USING POINTNET TO AUTOMATE LIDAR TILE CLASSIFICATION TECHNIQUES

A PREPRINT

Blake McClung
MS, Business Analytics
University of Iowa
Iowa City, IA
blake-mcclung@uiowa.edu

Brendan Sigale
MS, Business Analytics
University of Iowa
Iowa City, IA
brendan-sigale@uiowa.edu

Christopher Wenger*
PhD, Industrial Engineering
University of Iowa
Iowa City, IA
christopher-wenger@uiowa.edu

May 16, 2021

ABSTRACT

The purpose of this project is to automate area classification of lidar files using a robust model which in the future could be adapted to be much more specific or fill specific business needs. Using a deep learning methodology for obtaining these classifications could lead to a much greater ability to quickly categorize tiles. This will be especially useful as the world modernizes to support automated vehicles, including automated low-elevation flights, which could find this type of classification incredibly useful. In addition, this method will allow for adding new classification types quickly by altering the training data, reducing the need for manually classifying millions of tiles.

1 Introduction

Lidar (Light Detection and Ranging) is a technique which uses a laser and scanner to measure the time it takes for a laser beam to bounce back into the scanner. This, combined with a specialized GPS, allows the user to create high-quality 3D maps of virtually anything. By knowing how long it takes for the light to bounce back and where the laser is when the laser is shot, the calculation is as simple as dividing the time by the speed of light, and then dividing by two (one half on the way there, one half on the way back). Lidar gives the capability to map entire cities while also giving the user the capability to find objects or structures in perfect detail. Lidar is used in all sorts of applications - sensors on satellites to measure changes on the surface and construction sites to create accurate 3D models of a building or bridge.

Where the applications of lidar become extremely interesting, though, is object classification. In this scenario, a machine learning model will be able to take in a lidar scan of terrain and automatically classify it based on previous lidar scans it was trained on. Being able to dynamically classify terrain into predefined categories allow for quick mapping and modeling. This ability is a vast leap in machine learning, and is heavily supported using convolutional neural networks (CNN).

A CNN takes an image and parses through each pixel assigning it to a class and subsequently runs these pixels through multiple layers of our network. A CNN processes inputs through multiple layers, some which process the data and others which learn from the data. By passing through multiple layers, the model developer (and subsequently the user) can be assured the model has been trained on all parts of the data.

There are many different spins you can put on a CNN model, almost all CNN models run under the same assumption that the initial layers understand the low-level details and the later layers discover higher-level details. For tile classification purposes, we are unable to have fully-connected layers as it will cause us to lose our spatial information, so we look into using a fully convolutional network structure in our project. In this network, an encoder-decoder structure is used to capture the semantic information and spatial information. Skip connections are used to prevent stacking of encoder-decoder structure and allow us to make up for information lost from the image.

*Can contact listed author for more information or questions regarding paper via email.

Lidar is most often found for consumers in cars, where lidar sensors all around the car feed data into a powerful computer inside. The lidar sensors are generating a constantly updating 3D representation of the area around them, which the car can use to identify stop signs and stop lights, road boundaries, other cars, and people. By creating this constant scan, the driver is able to put some trust into the car to keep them safe. For example, a car with lidar may automatically prevent a lane change if there is a car in the driver's blind spot. In addition, because lidar works independently of visible light, a car with lidar may be able to identify a person or animal crossing the street in the dark, when the driver may not see them. Lidar has created a fundamental shift in the way we are able to think about consumer safety, especially in cars.

In this project, we will be implementing lidar to experiment with the first steps of using lidar to improve the safety of pilots and those on board an aircraft. We will be training a model which automates the classification of tiles to help build a robust map of the entire planet. This can be useful in many ongoing projects, especially ones such as Urban Air Mobility, a full-suite automated urban air-based public transport system. By using powerful eVTOL (electric vertical takeoff and landing) vehicles, they plan to move people around cities using compact, fast, and automated drones which can take a family of four across a city three times faster than cars, according to their website.[2]

There are thousands working to find new ways to apply lidar data. While we have not identified any using it in the way we are, there are many applying this data in different ways or much more advanced ways.

For example, researchers at ESRI, a global leader in geospatial analytics applications, recently unveiled their model and training process to use lidar data to identify power lines in dense urban areas. To accomplish this, they use PointCNN, a convolutional neural network architecture for processing lidar data. ESRI predicts their model will save 50,000 man hours of labor. Their research can be found at the link below.

`https://medium.com/geoai/pointcnn-replacing-50-000-man-hours-with-ai-d7397c1e7ffe`

2 Problem Definition

Problem Definition

The model will be used to automatically segment lidar LAS Point Cloud Data into nine categories, seen below:

Table 1: LIDAR Tile Classification Code Reference Guide

Class Code	Class Definition	Description
0	downtown	Strictly downtown, business district areas.
1	downtown-housing	Tightly packed housing near downtown
2	downtown-water	Downtown areas in which the tile contains water.
3	rural	Rural areas, such as forest or farmland.
4	rural-water	Rural areas in which the tile contains water.
5	suburban	Suburban housing and neighborhoods.
6	suburban-rural	Suburban housing in which the tile contains significant farmland or forest.
7	suburban-water	Suburban housing in which the tile contains water.
8	water	Tiles which exclusively contain water.

These classifications were developed by us to accommodate all types of tiles found in our data - our model will be trained to predict which classification is correct for each tile. We will verify this on a testing set once our model achieves appropriate loss and accuracy levels. For example, a 'downtown-housing' classification may contain several small buildings near one another organized in a clear housing structure and near downtown, compared to the 'suburban' classification, which is more spread out small buildings in areas which may have more winding roads, such as a subdivision. These classifications are diverse enough to accommodate almost all types of terrain which may be encountered.

3 Data

We acquired our data from the University of Illinois Geospatial Data Clearinghouse. The data was found as part of their Illinois Height Modernization Project (ILHMP). It contains LAS Point Clouds as well as processed Digital Surface Models (DSM), Digital Terrain Models (DTM), and Digital Elevation Models (DEM). While these can be useful for

[2]https://www.airbus.com/innovation/zero-emission/urban-air-mobility.html

general analysis, we were only interested in the LAS Point Cloud for our analysis. The data is aggregated by county as well as special areas of interest, such as large rivers.

The data can be found at the following link:

`https://clearinghouse.isgs.illinois.edu/data/elevation/illinois-height-modernization-ilhmp`

For this project, we chose to analyze the LAS Point Cloud for Peoria County. We chose Peoria County because it contains a diverse distribution of rural fields and farmland, suburban homes, industrial areas such as rail yards and warehouses, and tall buildings. It also lies on the Mississippi River, giving us ample data to train for all the classification codes we wish to predict.

The data we received can be visualized in ArcGIS Pro. Some examples of what the data looks like and comparisons to Peoria can be found below.

Figure 1: Google Maps 3D View of Peoria Terrain (Downtown Looking West).

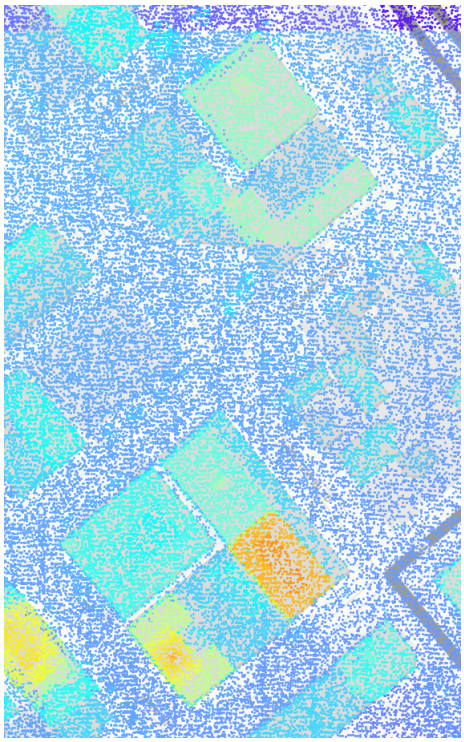

Figure 2: LAS Points Classified by Elevation (Note: Based on the Chase Bank Building - Bottom)

Figure 3: Google Maps 3D Render of the Chase Bank

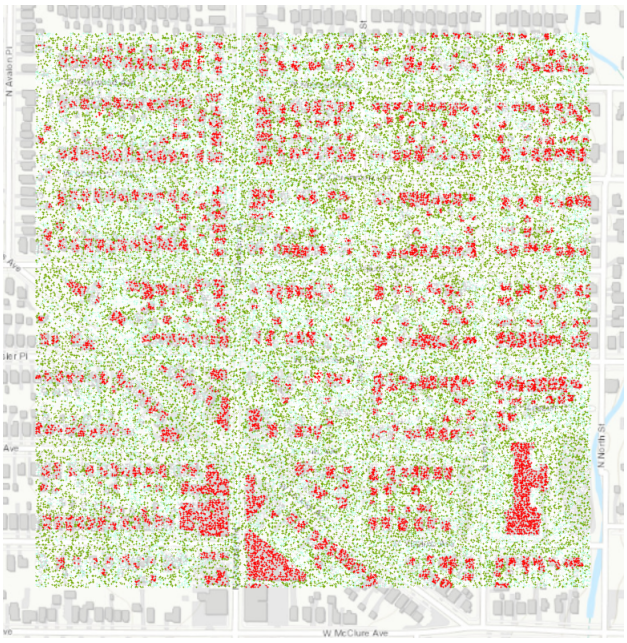

Figure 4: Classified Points in Suburban Peoria (Red = Buildings; Green = Ground

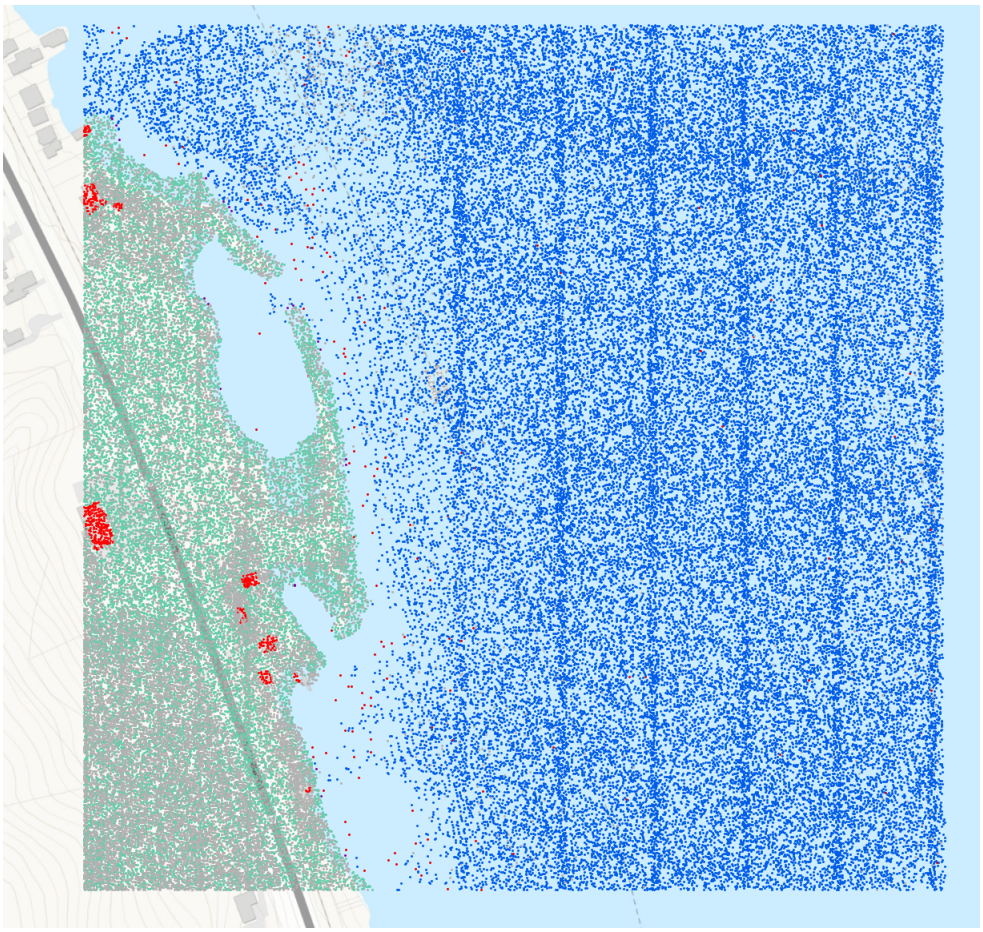

Figure 5: Bank of the Mississippi River (Red = Buildings; Green = Ground; Blue = Water

Once the LASer Point Cloud data set is downloaded and extracted from the Illinois Clearinghouse's website, a set of 180 LAS files is selected for the training and testing dataset (1.8GB - available on request). These tiles were strategically selected to ensure an appropriate amount of each classification was included for training and testing. Only a sample of the original dataset is used because the total data set contains 4,857 LAS files (84.4GB), which peaks the storage on the Google Colab server. Once the LAS files are downloaded and extracted to the Colab server they are scaled down to contain a consistent number of points. In the case study, the maximum number of points is 500,000, which was determined by the mean of total number of points contain in all LAS files within the data set. The first step taken after the data set is downloaded is to filter out the noise of the lidar packets by capturing the x-y-z coordinates and label information. This process is shown in Figure 6, where the lidar points are the raw coordinates of the lidar scan and the label is the classification of the file. After the coordinate and label information is collected it is packaged into a training and test dataset ready for the model to process.

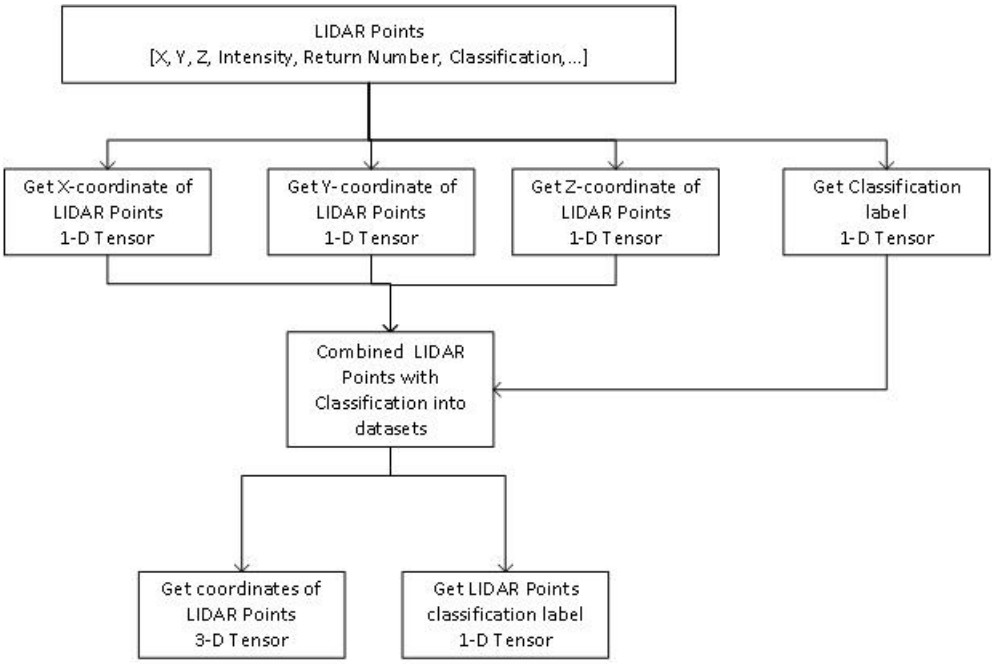

Figure 6: Processing Lidar Dataset

To better understand the data better, the first row with points 1,2, and 3 contain the same x-coordinate value but different y-coordinates. As we move down the rows, points 1, 4, and 7 will contain a different x-coordinate values but will contain the same y-coordinate value. This process begins to create an image over the x-y coordinate plane where each (x,y) combination contains a unique z-coordinate value. Instead of sorting the point cloud data, the software will use a symmetric function to aid in the model's prediction algorithm to save on computational time.

4 Method

This project uses the PointNet Architecture [4] that takes in lidar point net information to predict structural predictions of an aerial scan. A major benefit of using the current lidar dataset is the provider of the information already organizes the data around a particular axis. This is typical in aerial scans as the unmanned aerial vehicles scan along a fight plan path and point locations are known at the time of the scan. This saves on the processing time and removes the need for additional prepossessing of the point cloud data before feeding it into the deep learning model. Figure 8 showcases the prepossessing approach using the model presented in this paper. Once the information is captured from the lidar dataset it is fed into a PointNet deep learning model. This classification network takes the desired number of lidar points (n) and applies input and feature transformations and finally aggregates point features by max pooling. This architecture is shown in Figure 9 is the classification network designed by Charles R. Qi [4]. A key feature in the PointNet architecture is the use of a symmetric function applied to the transformed elements

$$f(x_1, ...x_n) \approx g(h(x_1), ..., h(x_n)$$

Where

$$f : 2^{\mathbb{R}^N} \rightarrow \mathbb{R}, h : \mathbb{R}^N \rightarrow \mathbb{R}^K, g : \mathbb{R}^K \times ... \times \mathbb{R}^K$$

The classification model is simple, *h* is approximated by a multi-layer perception network and *g* is composed of a single variable function and a max pooling function. While the model is collection h it can learn a number of f's to capture different properties of the area being scanned.

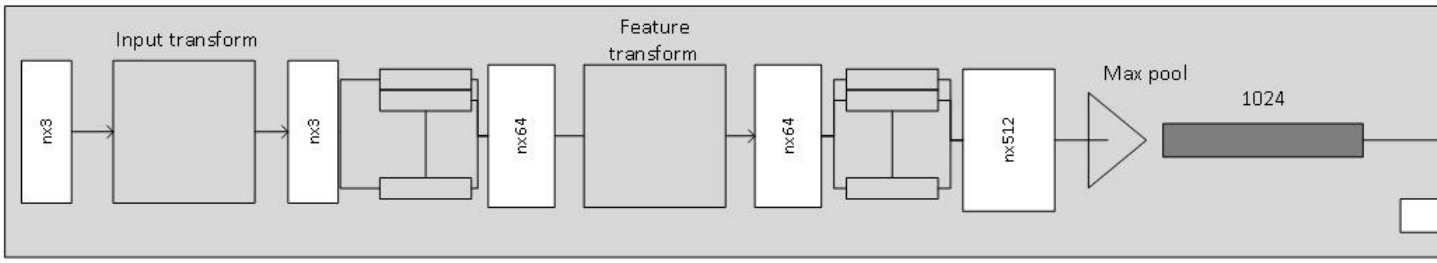

Figure 7: PointNet architecture

With the model defined it is important to understand the limits while parsing these large aerial images. There are over 500,000 points in each of the lidar files and while it is preferred to use all the points to identify land masses, cities, and etc. it is difficult to achieve efficient computational performance without high end hardware. So the sample size was limited to 2048, this was an optimal number as it speed up the prediction process and minimize the number of overflow and computational warnings that occurred while using Google Colab. An example of the dataset is shown in figure 8. There is a total of 81 lidar files each split up into their classification folder. In figure 8 (#) represents the number of lidar files in each of the classification folders. Since, the scan is over Peoria County most of the lidar files are over rural areas.

Classified_files.zip
Downtown (1)
Downtown-housing (10)
Downtown-water (4)
Rural (28)
Rural-water (7)
Suburban (15)
Suburban-rural (8)
Suburban-water (5)
Water (3)

Figure 8: Zip File DIR Structure

5 Discussion

5.1 Model Results & Interpretation

The model predicts the area of each scan by determining what feature it is currently sampling from the lidar dataset. Since multiple areas have similar features (i.e buildings, water, land, road, etc) the model makes its best determination to see which section of the area it is currently in. After running 20 Epochs the model was able to accurately predict the classification of areas using the smaller lidar samples. It has an accuracy between 58% to 70%. Shown in figure 9 is an example of the images the model is attempting to predict. The best approach to determine which region each object is located is based on the vertical representation of each point. This is critical to pilot's when determining their flight plans and is the main driving factor for this model. At or above 5,000 ft their is minimum detail and only the outline of cities, objects, and fields are visible. So being able to determine land masses is import for autonomous aircraft.

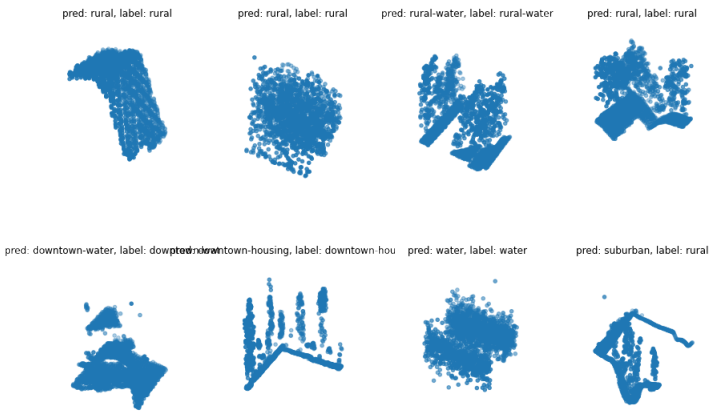

Figure 9: PointNet Model Results

To improve the accuracy of the classification predicts the data could have been parsed and split up into smaller LIDAR files. Using laspy as the util to pull in and parse LIDAR data is useful [6]. It has the ability to pull in the data as shown in figure 8 but it can also take that information and segment it into smaller LIDAR files. This would create smaller sub-files that can be used by the model to improve the accuracy. This approach can be shown in figure 10. This would reduce the number of sample points from an average of 500,000 to 500,000 divided by number of sub-segments.

$$Points_{Total} = LIDARPOINTS/segments$$

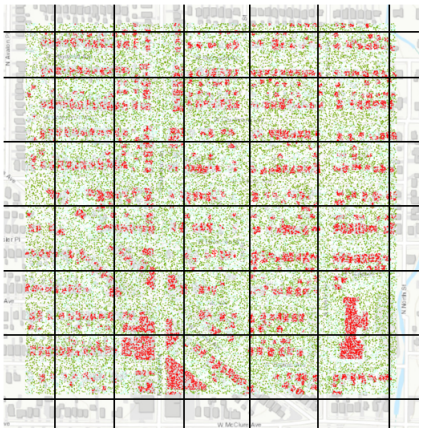

Figure 10: Segmenting LIDAR Image

The results produced by the model with the proposed dataset in this paper is a great starting point. The results met expectations but could be improved. Learning about the limitations of parsing too much data into the deep learning machine lead to the thought of parsing these giant LIDAR files into smaller segments. This is the next milestone for the development of an AI system to aid in helping pilot's or unmanned aerial system navigate through concentrated areas. Along with navigating through these areas the AI system be developed to classify object based on the height of a cluster of LIDAR points around the same area.

5.2 Future Opportunities

In the future, being able to perform more computationally intensive tasks would likely improve the accuracy of our model, as we would be able to train on more points per file to give the model a better idea of significant aspects of each classification type. This could possibility be achieved using a high-performance computing solution through an academic arrangement, such as the University of Iowa High Performance Computing (HPC) system, or premium

web-based processing tools, such as Google Colab Pro. This would also enable a greater amount of input training data - this project was partially limited by Google Drive's free-tier storage limit.

Being able to train on more tiles would greatly increase the model's accuracy and allow more specific classifications. For example, allowing the model to learn the difference between Downtown Peoria and Downtown Chicago. Downtown Peoria is barely a drop in the bucket compared to the amount of data collected for Cook County, where Chicago is located. However, because we have not tested the data in Chicago, it is possible the model would already be able to properly classify downtown based only on the z-axis differences between the ground and skyscrapers, even though those differences are drastically larger than found in Downtown Peoria. The only way to know would be to have the computational and storage capabilities to process this data, which would make our model more generalized and effective.

We believe this model begins to scratch the surface of a problem which will require solutions in just 10-20 years. Using AI to automatically navigate users without input is already somewhat common, as seen in Tesla's Autopilot Beta product, but this is only adapted to the ground, and only allowed in certain areas. To reach a truly autonomous future, we will need more adaptable and robust navigation systems. To accomplish Airbus's goal of Urban Air Mobility, the company (and others competing for the same goal) will need to have accurate classifications and mapping abilities in a 3D environment. Automatic driving using a Tesla is incredible technology, but adding the third dimension will be incredibly difficult, and will need new solutions to make it possible.

One such solution which will be necessary is an adaptation of content-based image recognition, which allows a model's developers to dynamically add new classification and categorizations to a model's training. Take, for example, a library with an app that allows customers to scan an item (from a screenshot, their camera, or elsewhere) and the app identifies the item and tells the user where in the library it's located. This model would work great until the library gets a new item (of which libraries might get dozens of per day). To accommodate these new items, the library would have to retrain their model by taking images of the new items, updating their model code, retraining the model, and deploying it. This is not a sustainable model for their item-location service.

Content-based image recognition is still an active area of research, but the general concept is that the neural network is trained to generate "keys" by scoring images using unique attributes, then compare inputs to the existing databases of keys. The model then returns the "key" which is closest to the input. For example, the book cover in their catalog which most closely resembles the picture input by the user. Adding new items to this model is as simple as using the same algorithm used for the original model to score the new item and add the new "key" to the database of existing keys. To our understanding, this technique has not been applied to Lidar files such as LAS Point Clouds, but would be crucial for adding new classifications as business needs change or the environment and terrain change.

In addition, becoming much more specific with classifications will be crucial for achieving Urban Air Mobility. The tiles used to train our model are relatively large, but using smaller tiles will be an important step in making smaller classifications. This would, in theory, require breaking LAS Point Clouds into significantly smaller chunks, which could be classified on additional factors such as building type or landing accessibility.

This would require the model to independently break LAS Point Clouds into significantly smaller chunks using a model which classifies chunks such as city blocks, parks, or complexes such as hospitals. A second model would then likely be required to break those chunks into even more specific classifications, such as apartments, malls, or emergency departments. This method, combined with an LAS-adapted Content-Based Image Recognition technique, would allow quick and efficient routing while also allowing for rapid iteration and addition to the model to account for new building types or newly developed areas.

While these models and techniques are well outside the scope of this project, they are important milestones to achieving fully autonomous, safe, urban transport.

6 Conclusion

This model performs well given the limitations experienced due to the scope of the project. Using LAS Point Clouds to classify tiles is a great starting point for solving problems which will need to be solved relatively soon. Our model performed adequately, ranging between 58% - 70%. In the future, further tuning the model or implementing additional technologies (some of which do not currently exist) would make the results of our project impactful. In the meantime, this project provides an excellent framework for classifying LAS files for terrain recognition.

References

[1] George Kour and Raid Saabne. Real-time segmentation of on-line handwritten arabic script. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 417–422. IEEE, 2014.

[2] George Kour and Raid Saabne. Fast classification of handwritten on-line arabic characters. In *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of*, pages 312–318. IEEE, 2014.

[3] Guy Hadash, Einat Kermany, Boaz Carmeli, Ofer Lavi, George Kour, and Alon Jacovi. Estimate and replace: A novel approach to integrating deep neural networks with existing applications. *arXiv preprint arXiv:1804.09028*, 2018.

[4] Charles Qi, Hao Su, Kaichun Mo, and Leonidas Guibas PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation *Proceedings of the IEEE conference on computer vision and pattern recognition.*, 2017

[5] "Laspy" Util - Laspy 1.2.5 Documentation, pythonhosted.org/laspy/util.html.

[6] "PointNet" Point cloud classification with PointNet, https://keras.io/examples/vision/pointnet/.