of the bounding box. Another advantage to this approach is the requirement of less computational resources, as comparing segmentation masks is far more taxing than checking several points. However, this now introduces the possibility for error at each of the four points, which then must have acceptable margins of error decided upon. Relying on the singular center point removes this potential issue, although at the cost of absolute precision.

Ultimately, the decision was made to make use of the bounding boxes' center point as the marker for evaluating the accuracy of a prediction. The rationale behind this decision was that the detection of a license plate is only part of an automated license plate recognition system, with the next step being the transcription of a given license plate. With this in mind, being able to accurately pinpoint the center of a license plate indicates a successful detection model, allowing the second step of transcribing the license plate to crop more or less of the detection's bounding box as is determined to be required. For stationary ALPR systems, a fixed crop size can be anticipated based on the camera's distance from the vehicle, however this becomes trickier with mobile systems. As these experiments are primarily concerned with the detection of a license plate in a given image, comparing center points is sufficient.

4.1.8 Results

Both models trained performed to a high degree, with the ResNet-50 backbone achieving an F1 score of 0.940 and the ResNet-101 reaching an F1 of 0.937. While neither reached the performance level of a deployable ALPR model, which tends to sit around a 98% capture success rate, these models demonstrate the ease in which an ALPR system can be quickly created **[28]**. Perhaps surprisingly, the ResNet-50 backbone slightly outperformed the ResNet-101 model, owing in part to the ResNet-50 model's higher precision, whose predictions were 3.7% more accurate, although the ResNet-101 model maintained a 3.2% higher recall. As noted in the data set exploration, there exist cases where license plates appearing in the images are not marked for some reason. Because of this, it is possible that the ResNet-101 model was accurately detecting these license plates but was being erroneously marked as inaccurate predictions. Another possibility is that the ResNet-101 network, having twice as many layers as the ResNet-50 network, overlearned the features of a license plate, causing it to raise more false positives. In a production model, false positives could be reduced by using a secondary model to identify vehicles, which would verify that the license plate detection was valid. This could alternatively create issues with the ALPR model's recall if the vehicle detection model was found to be of poor performance.

Table 1. Results from training license plate detection models.

	ResNet-50	ResNet-101
Total Predictions	91	98
Total Accurate	86	89
Precision	0.945	0.908
Recall	0.935	0.967
F1 Score	0.940	0.937

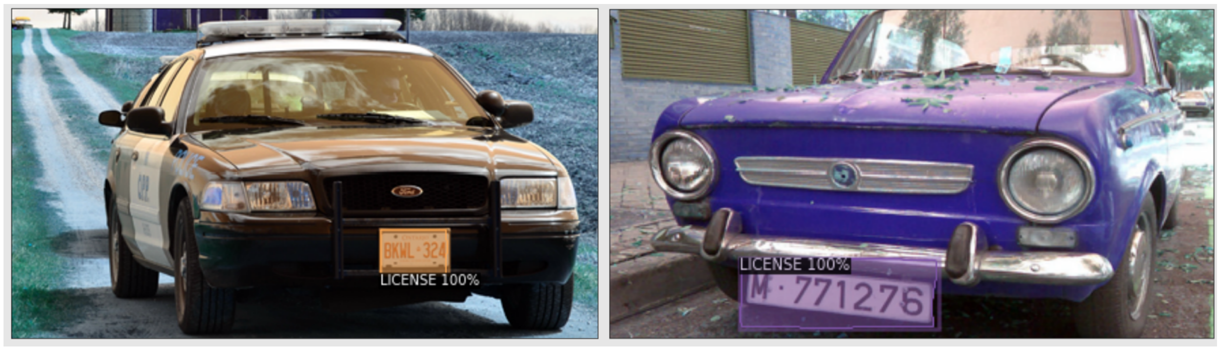


Figure 14. Example license plate detections.

4.2 Evaluation on Adverse Conditions

With two simple license plate detection models trained, both were then used to identify circumstances in which license plates could or could not be detected. Several images of vehicles in rain, snow and ice, low light conditions, or some combination of the three were used to evaluate the models.

4.2.1 Rain

In images containing heavy rain, those that were of high quality did not pose an issue to license plate detection by either model. Despite the increased noise from the rain, this would suggest that cameras capable of taking high quality images could successfully continue ALPR use even under heavy rain. However, in the case of **Figure 16**, a dirty camera or one covered in water would result in images where a license plate could not be detected by the models.

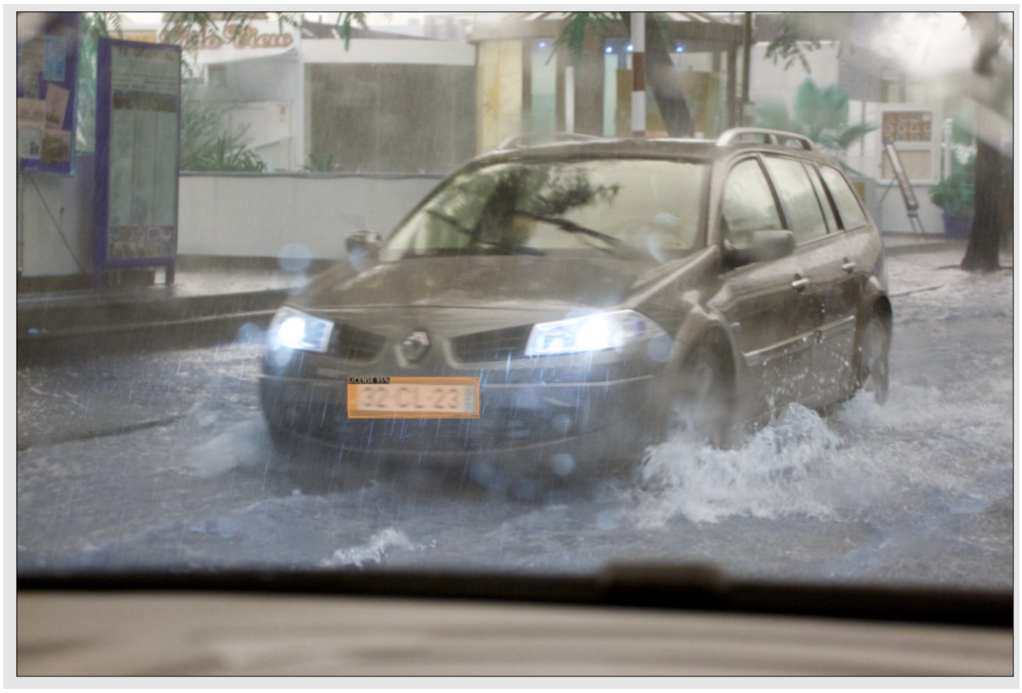


Figure 15. A successful detection during heavy rain.

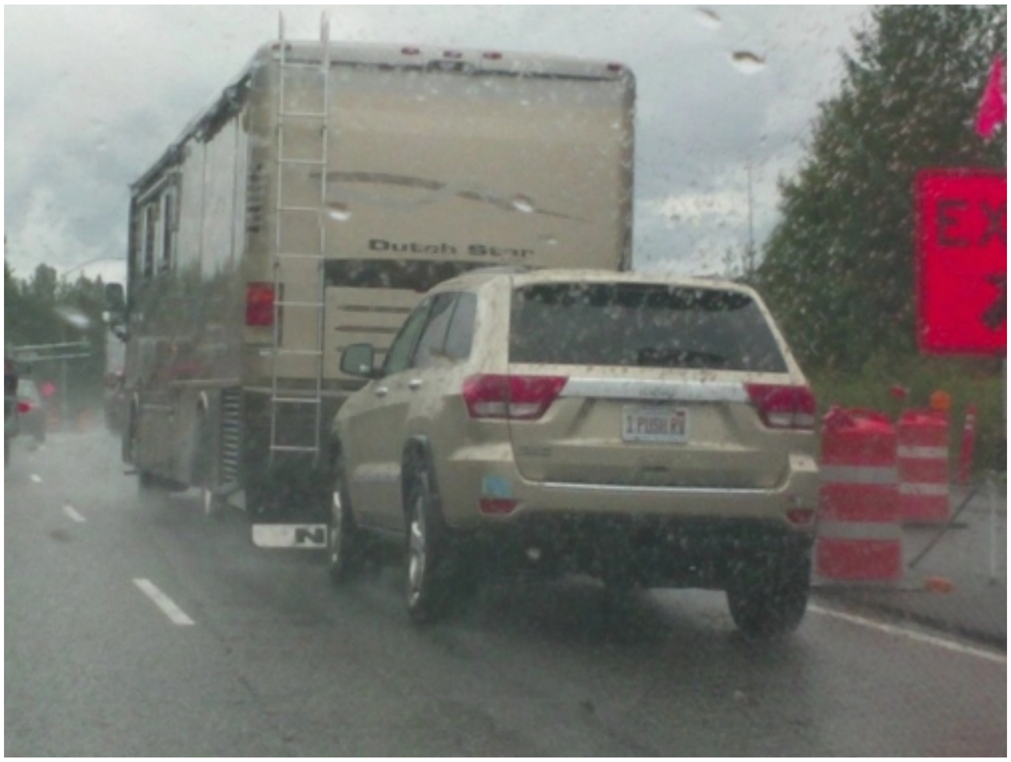


Figure 16. Images blurred by rain hinder detections.

4.2.2 Snow

Snow posed a decent challenge for the models. In cases of light snow, license plate detection was partially possible, however heavy snow severely crippled the models' ability to detect license plates. In **Figure 17**, this license plate was detected by the ResNet-101 model, but not by the ResNet-50 model, although both were able to detect the license plate in **Figure 18**. Heavy snow and ice, such as that in **Figure 19**, would almost always result in missed detections.

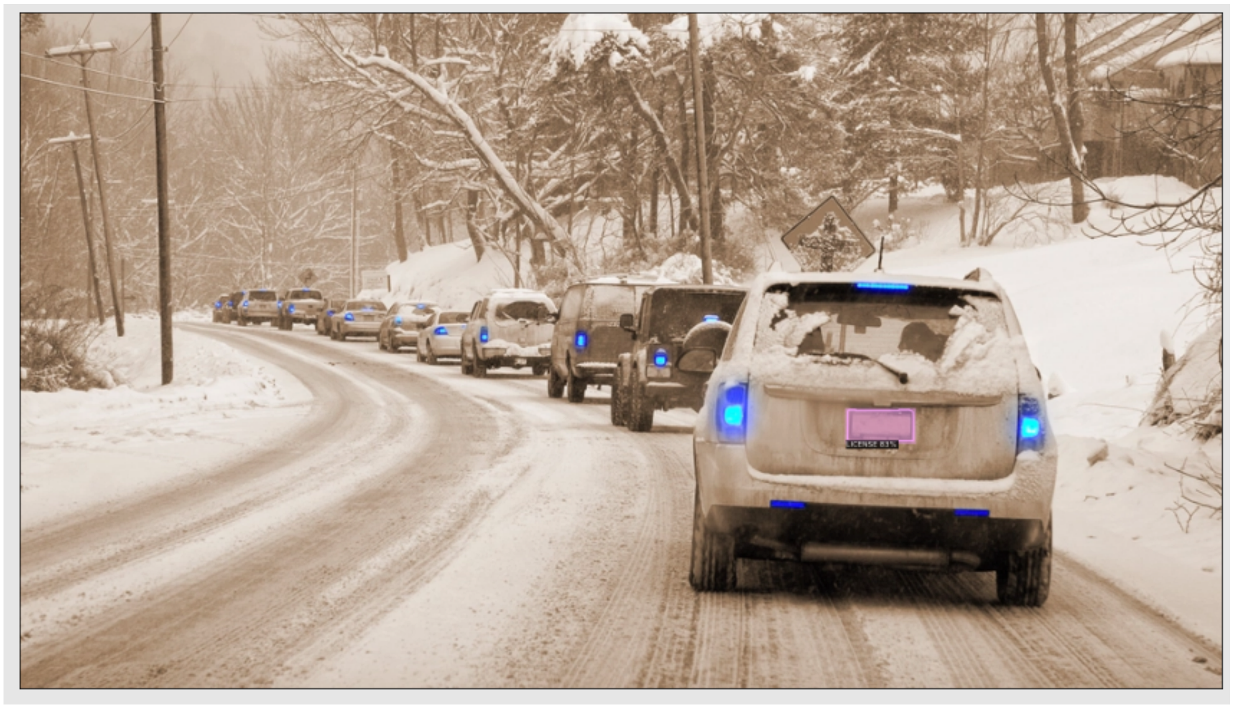


Figure 17. A license plate in light snow detected only by ResNet-101.

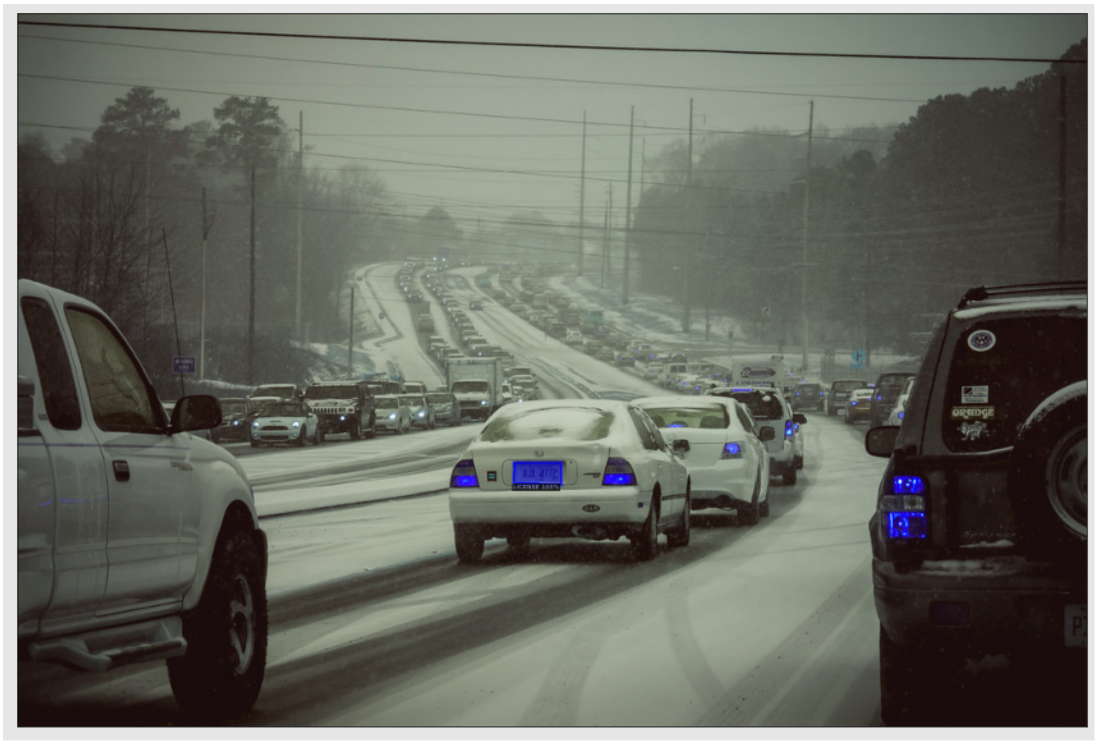


Figure 18. A license plate in light snow detected by both models.

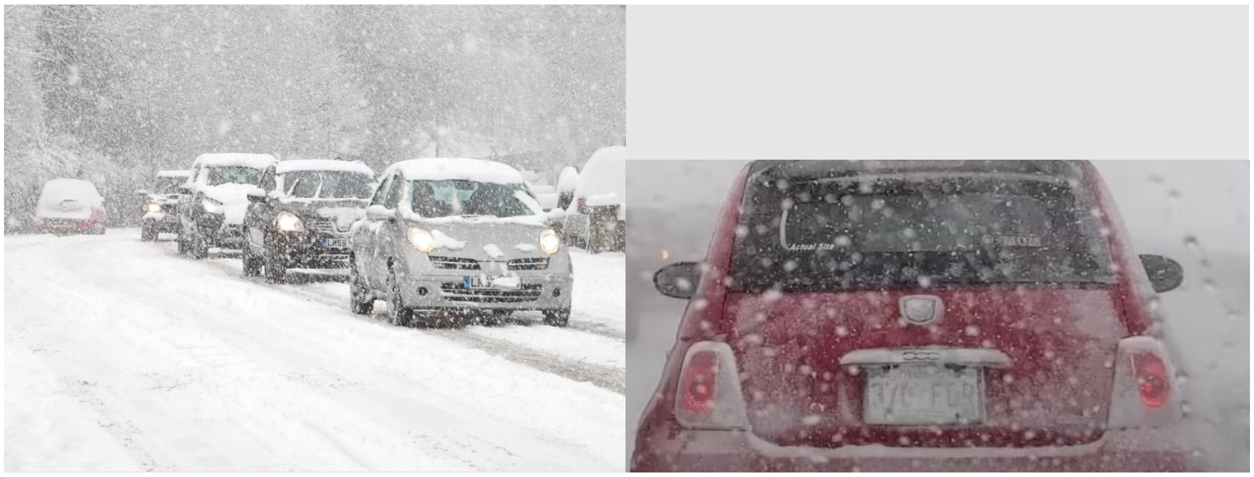


Figure 19. Heavy snow and ice prevent license plates from being detected.

4.2.3 Low Light

Low light seemed to present difficulties based on the vehicle's distance from the camera. In close-up images, especially those head-on, the vehicle's lights were enough for the model to be able to make a successful detection, however in images where the vehicle was further away from the camera detection would often fail. This could be due to one of two reasons – first, the data set the models were trained on were almost completely daytime images, and second, the vehicles in question may have been simply too far away to detect license plates even in daylight. For use at night, stationary ALPR cameras should be situated in such a way that it is possible to obtain a close-up view of a vehicle as it passes by.

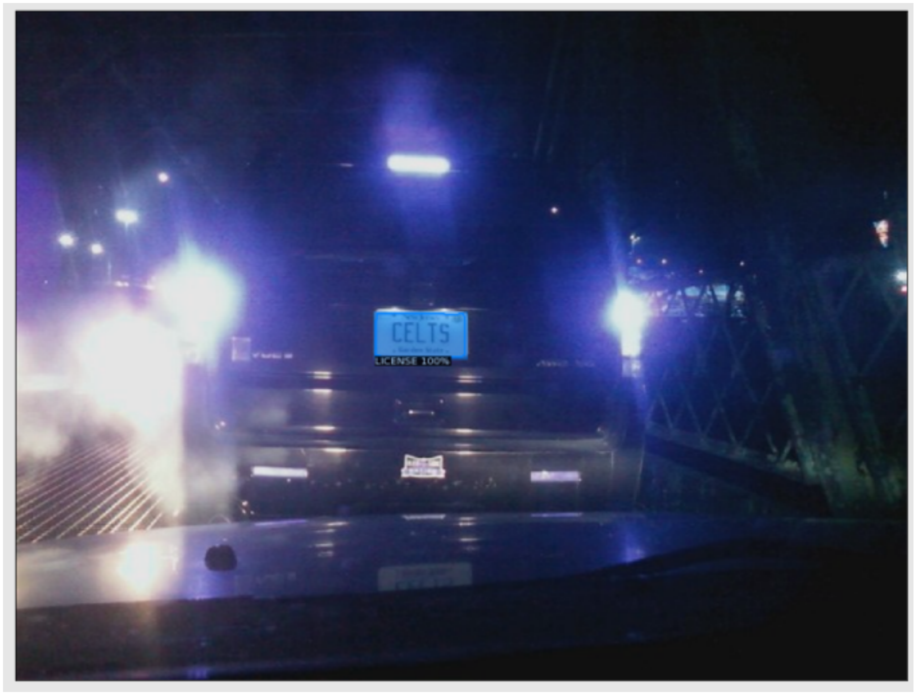


Figure 20. A successful license plate detection at night.

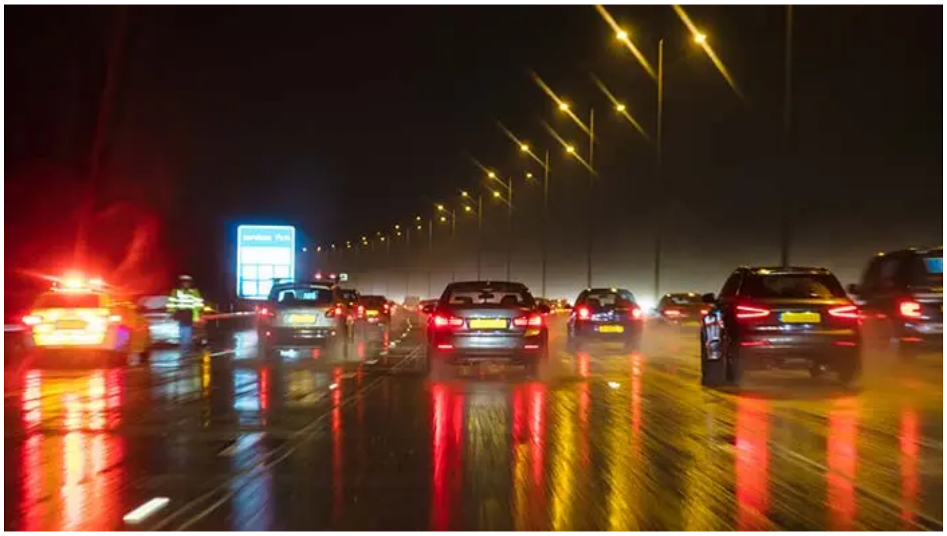


Figure 21. No license plates could be identified at this distance.

4.3 Limitations

The models appeared to have the most difficulties with identifying license plates when there are moderate to high levels of license plate obscuration as well as when vehicles are at a distance from the camera. Rain seems to only cause an issue in low-quality imagery, or when the camera becomes wet, causing blur and noise. This limitation can likely be overcome for stationary ALPR cameras by placing them in dry locations, such as under a roof overhang, however mobile ALPR systems will have to make do with makeshift covering if able. In cases of heavy snow, ALPR performance will likely see a drop if not operated in a controlled setting, such as larger toll booths. It should be noted that heavy snow accumulation may obscure portions of a license plate, which makes it impossible for either ALPR or human to correctly identify a license plate number. Furthermore, uncovered ALPR cameras themselves can suffer from snow accumulating on the lenses, further increasing obscuration. For regions that face heavy snowfall, this limitation needs to be taken into consideration before relying solely on an ALPR system for traffic enforcement. Low light environments do not seem to be a major issue for ALPR systems, and the Department of Homeland Security reports that their DeepVIEW ALPR system, a mobile ALPR, will be able to function at night **[26]**.

In a curious instance, bumper stickers were found to raise a large number of false positives. Seen in **Figure 22**, a vehicle that is covered with bumper stickers raised eight false positives from the ResNet-50 model and five false positives from the ResNet-101 model. This may suggest that the ResNet-101 model has learned more identifiable or distinct license plate features than the ResNet-50 model which prevents it from picking up some of the same false positives. This also raises questions as to whether or not active ALPR systems in use today are vulnerable to being attacked using similar confusers for purposes of raising false alarms or misidentifying a specific individual.

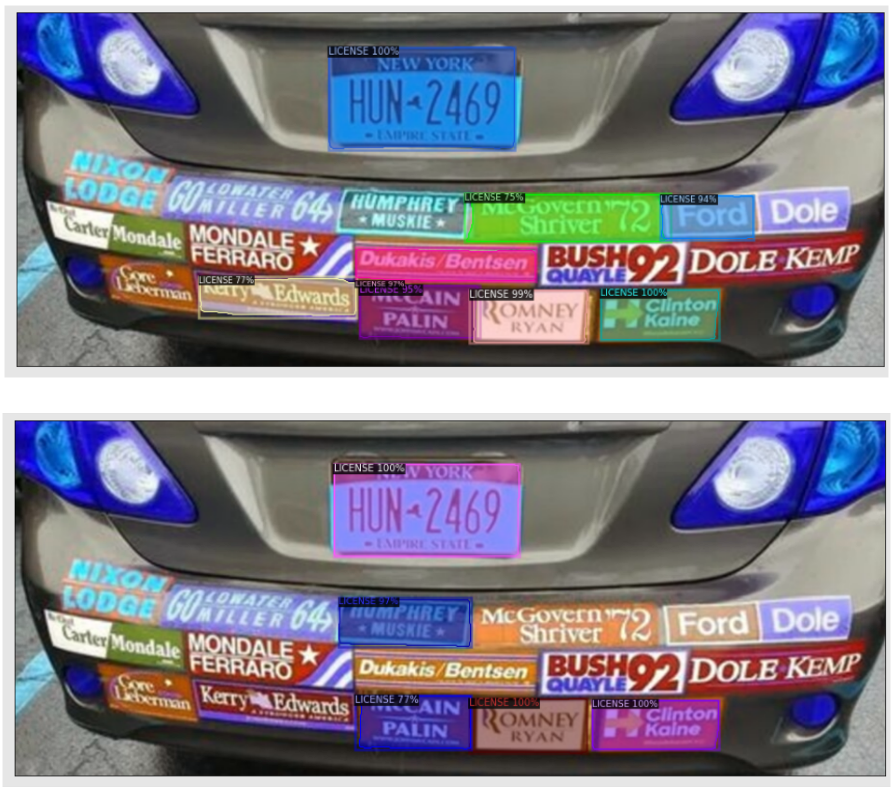


Figure 22. False positives caused by bumper stickers, top ResNet-50, bottom ResNet-101.

SECTION 5 – NEXT STEPS

In light of the results determined from the evaluations, further work into improving ALPR performance under heavy snow should be pursued, as well as investigation into the possibility of attacking ALPR systems with adversarial attacks. Improving performance under heavy snow is a difficult task, as falling snow creates obscuration which becomes greater at distance, however depending on the task at hand, an ALPR camera can be strategically placed so as to minimize this obscuration. Not all implementations have this luxury, however, which may require manual oversight to ensure accurate scans. Training models on data sets that include license plates partially obscured by snow at the anticipated use distance could help boost license plate detection, although transcription may remain a problem. The extent to which ALPR should be relied on in snowy conditions will likely be determined by the amount of snowfall a region typically receives, as requiring manual oversight a handful of days is less of a cost than requiring a full-time auditor during the winter season.

With the relative newness of machine learning in comparison to more traditional cyber security endeavors, little is understood in regards to a neural network's security. Although perhaps a fault in the models used, the ability for a bumper sticker to raise false positives is concerning, as it suggests there may be ways to intentionally trick ALPR systems into behaving in a way other than intended. While it would be simple enough to force an ALPR system to only accept one detection from a given vehicle, the question next becomes how to determine which detection is valid. A model needs to be trained to the extent that it only recognizes license plates, but at the same time maintain enough flexibility to detect uncommon license plates, such as those from a different state. One possible solution would be to maintain a database of common bumper stickers known to cause issues and use that to automatically check anomalous detections, however a malicious actor could easily find a way around this. The damage possible in accepting a false positive depends on the ALPR’s intended purpose and could range from racking up enormous fines to framing a targeted individual as being somewhere they were not. Additionally, adversarial patterns should be approached as well, as these threats are relatively unknown and could allow license plates to evade recognition, be it accidentally or intentionally.

SECTION 6 – CONCLUSION

Complex technologies such as automated license plate recognition systems are often build upon a foundation of numerous simpler concepts. This thesis contributes to the general understanding of automated license plate recognition systems by covering the concepts of image data and machine learning, including how image data is represented in binary and some commonly found data formats, as well as the basics of machine learning and how a neural network adapts to a given task. Furthermore, the effects of the deployment of such systems are contextualized through the examination of in-development technology as well as the demonstration of a license plate detection model and detailed steps for training one. This approach arms this thesis with the information needed to properly survey the scope and capabilities of modern automated license plate recognition systems, a prerequisite required to approach such topics as the ethics and potential concerns of using ALPR on a regular basis.

Powered by the advances in machine learning and neural networks in recent years, automated license plate recognition systems have seen steady growth in use. With the Department of Homeland Security's DeepVIEW in testing, ALPR appears to be here to stay. With that, the necessary legal protections need to advance as well, as concerns about individuals' and groups' privacy remain in question. Demonstrated in **Section 4**, a license plate detection model is easy to train, however the heuristics of the intended purpose and region deployed makes tailoring ALPR models difficult. To further add to this, as machine learning sees progress, so does adversarial machine learning, a field that will need to be taken seriously if ALPR systems are expected to be fielded for regular use nationwide. As the statistical complexity of neural networks makes it difficult if not impossible to reason why the outcomes are as they are, for ALPR systems to be trusted to function properly, further work is required to ensure that ALPR systems can operate under otherwise seemingly unfavorable conditions as well as defend itself against adversarial attacks.

References

[1] *Bit Depth Tutorial*. (2020). Cambridge in Color. https://www.cambridgeincolour.com/tutorials/bit-depth.htm

[2] League, C. (2017). *Image encoding*. LIUCS. https://liucs.net/cs101s17/n3-media.html

[3] Colors on a Computer Screen. (2022). *Mixing RGB to make colors* [Image]. https://www.chem.purdue.edu/gchelp/cchem/RGBColors/body_rgbcolors.html

[4] *Additive & Subtractive Color Models*. (2021, December 16). Pavilion. https://pavilion.dinfos.edu/Article/Article/2355687/additive-subtractive-color-models/

[5] Harmsen, N. (2019, December 5). *PNG vs JPG*. [Image]. Fstoppers. https://fstoppers.com/education/about-jpeg-images-and-their-quality-degradation-435235

[6] Hiwarale, U. (2019, October 19). *Bits to Bitmaps: A simple walkthrough of BMP Image Format*. Medium. https://medium.com/sysf/bits-to-bitmaps-a-simple-walkthrough-of-bmp-image-format-765dc6857393

[7] Chastain, S. (2020, February 10). *Vector vs. Bitmap Images*. Lifewire. https://www.lifewire.com/vector-and-bitmap-images-1701238

[8] McAnlis, C. (2016, April 26). *How JPG Works*. FreeCodeCamp. https://www.freecodecamp.org/news/how-jpg-works-a4dbd2316f35/

[9] McAnlis, C. (2016, April 6). *How PNG Works.* Medium. https://medium.com/@duhroach/how-png-works-f1174e3cc7b7

[10] Bourke, P. (1993, November). *A Beginners Guide to Bitmaps*. Paul Bourke. http://paulbourke.net/dataformats/bitmaps/

[11] Rossi, T. (2022, October 24). *What is Bit Depth?*. [Image]. Fix the photo. https://fixthephoto.com/8-bit-vs-16-bit.html

[12] Cueller, C. (2011, September 13). *How To Create A Bitmap Image File By Hand, Without Stencils.* art21 Magazine. https://magazine.art21.org/2011/09/13/how-to-create-a-bitmap0-image-file-by-hand-without-stencils/

[13] *The Growing Popularity of Machine Learning Technology*. (2021, August 20). Medium. https://fintelics.medium.com/the-growing-popularity-of-machine-learning-technology-bd217bb00d4f

[14] *Descending into ML: Training and Loss*. (2022, July 18). Google Developers. https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss

[15] *Descending into ML: Linear Regression*. (2022, July 18). Google Developers.

https://developers.google.com/machine-learning/crash-course/descending-into-ml/linear-regression

[16] Harrison, O. (2018, September 10). *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. Towards Data Science. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

[17] Hardesty, L. (2017, April 14). *Explained: Neural networks*. MIT News.

https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414

[18] Sharma, S. (2017, September 6). *Activation Functions in Neural Networks*. Towards Data Science. https://towardsdatascience.com/activation-functions-neural-networks-

1cbd9f8d91d6

[19] Zagoruyko, S., Komodakis, N. (2016). *Wide Residual Networks*. Paris, France: Université Paris-Est, École des Ponts, ParisTech.

[20] *Reducing Loss: Gradient Descent*. (2022, July 18). Google Developers.

https://developers.google.com/machine-learning/crash-course/reducing-loss/gradient-descent

[21] *Reducing Loss: Learning Rate*. (2022, July 18). Google Developers.

https://developers.google.com/machine-learning/crash-course/reducing-loss/learning-rate

[22] Singh, L. (2021, July 10). *Forward and Backward Propagation — Understanding it to master the model training process*. Medium. https://medium.com/geekculture/forward-and-backward-propagation-understanding-it-to-master-the-model-training-process-3819727dc5c1

[23] *ML Practicum: Image Classification*. (2022, July 18). Google Developers.

https://developers.google.com/machine-learning/practica/image-classification/

convolutional-neural-networks

[24] Smilkov, D., Carter, S. (2016). *A Neural Network Playground*. TensorFlow.

https://playground.tensorflow.org/

[25] *Automatic License Plate Readers*. (2018, June 25). ACLU of New York.

https://www.nyclu.org/en/automatic-license-plate-readers

[26] *Automatic License Plate Reader (ALPR)*. (2021, January). U.S. Department of Homeland

Security – Science and Technology. https://www.dhs.gov/sites/default/files/publications/2021_st_alprfactsheet_20210105_final508.pdf

[27] *The History of Toll Collection on the NYS Thruway*. (2022). New York State. https://www.thruway.ny.gov/oursystem/toll-collector-history.html

[28] *Technologies That Enable Congestion Pricing—A Primer*. (2021, March 26). U.S. Department of Transportation – Federal Highway Administration. https://ops.fhwa.dot.gov/publications/fhwahop08042/cp_prim2_04.htm

[29] *Traffic and Tolling*. (2021, March 15). Vaxtor. https://www.vaxtor.com/sectors/traffic-tolling/

[30] *Automated License Plate Readers (ALPRs)* (2017, August 28). Electronic Frontier Foundation. https://www.eff.org/pages/automated-license-plate-readers-alpr

[31] Greenberg, P. (2015, February). *Automated License Plate Readers*. National Conference of State Legislators. https://www.ncsl.org/research/telecommunications-and-information-technology/automated-license-plate-readers.aspx

[32] New Ontario license plate is blindingly reflective when sunlight hits it at the right angle. (2022, February 22). *A reflective license plate*. [Image]. https://www.reddit.com/r/ontario/ comments/f83fr9/new_ontario_license_plate_is_blindingly/

[33] Ismail, M. (2017). License Plate Recognition for Moving Vehicles Case : At Night and Under Rain Condition. *Second International Conference on Informatics and Computing (ICIC), 2017, pp. 1-4, doi: 10.1109/IAC.2017.8280649.*

[34] Kaur, S., Dhillon, K. K., Manvi, Chauhan, R. S. (2014, July). An Automatic System for Detecting the Vehicle Registration Plate from Video in Foggy and Rainy Environments using Restoration Technique *International Journal of Computer Applications, 97 (20)*.

[35] Kerekes, J. P., Hsu, S. M. (2014). Spectral Quality Metrics for VNIR and SWIR Hyperspectral Imagery. *SPIE 5425.*

[36] Irvine, J. M. (2003). National Imagery Interpretability Rating Scale (NIIRS). *Encyclopedia of Optical Engineering 1:1,1442 – 1456*.

[37] Wu, Y., Kirillov, A., Massa, F., Lo, W., Girshick, R. (2019). *Detectron2*. https://github.com/facebookresearch/detectron2

[38] Larxel. (2020, May 30). *Car License Plate Detection*. Kaggle. https://www.kaggle.com/datasets/andrewmvd/car-plate-detection

[39] Larxel. (2022). *Larxel*. Kaggle. https://www.kaggle.com/andrewmvd

[40] Lin, T. et. Al. (2014, May 1). *COCO Common Objects in Context*. COCO Dataset. https://cocodataset.org/

[41] Tony607. (2019). *labelme2coco.* https://github.com/Tony607/labelme2coco

[42] *Introduction to Resnet or Residual Network*. (2022, March 22). Great Learning. https://www.mygreatlearning.com/blog/resnet/

[43] Connor, S. (2019, January 24). *Introduction to ResNets*. Towards Data Science. https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4

[44] ul Hassan, M. (2019, January 23). *ResNet (34, 50, 101): Residual CNNs for Image Classification Tasks*. Neurohive. https://neurohive.io/en/popular-networks/resnet/

[45] Honda, H. (2020, January 5). *Digging into Detectron 2 – part 1*. Medium. https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd

[46] *Tutorial: Get started with Amazon EC2 Linux instances*. (2022). Amazon Web Services. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

[47] *t2.micro*. (2022). Vantage. https://instances.vantage.sh/aws/ec2/t2.micro

[48] *p3.2xlarge*. (2022). Vantage. https://instances.vantage.sh/aws/ec2/p3.2xlarge

[49] Manville, K., Leary, S., Jutras, M. A., Ward., C. M., Liang., E. (Forthcoming). Detection and Flagging of Digital and Physical Adversarial Patch Attacks. *AIPR 2022*.

[50] Carlini, N. (2022, October 13). *Keynote*. [Keynote presentation]. *AIPR 2022*.

[51] Young, D., Bradbury, M., Larson, E., Bigham, M., Thornton, M. (Forthcoming). Current Status and Overview of SMU-DDI Cyber Autonomy Range. *AIPR 2022*.

[52] Javidi, T. (2022, October 11). *Keynote*. [Keynote presentation]. *AIPR 2022*.

[53] Goodfellow, I. J., Shlens, J., Szegedy, C. (2014, December 20). Explaining and Harnessing Adversarial Examples. *ICLR 2015*.