CS 7643 Final Project Summer 2021: Building Detection and Location Intelligence in Aerial Satellite Imagery

Ahmed Bilal abilal7@gatech.edu
Christian Wiles cwiles7@gatech.edu
Sandeep Singh ssingh600@gatech.edu

Abstract

Building structures detection and information about these buildings in aerial images is an important solution for city planning and management, land use analysis. It can be the centerpiece to answer important questions such as planning evacuation routes in case of an earthquake, flood management, etc. These applications rely on being able to accurately retrieve up-to-date information. Being able to accurately detect buildings in a bounding box centered on a specific latitude-longitude value can help greatly. The key challenge is to be able to detect buildings which can be commercial, industrial, hut settlements, or skyscrapers. Once we are able to detect such buildings, our goal will be to cluster and categorize similar types of buildings together.

1 Introduction

We plan on reproducing semantic segmentation CNN models based on U-Net (Ronneberger et al., 2015) and Res-U-net (Diakogiannis et al., 2020) algorithms, trained by transfer learning using the ImageNet dataset. In addition, we will optimize these different models with focal loss, dice loss, cross entropy loss, hierarchical loss and differently weighted intersection-over-union (IoU) loss to overcome issues of scale difference [3] in building detection. To clearly delineate each individual member's contribution, we will be organizing our paper by each member's contribution.

2 Methodology

We have decided to work on three different models in parallel, which are inherently different on following parameters:

1. Architecture of models: We have decided to use different variations of UNet architectures, which might help us focus of different patterns easily.
2. Pre-training usage of the models: We have employed training from scratch as well using pre-trained weight for the encoder layers across out 3 of the models.
3. Loss functions being employed by models: We have used different loss functions in all the different models.
4. Image sampling and augmentations employed: We have used totally different techniques to samples the images from the data sets available to introduce the component of randomness in data distribution across models. While model 1 and 2 have used random resized crops of size 224x224 from big 5000x5000 images and masks; model 3 has resorted to fixed size split of big 5000x5000 images in 512x512 tiles. Both these sampling techniques have again used totally different set of transforms with different quantifiers.

While we have details of all the models provided in the sections 3,4 and 5, here is a quick overview of the ensemble technique exploited by us:

1. Get prediction from all 3 models. All predictions must of same size as input image. We output thresholded mask as tensor of softmax probabilities.
2. Chose the most confident pixel from each of these masks from 3 models for every pixel location's softmax probability. This is will be single merged mask of same size as input.
3. From merged mask predicted, drop anything that is not at least 0.75 confident. Rational behind doing this is that we have already chose most confident pixel locations from 3 models masks, So, all the softmax value at this stage must be pretty confident ones at least in one of the models.

As seen in the fig 1. We have final mask, which is without any less confident pixel.

3 Standard U-Net Architecture

As all three models are derived from U-net, we will begin with a discussion of the canonical U-net architecture. U-net is a modular model largely consisted of encoder blocks and decoder blocks.

3.1 Encoder blocks

The encoder layer is primarily responsible for detecting the 'what' elements of the images. The goal is to be able to extract features in the image at different scales and different levels of abstraction. As such, at every steps of the encoder, two 2D convolutional blocks are used to extract information from the image and double the size of the feature space. At each encoder layer, we used a maxpool layers of 2x2 kernel size and a stride of 2 for down sampling spatially. This allowed us to increase the number of filters at each of our encoder layers without being extremely computationally expensive and increase the receptive field of our filters with deeper layers allowing for segment detection at multiple scales.

3.2 Decoder blocks

The decoder layers are the up sampling layers in the model. The primary purpose of these layers is to localize the features extracted in the encoder block. This information is essential in our semantic segmentation in order to be able to output an image with the buildings detected localized in the right spaces in our output mask. For up sampling, we used Transposed Convolution layers. This allows us to ultimately assign class labels to each pixel in our image as part of our semantic segmentation.
At each of our decoder layers, we also make use of skipped connection given to us by the respective encoder layer for our decoder layer. The skipped connections cross from same sized part in the encoders to the decoders. The skipped connections allow us to overcome problem of vanishing gradient, increasing dimensionality and help regain the initial spatial information that we lost during the encoding path.

3.3 Integration

A full u-net model is composed of N encoder blocks, and N-1 decoder blocks. The feature space of the first encoder block is a hyperparameter but seems to be often set to 64 or 128. Save for the last encoder block, the output of the final convolutional layer in each encoder block is cropped and concatenated with the output of the transposed convolutional layer of the decoder block. At the end of the model, a 1x1 convolutional layer is used to create a classifier head with the same features space as the number of classes. This can be passed to a softmax layer to produce class probabilities.

4 Model 1 approach (Christian)

4.1 Data Pipeline and Exploration

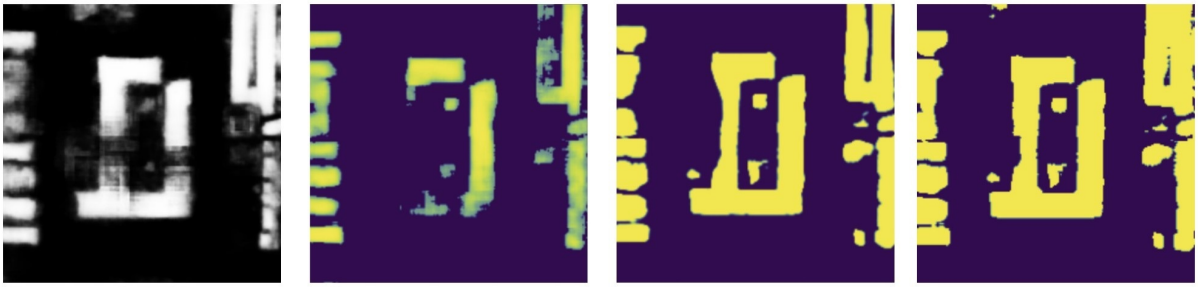
For this project, we used the Inria Aerial Image Labelling Dataset for training (Maggiori et al., 2017). The dataset consists of 360 (180 train and 180 test) 5000x5000 pixel full-color images with corresponding masks indicating the presence of building or non-building pixels. Images were taken from a variety of settings, including rural and urban cities from different continents. A few problems had to be solved to enable training with this dataset: data augmentation and quick random access.

4.1.1 Data augmentation

To generate more data for training, data augmentation was undertaken. Pytorch's standard transformation library does not make allowances for maintaining consistent transformations between an image and a segmentation mask, so the functional transforms library was used, which allows for the randomness to be provided by external variables, which can be held static between the ground truth and full-color images. For each of the 180 input training images, 350 224x224x3 image patches were created. This was intended to allow transfer learning for networks trained on ImageNet. A patch was taken from the image with random width (between 100 and 500 px), height (within +/- 10% of the random width), and image origin. This image was then randomly flipped horizontally and/or vertically and normalized by ImageNet standard deviation and mean.

4.1.2 Caching

It was found that performing the loading of the full-color images and performing transformations on the fly was too computationally intensive. To alleviate this problem, after the first time the transformations were done, the resulting tensors were saved to disk. This reduced the time to train significantly, as a 70 MB image did not have to be loaded and manipulated thousands of times per epoch, but instead a 600 KB tensor could be used.



(a) model 1 output *(b) model 2 output* *(c) model 3 output* *(d) Ensemble model's output*

Figure 1: Ensemble method being employed. Output softmax probabilities are compared and chosen to select most confident pixels of 2 models and thresholded to be at least 75% confident to be considered for final mask.

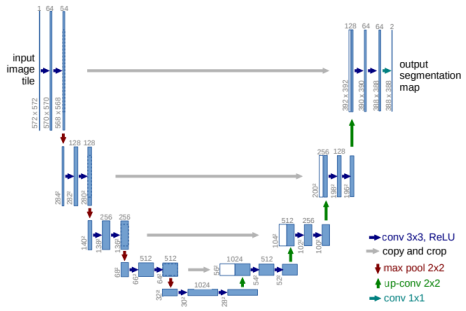


Figure 2: Example of overall organization of u-net model from (Ronneberger et al., 2015)

4.1.3 Other Considerations

It was also deemed important to add support for visualization of training-related metrics. Tensorboard support was added to the project to track training and validation set loss and accuracy, precision-recall curves for the validation set, and visualization of the forward pass of the model on the validation set. One last consideration made was the use of a seed when splitting the training and validation set. It was noticed that when resuming training from a checkpoint that the validation set was not the same as before the checkpoint. By maintaining the constant seed, a barrier was maintained between the two sets.

4.1.4 Dataset Statistics

As is often the case with segmentation tasks, the dataset was not balanced between building pixels and non-building pixels. The training dataset was analyzed to determine the prevelance of each class. The findings are below.

4.2 U-net from scratch in Pytorch

To test the hypothesis that building detection was a sufficiently specific domain to merit training from scratch, a u-net was created with random Xavier-initialized weights. Both the original U-net paper (Ronneberger et al., 2015) and Johannes Schmidt's blog posts (Schmidt, 2021) were consulted in the creation of the model.

Two major deviations were attempted from the models mentioned above. Both u-nets resulted in a cropped image with every convolution due to the use of valid padding. By using same padding, on convolutions and transposed convolutions, we can return an image that is of equal size to the input. This may result in slightly worse accuracy in the extremities of the image due to the extrapolation employed by same padding, but does simplify some aspects of the analysis, as every mask pixel has a corresponding prediction.

Secondly as the output of a 2-class softmax classifier only has 1 degree of freedom, it was attempted to perform classification as a single-class regression, with the output of the regression put through a sigmoid function. This one-channel output can then be interpreted as p(building). This approach was eventually discarded, as it had a very small impact on model size due to only affecting the final 1x1 convolutional layer, and adding a second classifier dimension increased model performance by a few percentage points.

Finally, batch normalization was added between the convolutional and activation layers in encoder and decoder blocks. These recenter the distribution of the output of the convolutional layers and add to stability in training as seen in (Santurkar et al., 2019).

Class	Number of Pixels	Percentage	Effective Number (β = 1 - 10E-9)
Building	7.1E8	1.58%	5.08E8
Not building	4.4E10	98.4%	1.00E9
Total	4.5E10	100%	1.00E9

Table 1: Class distribution of pixels in training set images.

To discourage overfitting, dropout was added between the output of the final decoder layer and the 1x1 convolution.

4.3 Loss function

3 different loss funcitons were attempted with this model. With the regression-based approach, weighted mean square error was used due to the class imbalance. On a per-batch basis, the effective number of building and non-building pixels was calculated on a per-batch basis, similar to the methodology in (Cui et al., 2019). This weighting was weigh the loss on the minority class (buildings) more heavily.

Once the model had progressed to a two-class method, two losses were pursued: weighted cross-entropy loss and dice loss. Dice loss was pursued due to its background in segmentation tasks and invariance with respect to class imbalance (as dice loss is related to the size of the true positive region). Binary cross-entropy weighted by the inverse of effective number was also explored. This provides a more convex loss function that should be easier and more stable to train.

4.4 Results

Loss Fn	Accuracy	IOU Score	F1-Score
Dice	%		
BCE	95.0%	0.726	0.841

Table 2: Model 1 Train Set results after 20 Epochs

Hold for train/val accuracy graph

As the loss function are inherently different in magnitude, loss function charts will not be compared here.

5 Model 2 approach (Ahmed)

5.1 Model Specifications

5.1.1 Double Convolution Blocks

The Unet build consisted of a double convolution layer, where each convolution layer consisted of a kernel size of 3, stride and padding of 1. We set the bias to false in order to add a BatchNorm layer, which is then followed by a ReLU activation layer. We settled on a small 3X3 kernel receptive field in our convolution layers in order to be able to detect very small edges and shapes in our aerial images. Doing so is especially relevant for our aerial images as there is a lot of noise in the images and our model needs to be able to use small edges and shapes to detect buildings as buildings appear in many different sizes in our input images.

5.1.2 Encoder Layers

The encoder layer is primarily responsible for detecting the 'What' elements of the images. The goal is to be able to detect multiple segments in the image at different scales. As such, at every steps of the encoder, we run our Double Convolution Blocks to start from the 3 RGB image channels and output 16,32,64,128 and 256 output channels at each our encoder layer respectively. At each encoder layer, we used a maxpool layers of 2x2 kernel size and a stride of 2 for down sampling. This allowed us to increase the number of filters at each of our encoder layers without being extremely computationally expensive and increase the receptive field of our filters with deeper layers allowing for segment detection at multiple scales.

The authors in *U-Net: Convolutional Networks for Biomedical Image Segmentation* (Ronneberger et al., 2015) recommend encoding layers with output channels 64,128,256, 512 and 1024. However, we found more success with output channel layers 16,32,64,128 and 256. We believe this is because lower output channels of 16 and 32 in the start allow us to detect really small building segments with a small receptive field. In addition, the 512 and 1024 channel layers were not leading to any significant performance gains in our testing.

5.1.3 Decoder Layers

The decoder layers are the up sampling layers in the model. The primary purpose of these layers is to return the 'Where' information back to out model that we lost as part of the maxpool down

sampling layers. The 'Where' information is essential in our semantic segmentation in order to be able to output an image with the buildings detected localized in the right spaces in our output mask. For up sampling, we used Transposed Convolution layers. This allows us to ultimately assign class labels to each pixel in our image as part of our semantic segmentation.
At each of our decoder layers, we also make use of skipped connection given to us by the respective encoder layer for our decoder layer. The skipped connections cross from same sized part in the encoders to the decoders. The skipped connections allow us to overcome problem of vanishing gradient, increasing dimensionality and help regain the initial spatial information that we lost during the encoding path.

5.2 Pre-trained ResNet34 Encoder Specifications

We now add ResNet 34 Encoder layers to the model. As such, we are now performing the Double Convolution blocks 3,4,6, and 3 times at each encoder layer level, using skipped connections between encoder layers, and using a higher stride to down sample instead of max pooling. These encoder layers are also pre-trained on image-net dataset.

5.2.1 Encoder Modifications

After finding success in our U-Net built with 16,32 and 64 output channel initial encoder layers, we replace the initial ResNet convolution, ReLU and Max Pool layers with our U-Net 16,32 and 64 output channel encoder layers with skipped connections in order to preserve a lot of the small shapes and edges information in our images.

5.2.2 Attention Mechanism

For the loss function, we will be using the Dice Loss to create cleaner mask segments to represent the buildings. In order to supplement our model in reducing the Dice Loss, we also include an attention mechanism using spatial and channel 'squeeze & excitation' Blocks. This is done to aid our encoder layers in spatial encoding for more accurate mask prediction and better network flow. The authors in *Recalibrating Fully Convolutional Networks with Spatial and Channel 'Squeeze & Excitation' Blocks* (Roy et al., 2018) found a reduction of 4-9% in the Dice Loss. We see similar results in our testing.

5.2.3 Results

In our testing, we saw the pre-trained image-net backbone significantly increase the model performance. After 15 epochs, we saw the following results.

Accuracy	Dice Loss	IOU Score	F1-Score
0.965	0.116	0.749	0.856

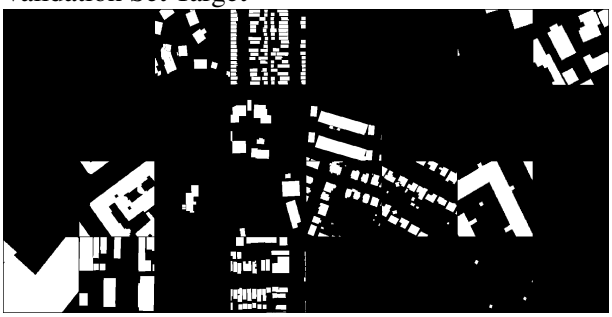
Table 3: Train Set results after 15 Epochs

Accuracy	Dice Loss	IOU Score	F1-Score
0.961	0.129	0.702	0.824

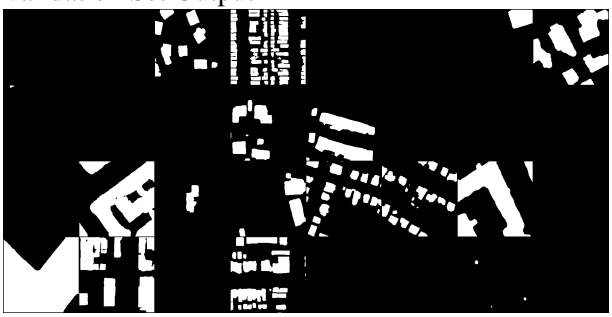
Table 4: Validation Set results after 15 Epochs

5.2.4 Model 2 Validation Set Mask Samples

Validation Set Target



Validation Set Output



5.2.5 Accuracy Results Per Epoch

6 Model 3 approach (Sandeep)

For third model was developed to address the aspects, which were almost orthogonal to the main characteristics captured by Model 1 and Model 2. This model was also variation on ResUnet model, which has tried to exploit typical Unet capabilities. Additionally, following were distinctive features(compared to model 1 and model 2) of model 3 as below:

Figure 3: 'Model 2 Train Accuracy'

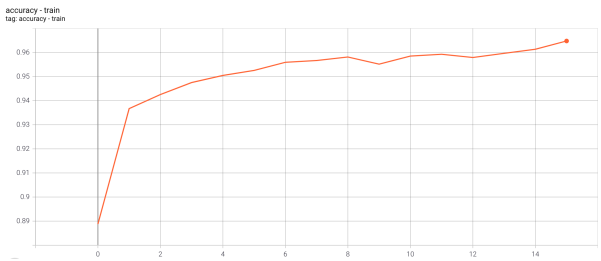
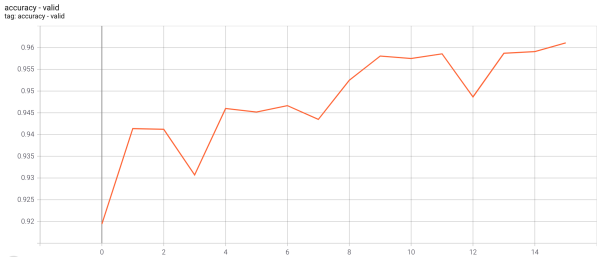


Figure 4: 'Model 2 Valid Accuracy'



6.1 Data Sampling Strategy

For third model, We have tried to employ "Progressive Resizing" for the training images. This is the training technique, where we purposefully change the contents of image by resizing the images to contain more area, while keeping input image size same. We have not used the randomized random crops of size 224x224 as in model 1 and 2. Instead, we have split the big images of 5000x5000 into smaller 512x512 tiles, which are non-overlapping and contiguous tiles. These tiles at the time of training get resized to 224x224 input images. So effectively, each tile has almost 4 times more buildings in each tile compared to model 1 and 2. That means this model learns to capture much more smaller buildings in more crowded localities. (Howard, 2018)

6.2 Architectural Considerations

For model 3, We have evaluated 3 types of encoders and choose the one, which has shown enough capacity and was fast enough per epoch. We have evaluated ResNet18, ResNet34 and ResNet50. ResNet18 has shown to adapt to bigger buildings very easily, but shown to be struggling to encode features of smaller buildings successfully. Between of ResNet34 and ResNet50, both have shown very similar performance in detected buildings without any marginal increase in performance mutually, While ResNet50 was taken significantly more training time than ResNet18 and ResNet34 encoders. That is why, We have decided to used ResNet34 as our eventual encoder layer in UNet architecture in model 3. One more significant improvement in model 3 was during Decoder(up sampling) head, where we have employed Pixel Shuffle ICNR up-sampling as provided by fast.ai implementation. This technique employed sequence of shuffleblock during up-sampling/decoder head. (Andrew Aitken*, 2018)

6.3 Training and Validation Split

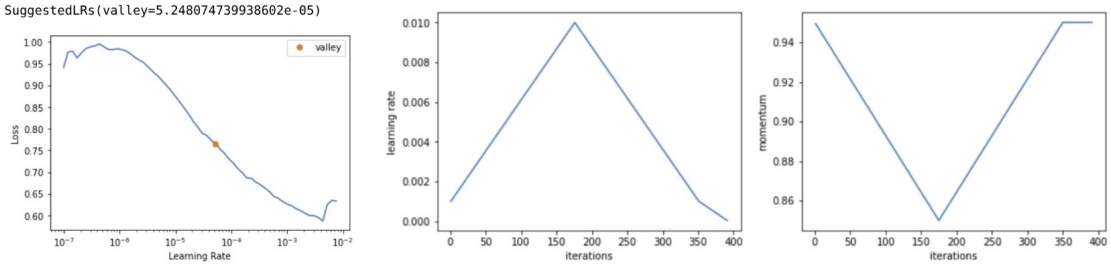
For model 3, we have done the data split on the basis of geography, instead of random ratio split of the whole training corpus. We had training data from 5 cities as: Austin, Chicago, Kitsap, Tyrol, Vienna. Instead of tiling contiguous parts of the one big images into the training and test tiles, different cities are included in each of the subsets. e.g., images over Chicago are included in the training set, but not on the test set. Also, images over San Francisco are included on the test set but not on the training set. This was done in order to assess the generalization capabilities of the features learned by the model. At the same time, we have tried to include the training data all type of structures of building. e.g. low rise vs high rise vs community living buildings apartment complexes.

6.4 Custom Loss Function Design

For this version of model, we have not used CrossEntropy loss variant. Instead, For model 3, we have written our own custom loss function, which has helped us predict foreground pixel with higher softmax confidence and making it very hard for the model to predict higher confidence for background pixel locations. We have opted for Combined Loss of Dice Loss and Focal Loss, both with equal weights. Focal loss part is quantitative feedback, which is penalizing more confident wrong predictions more heavily than less confident predictions. We have used gamma value of focal loss as 2. Also, Dice score has provided feed back to strive to keep precision and recall both highest possible. both Dice loss and Focal has used equal weights in order to not over-power each other given we just has single channel target. Also, For model 3, we have used the Dice Score metric as well, instead of using accuracy as metric for classification of pixels.

6.5 Training Convergence

For model 3, we have employed to distinctive techniques than model 1 and 2. First one is application of LR Finder scheduler before with start fine tuning



(a) LR Finder

(b) Fit One Cycle: Momentum is moved in opposite direction than LR during increase or annealing phase

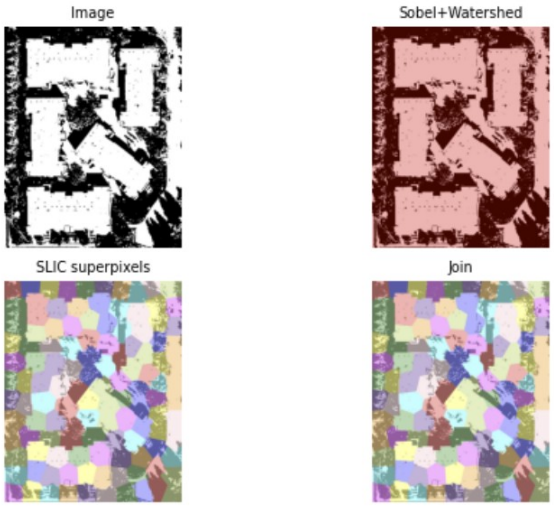
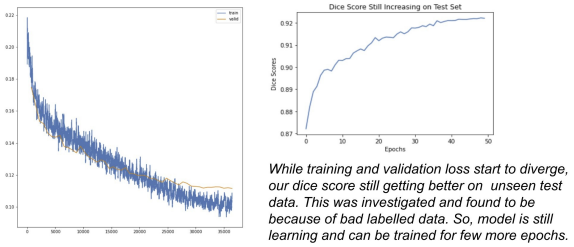


Figure 5: Fig. Post processing and merging of traditional segmentation techniques with Unet's mask and shape correcting polygons for buildings.



While training and validation loss start to diverge, our dice score still getting better on unseen test data. This was investigated and found to be because of bad labelled data. So, model is still learning and can be trained for few more epochs.

Figure 6: Fig. Model 3 Training Curves.

pretrained weights of ResNet34 encoder. This has helped of find the most appropriate maximum LR value. Second innovative technique employed by us was "Fit-one-cycle"(Smith, 2018) to achieve super convergence. In this technique, we increase LR to maximum value in initial batches before start to anneal the learning rates to eventually reduce the step size. Please refer to figure for LR finder and fit one cycle both.(Sylvain Gugger, 2018) This technique is taken from Leslie Smith iconic Super Convergence paper's implementation in fast.ai framework.

Also, We trained with initial 20 epochs without over-fitting and saved the model only when better score on validation was seen. then we have trained again for 20 more epochs while saving the better model as per score seen on validation set. We had still not seen over-fitting, while Dice score was 92+ percent almost. But marginal decrease in loss and marginal increase in Dice Score both were very small per new epoch. So, we have decided to test our model for unseen images from google static maps api , which has shown very good results and we had decided to stop training more at that point in time. Please refer to the loss table.

Train_Loss	Valid_Loss	Dice_Score
0.102414	0.115400	0.920870

Table 5: Model 3: Losses and Metrics Values

6.6 Post Processing Enhancements

Model 3 has employed some of the traditional computer vision techniques of segmentation and tried to fuse their results with Unet predicted mask. We have Have tried to experiments with Otsu's threshold, Watershed segmentation, SLIC Super pixel algorithm for segmentation. eventually we have decided to merge segments from SLIC Super pixel algorithm(Roerdink and Meijster, 2001) with our model's mask prediction. We have selected all the coincidental mask segments from super pixel algorithms' output with Unet's mask and tried to shape correct polygon for buildings. This techniques has helped us achieved sharp edges for the complex building shapes. mainly it has helped us fix the shapes of rounded or curved building edges of the buildings visible in satellite images. Please refer to the post processing segmentation merging figure 5.

7 Experiences and Challenges

7.1 Challenges

While, we had insinuations about the nature of the satellite images, we had discovered many new challenging aspects, which are specific to to the satellite images. Few of these aspects are:

- Satellite images are very noise and affected by weather factors like, heat, clouds, vegetation on the ground. (Wikipedia, 2018)
- Characteristics of particular satellite instrumentation vary very widely and different type of bands are available to be exploited for particular detection task. e.g. it is very easy to detect water bodies and plants with infrared bands, while this is not most useful for building detection.
- Zoom level has profound impact on any type of algorithmic work done on satellite images. e.g. 512x512 image tile with zoom level 20 has approx 4/5 bindings in it, while zoom level 18 has few hundred buildings in one tile. While all CNN models re space and scale invariant, but at variation of zoom level, features become too scarce to be characteristic of candidate task at hand.
- Cost of acquiring images on demand can be costly and in some cases acquiring latest satellite images are not possible or the quality of images are not up acceptable enough to do training or inference.
- Types of buildings in satellite images are very different. This difference becomes more challenging across countries and cultures. e.g. buildings in US and Europe are very different. Similarly difference between US/Europe and Africa are drastic. So, training data has to make considerations about all types of building structures.

7.2 Adaptive approach to challenges

Right from inception of the project, we had to change our approach or redefine the scope of the problem statement. Few accommodation made by us were are:

- We have evaluated between Google Static maps api vs Bing maps api and found that google static maps api are more clearer and useful for our purpose. So, we selected google static maps api for out on demand image needs for building detection.
- We have decided to use 3 channel RGB images with zoom level of 20 as our reference point. So, all the training data acquired was zoom level 20. This has helped us benchmark out baseline model performance although our model is adaptive to zoom level variance and agnostic to building size to a great extent.
- Because of greater variance in shape and style across buildings of different geographies, We have used training data for 6 metropolitan areas across US and Europe. Also, we have made sure out size cities, one city's data is never seen by training or validation set and that is used purely for testing and benchmarking purpose.

7.3 Project Success Criterion

We were able to achieve most of the planned milestones for the project with some scope modifications. few of the things we have planned and achieved were are as follow:

- We were able to build 3 good enough model with Dice score more than 90 percent on test set and model 2 achieving detection of more 96 percent ground truth pixels in valid set.
- With help of Softmax based adaptive selection and ensemble, we have achieved detection of more 93 percent ground truth pixels in test set.
- We were able to implement building shape correction algorithms during post-processing stem with help of SLIC super pixel algorithm.
- Our ensemble model had shown surprising capacity to detect building across US and Tanzania, while we had no Tanzania data in our training set.

7.4 Future Project Aspirations

We had initially planned to implement and productize the building detection models for following purposes as:

- Can we cluster the similar buildings on the basis of roof color profile? We have implemented this partially only.

- Can we classify detected buildings ad residential vs commercial properties?
- Can we predict the new constructions possibility on the basis of currently detected buildings?
- Can we building system to detect illegal construction or activity detection for preservation forest areas?

All of these are very much achievable products of our already built building detection ensemble algorithms and we can pursue them as subsequent derived projects.

References

Lucas Theis* Jose Caballero Zehan Wang Wenzhe Shi*Twitter Inc. Andrew Aitken*, Christian Ledig*. 2018. Checkerboard artifact free sub-pixel convolution.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. 2019. Class-balanced loss based on effective number of samples.

Foivos I. Diakogiannis, François Waldner, Peter Caccetta, and Chen Wu. 2020. Resunet-a: A deep learning framework for semantic segmentation of remotely sensed data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 162:94–114.

Jeremy Howard. 2018. Progressive resizing for better generalization of the computer vision models.

Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. 2017. Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE.

Jos B.T.M. Roerdink and Arnold Meijster. 2001. The watershed transform: Definitions, algorithms and parallelization strategies.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation.

Abhijit Guha Roy, Nassir Navab, and Christian Wachinger. 2018. Recalibrating fully convolutional networks with spatial and channel 'squeeze & excitation' blocks. *CoRR*, abs/1808.08127.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. 2019. How does batch normalization help optimization?

Johannes Schmidt. 2021. Creating and training a u-net model with pytorch for 2d & 3d semantic segmentation: Dataset...

Leslie N. Smith. 2018. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate,batch size, momentum, and weight decay.

Jeremy Howard Sylvain Gugger. 2018. Adamw and super-convergence is now the fastest way to train neural nets.

Wikipedia. 2018. Satellite imagery and challenges associated with them.

A Code Repository

We have work on total of 4 repositories during our project life cycle. 3 repositories where used by each of us individually and 1 final repository was created as a place to perform ensemble and integration of all models together and do all the needed post-processing. Here the repositories as below:

Integrated Final repo:
https://github.com/sandeepsign/building_footprint_ensemble
Admed's repo:
https://github.com/abilal19/DL_FinalProject_Draft
Christian's repo:
https://github.com/cswksu/aerialDetection
Sandeep's repo:
https://github.com/sandeepsign/building_footprints_cs7643

B Individual Contributions

Contributor	Contribution
Ahmed	Full Model 2 (best performing), Project Report
Christian	Full Model 1, data loader, visualization, Project Report
Sandeep	Full Model 3, ensembling, post processing, Geo Coding, VM Setup, Clustering of polygons, Project Report

Table 6: Individual team member contributions.

C Data Source

We have used INRIA's spacenet challenge data from:
https://project.inria.fr/aerialimagelabeling/

D Train Infrastructure

We have used combination of techniques to execute this project.

Each had used individual hardware for setup and eventually to run long time training on google cloud VM instance.

GPUs Used are:
nVIDIA QUADRO RTX 5000 16GB
nVIDIA T4 16GB
nVIDIA 1080ti