Semantic Segmentation on Satellite Imagery: Land Classification and Building Detection

Harold Gamarro , Joe Brian Malubay , Said Mejia , Juan Pablo Montoya

May 26 , 2021

Abstract

Map features such as roads, building footprints, and points of interest are primarily created through manual techniques. Automatic categorization of land cover can play a role in forest preservation, extreme events damage assessment, and urban growth. Advancements in automated feature extraction can be applied to efforts such as humanitarian and disaster response, such as the events that took place with Hurricane Maria in Puerto Rico. The task was approached by attempting to solve two sub-tasks: land classification and building detection. The metrics assessed across both problems are IOU and F1 scores.

Land classification of seven distinct classes was explored through two architectures: UNet, SegNet, and PSPNet in combination with two backbones VGG16 and Resnet50. These models' performances were further compared to adjustments in the dataset via balancing the number of classes and applying weights to training instances based on their classes. The balanced dataset performed better, although its predictions were not as continuous. This resulted in the best model VGG16 with a PSPNet backbone. Building detection was explored through FPN, UNet, and LinkNet architectures with VGG16 with backbone, UNet with EfficientNetB3 backbone, and Pix2Pix. Each Neural Network performed better than the Random Forest baseline with the best model being the second version of our Pix2Pix model. It outperformed better by a large margin, able to generate fine corners and solid outlines.

I. Introduction

Feature extraction via satellite imagery has the potential to map currently un-mapped areas at scale, rapidly modify maps during times of disaster to help coordinate relief, and track global urban development in more detail. Satellite imagery is an important class of imaging data that has remained largely underutilized. Government agencies such as NASA or ESA and companies such as DigitalGlobe have access to terabytes of satellite images. However, there are few large-scale publicly available datasets, and data labeling is always a bottleneck for segmentation tasks. Two separate datasets of large resolution satellite images were identified for land classification and building detection tasks. After cursory exploration of the dataset, research was conducted into applications of architectures with similar project goals. The architectures of interest chosen were: UNet, Segnet, PSPNet, LinkNet, FPN, and Pix2Pix.

Comparison of the traits among architectures and their implications on our dataset are first explored. Comparison of advantages and disadvantages as well as modifications from source material in an attempt to scale and conform with the characteristics of our datasets are explained. Land classification is first examined at length, followed by similar examination of building detection of different phases of exploration. Data processing decisions along with initial experiments. The results of which impacted the model adjustments. In turn, the adjustments were informed by visual inspection of predictions as well as metric evaluation of mean Intersection-Over-Union, Precision and Recall, and F1 scores.

Finally the models of the two sub-tasks are applied to create a large image from stitching together the results as well as on a completely new image.

II. Related Work

Before getting into the model selection, it was important to do some review of the best models used for image segmentation. Jonathan Long et al. [Long et al., 2015] developed the fully convolutional network by replacing the fully connected layers in a convolutional neural network (CNN) with standard convolutional layers. Chen et al. [Chen et al., 2017] found that having consecutive convolutional and pooling layers caused a decrease of the spatial detail and with

that loss of spatial information. Xuedong Yao et al. [Yao X et al., 2019] found that 3 of the most accurate deep convolutional neural networks for image segmentation classification are U-net, Deeplab-v3, and SegNet. Hengshuang Zhao et al. [Hengshuang Zhao et al., 2017] proved that PSPnet has a good result in semantic segmentation problems with an overall IOU of 80.2%.

III. Network Architecture

i. Random Forest

A simple random forest classification model was chosen to serve as a baseline case study for the building detection problem. With only two categories (building and no building), we did not feel the need to extend this approach to the land classification problem. For the data input, two additional texture features were created that are often used in images classification problems: Haralick Features [Haralick et al., 1973] and Local Binary Pattern [Guo et al., 2010].

ii. UNet and SegNet

These two models are fully convolutional networks and their main characteristic is that they are composing of a contracting part called the encoder and an expansive path called the decoder. The contracting part is based on 3x3 convolutional networks, rectified linear units, and a 2x2 max-pooling operation. For UNet (Figure 1), the encoder or expansive path consist an upsampling feature map, 2x2 convolution, a concatenation, 2 convolutions layer of 3x3 each one followed by a rectified linear unit (ReLU) and at the end, a layer of 1x1 convolution is used to map each of the component feature vectors to the number of classes chosen [Olaf Ronneberger and Brox, 2015].

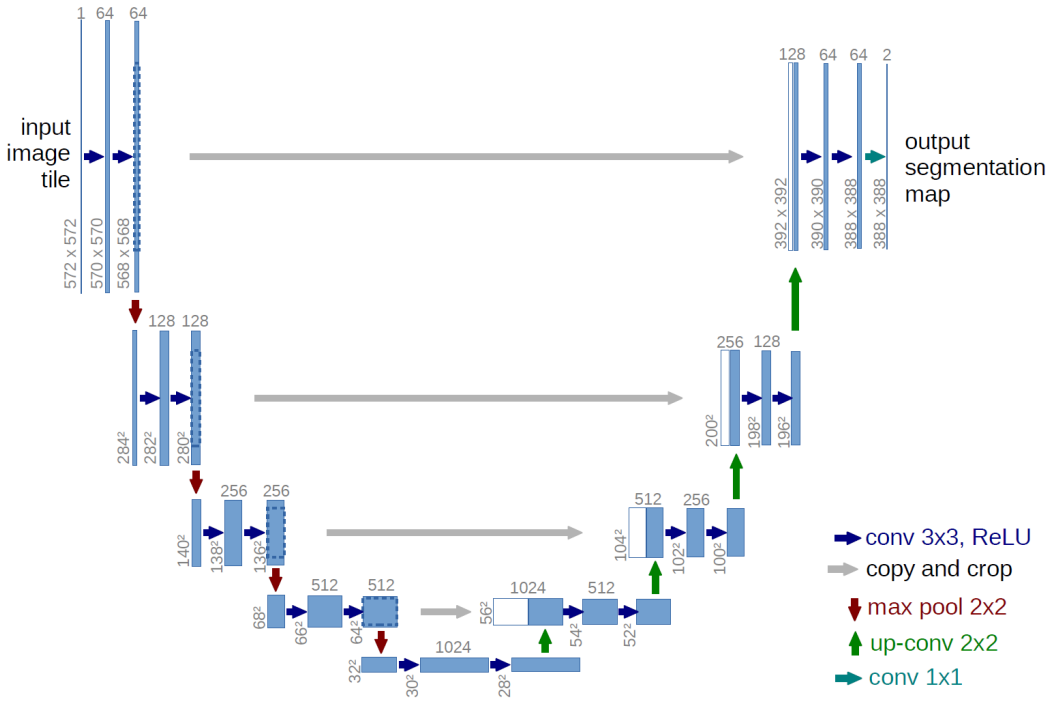


Figure 1: *UNet architecture*

On the other hand, SegNet (Figure 2) has the same characteristic as UNet architecture (an encoder and a decoder). The main difference of this model in comparison to UNet is that the decoders use the max-pooling indices from the corresponding encoder to compute the non-linear upsampling of their input feature maps. The advantages of this are increased performance of the boundary delineation and also reduces the number of parameters in the training process[Vijay Badrinarayanan and Cipolla, 2016]. This means that in SegNet the pooling indices are transferred from the encoder to the decoder using less memory, whereas in UNet the whole feature map is transferred using a lot of memory. These two models served as the first two CNN's used in the land classification problem.

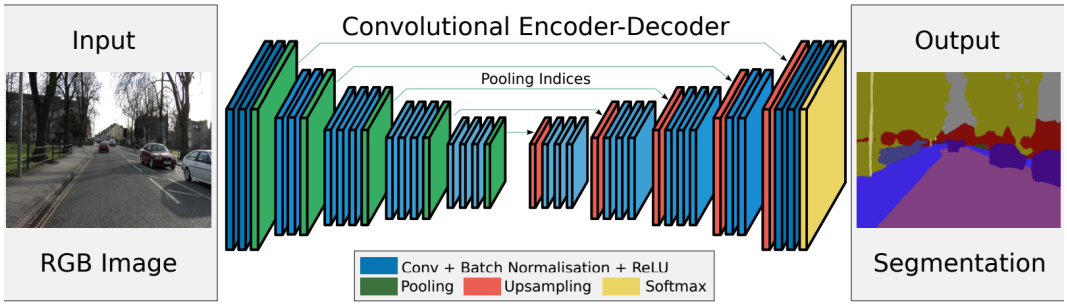


Figure 2: *SegNet architecture*

iii. Pyramid Scene Parsing Network (PSPNet)

PSPNet model (Figure 3) is another model that was found useful to make a prediction for image segmentation. This model uses convolutional neural networks to extract the feature map of an input image and then passes the features map by a 4 level pyramid where each of the pooling kernels covers a small, medium, and a large portion of the images, then it concatenates level pyramid information with the original feature map and finally, a convolution layer is used to make the prediction [Hengshuang Zhao et al., 2017]. One of the advantages of this model is that it can be used some pre-trained encoder such as ResNet-50 or VGG-16 to extract the feature map. This model served as the last CNN in the land classification problem.

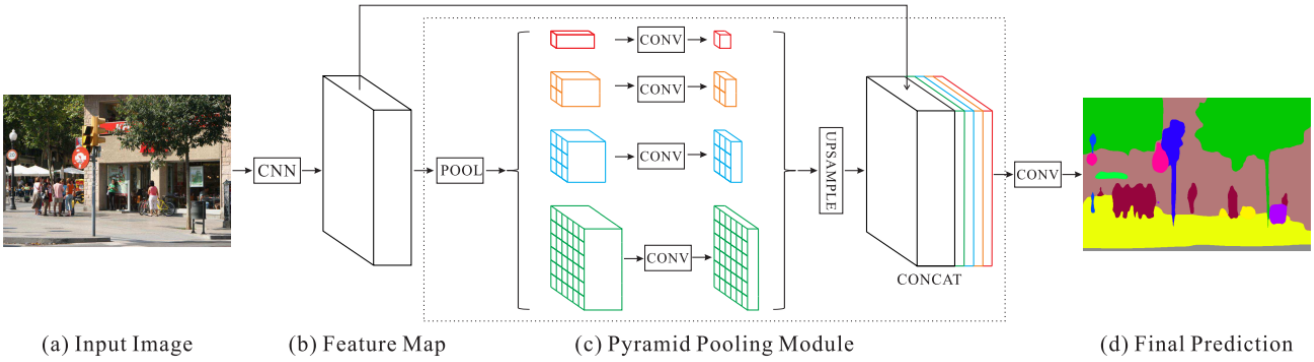


Figure 3: *PSPnet architecture*

iv. LinkNet and FPN

For the building detection case study, we adopted three models in this category of CNN's including the UNet, LinkNet, and Feature Pyramid Network (FPN) as shown in Figure 4 .

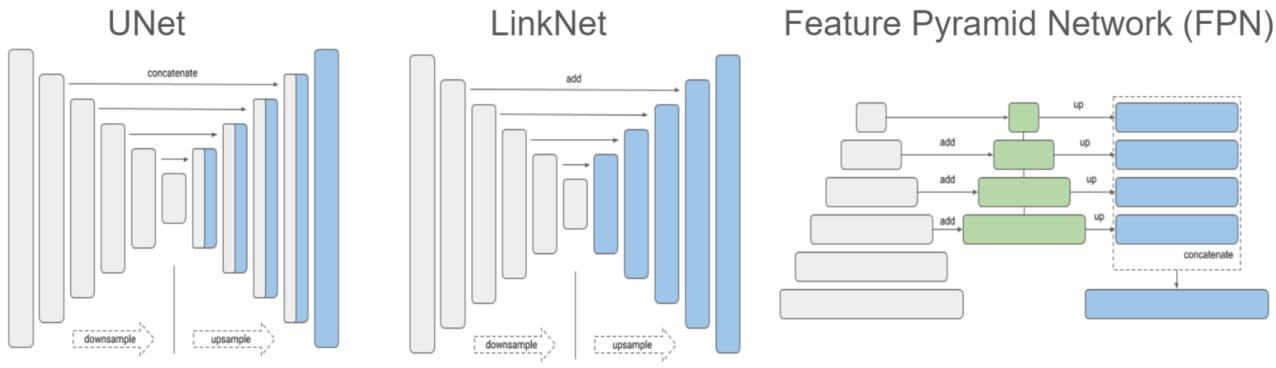


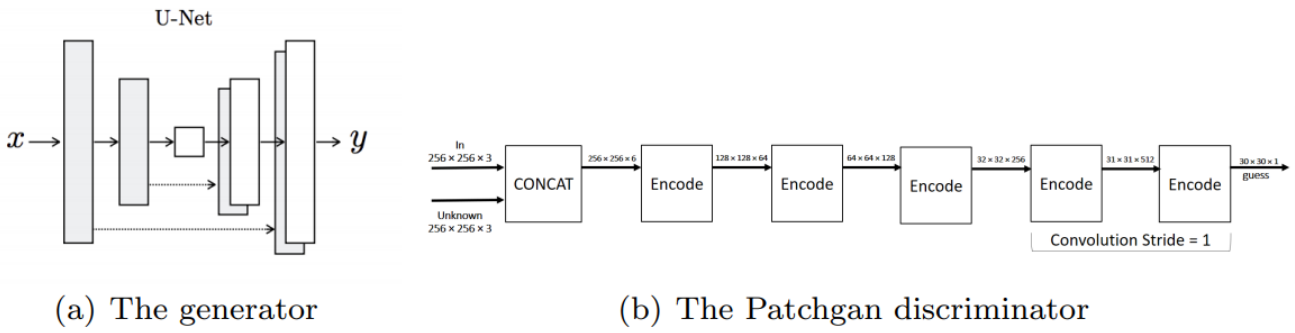
Figure 4: *Encoder Decoder Networks for Building Detection.*

LinkNet [Chaurasia and Culurciello, 2017] is a variation of the UNet network with two distinct differences. Firstly, LinkNet uses a residual module instead of the traditional convolutional block that is found in the UNet. Secondly, LinkNet skip connections get added in the decoding part of the model instead of stacking as found in the UNet.

FPN [Lin et al., 2017] share a similar structure to the UNet, in that the FPN has laterals connection between the bottom-up pyramid and the top-down pyramid as shown in Figure 4. But instead of appending the features the FPN applies a 1x1 convolution layer before adding them to the top-down pyramid.

v. Pix2Pix

Pix2Pix [Isola et al., 2017] is a conditional adversarial network that learns a mapping to transform the RGB satellite image into an output mask of buildings while also learning a loss function to train this mapping. The Pix2Pix network is built by using two components that includes the generator and the discriminator as shown in Figure 5. This model served as the best model in the building detection problem.



(a) The generator (b) The Patchgan discriminator

Figure 5: *Pix2Pix Generator and Discriminator.*

The goal of the generator is to take an input RGB satellite image and convert it to a generated building mask by using a traditional UNet architecture. In its initial form, the UNet did not have any pre-trained backbones with weights and it also used a tanh activation function for its last convolutional layer. A custom generator loss function was also built following the work of [Isola et al., 2017] where the loss function minimized a combination of the L1 loss and a sigmoid cross entropy loss between the generated images and a array of ones as shown in Figure 6a.

The discriminator on the other hand, had the task of measuring the similarity between the generated image and the ground truth mask. This part of the model was based on the PatchGan, a convolutional neural network that classifies an image in patches instead of individual pixels for computational cost reductions. For examples, each 30x30 patch of the output classifies a 70x70 portion of the input image as shown in Figure 7. It was built to receive 2 input building masks and determine the sigmoid cross entropy loss of the real images and an array of ones called the real loss. It then determined the sigmoid cross entropy loss of the generated images and an array of zeros called the generated loss. The task then trains to minimize the sum of these two losses. This processes is shown in Figure 6b.

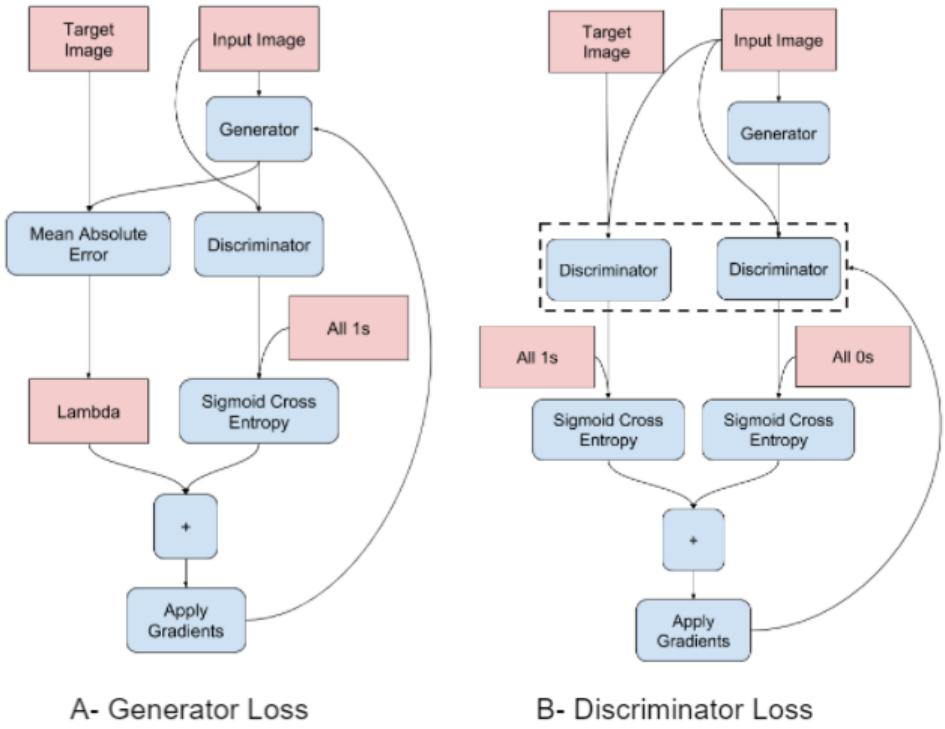


Figure 6: *Pix2Pix Generator Loss and Discriminator Loss.*

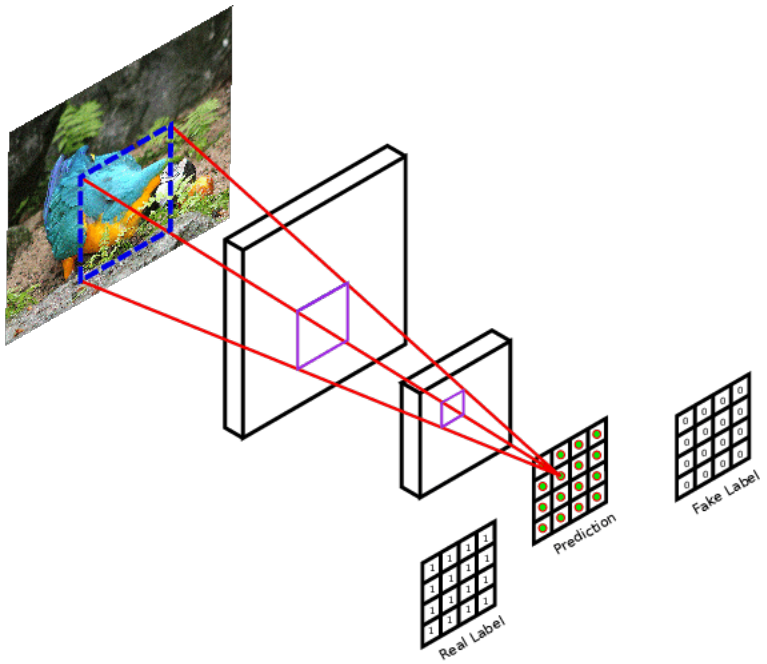


Figure 7: *Pix2Pix Discriminator Patch.*

vi. Metrics and Loss Function

The metrics selected for land classification and building detection were IoUScore and FScore. Intersection over Union (IoU) is one of the most popular metrics used for object category segmentation problems and it is highly implemented in semantic image segmentation. this metric measures the similarity between the predicted portion and the ground truth portion for an specific object located in the set of images, the IoU score is defined by the following equation [Rahman and Wang, 2016]:

$$IoU = \frac{TP}{FP + TP + FN} \tag{1}$$

Where TP is true positive, FP is false positive and FN false negative

IoU can simply define as the ratio of intersection over the union between the ground truth and the predicted region as it is shown in figure 8:

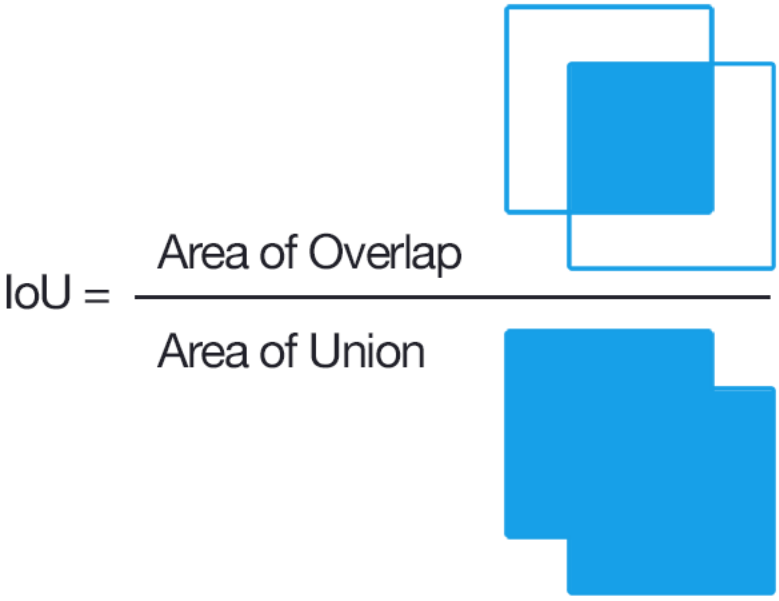


Figure 8: *IoU Representation*

Another metric that was used FScore which basically is the weighted average of the precision and the recall and this metric works for multi-label and multi classification problems, its formula is:

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{2}$$

Regarding the losses, it was used a dice loss in combination with focal loss. The reason is that for land classification and building detection were found some imbalance in some of the classes, therefore, it was necessary to use other types of losses different to cross-entropy loss functions which give equal importance to all of the classes. On the other hand, Dice and focal loss address the imbalance class by reducing the weights for samples that are easy to predict.

For land classification the following loss function were used:

$$Loss = dice\_loss + (1 * CategoricalFocalLoss) \tag{3}$$

For building detection the following loss function were used:

$$Loss = dice\_loss + (1 * BinaryFocalLoss) \tag{4}$$

the difference between binary and categorical focal loss is that the latter is used when the classes are more than two.

IV. Land Classification

i. Data Processing

The data used for the land classification side of the project was the DeepGlobe challenge dataset [Demir et al., 2018]. The DeepGlobe imagery is collected by the DigitalGlobe's satellite. All the images have RGB channels, 2448 wide x 2448 height and have a pixel resolution of 50cm. The dataset contains 803 train, 171 validation and 172 test images. Each of the training satellite images comes with its respective mask. However, the validation and test datasets did not contain any mask for the images.

Given that we only have masks for the train dataset, we divided the 803 images into train, validation and test. To get a bigger dataset, the images were split into 16 pieces of 612 x 612 pixels each, resulting in a total of 12856 images. Details on the size of the train, validation and test split size are provided on Table 1.

Table 1: *Train, Validation and Test Size.*

Dataset	Train	Val	Test	Total	Image Size
Raw	493	160	150	803	2448
Split	8221	2570	2065	12856	612
Split (%)	63.9	20.0	16.1	100	-

The first step for the pre-processing of the satellite images was the splitting. Second, each file was re-scaled to 512 width by 512 height. Finally, the images were normalized, ending up with three bands with a range of 0 to 1 for each image.

The DeepGlobe masks are also RGB files, where a different combination of the three bands represent each class, as shown in Table 2. The first step on the processing of the masks was to split them similarly to the satellite images, in 16 pieces. Second, we re-scale the images to to 512 width by 512 height. Then, we convert them to one band images, where the value of the pixel is the label for each land class. Finally, each mask was hot encoded, ending up with a file with 7 bands, each band for a different land class.

Table 2: *Land Classification RGB Mask Values and Labels.*

Class	R	G	B	Label
Unknown	0	0	0	0
Urban Land	0	255	255	1
Agriculture Land	255	255	0	2
Rangeland	255	0	255	3
Forest Land	0	255	0	4
Water	0	0	255	5
Barren Land	255	255	255	6

Moreover, we looked into the count of pixels for each of the classes throughout the dataset. The resulting histogram (Figure 9) shows a very unbalanced dataset, with the agricultural land cover (Label 2) being almost 6 times bigger than all the other classes. To solve this problem we propose two approaches.

The first one was to balance the dataset by deleting images with agricultural class pixel count over a certain threshold. Specifically, images with more than 80% and 90% of agricultural land cover. The 80% threshold resulted in a drastic reduction in the dataset size, ending up with just 2743 images in the train dataset. Furthermore, using the 90% reduction the train dataset reduced to 4332 images. The latter threshold was selected because the train dataset size was still big enough after the image deletion, and as shown in Figure 10, the unbalanced class is now in the same range as the other classes.

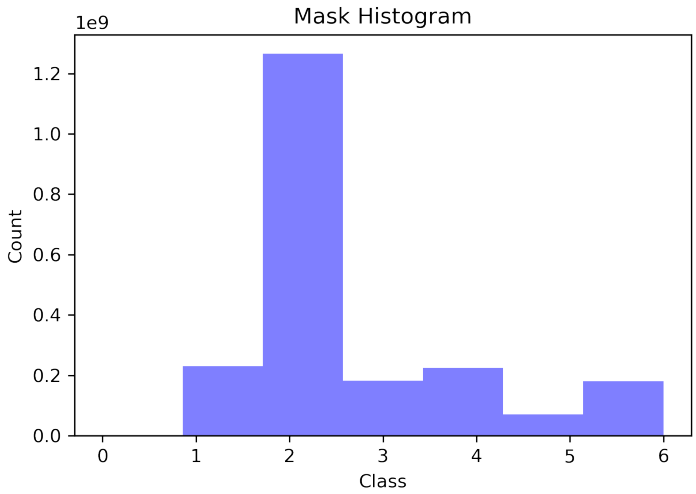


Figure 9: *Balanced Dataset Histogram.*

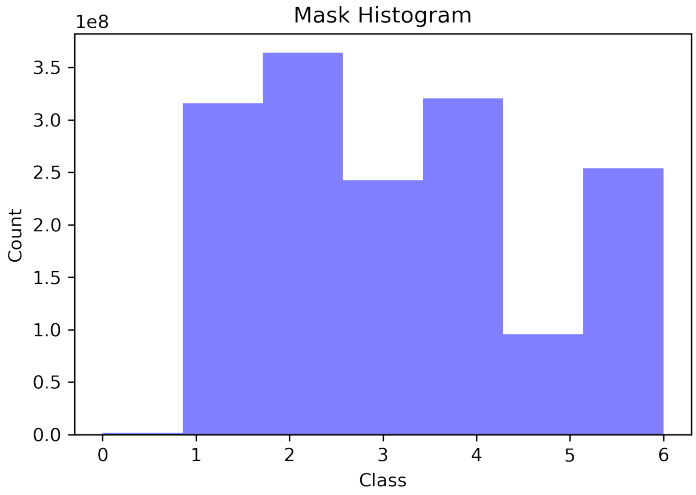


Figure 10: *Balanced Dataset Histogram, 90% Threshold.*

The second approach was to use weights in the training process for each model. It was

not possible to use class weights in the model training because we had more than 3 classes in our problem. Consequently, we estimated a weight for each sample image, depending on the percentage of agricultural land such image had. Moreover, these weights were incorporated inside the image generator code as a third input to the model training.

ii. Model Selection

The first step on the model selection was trying to find base architectures for SegNet, U-net, and PSPnet. After having the architectures, we started adjusting them into the land use land cover classification using the dataset previously processed. The following modifications were made to the base code in order to improve the metric scores:

1. The encoder part of the UNet and SegNet model was changed by ResNet50 and VGG-16. The main reason for this implementation was the need to pretraining the models using a large subset of images dataset such as Imagenet in a way to help to get better results in their training process. These networks have been trained on more than a million images allowing classifying images in more than 100 categories [Michalis Giannopoulos et al., 2020]. At the beginning of the project, some of the models had been trained using just the original dataset and the score did not ascend from 42% IoU. Resnet50 and VGG-16 also were used in combination with PSPnet model to also introduce some pretrained data into the model using Imagenet, as a result, the feature maps were gotten with a high accuracy. As a consequence the following combinations were obtained:

2. For the original Resnet-50 that was used, it has 5 stages with convolutional and identity blocks and each one of them has 3 convolution layers. however, for the VGG-16 a convolution block with a max pooling were create to decrease the latent space from 32x32x512 to 16x16x512, as a result, specific and important features can be filtered to get the important layers of the latent space.

Table 3: *All Model Combinations.*

Models
VGG16-UNet
VGG16-SegNet
VGG16-PSPNet
Resnet50-UNet
Resnet50-SegNet
Resnet50-PSPNet

3. The UNet and SegNet decoders were modified specifically before the last 2D convolution was added an upsampling layer to increase the output size from 256x256 to 512x512 .

After making some modifications to the base model. The next steps were to define the following callbacks:

- ModelCheckpoint was used to save the best weights monitoring the variable $'val\_iou\_score'$
- EarlyStopping was defined to stop the training processes when the $'val\_iou\_score'$ repeats 5 times.
- ReduceLROnPlateau was used to reduce the learning rate to help to converge and avoid the oscillation in the metric values

Table 4 contains the quantity of total trainable parameters and non-trainable parameters for each of the models. It should be noted that the number of trainable parameters corresponds mostly to the models which have UNet as a decoder. The principal reason is associated because for this type of decoders the whole features maps are transferred from the encoder, whereas with SegNet just the pooling indices.

Table 4: *Total Parameters for All the Models.*

Models	Trainable Parameters	Non-trainable Parameters
VGG16-UNet	24,123,463	2,944
VGG16-SegNet	11,549,767	1,920
VGG16-PSPNet	17,111,367	5,120
Resnet50-UNet	43,117,831	56,064
Resnet50-SegNet	14,832,391	32,512
Resnet50-PSPNet	29,863,431	58,240

As it was explained previously, two dataset were created to achieve this land use classification the first one was just balancing the dataset and the second one was adjusting the weights. Several combinations of models were trained using these two dataset. as a result, it was found that the best model that got outstanding results in IOU scores were VGG16-PSPNet using the balanced dataset and VGG16-SegNet employing the modified weight dataset, Table 5 summarize the IOU and F1 Scores of the different models executed.

Table 5: *Score Comparison for Model Selection.*

	IOU Score		F1 Score	
Models	**Balanced**	**Weights**	**Balanced**	**Weights**
VGG16-UNet	0.5204	0.57147	0.6625	0.6704
VGG16-SegNet	0.6534	0.68883	0.7770	0.7790
VGG16-PSPNet	0.70051	-	0.8086	-
Resnet50-UNet	0.5799	0.5845	0.7132	0.7203
Resnet50-SegNet	0.5949	0.64061	0.7292	0.7563
Resnet50-PSPNet	0.6348	-	0.7555	-

It is important to emphasize that in the training process, most of the models were trained in one Jupiter notebook, for this, it was necessary to save each of the trained models after the training process to save the best weights and the model at the same time, allowing us to make predictions in the future. In Figure 11, it can be seen that some of the images from the test dataset were used to make the prediction, to do this was necessary to load each of the

models, normalize each image and reshape them from 612x612 to 512x512 which was the same dimensions that were used to train the model.

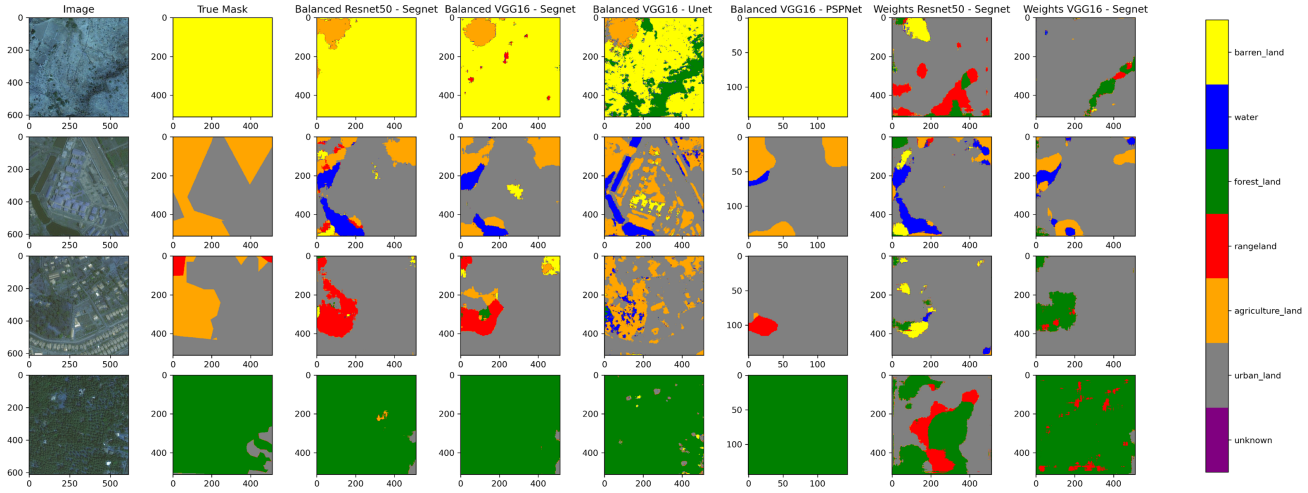


Figure 11: *Model Comparison.*

Focusing on the best models that got the best IOU score which was VGG16-PSPNet using the balanced dataset and VGG16-SegNet employing the modified weight dataset. it can be observed those models try to get a good prediction based on the true mask rather than the image, the cause of this problem is because that the true mask or ground truth does not capture the image's land classification with high definition, you can see that there is some forest land area in the first image that was not captured in the ground truth. Also, it was found the Balanced VGG16-UNet produces stunning results as regards land classification in comparison to the other models, it basically does a good job predicting the classes with really good details focusing more on the image than the ground truth, for this particular reason and the previously described reason related to the true mask problem justify why Balanced VGG16-UNet model got a low IOU score.

For the reason mentioned above, it was decided to choose the best models balanced VGG16-PSPnet and the Balanced VGG-UNet (Figure 12), the first one because it focuses more on the ground truth mask rather than on the image to make a prediction, and the latter the opposite side, it concentrates in predicting the details from the images in special outlining the edges.

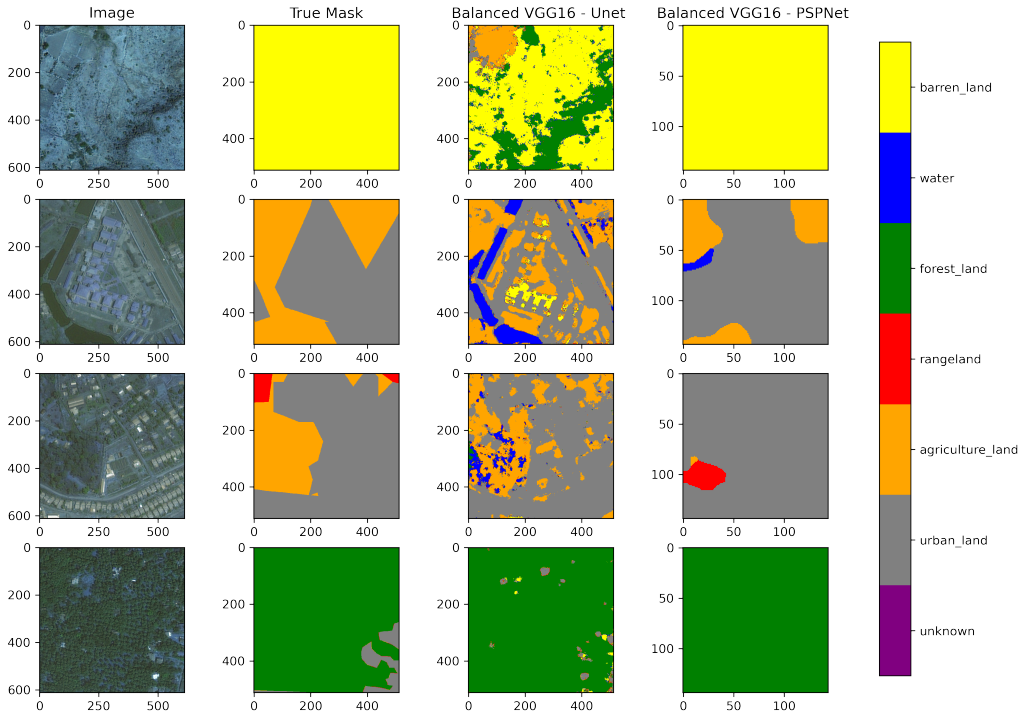


Figure 12: *Best Models Comparison.*

iii. Evaluation

The test dataset with 2065 images was used to evaluate the selected model (VGG - UNet - Balanced). Moreover, we used the classification report from sklearn to calculate the precision, recall and F-Score in the test data. Then, we plotted the report using a heat map from seaborn to better visualize the scores.

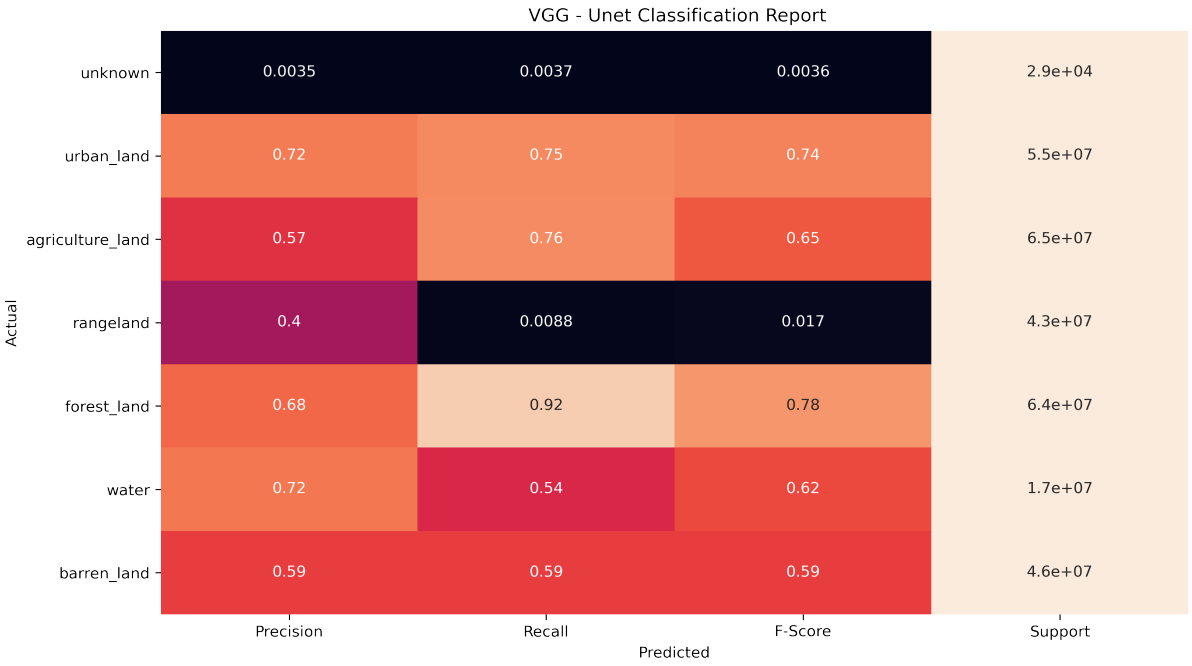


Figure 13: *Classification Report for VGG-UNet-Balanced.*

Looking at the classification report in Figure 13, we can see that the urban land and the forest were over all well classified, with a relatively high precision and recall. Furthermore, agricultural land shows a high recall with a low precision, meaning that the model is incorrectly over-classifying pixels as agricultural land. On the other hand, the water has a low recall with a high precision, meaning that the model is under-classifying that class. The rangeland class performed poorly in the three metrics, this can be due to most of the rangeland being difficult to differentiate from the agricultural land.

iv. Results

To showcase the ***Capacity*** of the model, we wanted to classify one complete image from the original test dataset that did not have a corresponding mask. A random image from the original test data was selected, with a size of 2448 wide x 2448 height. Similarly to the training data, we split the image into 16 pieces. Each piece was then normalized and re-scaled to 512 x 512.

VGG - UNet - Balanced model was used to classify the 16 images. Then, the predicted masks were put together to re-ensemble the original image.

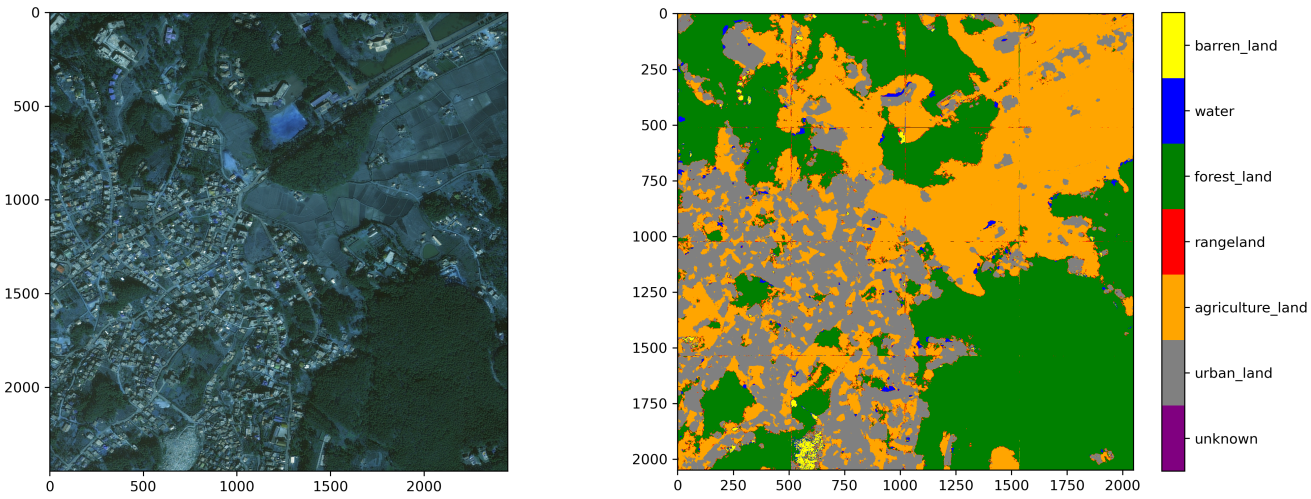


Figure 14: *Predicting Big Image with VGG-UNet-Balanced.*

Looking at the predicted mask on Figure 14, we can see that the forest and urban classes were correctly classified in the most part. Moreover, there were some water pixels incorrectly predicted near buildings. Agricultural land seems to be correctly predicted overall. However, there is still some overestimation near the urban land.

iv.1 Activation layers

Only 10 of the features maps were plotted because they have a lot of them

In figure 3 shows the following characteristics:

- the 3 convolution layers of block number 3 highlighting several features of the images.
- the latent space wherein most of the images are distorted, however, in two of them can be seen clearly a triangular area showing the urban area and that area is converted to some building and urban areas in the following upsampling layers

Figure 4 shows one convolution layer for each of the 5 blocks. It can be observed how some of the features in the image such as building forest and agricultural areas have more intensity

(they are key characteristic features). It was also noticed that the block of convolutions increases the images start looking distorted.

Figure 5 shows each of the 5 upsampling layers. it can be observed how some incredible features start to appear from the low-resolution images and the resolution and size start increasing with each of the upsampling layers

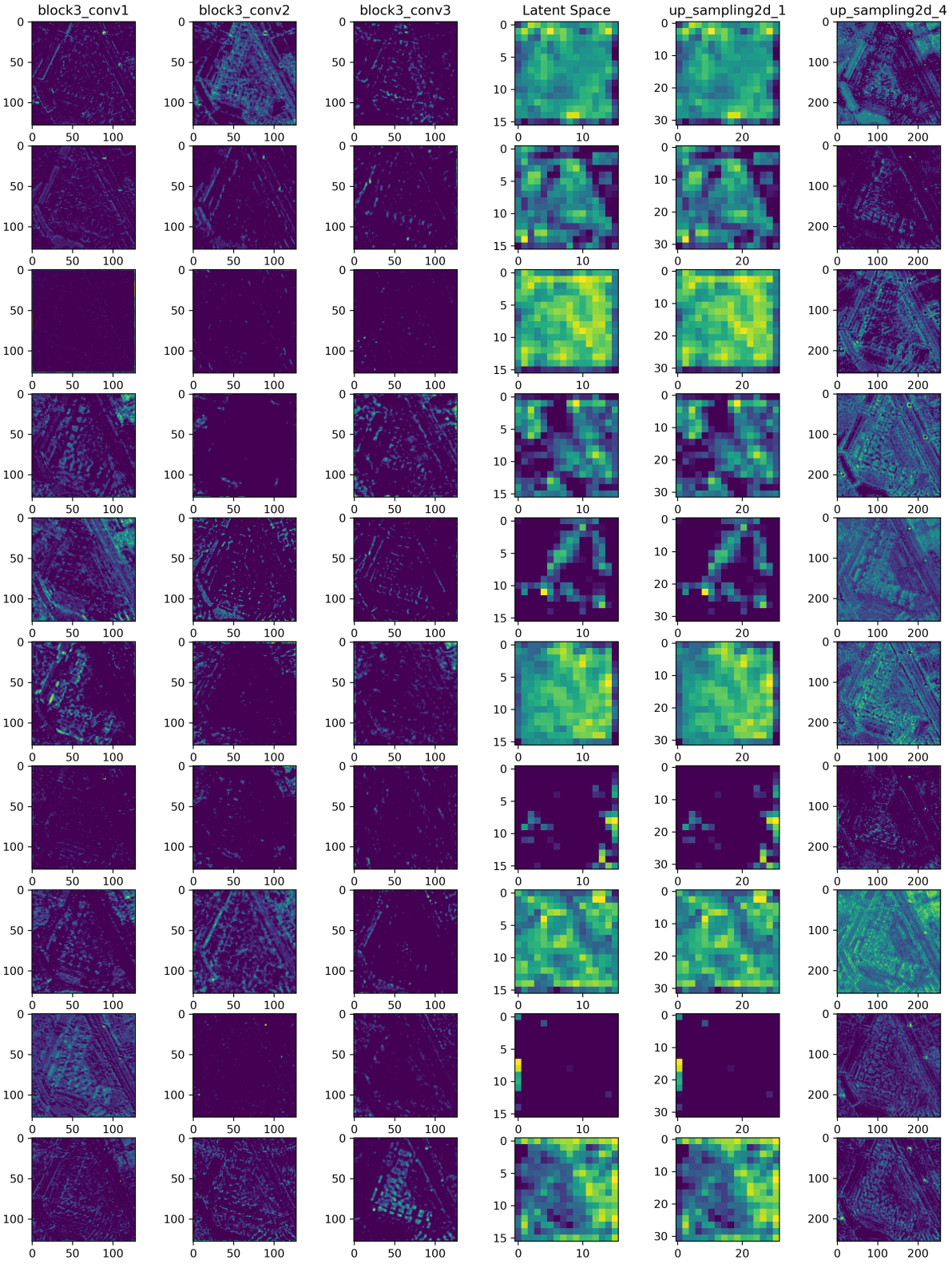


Figure 15: *Several activation layers*

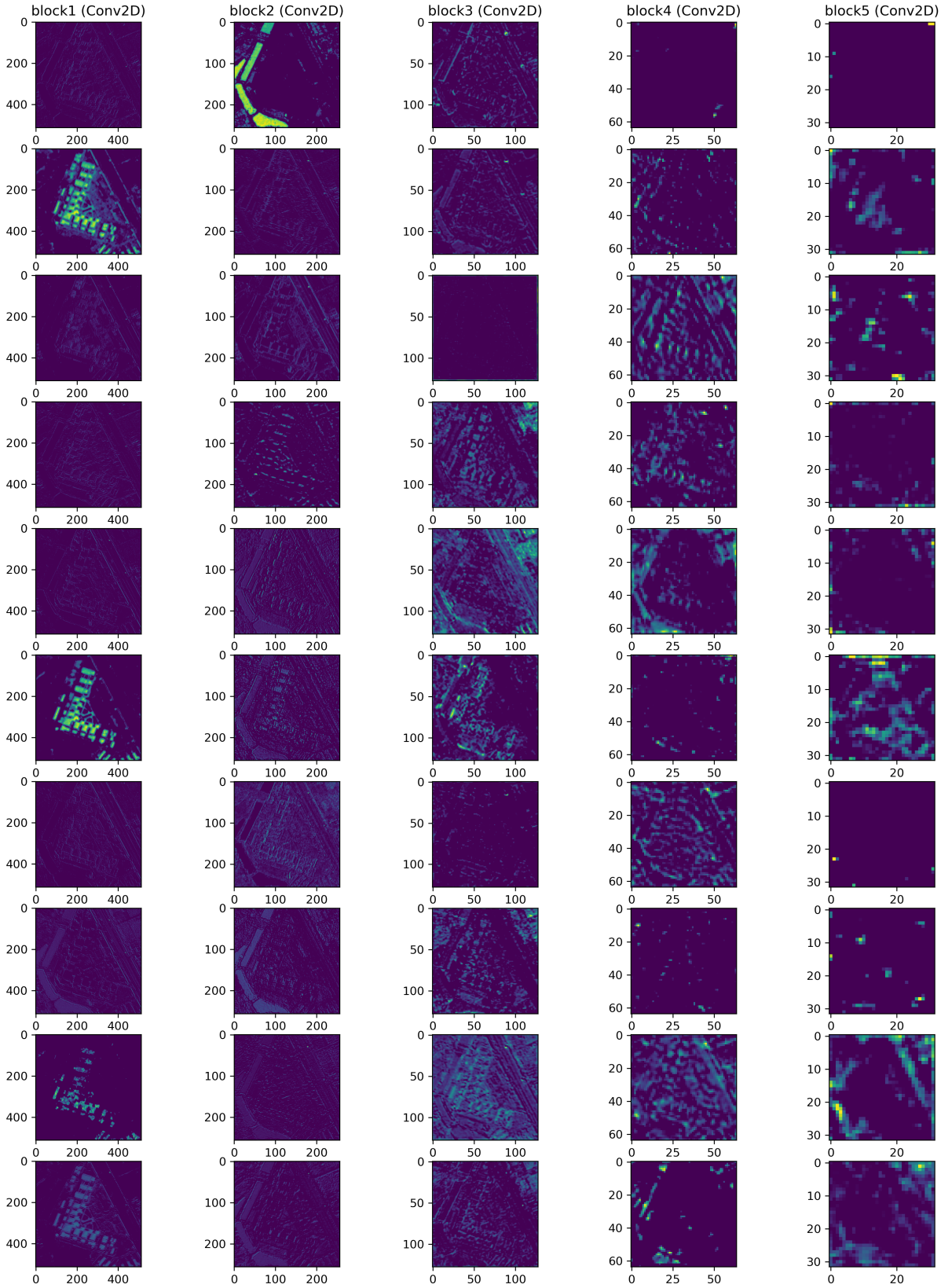


Figure 16: *Activation layers of the convolution blocks*

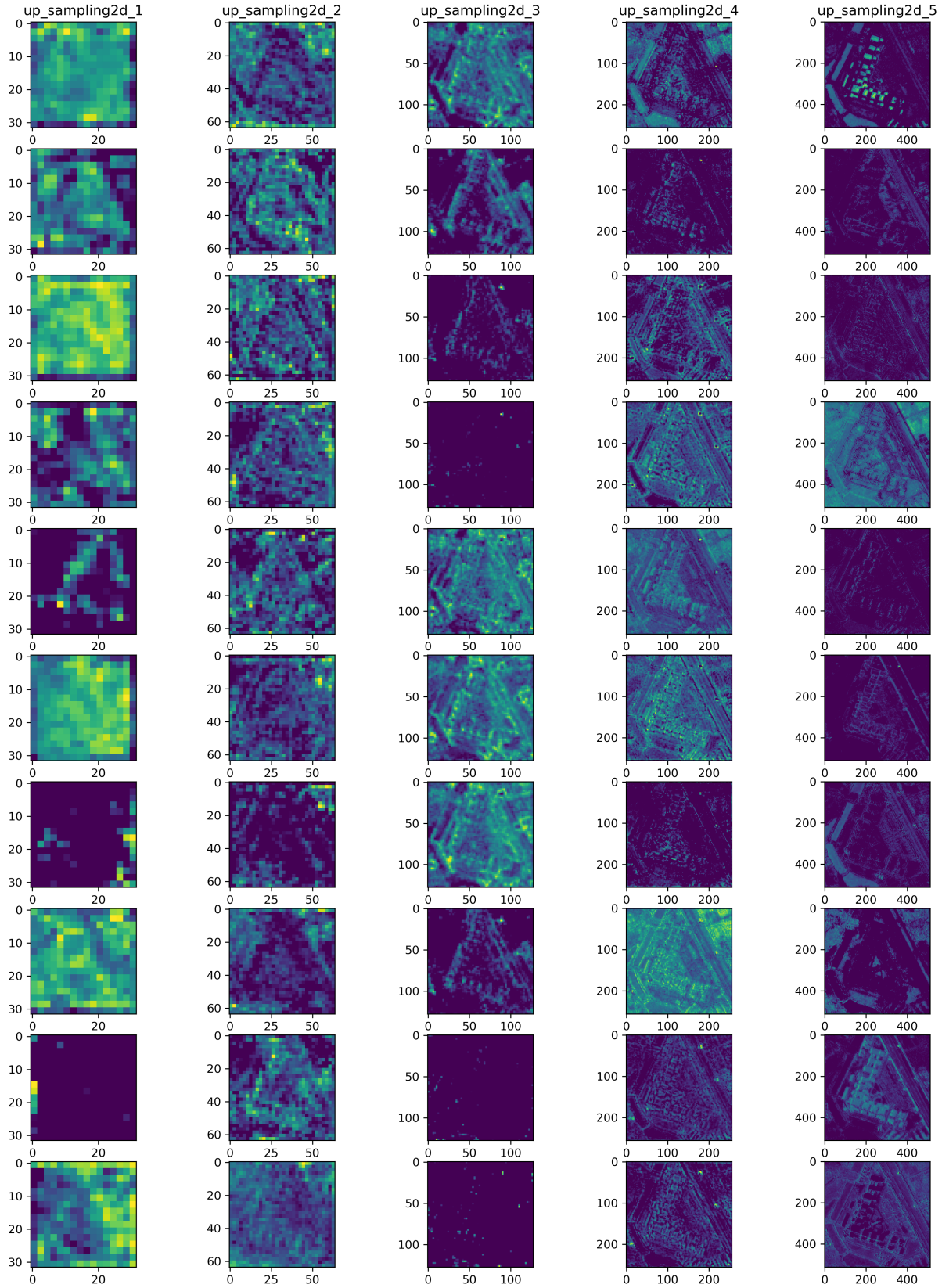


Figure 17: *Activation layers of the upsampling layers*

V. Building Detection

i. Data Processing

The data used to train and detect buildings in satellite imagery was the SpaceNet.ai Building Detection v2 dataset (Spacenet) [Van Etten et al., 2018] which contained over 150,000 building footprints over the city of Las Vegas, NV and was released to support an online competition. Each image sample covered an area of 200m x 200m and included an associated 650 x 650 pixel RGB satellite image. A series of pre-processing steps had to be performed to use this dataset in training. In its original form, the RGB satellite image came as a GeoTIFF raster dataset with an associated building footprint mask as a GeoJSON file. We had to extract the 3-band data from the GeoTIFF, re-scale the data into images of size 512 x 512 pixels, and finally normalize the re-scaled data from an image that had values up to 255 to final images with values from 0 to 1. For the ground truth masks, we had to convert the georeferenced GeoJSON files into binary masks with values of ones for pixels where a building was present and zero otherwise. An example of the final images and masks can be seen in Figure 18. We then split the resulting dataset into groups of training, validating, and testing using the following ratios of 75%, 15%, and 10% respectively. Finally, the data was grouped into batches of 60 images to help avoid out of memory errors in each of our compute systems due to limited memory.

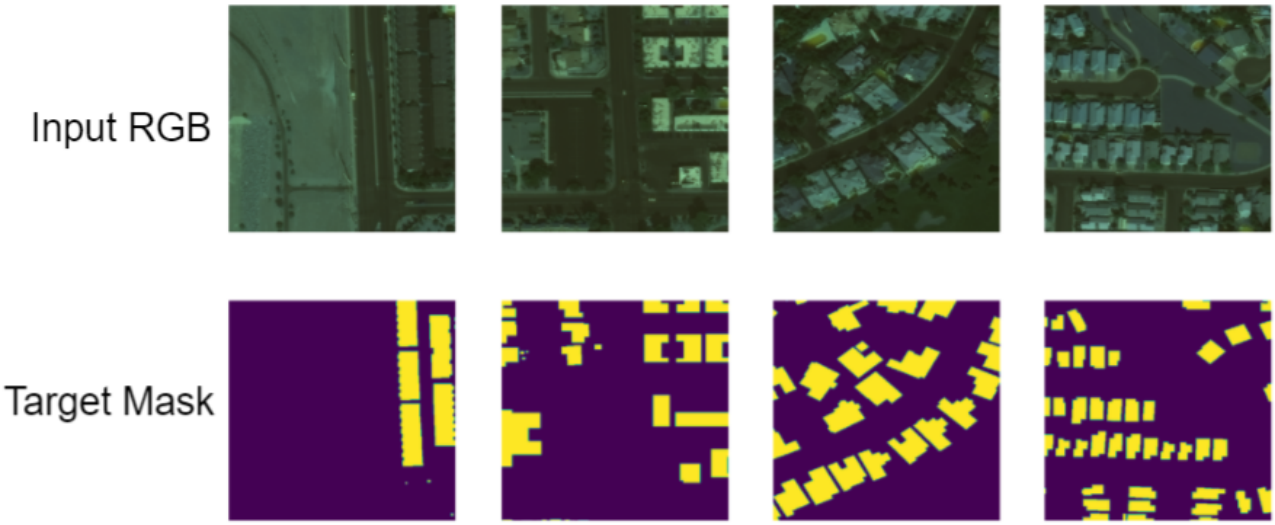


Figure 18: *Building Detection input images and target masks.*

ii. Methods

The main objective of this experiment was to explore semantic segmentation of satellite images and generate accurate building footprints based on a various deep learning architectures. To determine the best approach, we proposed a series of experiments listed below:

1. Random Forest for baseline case
2. Using an encoder decoder model (UNet architecture)
3. Adding a pre-trained encoder backbone to the UNet
    - VGG16
    - EfficientNetB3
    - Resnet50
4. Adjusting the Loss function and model metrics in case 3
5. Using LinkNet and FPN instead of UNet
6. Using a Conditional GAN (Pix2Pix) with simple UNet generator & PatchGan discriminator
7. Modifying the discriminator in case 6 by updating the patch window size
8. Modifying the generator's activation function in case 6
9. Replacing the generator in case 8 to use the case 4 UNet with pre-trained backbones

ii.1 Encoder-decoder network modifications

The initial strategy was to use an encoder decoder network without any pre-determined weights, but in practice the best approach was to use transfer learning within the encoder part of the network. In previous works, this was shown to enhance the performance of classification on remote sensing data [Marmanis et al., 2015]. The three encoder backbones that were chosen included the VGG16, EfficientNetB3, and Resnet50 which were previously trained on the 2012 ILSVRC ImageNet dataset. We wanted to see what effect these weights had in the UNet model without any further training, thus we loaded up the weights and passed a training examples

through the network as shown in Figure 19. From this example it was clear that the Vgg16 and EfficientNetB3 could outline the edges of the image so we chose to only use these two pre-weights.

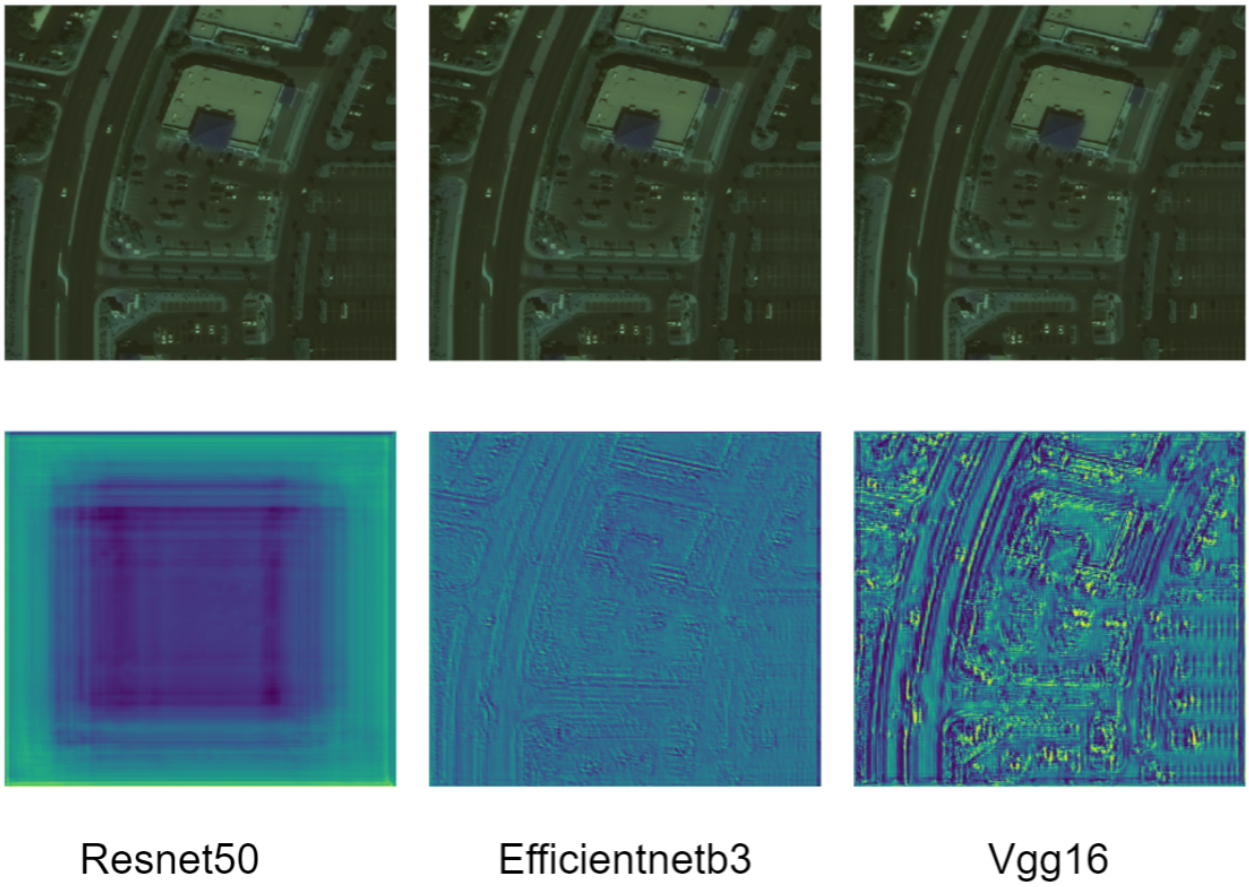


Figure 19: *Model outputs using pre-trained weights before any further training.*

We then tried using the model with a sparse categorical cross entropy loss function and l1 error metric without any success. This led us to use an improved dice loss function and focal loss as described previously. We also chose to use the IoU error metric to minimize, due to its performance in image segmentation problems. Finally like previously mentioned, we experimented with different encoder decoder models like the Unet, LinkNet, and FPN models. EarlyStopping and ReduceLROnPlateau were also used following the same setup as mentioned in the land classification section.

ii.2 Pix2Pix Modifications

Experimentation was done on the effect of the discriminator resolution on training. As previously mentioned, the discriminator classified an image patch instead of an individual pixel, so we experimented with the size of this patch. The initial size of 32 x 32 was increased to 128 x 128, resulting in better predictions on the edges of buildings, but the cost of time in training was dramatic. We decided to use an output patch of 70 x 70 for the discriminator instead, serving as a middle ground choice between the two extremes. We also noticed that the Pix2Pix original generator used a tanh activation function within the simple Unet. But from initial testing, this was causing the generator to create building masks that had values between negative one and one causing the discriminator to "win" more frequently in discerning the generated image. To correct this imbalance, we chose to used a sigmoid activation function instead. Finally, the most significant change in the Pix2Pix model was replacing the simple Unet all together with the pre-trained Unet from the previous example (case 4). For evaluation, Pix2Pix v1 was the model that had no modifications and Pix2Pix v2 had all the mentioned modifications.

iii. Results

iii.1 Random Forest

Initially a forest of 250 trees of depth 12 was evaluated on fifty images, however this failed to scale onto the entire dataset. The next iterations used 1,000 and 2,000 trees. There was no difference in their metrics score, indicating that the optimal number of trees might exist between 250 and 1,000 trees. As can be seen in 20, the model has trouble predicting among the three types of residential houses depicted in the input.

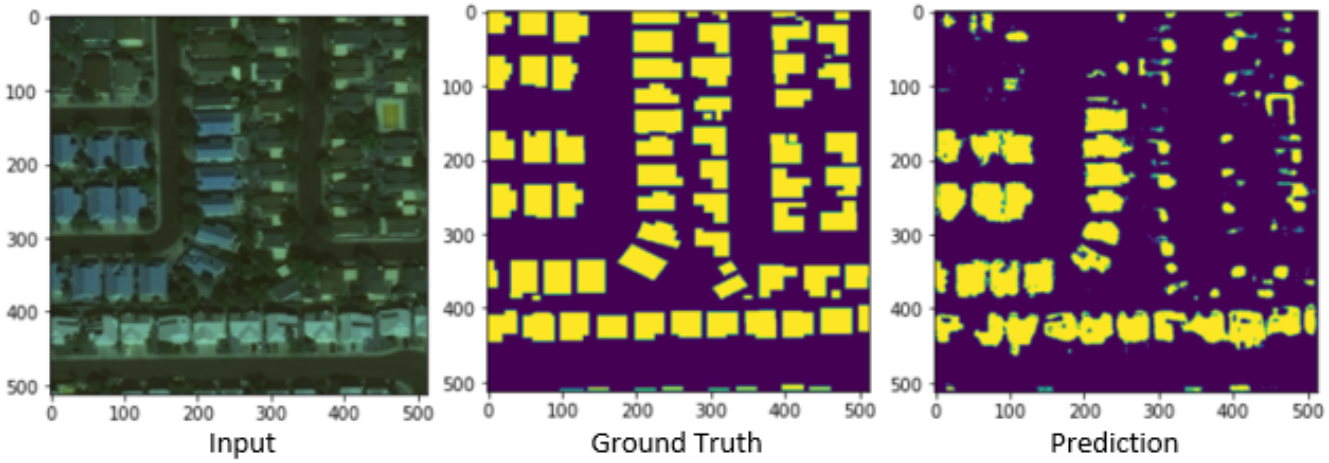


Figure 20: *Sample Random Forest Prediction*

Table 6: *Score Comparison for Model Selection.*

Model version	mIOU	Recall	Precision	F1
Random Forest	78.00	88.00	87.00	87.49
Pix2Pix v1	87.62	89.07	89.39	89.23
EfficientNetB3 - UNet	88.96	93.33	87.69	90.42
VGG16 - FPN	89.44	91.99	89.87	90.91
VGG16 - UNet	89.66	92.25	90.02	91.12
VGG16 - Linknet	89.71	91.61	90.76	91.18
Pix2Pix v2	**94.26**	**95.04**	**95.44**	**95.24**

The primary metric of comparison across the models is the mean IOU score. After some model tweaking, the Pix2Pix v1 model was able to outperform the Random Forest. The combination EfficientNetB3 with UNet backbone had increased performance by over 1 percent more. The combination VGG16 with various backbones had increased performance by less than 1 percent more. However, the Pix2Pix v2 with modified UNet generator model performed almost 5 percent greater than the next best model. Furthermore, it appears that the Pix2Pix architectures favor predicting ground truths with higher precision than recall. This is exhibited when inspecting the predictions, in that they are able to resolve acute geometries and generate uninterrupted shapes with sharp contrast to a quality that is unmatched by the other models.

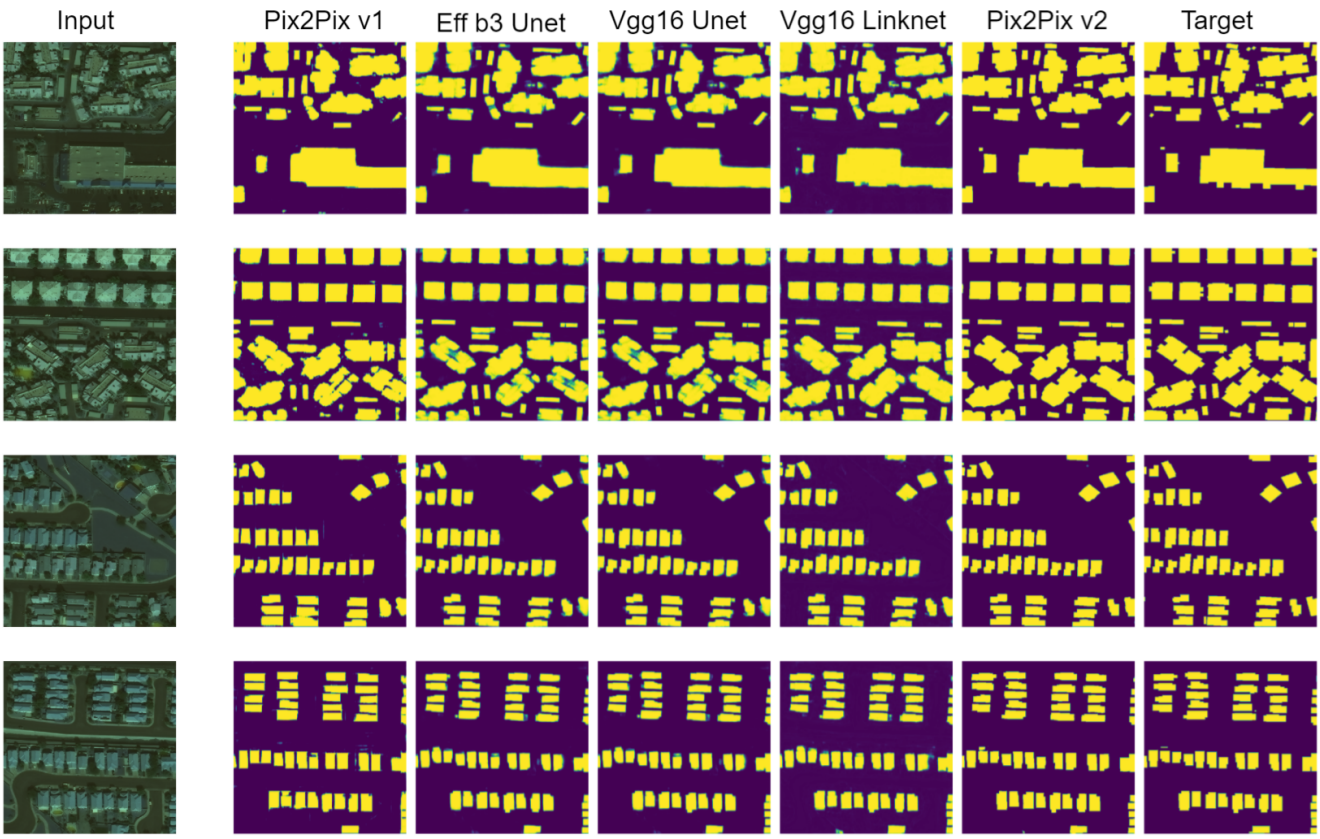


Figure 21: *Building Detection Results for All Models.*

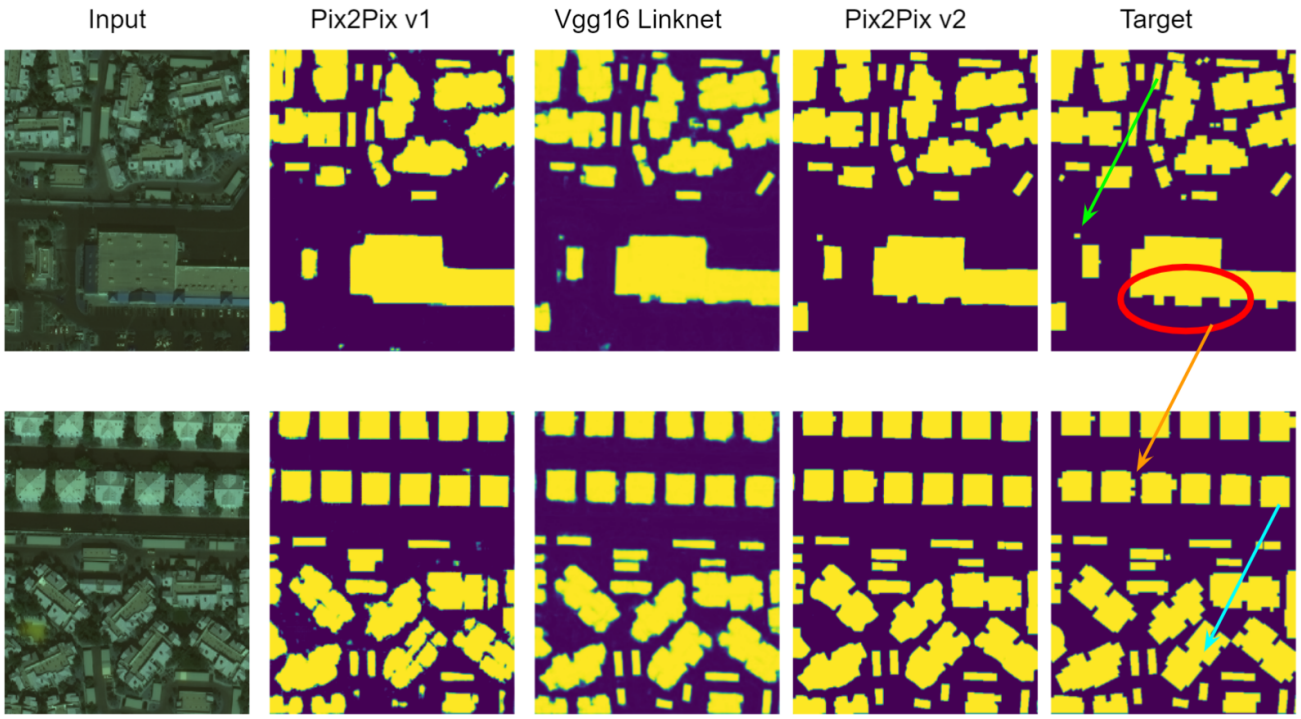


Figure 22: *Zoomed in Building Detection Results.*

Figure 21 shows a set of images from the test dataset that never seaw any training or verification. The models are ordered from worse performer to the best (left to right) in the figure. From here one key result was the performance of the pre-weight selection, the Vgg16 models had the best results out of all the encoder decoder networks. The linknet also performed better than the Unet, where the model was starting to output small features of each building. The best model ended up being the Pix2Pix v2 which included the most modifications. Figure 22 highlights this very well when looking at the key features marked by circles and arrows. Within the circle, the Vgg16 Linknet started to distinguish some of the smaller features but, the pix2pix v2 could easily determine it. the orange arrow highlights the same point in the second image below. Small features too performed better in pix2pix v2 as shown in the green arrow. Finallt, the pix2pix v2 model could segment out complete buildings without any trouble as shown by the cyan arrow.

VI. Merging the Two Models

The merging of the projects was done by first extracting the building labels from the building detection model output. Then, the building labels were incorporated to the land classification mask as another class. We took the same approach described before for predicting on bigger images for both of the models, slicing and predicting on pieces of the image. Figure 23 shows the result of the merged product.

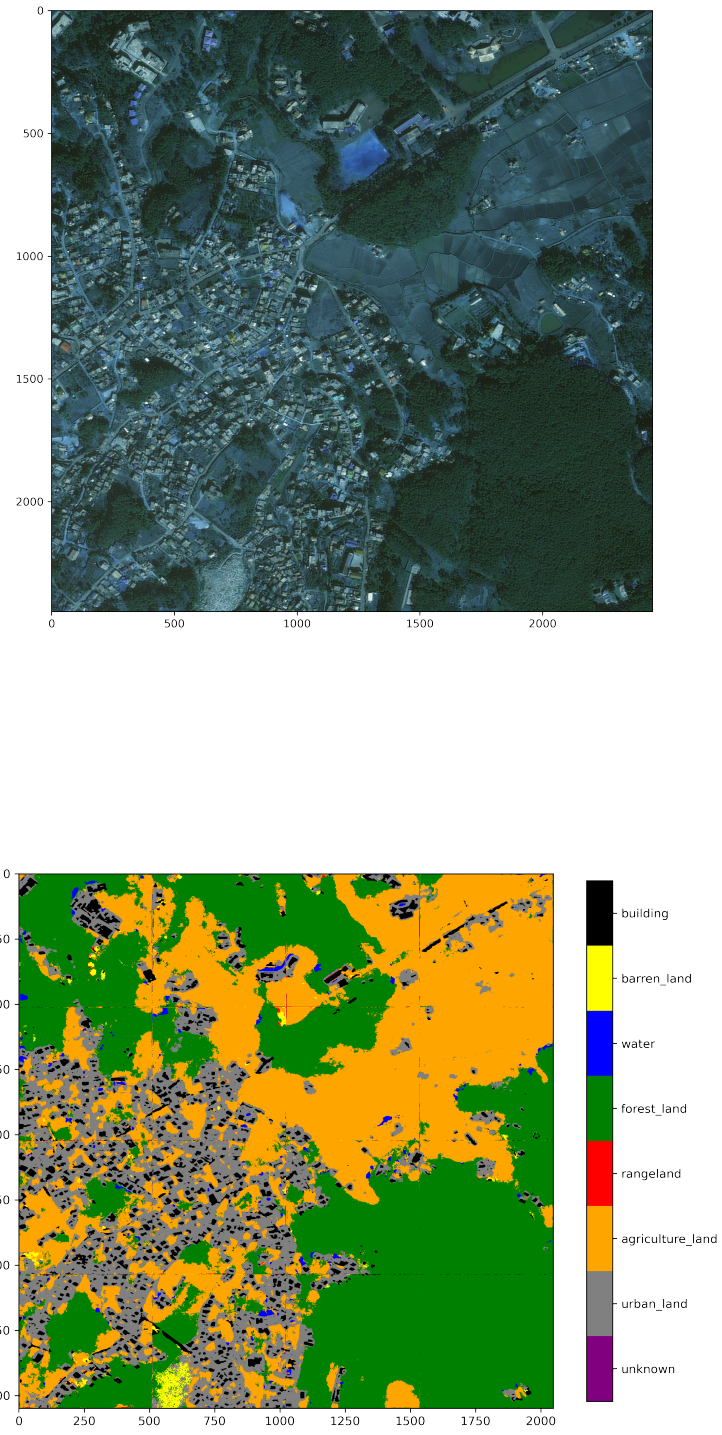


Figure 23: *Predicting Big Image with Merged Product.*

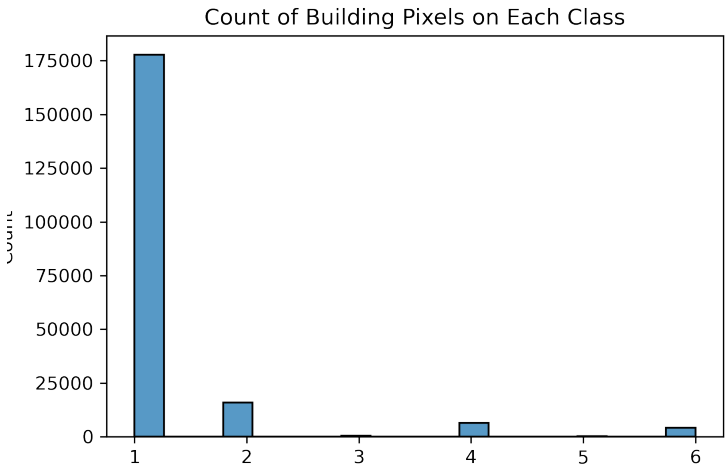


Figure 24: *Count of Building Pixels in the Land Classes.*

Finally we wanted to test the models in an area where urban land cover was in development, and see how the models help us identify changes in the land cover and urban expansion. To test this we looked for different sources of satellite RGB images, such as Sentinel, Landsat and Ikonos. The images from these satellites did not have a good enough spatial resolution for us to use them in the project. However, we were able to extract images from Google Earth with a sufficient resolution to test in our models.

The selected area of study was a residential area near Washington. We were able to download images for 2003, 2010 and 2014. These three years capture the change in land cover of the area. The results for this study are shown in Figure 25. For 2003 the land classification is doing a good job of in classifying the forest and barren land. However, there are some areas in the middle of the image where the pixels were misclassified as urban and water. The building detection was able to get most of the buildings in the urban areas on the left side of the image. The center of the image was also a problem for this model, the prediction is showing some nonexistent buildings near the barren land. Both of the models were able to capture the change from 2003 to 2010. The building detection performed better in this year, correctly predicting the newly constructed buildings. The land classification model was also able to capture the new urban areas on the image. Nevertheless, the model had some problems with the shadows of the buildings, and started to predict water in those areas. 2014 was very similar to 2010, with some minor differences. Some roads were now misclassified as buildings and the water

prediction on the building shadows reduced by a small amount.

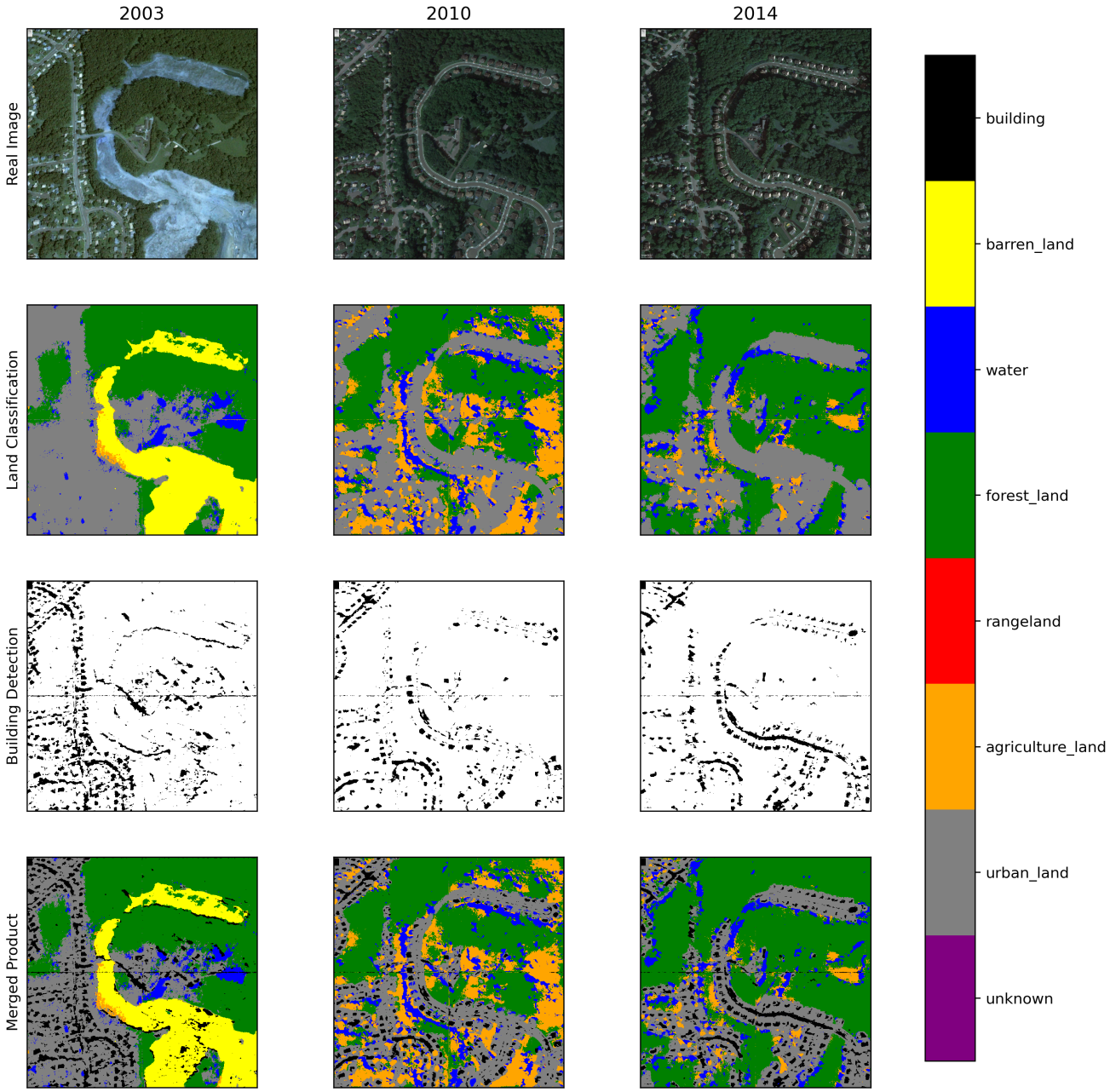


Figure 25: *Change of Land Cover Over the Years.*

VII. Contributions

Total github commits: 330

- Said: 88
- Juan Pablo: 82
- Harold: 78
- Joe Brian: 73 *initial commits weren't under contributor account

Total hour for project

- Said: > 100
- Juan Pablo: > 100
- Harold: > 100
- Joe Brian: > 100

Final Paper contribution

- Said: 25 %
- Juan Pablo: 25 %
- Harold: 25 %
- Joe Brian: 25 %

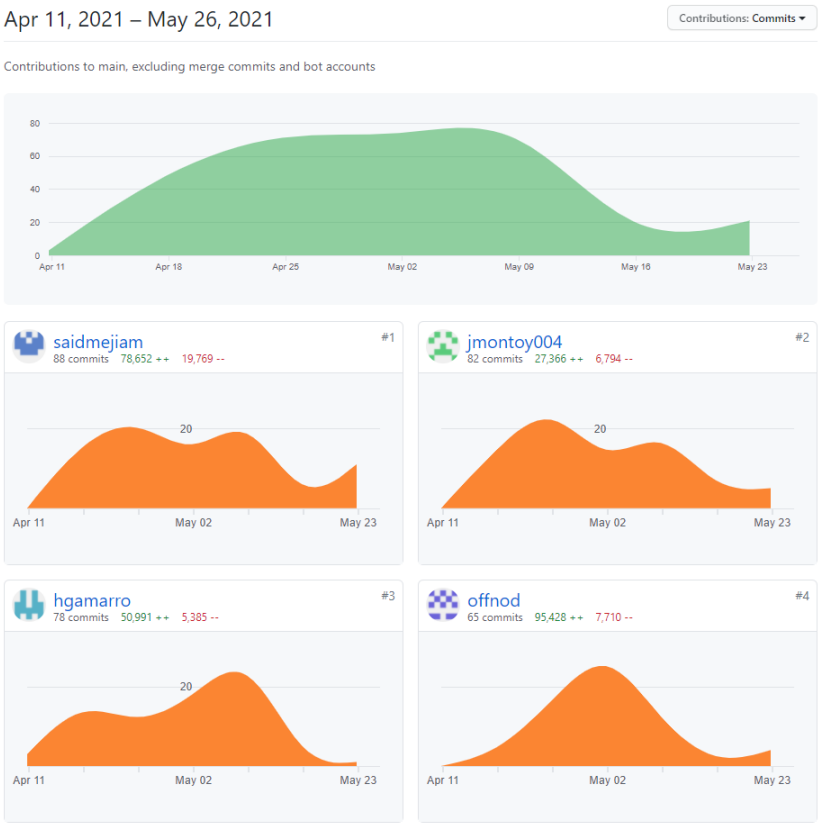


Figure 26: *Github Commit Contributions.*

NOTE: Most of these notebooks below are in /notebook/scratch folder the main notebooks are in /notebook and we contributed equally for all of those

i. Harold Gamarro

Notebook Proof:

`Tutorials`

`00-HG-multi_segmentation_(camvid).ipynb`

`00-HG-Classifier_tutorial.ipynb`

`Data generation and testing`

`01-HG-datagenerator_test_1.ipynb`

`01-HG-datagenerator_test_2.ipynb`

```
01-HG-datagenerator_test_3.ipynb
01-HG-Data_explore_1.ipynb

Exploring models
02-HG-Exploreing_1st_model.ipynb
02-HG-Exploreing_2nd_model_NCAR.ipynb
02-HG-Exploreing_2nd_model_NCAR_v2.ipynb
02-HG-exp_diff_unet_backbone.ipynb
02-HG-Main_Encode_Decoder_train_FINAL.ipynb
02-HG-UNET_test1.ipynb
02-HG-UNET_test_2.ipynb
02-HG-UNET_test_3_NCAR.ipynb
02-HG-v1_pix2pix_Sat_FINAL.ipynb
02-HG-v2_pix2pix_FINAL.ipynb
02_HG_SM_ALL_MODELS_true_NCAR.ipynb

Final results
03-HG-all_model_plot_FINAL.ipynb
03-HG-Encoder_Decoder_eval_FINAL.ipynb
03-HG-UNET-explore.ipynb
```

Other tasks:

- Project manager and meeting coordinator
- **Secretary**
- Setup and manage github repo
- Read multiple academic papers
- Setup CPU cluster for training
- Setup GPU cluster for training
- Wrote the src code "Library" for shared function including data processing

ii. Said Mejia

Notebook Proof:

```
Tutorials
00-SM-Tutorial_1.1_All_Dataset.ipynb
00-SM-Tutorial_1.ipynb
01-SM-Big_Images_Sentinel2
01-SM-ResNet50.ipynb
01-SM-ResNet50_pretrained_imagenet.ipynb
01-SM-VGG16_pretrained_imagenet.ipynb
01-SM_RESNET_SEGNET.ipynb
01-SM_RESNET_SEGNET_80_balanced.ipynb
01-SM_VGG16_UNET.ipynb
01-SM_VGG16_UNET_V2_of.ipynb
01-SM_VGG16_UNET_V3.ipynb
01-SM_VGG16_UNET_V4.ipynb
01-SM_VGG16_UNET_Version2.ipynb
01-SM_VGG_SEGNET.ipynb
01-SM_VGG_SEGNET_80_balanced.ipynb
01-SM_VGG_UNET_weights_balanced.ipynb

Exploring models
02-SM_ResNet_UNET_V1.ipynb
02_SM_ALL_MODELS_false.ipynb
02_SM_ALL_MODELS_true.ipynb
02_SM_ALL_MODELS_V1_true_NCAR-Copy1.ipynb
02_SM_ALL_MODELS_V1_true_NCAR.ipynb
02_SM_RESNET50_SEGNET_FINAL_Weights.ipynb

Final results
03-SM-Models_Validation.ipynb
```

03-SM-ResNet50_Validation.ipynb

03-SM_All_Models_Balanced_80.ipynb

03-SM_All_Models_Balanced_80_Activation_Layers.ipynb

03-SM_All_Models_Balanced_80_Architecture.ipynb

03-SM_VGG16_Unet_Activation_Layers.ipynb

Other tasks:

- Read some paper of the model implementations
- find the dataset
- Setup Computer GPU for training process (using CUDA)
- Write some functions to evaluate the results
- Modify the Unet, Segnet and VGG-16 base models to increase the performance of them and also to make them work for our problem

iii. Juan Pablo Montoya

Notebook Proof:

Tutorials

00-JM-Tutorial_1.ipynb

00-JM-Tutorial_2.ipynb

Data Preprocessing

01-JM-Data_Generator_V1.ipynb

01-JM-Data_Preprocessing.ipynb

01-JM-Image_split.ipynb

01-JM-ImgSlice.py

01-JM-Mask_to_Label.py

JM-01-PCA.ipynb

JM-01-Balancing_Classes.ipynb

```
Exploring Models
01-JM-Resnet50_Unet.ipynb
01-JM-Unet.ipynb
02-JM-RandomForest.ipynb

Model Evaluation
01-JM-Model_Prediction.ipynb
01-JM-Model_Testing.ipynb
JM_All_Models_Weights.ipynb

Final Results
01-JM-Big_img.ipynb
01-JM-Big_img_Final.ipynb
3-JM-Merge_Models.ipynb
```

Other tasks:

- Read papers for model options
- Look for different options of datasets
- Write functions for image slicing and mask processing on 'src'
- Write function for mask visualization on 'src'
- Modified image generator function to include sample weights
- Set up tensorflow gpu with CUDA

iv. Joe Brian Malubay

Notebook Proof:

```
Exploring Data Modeling
01-JB-Unet.ipynb
01-JB-Unet2.ipynb
01-JB-Unet3.ipynb
```

```
01-JB-Unet4.ipynb
01-JB-Unet4s.ipynb
01-JB-Unet5s.ipynb
01-JB-Unet6s.ipynb
02-JB-pix2pix.ipynb
02-JB-pix2pix1s.ipynb
02-JB-pix2pix2s.ipynb
02-JB-pix2pix4s-RF.ipynb
03-JB-randomforest1.ipynb

Finalize
02-JB-pix2pix3s.ipynb
02-JB-pix2pix4s.ipynb
03-JB-randomforest1-Copy1.ipynb
```

Other tasks:

- Academic paper research
- Shared code for activation layer visualization
- Initial Pix2Pix model exploration

v. Github folder structure

```
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── meeting_notes      <- Log of each meeting
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│   └── scratch        <- General scratch notebook stored here.
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models to make
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented visualizations
│       └── visualize.py
```

Figure 27: *Github Folder Structure.*

References

[Chaurasia and Culurciello, 2017] Chaurasia, A. and Culurciello, E. (2017). Linknet: Exploiting encoder representations for efficient semantic segmentation. In 2017 IEEE Visual Communications and Image Processing (VCIP), pages 1–4. IEEE.

[Chen et al., 2017] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs.

[Demir et al., 2018] Demir, I., Koperski, K., Lindenbaum, D., Pang, G., Huang, J., Basu, S., Hughes, F., Tuia, D., and Raskar, R. (2018). Deepglobe 2018: A challenge to parse the earth through satellite images. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops.

[Golovanov et al., 2018] Golovanov, S., Kurbanov, R., Artamonov, A., Davydow, A., and Nikolenko, S. (2018). Building detection from satellite imagery using a composite loss function. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 229–232.

[Guo et al., 2010] Guo, Z., Zhang, L., and Zhang, D. (2010). A completed modeling of local binary pattern operator for texture classification. IEEE transactions on image processing, 19(6):1657–1663.

[Haralick et al., 1973] Haralick, R. M., Shanmugam, K., and Dinstein, I. H. (1973). Textural features for image classification. IEEE Transactions on systems, man, and cybernetics, (6):610–621.

[Hengshuang Zhao et al., 2017] Hengshuang Zhao, J. S., Xiaojuan Qi, X. W., and Jia, J. (2017). Pyramid scene parsing network.

[Isola et al., 2017] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1125–1134.

[Li et al., 2018] Li, W., He, C., Fang, J., and Fu, H. (2018). Semantic segmentation based building extraction method using multi-source gis map datasets and satellite imagery. In Proceedings

of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 238–241.

[Lin et al., 2017] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2117–2125.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation.

[Marmanis et al., 2015] Marmanis, D., Datcu, M., Esch, T., and Stilla, U. (2015). Deep learning earth observation classification using imagenet pretrained networks. IEEE Geoscience and Remote Sensing Letters, 13(1):105–109.

[Michalis Giannopoulos et al., 2020] Michalis Giannopoulos, A. A., Anastasia Pentari, K. F., and Tsakalides, P. (2020). Classification of compressed remote sensing multispectral images via convolutional neural networks.

[Olaf Ronneberger and Brox, 2015] Olaf Ronneberger, P. F. and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. Medical Image Computing and Computer-Assisted Intervention.

[Rahman and Wang, 2016] Rahman, M. A. and Wang, Y. (2016). Optimizing intersection-over-union in deep neural networks for image segmentation.

[Van Etten et al., 2018] Van Etten, A., Lindenbaum, D., and Bacastow, T. M. (2018). Spacenet: A remote sensing dataset and challenge series. arXiv preprint arXiv:1807.01232.

[Vijay Badrinarayanan and Cipolla, 2016] Vijay Badrinarayanan, A. K. and Cipolla, R. (2016). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE.

[Xu et al., 2019] Xu, J. Z., Lu, W., Li, Z., Khaitan, P., and Zaytseva, V. (2019). Building damage detection in satellite imagery using convolutional neural networks. arXiv preprint arXiv:1910.06444.

[Yao X et al., 2019] Yao X, Y. H., Wu Y, W. P., Wang B, Z. X., and S, W. (2019). Land use classification of the deep convolutional neural network method reducing the loss of spatial features. Sensors (Basel, Switzerland).