12-16-2020

Multi-modal Data Analysis and Fusion for Classification in 2D/3D Sensing

Jonathan P. Schierl
University of Dayton

Multi-modal Data Analysis and Fusion for Classification in 2D/3D Sensing

Honors Thesis

Jonathan P. Schierl

Department: Electrical and Computer Engineering

Advisor: Theus Aspiras, Ph.D.

November 2020

Multi-modal Data Analysis and Fusion for Classification in 2D/3D Sensing

Honors Thesis

Jonathan Schierl

Department: Electrical and Computer Engineering

Advisor: Theus Aspiras, Ph.D.

November 2020

Abstract

This research would develop a method of more accurately detecting objects using machine learning. There is plenty of current research and algorithms to tackle this problem. Our approach would use a dataset gathered with 2-Dimensional Infrared Imagery as well as 3-Dimensional LiDAR Data. We would develop a deep learning network with the ability to "learn" using both of these datasets. This proposed fusion network will perform better than either of the individual networks.

Acknowledgments

Special thanks to my program director, Dr. Vijay Asari, for working closely with me throughout my time at the Vision Lab. Also, thank you to my advisor, Thues Aspiras, for guiding me throughout this process. Also, thank you to my coworker, Nina Varney, for all of the technical advice and help.

University *of* Dayton

Table of Contents

1. Introduction

Currently, there are many algorithms used to classify objects. Many of the approaches to this problem, use deep learning and neural networks. The most common approach uses 2D imagery to train and test the network. There is lots of research on this topic, and many deep learning architectures to optimize this approach. A growing field uses 3D LiDAR data to solve the same problem, which requires a dataset comprised of points in a 3D space. There is increasing work in this space, a lot of which, also uses deep convolutional networks. There are pros and cons to each approach, which will be expanded on later.

We have an aerial dataset of both 3D LiDAR captures and 2D RGB imagery of plots of land in Surrey, BC. To classify these plots of land in categories, such as, “Farm,” “Commercial,” “Residential,” “Forest,” etc. we want to take advantage of both datasets. Our approach uses a neural network, capable of handling both datasets. To do this, these datasets, are individually processed, using respective 2D and 3D open-source architectures. This is done until the features (a reduced dimension of the original piece of data) are extracted for each modality. At this point, the features are fused, and the combined features are processed through the remainder of the 3D architecture. In this way, we can use information from both sources to outperform the current state-of-the-art architectures.

My contribution includes:

- An aerial dataset labeled for classification, including 3D pointclouds and respective 2D aerial imagery
- A fusion architecture capable of using both 2D and 3D information for classification
- This network outperforms a state-of-the-art network, KPConv in classifying land area

2. Related Work

2.1. 3D and 2D Deep Learning Comparison

2D deep learning strategies have been around for far longer, so there is more research done in this area. Some of the reasons that 3D processing has started to become more popular, is there is reason to believe it has a higher degree of accuracy of detection, and in this case classification. One of the reasons it performs better, is that it simply contains more information. For example, with the dataset we used, for one plot of land the pointcloud is 31.1 MB and the respective image is 138.9 kB. Another reason, is that LiDAR scans are not tricked by sunlight or reflections. A picture, taken pointed in the direction of sunlight, result in images that have high contrast or can look overly saturated. This is similar to how our eyes work, when we have to squint when trying to make out something in front of heavy sunlight. LiDAR, which uses a laser scan to determine it's points is not thrown off by sunlight. Again, with capturing images with high reflectance, 3D will outperform 2D. As far as processing time is concerned, 2D is much faster than 3D. Once again, this is because of the respective amounts of information for data captures. Processing 3D deep learning architectures can take magnitudes more computer power and time.

2.2. 2D Processing: ResNet Backbone

The 2D processing portion of the fusion network is done with Mask R-CNN, the architecture most commonly used for 2D instance segmentation. For our purposes, we just need the 2D processing portion to extract a feature map from the original image. The part of this architecture that we utilize to accomplish this, is the feature extraction. In Mask R-CNN, this is done by passing this image through a deep CNN (Convolutional Neural Network), called a backbone, and is labeled as "CNN" in the Mask R-CNN architecture below.

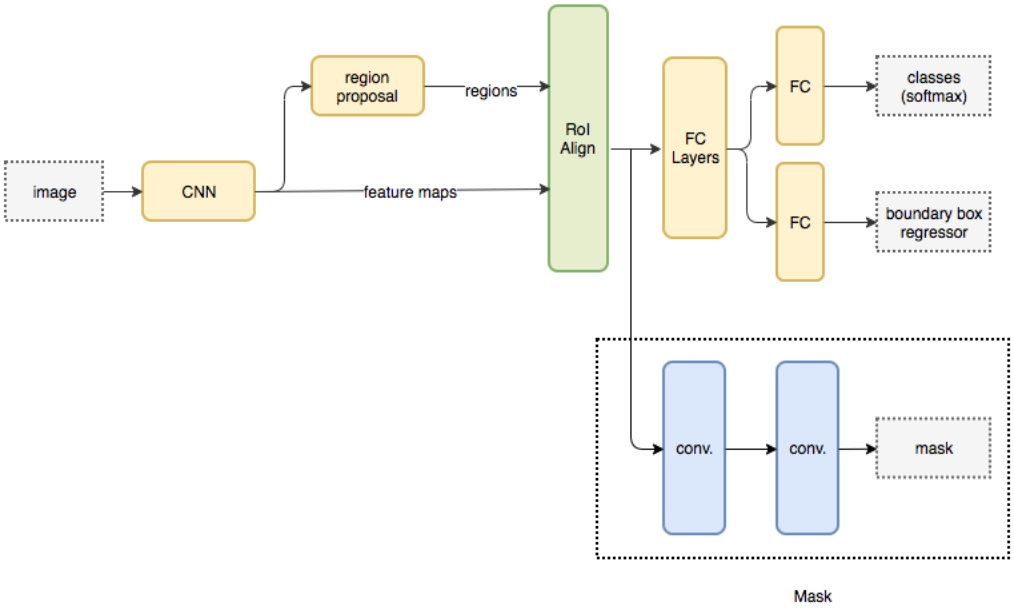

Figure 1: Mask R-CNN Architecture

The backbone of choice for this experiment is ResNet, because of it's success in solving the vanishing gradient problem. This was a problem, where increasing the depth of CNN's, lead to a steep drop in performance, due to over-saturation. Their solution was to create a shortcut connections that would skip a variable amount of layers, as seen in Figure 2. They were able to successfully create networks of depth, 34, 50, 101, and 152 convolutional layers deep that outperformed their competitors.

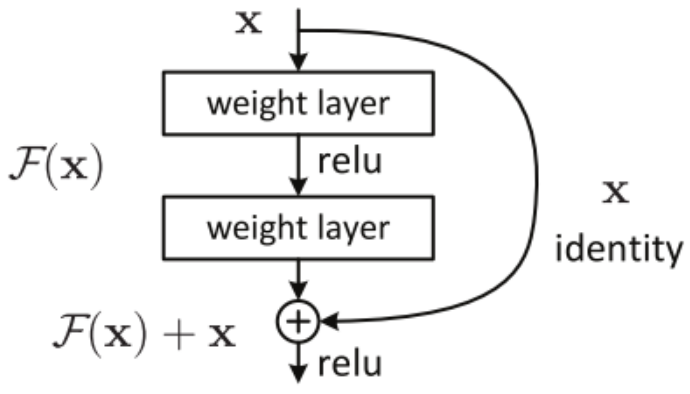

Figure 2: A Residual Block of the ResNet Backbone

The ResNet model we used, is “ResNet-101,” meaning that is has 101 convolutional layers. The deeper networks have increased accuracy, but take longer to process. We chose “101” as a middle ground. The network we used was pretrained on a giant dataset, called ImageNet, which is comprised of 14 million images and 200 classes. Because it is pretrained on such a robust dataset, it is extremely good at extracting high-level features.

2.3. 3D Processing: KPConv (Kernel Point Convolution)

KPConv was chosen to be the architecture for 3D processing. This is because it outperformed many of it’s competitors in classifying the ModelNet40 dataset. KPConv operates by applying a weighted kernel to the point clouds for convolution where the weights within the kernel filter are weighted spatially. Prior work with 3D neural networks that operated with kernels often performed poorly due to the fact that kernels are often rigid 2D or 3D structures. A rigid filter applied to a point cloud meant that it was often poorly representing the cloud in the learning process as the rigid filters were not able to conform to the shape of certain features. Point subsets within the cloud that were represented by a large number of normalized planes of points were able to be successfully learned using the rigid kernels, but clouds with complex dimensionality could not fit properly within the kernel filter. To alleviate this rigid kernel problem, KPConv introduces a deformable kernel that is able to fit a specific subset of points rather than a specific space. In this kernel weights are assigned spatially such that the deformation of the kernel impacts the weights. By doing this KPConv is able to perform kernel based convolution on irregular surfaces and features. The dataset used to train this network, discussed in detail in section 3.1, focuses on the classification of land area, which has very irregular shapes, meaning that KPConv is expected to perform well on this dataset using the deformed kernel approach. The deformable kernel structure can be seen in Figure 3 below.

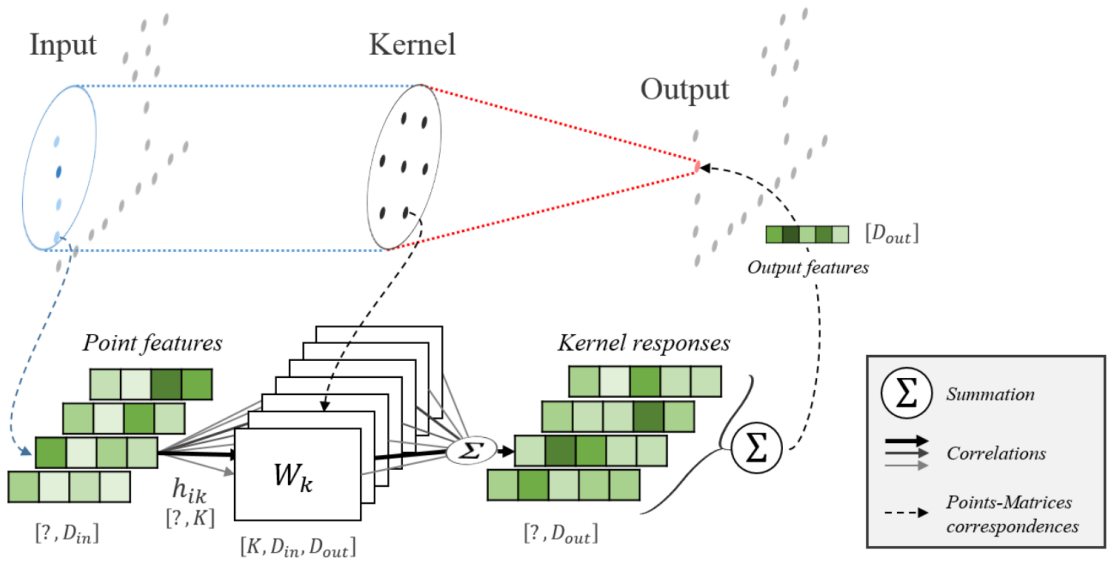

Figure 3: KPConv Architecture with the Deformable Kernel

3. Methodology

3.1 Surrey Dataset

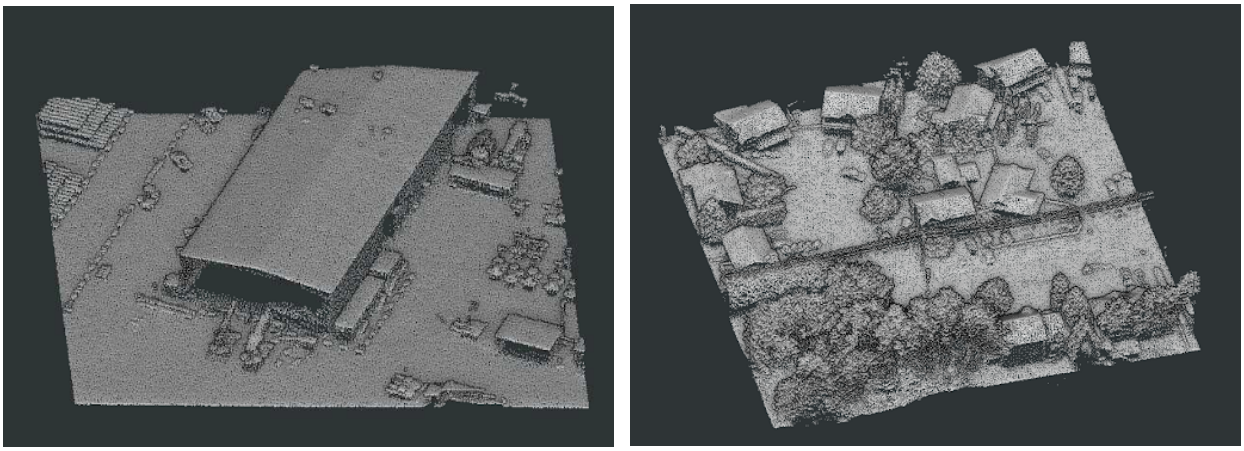

Figure 4: Sample Surrey Pointcloud Data (Commercial: Left, Residential: Right)

The dataset, used for this experiment, is a large-scale aerial LiDAR dataset, with respective RGB captures. The LiDAR pointclouds were used from Surrey, British Columbia's open source data release as part of their "Open Data program," and are captures across the city of Surrey. The first step was sending a hard drive to Surrey, to gather the dataset used for classification. Each plot of land was very large, and needed to

be split up into sections of 4. Once this was done, in an effort to label them according to their category of “cemetery, commercial, construction, farmland, field, forest, golf course, highway, park, recreation, residential, school, undeveloped, viaduct, or water,” a representation of them would have to be created to overlaid onto a map, to determine their class. Once completed, the setup on QGIS, can be seen in Figure 5 below.

Figure 5: QGIS Setup of LiDAR Representations for Labeling

To do this, the pointcloud files were converted to UTM Zone 10N, a system for describing the points on a longitude and latitude scale. Then, using their geolocation information, see-through shapefiles were created with a red border, to indicate their position on a map. These were then, parsed through and labeled according to class.

To get the respective aerial images, the location of the pointclouds were compared to a set of TIFFs (Tagged Image Format Files), also provided by Surrey. The LiDAR plots were much smaller, so the respective TIFF’s were cropped accordingly, and converted to JPGs. Once the 2D and 3D dataset was compiled, they were split by class, evenly, into an 80/20 train/test split. This dataset is comprised of 318 pointcloud, image pairs, with 254 training and 64 testing samples each. The class breakdown, can be seen below, in Figure 6.

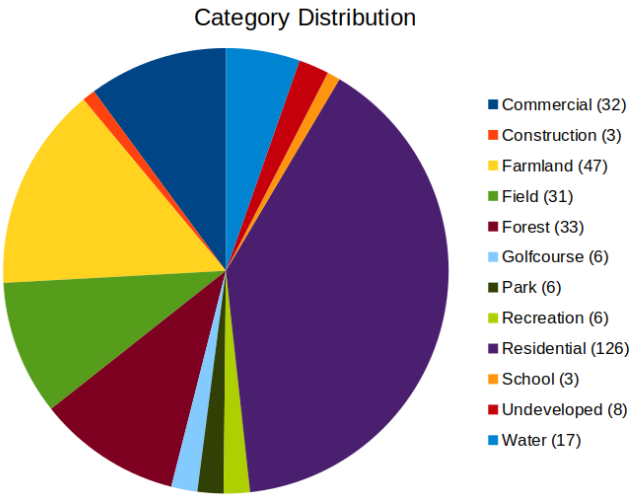

Figure 6: Surrey Dataset Category Distribution

Also, to note, if there was multiple classes within a section of land, it was labeled according to the dominating region. For example, in Figure 7 below, this would be classified as farmland because it has the most land area, even thought there are other classes present.

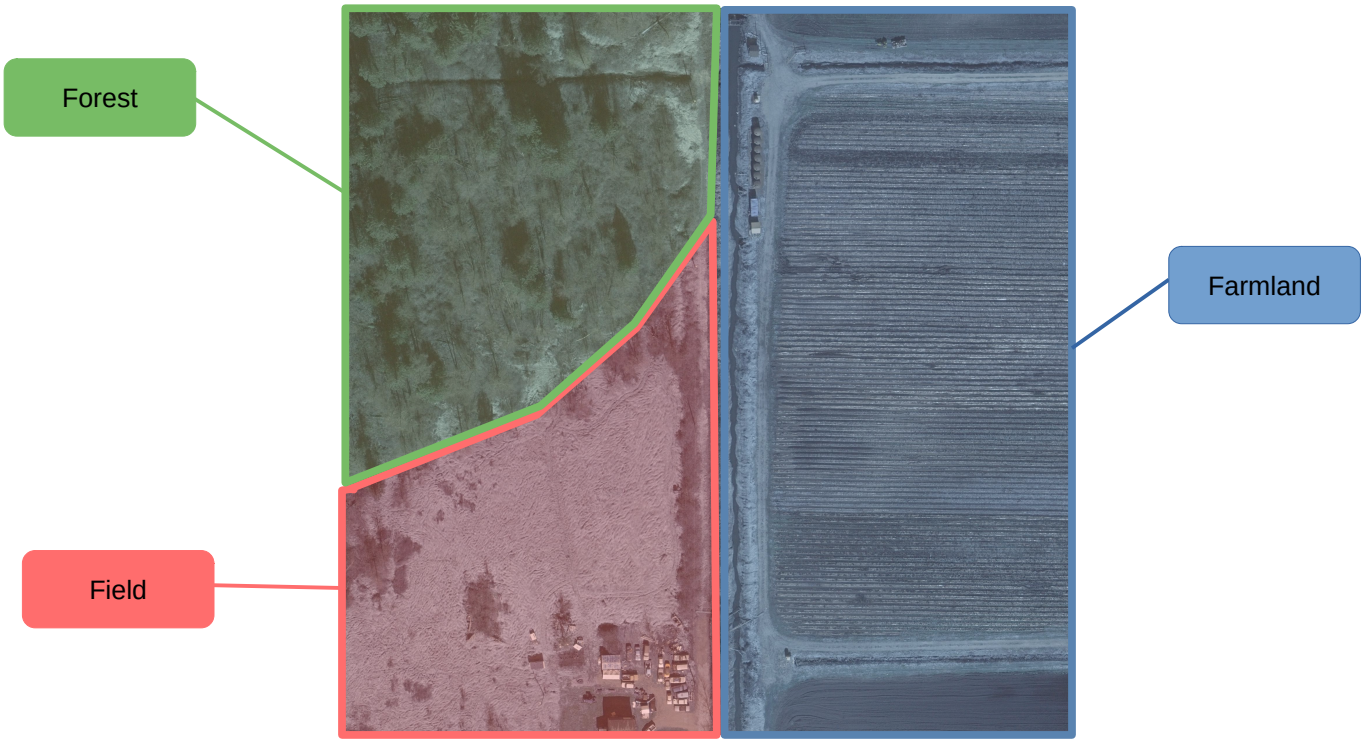

Figure 7: Classification by Land Area

3.2. Architecture Integration

The goal of this project is to process the 2D and 3D datasets separately, through Mask R-CNN and KPConv respectively. This would involve running these architectures in parallel. Instead, we decided to use KPConv as the general architecture, and before it processes the 3D features, fuse the 2D features, and processes these together through the rest of the architecture.

One of the first steps to be able to process these features in parallel is to run these architectures successfully on the same version of TensorFlow. This happened to be TensorFlow version 1.15.0. KPConv is strictly a 3D architecture, so the next thing was to add a way for KPConv to handle 2D images. The start of this was creating a folder in KPConv to store the images, and name the images the same as their matching 3D pointcloud. When the 3D pointclouds are loaded, KPConv stores this as a pickle file, which saves the data as a Python object, for accessibility. The list of elements stored in this pickle file had to be adjusted in the model to store the JPG images as well. In the architecture, where the 3D pointclouds were loaded, the inputs was adjusted to handle 2D images, and these were loaded in parallel with the pointclouds.

Also, when this information is sent to the GPU for processing, the dataset is to large to send all at once. The architecture resolves this by sending information in batches. These batches are specific to the input data and architecture, so they had to be adjusted to handle the additional information.

3.3. Feature-level Fusion

There are three main types of fusion: data-level fusion, feature-level fusion, and decision-level fusion. To best utilize both datasets, we decided to fuse them at the feature-level. This is because KPConv and Mask R-CNN are both state-of-the-art architectures for 3D and 2D data, respectively. In this way, we take advantage of the high performance of both architectures for each modality, while classifying based on both sets of features.

As mentioned, the way Mask R-CNN extracts features from the input images, is passes them through a ResNet backbone. To do this, while simultaneously running KPConv, Mask R-CNN was embedded inside this architecture. Then, when KPConv was

generating the 3D features, Mask R-CNN was loaded as a Python module, and the ResNet processing step, was called like a function. One image was passed at a time, and it returned 5 sets of features: C1, C2, C3, C4, and C5. These are the last layers for each stage of ResNet-101.

The features returned, can not be directly concatenated with the 3D features, for a number of reasons. They are Keras layers, so must be converted into TensorFlow tensors. Also, they are a different shape than the 3D features. This is an issue, when passing these features to the Convolution Head of the model. To solve this, we reduced the number of dimensions using a TensorFlow mean function across ‘axis 1’ and ‘axis 2.’ The 2D and 3D features are then able to be concatenated and sent to the remaining portion of the network.

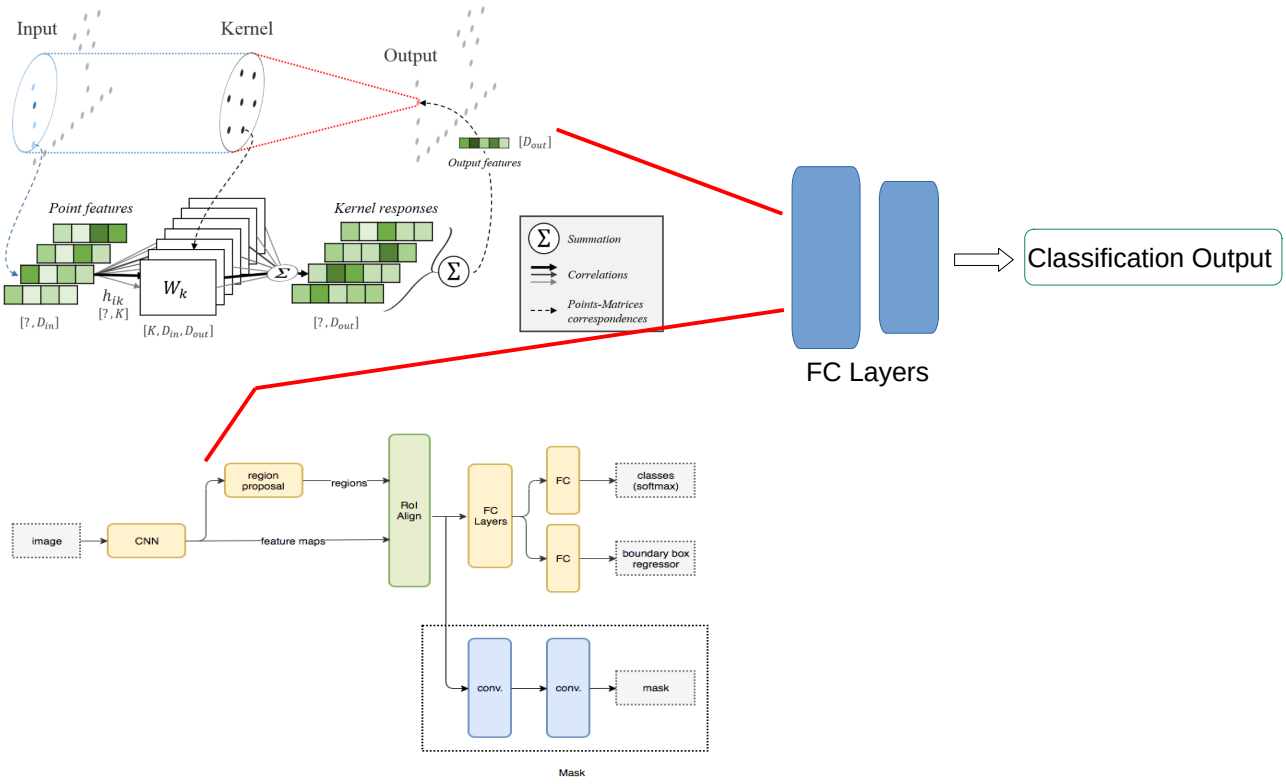

Figure 8: Fusion Architecture

4. Results

4.1. Experimental Setup

This experiment was meant to compare the original KPConv architecture to the KPConv network with additional fused feature (Fusion Network). To do this, each network was run individually using a 2080 Ti NVIDIA Graphics Cards. All the parameters, including the training and testing set, were held constant. The only thing that changed in the Fusion Network was the features passed into the Classification Head of the KPConv network. For the KPConv architecture, this was just the 3D features, and the Fusion Network, used the concatenated features. Each architecture was trained for 150 epochs, and the accuracy was computed across all classes.

The key parameters of this architecture is the first subsampling cell size dl_0 and the density parameters. In KPConv, the input clouds are subsampled in a grid-like fashion, to section off the clouds into more manageable sizes. For the most part, the bigger this parameter is, the larger the section of the cloud is being looked at, which usually correlates to a lower performance of the architecture. The input clouds are very large, so the subsampling parameter was set at a rather high value of 2.5 meters, for the GPU to be able to handle. Also, the density parameter is specified as the density of neighborhoods for deformable convolutions. This was set somewhat low at 6.0, where a higher density, such as 10.0 would be preferred. Again, this was to mitigate some of the computation cost put on the GPU.

4.2. Experimental Results

As expected, the feature-fusion architecture outperformed KPConv. This is because it had additional information, and was setup in a way to utilize it effectively. This can be seen in the overall classification accuracy, below in Table 1.

Network	Accuracy (%)
Fusion Network	**70.98**
KPConv	62.90

Table 1: Overall Classification Accuracy

Furthermore, this can be realized on a per-class basis, with a confusion matrix. With this, the classification of each piece of testing data can be seen. The columns in this table represent the predicted classes and the rows represent the actual classes. This means that the “2” under the residential class was predicted as residential, while it was actually commercial, because it is in the commercial row. In this way, all the correct classifications can be seen in the green diagonal, and all other values are mis-classifications. This was created for each network, with Table 2 representing the Fusion Network and Table 3 representing KPConv.

	Commercial	Construction	Farmland	Field	Forest	Golf Course	Park	Recreation	Residential	School	Undeveloped	Water
Commercial	4	0	0	0	0	0	0	0	2	0	0	0
Construction	0	0	0	0	0	0	0	0	0	0	0	0
Farmland	1	0	9	2	0	0	0	0	0	0	0	1
Field	0	0	0	1	0	0	0	0	1	0	0	0
Forest	0	0	0	0	4	0	0	0	1	0	0	0
Golf Course	0	0	1	0	0	0	0	0	1	0	1	0
Park	0	0	0	0	0	0	0	0	1	0	0	0
Recreation	0	0	0	1	0	0	0	0	0	0	0	0
Residential	0	0	0	1	2	0	0	0	22	0	0	0
School	0	0	0	0	0	0	0	0	0	0	0	0
Undeveloped	0	0	0	0	0	0	0	1	0	0	0	0
Water	0	0	0	0	1	0	0	0	0	0	0	4

Table 2: Fusion Network Confusion Matrix

	Commercial	Construction	Farmland	Field	Forest	Golf Course	Park	Recreation	Residential	School	Undeveloped	Water
Commercial	1	0	0	0	0	0	0	0	5	0	0	0
Construction	0	0	0	0	0	0	0	0	0	0	0	0
Farmland	0	0	8	5	0	0	0	0	0	0	0	0
Field	0	0	0	0	0	0	0	0	1	0	0	1
Forest	0	0	0	0	3	0	0	0	2	0	0	0
Golf Course	0	0	1	0	0	1	0	0	1	0	0	0
Park	0	0	0	0	0	0	0	0	1	0	0	0
Recreation	0	0	1	0	0	0	0	0	0	0	0	0
Residential	1	0	0	1	1	0	0	0	22	0	0	0
School	0	0	0	0	0	0	0	0	0	0	0	0
Undeveloped	0	0	1	0	0	0	0	0	0	0	0	0
Water	0	0	0	0	1	0	0	0	0	0	0	4

Table 3: KPConv Confusion Matrix

To strictly compare the networks on a per-class basis, the accuracies for each class with testing data present were computed. This can be seen below in Table 4 and 5, for each architecture, with the winning classes highlighted.

Category	Accuracy (%)
Commercial	66.7
Farmland	69.2
Field	50.0
Forest	80.0
Golf Course	0.0
Park	0.0
Recreation	0.0
Residential	88.0
Undeveloped	0.0
Water	80.0

Table 4: Fusion Network Per-class Accuracy

Category	Accuracy (%)
Commercial	16.7
Farmland	61.5
Field	0.0
Forest	60.0
Golf Course	33.3
Park	0.0
Recreation	0.0
Residential	88.0
Undeveloped	0.0
Water	80.0

Table 5: Fusion Network Per-class Accuracy

4.3. Discussions

While KPConv uses only 3D data, it mostly bases its classification on shape. This is because it learns based on the arrangement and density of points. This is great for objects that have very specific shapes and defined structures. However, it lacks other information found in pictures, such as shadows, textures, and colors. This added information is a huge advantage in classes with similar shapes. In this experiment, this is exactly where the fusion network outperforms KPConv, in the farmland and field classes especially. The 2D information is especially valuable in these classes, to distinguish between each other, because it adds color information of a brown farm instead of a green field. Similarly, with KPConv, many of the Commercial classes were mis-classified as residential. The aerial imagery helped prevent the rate of this mis-classification. Again, the shape of these classes are very similar, with sidewalks, buildings, and foliage. The

only difference is the nature of these buildings as being commercial building or having parking lots nearby that make it commercial.

Figure 9: Sample Residential Aerial Image

A complicated part of this problem, with the dataset used, is that it's sometimes hard to distinguish between classes, for a couple of reasons. One, is that it's hard to draw a line between some of the classes, such as field and park. So someone else may label the dataset slightly differently. Second, there are many sections of land that contain multiple classes within the sections. As mentioned, in Section 3.1, when there are multiple classes in each section, they are labeled based on the dominant class, or the class with the majority of land area. This makes it hard to classify if there are 3 classes present, for example, such as, farmland, forest, and residential and the split it is 30%, 30%, and 40%, respectively. While this class is labeled as residential, because it makes up 40% of the land cover, the entire land mass is labeled as residential. This is confusing to the network, if it has high certainty that it is farmland or forest, while it is not necessarily wrong, it will have a large affect on the backpropagation of the network. This also makes it challenging for the network to converge.

One other thing to note, is that some of the pointclouds overlapped with multiple TIFFs and this lead to some of the aerial images being cropped. The missing section was filled in with black, as seen below in Figure 10. While this did not lead to a full

utilization of the 2D dataset, if we consider this dataset added information, it was still advantageous.

Figure 10: Cropped TIFF Image

4.4. Future Works

To be able to deal with sections of land that have multiple classes present, instead of classifying based on the dominant class, these should be labeled using all the classes present. This could be done with pointclouds being able to be labeled with multiple classes in order of prevalence. To handle multiple classes, the architecture would have to be slightly modified to handle classes of multiple labels, and the nature of the classifier would need to be adjusted. Currently the classifier uses a sigmoid function, which outputs confidence for each class, with all of them adding up to 1.00. Such as if the network were 0.6 (60%) confident that it was farmland and 0.4 (40%) confident that it was field, all other classes would be 0. Instead, using a softmax classifier, all classes could be weighted on a scale of 0 to 1.0 individually. This network would then be able to assert that it thinks 1.0 (100%) confidence for farmland and 1.0 (100%) confidence for field.

Also, for handling pointclouds that overlap multiple TIFF files, there needs to be a way to merge the TIFF files, before finding the overlap. In this way, we would be able to recreate the complete 2D aerial capture for each pointcloud.

As far as the fusion network is concerned, it performs well as is, but there should be other ways tested of fusing the information, to verify this is the best method. There are

other places in the KPConv architecture that the features could be fused. Another good candidate would be at the input section of the network, that could demonstrate better classification.

5. Conclusions

In this work, we propose a dataset, consisting of 2D images as well as 3D pointclouds, labeled for classification. Using this dataset, we created a feature-level Fusion Network, capable of classifying this dataset, with state-of-the-art results. As demonstrated by this experiment, feature-level fusion yields better results than strictly classifying using KPConv. We believe this can work with other datasets, that have registered 2D and 3D captures.

References

[1] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, Leonidas J. Guibas “KPConv: Flexible and Deformable Convolution for Point Clouds,” arXiv:1904.08889, ICCV (2019).

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, “Mask R-CNN,” arXiv:1703.06870, ICCV (2017).

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, “Deep Residual Learning for Image Recognition,” arXiv:1512.03385v1 (2015).

[4] Krizhevsky, A., Sutskever, I., and Hinton, G., E., “ImageNet classification with deep convolutional neural networks,” Advances in Neural Information Processing Systems 25, 1097-1105 (2012).

[5] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K. and Li, F. F., “ImageNet: a large-scale hierarchical image database,” CVPR (2009).