Final Project COMP433

December 18, 2022

Michelle Lin
michelle.lin2@mail.mcgill.ca
McGill University
Montréal, Québec, Canada

Silas Kalinowski
silas.kalinowski01@gmail.com
Concordia University
Montréal, Québec, Canada

Lin Ling
jane.ling912@gmail.com
Concordia University
Montréal, Québec, Canada

ACM Reference Format:
Michelle Lin, Silas Kalinowski, and Lin Ling. 2022. Final Project COMP433: December 18, 2022. In *Proceedings of (NA).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 NOTE TO READER

Please note, there is an issue with the Overleaf template code where the citations were not rendering properly, but we have included listed the references and included a link to the overleaf project in the last section

This report is not formatted according to a traditional research paper. Instead, to adhere to the assignment instructions, it is split up into 3 primary parts, A) the Discussion section, where we provide a literature review providing a high level overview of our methods with justification and implementation considerations – similar to that of an extended abstract and table of contents where the reader can quickly look up the desired section to read into B) We describe the research context and motivation (this Part contains the most similarities as a research paper) and C) we describe our investigation and detail our exploration in this project.

2 PART A

3 DISCUSSION

Adhering to the assignment instructions – but adapting the requirements to our research project format, this section will summarize and justify the methods used.

Semantic Segmentation Semantic segmentation is a fundamental and well-established technique in computer vision. This classification method has been used in a variety of data-abundant remote sensing problems, including tasks with multi-band hyperspectral satellite imagery, such as tree and vegetation classification [**tree**], crop cover and type analysis [**yang2016adaptive**] and environmental monitoring [**blaschke2000object**]. In addition, segmentation techniques have demonstrated remarkable success in geolocalization tasks such as improving localization and mapping on slums and small-scale urban structures [**WURM201959**].

U-Net A well-known and often cited machine learning algorithm, the U-Net, is commonly used to perform semantic segmentation tasks. Originally developed for medical image segmentation, the U-Net [**DBLP:journals/corr/RonnebergerFB15**] is known as a fully convolutional neural network (FCN), that has been used in a variety of problems, Examples include, road extraction [**Zhang_2018**] and greenhouse detection [**government**]. This algorithm has seen a lot of success due to its ability to perform image and semantic segmentation with minimal training data. While the U-Net is not considered the state-of-the-art algorithm for semantic segmentation such, an investigation into the future work would greatly benefit from it was selected for the reasons described above of being applicable and useful in a variety of domains. For more information, refer to 6

The model implementation inspiration was from the following Kaggle repository: [**kaggle_unet**].

Data Augmentation Deep convolutional neural networks are heavily reliant on big data to avoid overfitting [**Overfitting**], a survey focused on Data Augmentation [**Data_Augmentation**], a data-space solution to the problem of limited data. The image augmentation algorithms we could chooses to test the limited dataset include geometric transformation, color space augmentations, kenrnel filters, mixing images, random erasing etc [**Data_Augmentation**]. There are several image data augmentation libraries, which we choose is Albumentations [**info11020125**] which is a computer vison tool designed to perform fast and flexible image augmentations, and it have the largest set of transformation functions which will help us to see the difference results between different techniques we applied. For more information and results,refer to 8.1.

Deep Lab V3 Another model which used for segmentation task is the DeepLabv3 model. Proposed by Liang-Chieh, Papandreou, George and Adam in 2017 [**DeepLabV3**], it uses a new approach mainly relying on atrous convolution. The model achieves comparable performance with other state-of the art models on both the Cityscape and PASCAL VOC 2012 Test set [**reviewdeeplabv3**]. Also the model is designed to be efficient at inference time and therefore perfectly suitable for our project. Additonally, the DeepLabv3 has shown to be versatile, enabling the use in a wide range of tasks (e.g. semantic segmantating, instance segmentation, ...). Lastly, the model was easily accessible through the Segmentation Modely python library with different initial weights, encoders and other possible modifications [**smp_pytorch**]. For the reasons mentioned above and mainly to compare the performance of two different approaches to semantic segmentation we chose this model. For more information, refer to 6.1

Comparison of Pre-trained and trained models Transfer Learning described the phenomena that after solving a Task A,

NA, 2022, Montréal, QC

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

solving a similar Task B is easier because of the learning made on Task A [**lecture9**]. One of the most popular variants of Transfer Learning is using a pretrained model and fine tuning it [**lecture9**]. We used models that were pretrained on the image net dataset. One of the reasons for that is that is has been shown that those models are able to recognize edges and patterns and even objects [**VisualizingandUnderstandingConvolutionalNetworks**]. This ability is very useful for our tasks. We finetuned the entire model, thereby using the pretrained model as a good initialization. For more information see 7

Batch Normalization Batch Normalization is a popular technique enabling faster and more stable training of Deep Neural Networks [**Batch**]. Batch Normalization reduces the sensitivity to initialization, is robust to hyper parameters and may lead to higher testing accuracy [**lecture6**]. It can even make Dropout or other regularization measures unnecessary [**BatchNorm2**]. The reasons behind the effects are not fully understood. For reasons mentioned above and because the pretrained pytorch model already used batch normalization be default [**smp_pytorch**] we wanted to test which effects it had in our experiments. For the results refer to 8.2.

Dropout Dropout is a regularization technique where some outputs at each layer are zeroed out with a probability p. This can be seen as introducing noise to the training process. Therefore a increase in robustness is expected. Additionally it is avoided that a single weight has a high influence on the output [**lecture6**]. In general Dropout aims to prevent over fitting and can be seen as a way of combining exponentially many different neural networks efficiently [**Overfitting**]. Dropout is mostly replaced by Batch Normalization in current models. [**BatchNorm2**]. To test what effects dropout has on our model and for the reasons mentioned above we tested Dropout layers with different probability. For the results refer to section 8.3. Implementing Dropout was more difficult than expected, since the [**smp_pytorch**] models do not have a attribute for Dropout by default. We had to add the drop out layers manually to the model, which required more time than previously expected.

Future work Methods With respect to the research context of this project, a future application of this project would be the ability of the algorithm to generalize to various geographical regions with little to no additional data. Meta-learning methodologies such as model-agnostic meta-learning (MAML) [**finn2017model**].

3.1 Other Implementation Considerations To Highlight

For more details on each highlighted consideration, please refer to their respective sections. A major implementation consideration is the need for compute power to run these models, especially for larger datasets. We initially planned to use kaggle to run all of our experiments because it provides enough free computing resources. Unfortunately, the import for the [**smp_pytorch**] library did not work on kaggle. Therefore we had to transfer all of our experiments to google Colab. This included downloading and uploading the datasets from kaggle to google Drive. The computing resources in google Colab were limited which posed a major challenge for us.

4 PART B: RESEARCH CONTEXT

4.1 Motivation

Our objective in this project is to a) explore the Deep learning Pipeline from data processing of the raw data to model ingestable data while b) exploring the usage of semantic segmentation methods and deep learning optimization techniques on geospatial data.

4.2 Application Context: Machine Learning for Remote Sensing Applications

The use of machine learning techniques applied to remote sensing has come to prominence over the past few years. The success of these applications is apparent with algorithms being able to predict, classify, or monitor objects from images, ranging from different types of crops [**timl**] to tracking large populations of bird migrations. As such, private companies such as Esri have created their own proprietary implementations of machine learning algorithms, such as illegal oil and gas well monitoring [**timl**] and locating nuclear facilities [**tang_matthew**].

To preface, *remote sensing* is a widely used technique done for object detection, image segmentation, analysis, and other purposes through the use of satellite imagery, aerial imagery, or drone imagery [**Zhu_2017**]. In this project, satellite imagery as the primary data source. The reasoning behind this choice is that the continuing and recent advent of higher resolution, and more accessible, satellite imagery has made it easier to localize refugee settlement camps and differentiate other characteristics [**Quinn_2018**]. The high resolution imagery is on the magnitude of a few dozen centimeters per pixels - with 30cm per pixel being the state-of-the-art resolution, with more publicly available, but lower resolution images on the magnitude of 50-100 metres per pixel.

4.3 Datasets

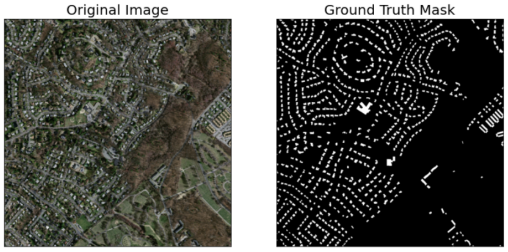


Figure 1: Figures of mask and satellite image of building dataset

In this section, we describe our selected datasets here. Our baseline datasets comprise of the *Buildings* and *Footpath*, publicly available datasets with well established metrics of. Outside of our baseline dataset we include our own dataset, *The Darien Trees* dataset.

As the first dataset we used a Building Dataset that consist of aerial images of buildings in Massachusetts [**building_dataset**]. We chose this dataset for three main reasons: it consist of aerial images (similar to the satellite images of our dataset), it contains only 151 samples (similar size as our dataset) and it includes binary masks as targets (same as our dataset).

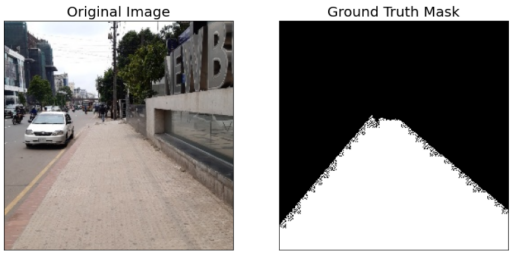


Figure 2: Figures of mask and image of footpath dataset

The second dataset we used was a footpath dataset that conists of 3000 images of Bangladeshi footpath [**footpath_dataset**]. We chose this dataset for these three reasons: It includes the same binary masks as our dataset, it contains a higher amount of images (as a comparison to a the smaller dataset) and the task is quite different to the aerial segmentation (comparison if our methods improve performance of different tasks).

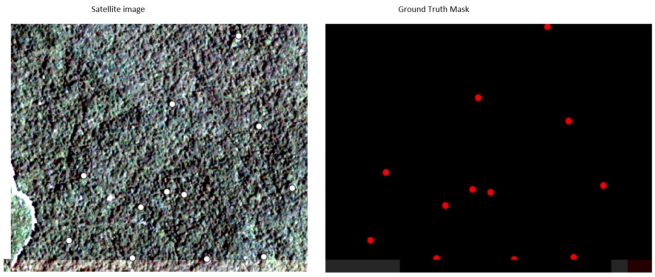


Figure 3: An example of the *Darien Trees* mask.

The data used in the construction of the *Darien Trees* was provided courtesy of Potvin lab [**Potvin**]. The labels consist of geographic coordinates of a rare specie of recognizable large tree crowns, of approximately 10m radius, in the Darien region of Panama. Satellite imagery data from Planet Labs is made up of high-resolution, multi-band, geospatial images on the scale of 3 metres per pixel, to detect features of a small spatial size. See 3 for a sample image and its mask from the dataset More details are provided in Section *Data Processing*.

Considerations Initially, we selected a dataset comprised of Planet Labs satellite imagery of planes [**planes_dataset**]. We thought this dataset would be a good choice for transfer learning and pretraining for the *Darien Trees* dataset as the image resolution is the same as the *Darien Trees*. Unfortunately, the dataset did not have binary segmentation masks but instead only had binary labels for each image. Another dataset we initially planned to test our model on was the forest dataset [**forest_dataset**]. The dataset is very similar to the *Darien Trees* with the only difference being that the all typed of trees are belong to one class. The model just needs to differentiate between background and tree. For our dataset only a certain type of tree should be recognized. We could not run any experiments on the forest dataset because we faced computing issues.

5 PART C: EXPLORATION

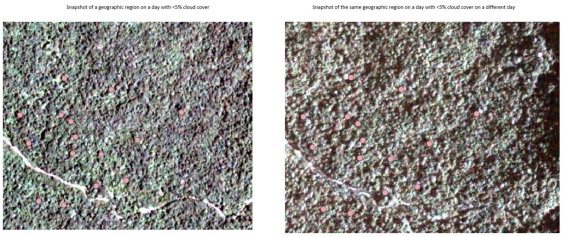


Figure 4: Comparison of the use of the same geographic region taken at different times in the *Darien Trees* dataset

5.1 Dataset and Data Processing of the *Darien Trees*

Other than the described data augmentations 8.1, no other data manipulation was required for the Footpath [**footpath_dataset**] and Buildings [**building_dataset**] datasets as they are already model-ready.

On the *Darien Trees* dataset needed to be cleaned, organized, pre-processed and constructed based off geographic coordinates and satellite imagery. In the construction of the *Darien Trees* dataset, geospatial analysis and visualization tools (QGIS), image processing (PIL) and Python libraries (Rasterio, GDAL) were used to create the segmentation masks [**gdal**] . These tools and libraries were logistically difficult to install and use with limited documentation. The necessity of *domain expertise of these geospatial tools* is integral for future progress.

At first, the masks were attempted to be generated from scratch using the above mentioned libraries by converting the label information in the form of a .shp file, a vector data storage format for storing the location, shape, and attributes of geographic feature [**shp**]. After numerous issues associated with the extent and specifications of the image mappings, a built-in Rasterio function was ultimately used to generate a binary segmentation GeoTIFF mask, a standard .tif or image file format that includes additional spatial(geo-referencing) information embedded in the .tif file [**shp**].

Next, the satellite imagery and the masks in the GeoTIFF format were then "tiled" in a sliding window manner from their 1668 x 1668 size into 512 x 512 images, where the GeoTIFF format containing 4 bands of RGB and NIR (near-infrared) would then be converted to a PNG format with 3 bands as the input to the model [**gdal**]. A similar trial and error method was used as above, with similar complications in particular with respect to the *gdal2tiles* library which had Windows incompatibilities.

Compute power became a major bottleneck, as training could not be completed and metrics could not be yielded. See **??**

5.1.1 Lack of Labelled data and Data Imbalance. To account for the limited amount of labelled data, the dataset samples multiple images of the same region taken at different time steps. See **??** for a visualization. It would be interesting to employ an unsupervising learning method to account for this issue in the future.

Similarily, the dataset contains more negative examples than postive examples, as tree crowns are only a small part of each satellite image. In addition to the sampling mentioned above, it would be interesting to employ sampling preprocessing techniques such as Synthetic Minority Over-sampling TEchnique (SMOTE) [**SMOTE**]

5.1.2 Robustness of The Labels. The geographic locations of the trees are approximate with the annotations being generated from an approximate shape of a polygon (circle in this case) with a radius of 20m. As such, some of the labels will be inaccurate to varying degrees. However, we anticipate that the neural network will able to ignore certain amounts of label noise according to [**DBLP:journals/corr/RolnickVBS17**])

6 FIRST ARCHITECTURE: U-NET

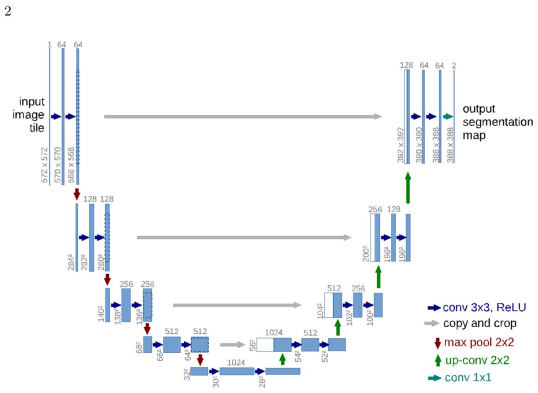


Figure 5: Unet architecture visualized [Unet]

As our first architecture we chose the U-Net model.[**Unet**] We found it interesting that the U-Net model can be trained with very few training images and still produces satisfying results. This is especially relevant since our Data set only consist of approximately 150 images. The effectiveness with a low amount of samples is achieved by the architecture. One path of the architecture is a usual contracting network the other path is an expansive path used for up sampling. Secondly this is achieved by different data augmentation methods. The authors used excessive data augmentation [**Unet**], that yielded better generalization and robustness to noise. Also the architecture improved on drawbacks of previous architecture like deep CNNs, that for example were trained with a sliding-window setup for segmentation. The major drawbacks of the older architectures were: slowness, redundancy and that there is a trade-off between localization accuracy and the use of context.

Regarding the Training of the model we found it interesting that the batch sized was reduced to a single image. The reason for that is the overhead of the input image and the goal to use as much of the GPU memory as possible. With that batch size a high momentum is used so that many of the previously seen images are used to make the current optimization step.

Also the initialization for the weights was interesting since we saw in the first Assignment how that can influence the training efficiency and convergence of the model. The initialization they used a similar approach as the Xavier initialization which we had previously used. The authors also made the initial value of on node depended on the number of incoming nodes.

Lastly, in the Data augmentation part of the paper the authors also mentioned the use of Drop-out layers at the end of the contracting path. This made me realize that Drop-out layers at the end of the contracting are also just a form of implicit augmentation. Before I did not make that connection, I saw Data Augmentation and Drop-Out layers as two separate concepts.

We used the an untrained Unet architecture and it produced the following results:

6.0.1 Train performance. The dice loss stays nearly the same over 10 epochs. The IoU Score improves at the beginning (epoch 1 and 2) and then remains nearly the same for the rest of the epochs. To sum up the model converges fast and does not improve the performance after the first few epochs. This is explained by the fact that the data set is very small and the model adapts fast to it.

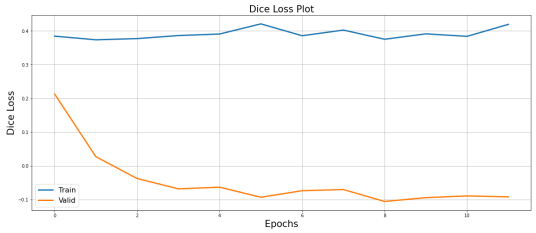


Figure 6: Train Dice Loss

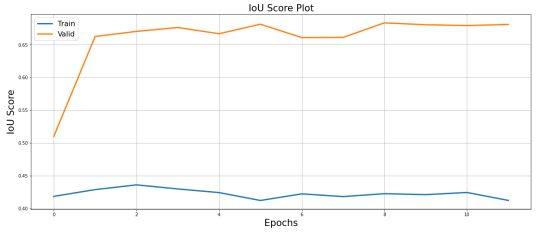


Figure 7: Train IoU Score

6.0.2 Test Performance. The Unet model had a mediocre test performance on the Building Dataset. The mean IoU score was 0.4648 and the mean dice loss was -0.3423 on the Buildings data set. Considering that the building data set is very small the performance is surprisingly good.

Figure 8: visualization of one of the outputs

6.1 Second Architecture: Deep Lab V3

As our second choice of architecture we used the Deep Lab V3 model. The model was proposed by Liang-Chieh, Papandreou, George and Adam in 2017 [**DeepLab**]. Therefore it's a more recent approach than the UNet architecture [**Unet**]. One of the problems DCNNs face is the reduced feature resolution caused by consecutive pooling operations or striding. To tackle this issue the authors proposed the use of atrous convolution [**DeepLab**]. In [**lecture7**] we also talked about dilation in Filters and the advantages of dilation. Especially advantageous for Segmentation Tasks is that the receptive field grows exponentially without losing resolution. This is a perfect solution of the problem mentioned above. The second challenge the authors faced were that objects can exist in multiple scales. The model should be able to confidently recognize the objects at different scales. This challenge is tackles by first the application of the model to an image pyramid, second the encoder-decoder structure, third the cascading of extra modules on top of the original network, fourth spatial pyramid pooling.

For training the authors experimented with different crop sizes, upsampling logits during training and fine-tuning batch normalization. Smaller crop sizes, downsampling the logits and not finetuning the batch normalization decreased the performance of the model. Also larger batch sizes proved advantageous. Lastly, selecting a appropriate value for output strides improved the performance.

6.1.1 Train Performance. When looking at the training metrics the model makes huge improvements in the first two epochs and then the improvement slows down considerably. After epoch five the model's performance converges. The Dice loss is over all training epochs very stable.

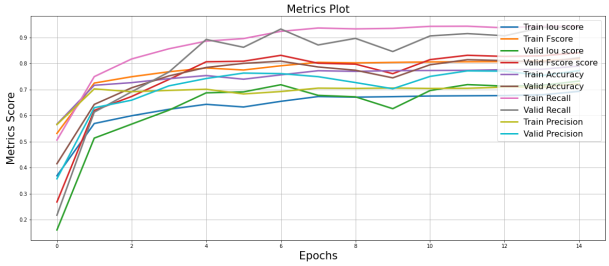


Figure 9: Train IoU score Building Dataset

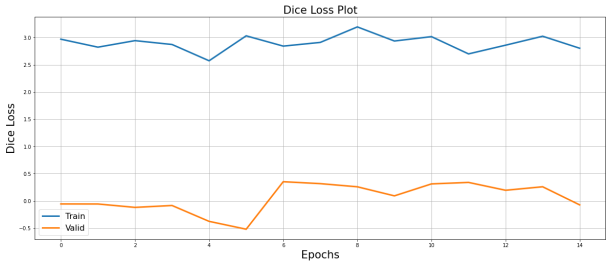


Figure 10: Train Dice loss Building Dataset

6.1.2 Test Performance. We tested a pretrained DeepLabv3 model with the following results: Overall the model performs very well on the Building Dataset. The model reaches a IoU score of 0.7324 on the test data. The other measures like accuracy, recall or precision are even higher. The model outperforms the Unet pretrained model by 0.1181 (mean IoU score) and by 0.1643 (mean Dice loss).

Metric	Mean on Test Data
IoU Score	0.7324
Dice Loss	0.2117
Fscore	0.8410
Accuracy	0.8217
Recall	0.9376
Precision	0.7723

Figure 11: Test measures Building

7 TRANSFER LEARNING

7.1 not pretrained vs pretrained

As mentioned in 3 we focused on finetuning the entire model. We imported the model using the [**smp_pytorch**] library. Specifically, we used both of our models (Unet, DeepLabV3) with a pretrained Resnet encoder. The Resnets were all pretrained on the image net dataset.

As expected the performance of the pretrained models was much better than the performance of the not pretrained models (IoU score 0.4648 to 0.6140) 21. Also the predictions had much more detail. What was also interesting is that the pretrained model metrics in training in the early metrics are higher compared to the untrained model. This can be explained by Transfer Learning. See 3 for further explanation.

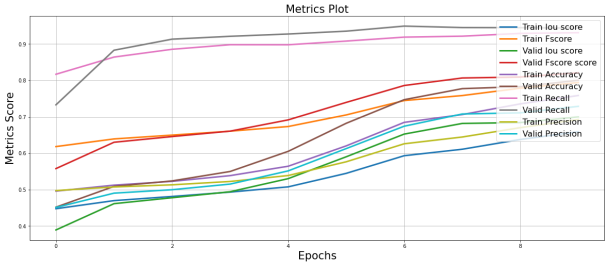


Figure 12: Train metrics

Metric	Mean on Test Data
IoU Score	0.6140
Dice Loss	0.3750
Fscore	0.7589
Accuracy	0.7250
Recall	0.9017
Precision	0.6559

Figure 13: Test performance

7.2 Deeper Models/ More parameters

As an encoder we used two different Resnet versions. The main idea behind Residual Networks are shortcut connections [**Resnet**]. They were originally introduced to fight vanishing and exploding gradients. But it turned out that skip connections have even more advantages. ResNets are hypothesized to have inductive biases that help generalization. Because of these advantages and others. ResNets are a popular choice for Segmentations tasks, specifically they are used as Encoders. For these reasons we chose to use Residual Networks in our experiments. We originally planned only to use the Resnet34. But in the paper about the DeepLabV3 model the authors discovered that using deeper Resnets improved the performance of their model [**DeepLabV3**]. Therefore we also tested Residual Networks with different depth [**DeepLabV3**].

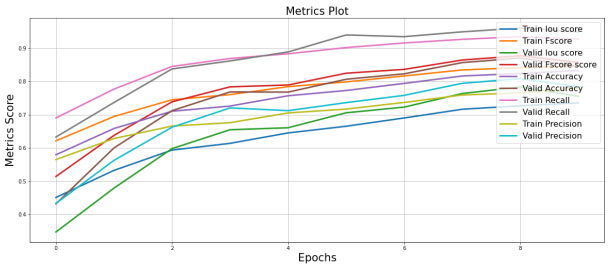


Figure 14: Test metrics Unet

We discovered that using the the Resnet50 instead the Resnet34 as the UNet encoder improved our testing performance by approximately 0.08 on the Building data set. We did not test Resnet101 or even deeper Resnet versions since our computing resources were limited and we expected the same decreasing improvement in performance for deeper versions as the authors from the Deep Lab V3 paper saw [**Deeplav3**].

Metric	Mean on Test Data
IoU Score	0.6898
Dice Loss	0.3045
Fscore	0.8144
Accuracy	0.7961
Recall	0.9280
Precision	0.7268

Figure 15: Test metrics Unet

8 REGULARIZATION

8.1 Data Augmentation

Data augmentation [**Data_Augmentation**] is a technique used in machine learning to increase the amount and diversity of data available for training models. This is important because in many cases, the performance of a machine learning model is directly related to the amount and quality of the data it is trained on. By using data augmentation, we can effectively increase the size of our data set and make it more diverse, which can lead to better performance from our model. Additionally, data augmentation can help to reduce overfitting [**Overfitting**], which occurs when a model is trained on a small data set and ends up memorizing the training data rather than learning to generalize to new data.

To specific speaking about Image data augmentation, this can be accomplished by applying a variety of random transformations to the training images, such as cropping, scaling, rotating, or flipping the images. These transformations can help the model learn to recognize objects in a wider range of positions, scales, and orientations, which can improve its performance on new data. Image data augmentation is often used in combination with other techniques, such as training on large data sets and using regularization methods, to further improve the performance of image recognition and segmentation models.

In order to improve the performance of the model at the very beginning, we decided to test a U-Net Model, which is well explained in Section 2.1, with different augmentation techniques. We trained the model on a Building data set which is a small data set with 302 training images, 4 validation images and 10 test images. One test augmentation method includes HorizontalFlip/VerticalFlip and Random Rotate, the other test augmentation method besides these three techniques were added more techniques, CLAHE,Random Gamma,Random Brightness,IAASharpen,Blur and its variants,Random Contrast, HueSaturationValue.

8.1.1 Without data augmentation techniques. Without using any data augmentation techniques, the validation performance of the building dataset is much better than the train's. From figure below 20, is shows that the IoU score of validation which is changed between 0.56 and 0.61, is around 10 percentage higher than the IoU score of train which is changed between 0.47 and 0.50. In the meantime, it shows that the dice loss of train which is changed between 0.17 and 0.21, is around 18 percentage higher than the dice loss of validation which is changed between 0.0 and 0.08.

After using basic data augmentation techniques, the gap between the performance of train and validation is narrowed.It could clearly present in .20

8.1.2 More data augmentation techniques. After adding more techniques of data augmentation, the results shows us the gap between the performance of train and validation is gotten bigger comparing to the basic techniques.

8.1.3 Evaluation on Test data. From the table of Evaluation on Test data, it shows us the performance of test when use the more data augmentation is better than the others two.

	Without data augmentation	basic data augmentation	More data augmentation
Mean IoU Score	0.4892	0.4648	0.5631
Mean Dice Loss	0.1504	-0.3423	0.0752

Figure 16: Evaluation on Test data

8.2 Batch normalization

The pretrained Unet model we imported from [**smp_pytorch**] already includes batch normalization in the layers of the decoder. To find out what effects batch normalization has we turned it off and trained and tested again. Batch norm has different behaviors

at train and test time. At train time the running mean and variance are tracked. In test time these values are used. We made sure the model was set into train and eval mode at the appropriate times.

8.2.1 Performance. The model with batchnorm outperformed the model without batchnorm slighty (0.6898, 0.6712 Mean IoU score). Also on all other measures the model with batchnorm scored higher 23. The only slight increase in performance could be explained by the small size of the dataset and the rather small batch size which increases the uncertainty of the running mean and variance. The example predictions below show an even clearer picture:

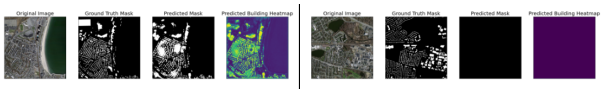


Figure 17: left: model with batchnorm, right: model without batchnorm

the model with batchnorm produces a very accurate result while the model without batchnorm only returns a black prediction mask.

8.3 Dropout

. As our second Regularization method we decided to use Dropout. Specifically we added dropout layers to encoder of the pretrained Unet models. We did not add any dropout layers to the decoder since the decoder used batch normalization. We wanted to test if combining both dropout and batch normalization yields even better results than just using batch normalization.

8.3.1 Performance. Different values for p had a major influence on the performance of the model. The worst performance was obtained with a dropout rate of 0.5. The model performed far worse than the model without any dropout (Test IoU 0.6900 and 0.4515). The best model dropout rate according to the test metrics was 0.01, with a Test IoU from 0.7690 compared to 0.6900 of the base model. Also the Test loss was greatly improved from 0.3081 to 0.2507. The second best model was the model with a dropout rate of 0.8. Even though it had a much higher IoU score (0.7649 to 0.6900) and Test loss (0.2794 to 0.3081), the predicted masks were nearly all just black images. We concluded that the IoU score was sometimes misleading because we set the treshhold to 0.5, making it possible to obtain a high IoU score without producing good masks. When considering this smaller dropout rates between 0.01 to 0.375 produced the best results (see 22). The results match our expectation that a rather small dropout rate should be used. A higher dropout rate combined with batch normalization decreases the performance.

Dropout rate	0.8	0.5	0.375	0.25	0.125	0.06125	0.01	0
Test IoU	0.7649	0.4515	0.7378	0.7356	0.7585	0.7540	0.7690	0.6900
Test loss	0.2794	0.4519	0.2704	0.2781	0.3803	0.3247	0.2507	0.3081

Figure 18: Test metrics sorted by Dropout Rate

Dropout rate	0.01	0.8	0.125	0.06125	0.375	0.25	0	0.5
Test IoU	0.7690	0.7649	0.7585	0.7540	0.7378	0.7356	0.6900	0.4515
Test loss	0.2507	0.2794	0.3803	0.3247	0.2704	0.2781	0.3081	0.4519

Figure 19: Test metrics sorted by performance

9 LOSS FUNCTION AND EVALUATION METRICS

There are several reasons why the Dice loss function[**kaggle_loss_function**] is commonly used for image segmentation tasks. One reason is that it is differentiable, which means that it can be used to train a model using gradient descent. This is important because it allows the model to learn from the data and improve its performance over time. Another reason is that the Dice coefficient is a well-established metric for evaluating the performance of image segmentation models, and the Dice loss function is simply the negative of this metric. This means that minimizing the Dice loss is equivalent to maximizing the Dice coefficient, which is a desirable property for a loss function. Finally, the Dice loss is sensitive to the overlap between the predicted and ground truth masks, which is an important factor in many image segmentation tasks. Overall, the Dice loss function is a widely used and effective loss function for image segmentation tasks.

The IoU (intersection-over-union) score [**IoU_evaluation_metric**]is a common metric used to evaluate the performance of image segmentation models. It measures the overlap between the predicted segmentation mask and the ground truth mask, and is defined as the ratio of the number of pixels in the intersection of the two masks to the number of pixels in the union of the two masks. The IoU score is a useful metric because it ranges from 0 (no overlap) to 1 (perfect overlap), which allows for easy interpretation of the model's performance.

To explore more about the evaluation metrics[**Segmentation_metrics**], we added more metrics such as Fscore, accurary, recall, and precious. For more details of the results, please refer to 21

10 DIFFERENT DATASETS

We tested our methods as mentioned in 3 also on different datasets with the following results:

We could find that the IoU score on building datasets 0.4648 and the footpath is 0.5374. For the Unet not pretrained, the performance of footpath dataset is slightly higher(around 0.07) than the building dataset.

When looking at the performance it was noteworthy that using pretrained models on the footpath lead to a much higher increase in performance than using pretrained models on the Building dataset (footpath improvement of 0.3771, building improvement of 0.2252). One of the reasons for that could be that the image net data is more similar to the footpath dataset than to the Building Dataset. The Building Dataset consists of aerial images of buildings with different textures and shapes [**building_dataset**]. The image net dataset does not contain many aerial images. On the other hand the Footpath dataset consist of images from footpaths from the perspective of a pedestrian [**footpath_dataset**]. Similar images are also part of the image net dataset. Also sometimes signs or

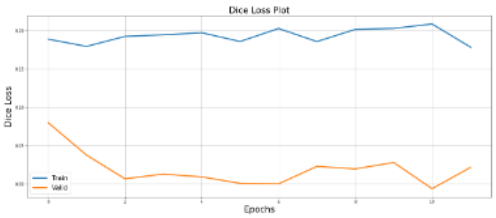
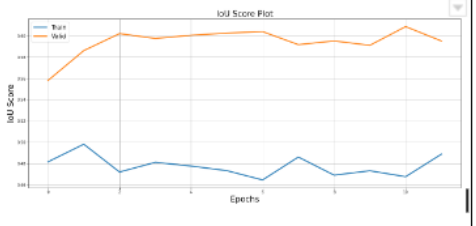
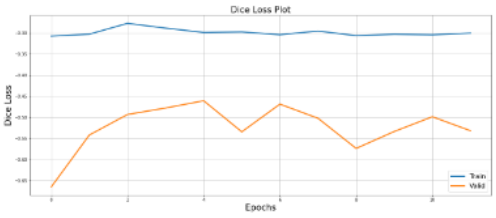
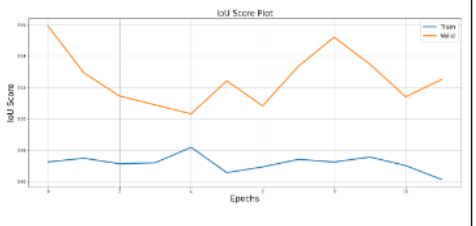
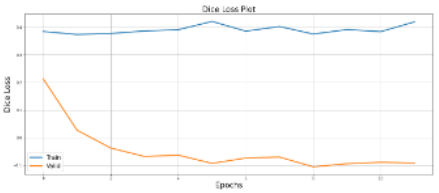
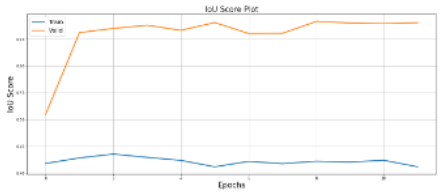
	Dice loss	IoU Score
Without use data augmentation		
With base techniques of data augmentation		
With more techniques of data augmentation		

Figure 20: Comparison table of data augmentation

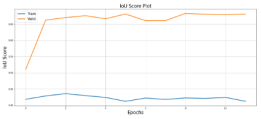
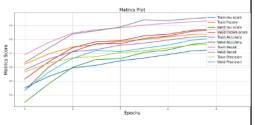
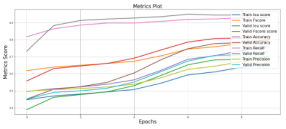
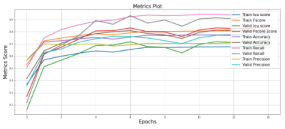
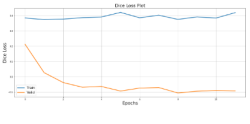
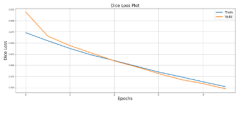
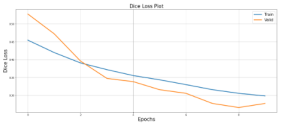
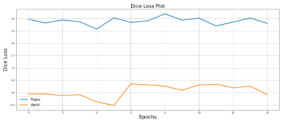
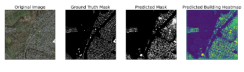
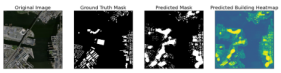
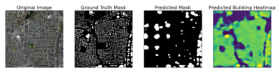
	UNet not pretrained	UNet pretrained (Resnet50)	Unet pretrained (Resnet 34)	Deep LabV3 (Resnet50)
Test IoU Building	0.4648	0.6900	0.6140	0.7321
Test loss Building	-0.3423	0.3081	0.3750	0.2117
Train Metrics				
Dice Loss				
Prediction				

Figure 21: Comparison Pretrained vs not pretrained

other objects that are part of the image net dataset are part of the footpath samples. The pretrained model is able to recognize these objects and therefore is able to recognize the footpath more easily.

In conclusion the methods worked even better on the footpath dataset.

11 FUTURE METHODS TO CONSIDER

For the modeling architectures part, we could consider to use LinkNet [**Chaurasia_2017**] since it's fast and memory efficent. For the loss funtion, we could try some different loss functions like weighted boundary loss whose aim is to reduce the distance between the predicted segmentation and the ground truth.For the training part, we could consider to do more hyperparameter tuning, try to track of our experiments using Neptune [**Neptune**]. Another method which could be considered for future experiments would be training a model on the building [**building_dataset**] or any similar datasets and then training the model on our dataset. The results can then be compared to an untrained model or a model pretrained on the image net dataset. Is there a positive or a negative transfer learning effect? Also different variants of fine tuning could be considered. For example only fine tuning the top layer[**Transfer**].

12 OVERALL EXPERIMENT EVALUATION OF INVESTIGATION

When evaluating the overall performance – specifically IOU score, it is clear that the difference between using pre-traine models was the most impactful method – improving the score by 20%Às expected, when all the methods (using Data augmentation, using a pretrained ResNet50 as an encoder, applying small drop out rate of 0.01 and using batch norm) the model performed the best on both datasets with the IOU score on the *Buildings* data set nearly doubling from 0.46 and the FootPath dataset also making a visible improvement from 0.77 to 0.91.

In this project, while investigating the usage of semantic segmentation methods and deep learning optimization techniques on geospatial data, we established baselines metrics for geospatial segmentation tasks, comprised of the Footpath and Buildings dataset, gained insight into pre-processing real-world raw data into machine learning ready format.

13 STATEMENT OF CONTRIBUTION

Joint: Part A.3 (Discussion), Part C6, 13
Michelle: Part B (Research Context), Part C.7, C14
Silas: Part C (Exploration) - Section 8, 9, 10.2, 10.3, 12*
Lin: Part C - Section 9, 10.1, 11, 12*

14 APPENDIX:

14.1 Link to code notebooks

Data augmentation notebooks
Data processing notebooks
Building Dataset
Footpath Dataset
Forest Dataset
Notebooks for Dropout, Batchnorm, Transfer Learning, Different Datasets

Overleaf

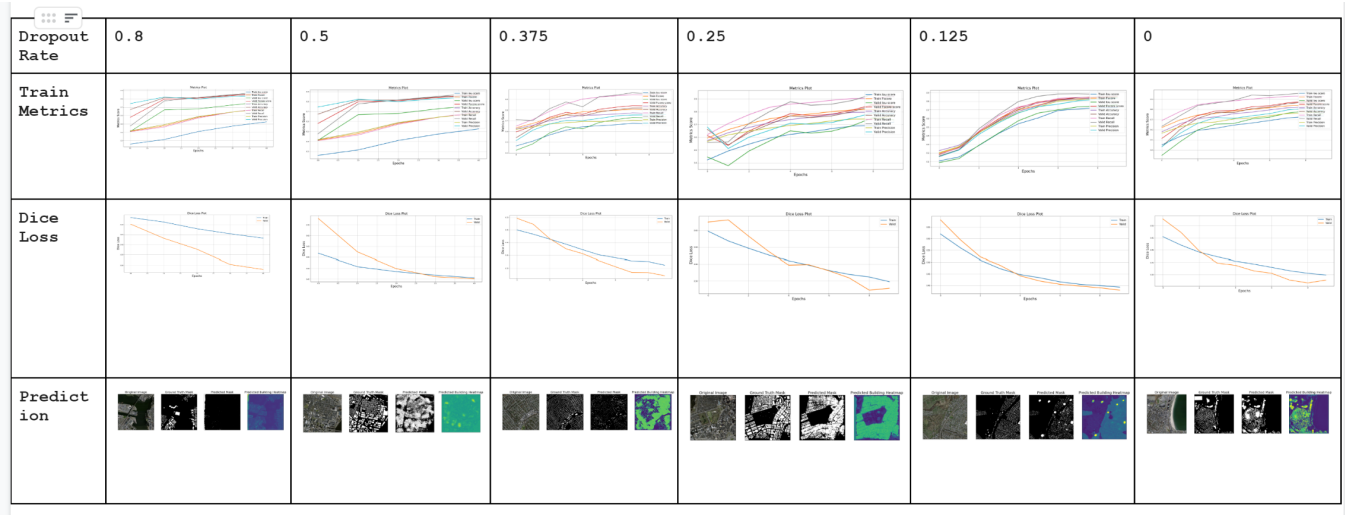


Figure 22: Train Metrics sorted by Dropout Rate

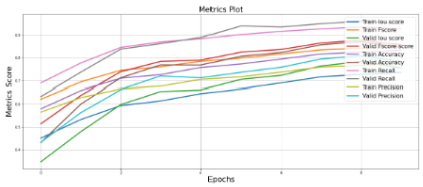
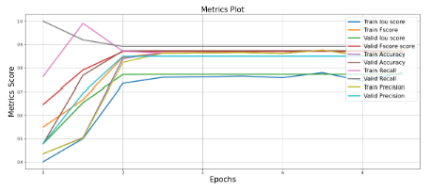
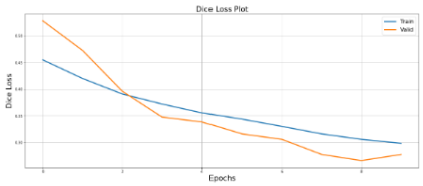
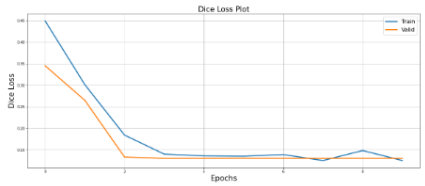
Model	Unet pretrained Resnet 50 decoder with batchnorm	Unet pretrained Resnet 50 decoder with no batchnorm
Test Metrics	Mean IoU Score 0.6898 Mean Dice Loss: 0.3045 Mean Fscore: 0.8144 Mean Accuracy: 0.7961 Mean Recall: 0.9280 Mean Precision: 0.726	Mean IoU Score 0.6712 Mean Dice Loss: 0.2946 Mean Fscore: 0.7900 Mean Accuracy: 0.7657 Mean Recall: 0.9210 Mean Precision: 0.6921
Train Metrics		
Train Dice Loss		

Figure 23: Comparison Batchnorm and no Batchnorm

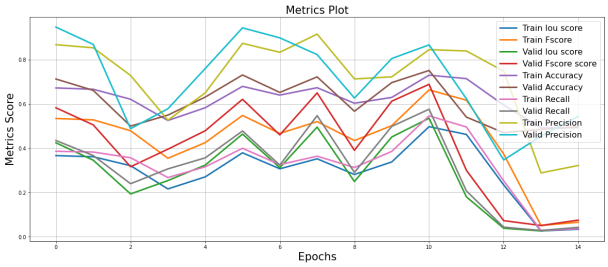


Figure 24: Train measures Footpath Dataset untrained Unet

	UNet not pretrained	UNet pretrained (Resnet50)	Unet pretrained (Resnet 34)
Test IoU footpath	0.5374	0.9145	
Test loss footpath	-0.6773	0.0727	
Test IoU Building	0.4648	0.6900	0.6140
Test loss Building	-0.3423	0.3081	0.3750

Figure 25: Comparison Test performance pretrained vs not pretrained

References

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. *Model-agnostic meta-learning for fast adaptation of deep networks*. International Conference on Machine Learning, pp. 1126–1135, 2017.

[2] Xiao Xiang Zhu, Devis Tuia, Lichao Mou, Gui-Song Xia, Liangpei Zhang, Feng Xu, and Friedrich Fraundorfer. *Deep learning in remote sensing: a review*. arXiv: Computer Vision and Pattern Recognition, 2017.

[3] John Quinn, Marguerite Nyhan, Marguerite M. Nyhan, Celia Navarro, Davide Coluccia, Lars Bromley, and Miguel Luengo-Oroz. *Humanitarian applications of machine learning with remote-sensing data: review and case study in refugee settlement mapping.*. Philosophical Transactions of the Royal Society A, 2018.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015.

[5] Eugene Belilovsky. *COMP 433 Lecture 7: Convolutional Neural Networks*. .

[6] Eugene Belilovsky. *COMP 433 Lecture 9: Multi-Task and Transfer Learning*. .

[7] Eugene Belilovsky. *COMP 433 Lecture 6: Training Deep Networks in Practice*. .

[8] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. *Rethinking Atrous Convolution for Semantic Image Segmentation*. 2017.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015.

[10] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. Journal of Machine Learning Research, **15**, pp. 1929–1958, 2014.

[11] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015.

[12] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. *Albumentations: Fast and Flexible Image Augmentations*. Information, **11**, 2020.

[13] Gabriel Tseng, Hannah Kerner, and David Rolnick. *TIML: Task-Informed Meta-Learning for Agriculture*. CoRR, **abs/2202.02124**, 2022.

[14] Even Rouault, Frank Warmerdam, Kurt Schwehr, Andrey Kiselev, Howard Butler, Mateusz Łoskot, Tamas Szekeres, Etienne Tourigny, Martin Landa, Idan Miara, Ben Elliston, Chaitanya Kumar, Lucian Plesea, Daniel Morissette, Ari Jolma, and Nyall Dawson. *GDAL*. 2022.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. *SMOTE: Synthetic Minority Over-sampling Technique*. Journal of Artificial Intelligence Research, **16**, pp. 321–357, 2002.

[16] Kimberly Stoner. *Approaches to the biological control of insect pests*. .

[17] Joseph Goeb, Jenny Smart, Jason Snyder, and David Tschirley. *Information, pesticide safety behaviors, and toxicity risk perceptions evidence from Zambia and Mozambique*. , 2022.

[18] Claire Lamine, Marc Barbier, Julien Blanc, Jan Buurma, Isabelle Scherer-Haynes, Jozsef Lehota, Elisa Maraccini, Egon Noe, Rejane Paratte, Zoltan Szabo, and Anna Wierzbicka. *Reducing the dependence on pesticides: a matter of transitions within the whole agri-food system*. 2010.

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional networks for biomedical image segmentation*. International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 234–241, 2015.

[20] Ahmed Nassar, Karim Amer, Reda ElHakim, and Mohamed ElHelw. *A Deep CNN-Based Framework for Enhanced Aerial Imagery Registration With Applications to UAV Geolocalization*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2018.

[21] Thomas Blaschke, Stefan Lang, Eric Lorup, Josef Strobl, and Peter Zeil. *Object-oriented image processing in an integrated GIS/remote sensing environment and perspectives for environmental applications*. Environmental information for planning, politics and the public, **2**, pp. 555–570, 2000.

[22] *Segmentation metrics*. .

[23] Shuai Yang, Qihao Chen, Xiaohui Yuan, and Xiuguo Liu. *Adaptive coherency matrix estimation for polarimetric SAR imagery based on local heterogeneity coefficients*. IEEE Transactions on Geoscience and Remote Sensing, **54**, pp. 6732–6745, 2016.

[24] Matthew D Zeiler and Rob Fergus. *Visualizing and Understanding Convolutional Networks*. 2013.

[25] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. *Do Better ImageNet Models Transfer Better?*. 2018.

[26] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015.

[27] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. *Road Extraction by Deep Residual U-Net*. IEEE Geoscience and Remote Sensing Letters, **15**, pp. 749–753, 2018.

[28] Government of Alberta. *Landowner and indigenous community site nomination*. 2021.

[29] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. *The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition workshops, pp. 11–19, 2017.

[30] EarthLab. *Introduction to multispectral remote sensing data in Python*. 2018.

[31] Irem Ülkü and Erdem Akagündüz. *A Survey on Deep Learning-based Architectures for Semantic Segmentation on 2D images*. arXiv preprint arXiv:1912.10230, 2019.

[32] Irem Ulku, Panagiotis Barmpoutis, Tania Stathaki, and Erdem Akagunduz. *Comparison of single channel indices for U-Net based segmentation of vegetation in satellite images*. Twelfth International Conference on Machine Vision (ICMV 2019), pp. 338–345, 2020.

[33] Michael Wurm, Thomas Stark, Xiao Xiang Zhu, Matthias Weigand, and Hannes Taubenböck. *Semantic segmentation of slums in satellite images using transfer learning on fully convolutional neural networks*. ISPRS Journal of Photogrammetry and Remote Sensing, **150**, pp. 59–69, 2019.

[34] Jakaria Rabbi, Nilanjan Ray, Matthias Schubert, Subir Chowdhury, and Dennis Chao. *Small-Object Detection in Remote Sensing Images with End-to-End Edge-Enhanced GAN and Object Detector Network*. 2020.

[35] Statistics Canada Government of Canada. *Greenhouse detection with remote sensing and machine learning: Phase one*. 2021.

[36] Ling Tang and Sangeet Matthew. *Using Machine Learning and Deep learning with Imagery in ArcGIS*. .

[37] Rachel K. E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, Seema Nagar, Karthikeyan Natesan Ramamurthy, John T. Richards, Diptikalyan Saha, Prasanna Sattigeri, Moninder Singh, Kush R. Varshney, and Yunfeng Zhang. *AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias*. CoRR, **abs/1810.01943**, 2018.

[38] David Rolnick, Priya L. Donti, Lynn H. Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, Alexandra Sasha Luccioni, Tegan Maharaj, Evan D. Sherwin, S. Karthik Mukkavilli, Konrad P. Kording, Carla P. Gomes, Andrew Y. Ng, Demis Hassabis, John C. Platt, Felix Creutzig, Jennifer Chayes, and Yoshua Bengio. *Tackling Climate Change with Machine Learning*. arXiv: Computers and

Society, 2019.

[39] Isabelle Tingzon, Niccolo Dejito, Ren Avell Flores, Rodolfo De Guzman, Liliana Carvajal, Katerine Zapata Erazo, Ivan Enrique Contreras Cala, Jeffrey Villaveces, Daniela Rubio, and Rayid Ghani. *Mapping New Informal Settlements using Machine Learning and Time Series Satellite Images: An Application in the Venezuelan Migration Crisis*. arXiv: Computers and Society, 2020.

[40] Mohammad Mehedy Hassan, Audrey Culver Smith, Katherine Walker, Munshi Khaledur Rahman, Munshi Khaledur Rahman, and Jane Southworth. *Rohingya Refugee Crisis and Forest Cover Change in Teknaf, Bangladesh*. Remote Sensing, 2018.

[41] John Quinn, Marguerite Nyhan, Marguerite M. Nyhan, Celia Navarro, Davide Coluccia, Lars Bromley, and Miguel Luengo-Oroz. *Humanitarian applications of machine learning with remote-sensing data: review and case study in refugee settlement mapping.*. Philosophical Transactions of the Royal Society A, 2018.

[42] Emilio Zagheni, Ingmar Weber, and Krishna Gummadi. *Leveraging Facebook's Advertising Platform to Monitor Stocks of Migrants*. Population and Development Review, 2017.

[43] Joshua Blumenstock. *Inferring patterns of internal migration from mobile phone call records: evidence from Rwanda* . Information Technology for Development, 2012.

[44] Katherine Hoffmann Pham, Jeremy Boy, Miguel Luengo-Oroz, and Miguel Luengo-Oroz. *Data Fusion to Describe and Quantify Search and Rescue Operations in the Mediterranean Sea*. 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), 2018.

[45] Stefan Voigt, Thomas Kemper, Thomas Kemper, Torsten Riedlinger, Ralph Kiefl, K. Scholte, and Harald Mehl. *Satellite Image Analysis for Disaster and Crisis-Management Support*. IEEE Transactions on Geoscience and Remote Sensing, 2007.

[46] Michael Hagenlocher, Stefan Lang, Stefan Lang, and Dirk Tiede. *Integrated assessment of the environmental impact of an IDP camp in Sudan based on very high resolution multi-temporal satellite imagery*. Remote Sensing of Environment, 2012.

[47] S. Giada, T De Groeve, Daniele Ehrlich, and Pierre Soille. *Information extraction from very high resolution satellite imagery over Lukole refugee camp, Tanzania*. International Journal of Remote Sensing, 2003.

[48] Xiao Xiang Zhu, Devis Tuia, Lichao Mou, Gui-Song Xia, Liangpei Zhang, Feng Xu, and Friedrich Fraundorfer. *Deep learning in remote sensing: a review*. arXiv: Computer Vision and Pattern Recognition, 2017.

[49] Reik Leiterer, Urs Bloesch, Hendrik Wulf, Sebastian Eugster, and Philip C. Joerg. *Vegetation monitoring in refugee-hosting areas in South Sudan*. Applied Geography, 2018.

[50] David Rolnick, Priya L. Donti, Lynn H. Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, Alexandra Luccioni, Tegan Maharaj, Evan D. Sherwin, S. Karthik Mukkavilli, Konrad P. Kording, Carla Gomes, Andrew Y. Ng, Demis Hassabis, John C. Platt, Felix Creutzig, Jennifer Chayes, and Yoshua Bengio. *Tackling Climate Change with Machine Learning*. 2019.

[51] Isabelle Tingzon, Niccolo Dejito, Ren Avell Flores, Rodolfo De Guzman, Liliana Carvajal, Katerine Zapata Erazo, Ivan Enrique Contreras Cala, Jeffrey Villaveces, Daniela Rubio, Rayid Ghani, and et al.. *Mapping new informal settlements using machine learning and Time Series Satellite Images: An application in the Venezuelan migration crisis*. 2020 IEEE / ITU International Conference on Artificial Intelligence for Good (AI4G), 2020.

[52] Mohammad Mehedy Hassan, Audrey Culver Smith, Katherine Walker, Munshi Khaledur Rahman, and Jane Southworth. *Rohingya Refugee Crisis and Forest Cover Change in Teknaf, Bangladesh*. Remote Sensing, **10**, 2018.

[53] John A. Quinn, Marguerite M. Nyhan, Celia Navarro, Davide Coluccia, Lars Bromley, and Miguel Luengo-Oroz. *Humanitarian applications of machine learning with remote-sensing data: Review and Case Study in Refugee Settlement Mapping*. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, **376**, pp. 20170363, 2018.

[54] Emilio Zagheni, Ingmar Weber, and Krishna Gummadi. *Leveraging facebook's advertising platform to monitor stocks of migrants*. Population and Development Review, **43**, pp. 721–734, 2017.

[55] Yutong He, Dingjie Wang, Nicholas Lai, William Zhang, Chenlin Meng, Marshall Burke, David B. Lobell, and Stefano Ermon. *Spatial-Temporal Super-Resolution of Satellite Imagery via Conditional Pixel Synthesis*. CoRR, **abs/2106.11485**, 2021.

[56] *UNHCR Global Trends - forced displacement in 2020*. 2021.

[57] Abhishek Chaurasia and Eugenio Culurciello. *LinkNet: Exploiting encoder representations for efficient semantic segmentation*. 2017 IEEE Visual Communications and Image Processing (VCIP), IEEE, 2017.

[58] Mary Kang, Shanna Christian, Michael A. Celia, Denise L. Mauzerall, Markus Bill, Alana R. Miller, Yuheng Chen, Mark E. Conrad, Thomas H. Darrah, and Robert B. Jackson. *Identification and characterization of high methane-emitting abandoned oil and gas wells*. Proceedings of the National Academy of Sciences, **113**, pp. 13636–13641, 2016.

[59] James P. Williams, Amara Regehr, and Mary Kang. *Methane Emissions from Abandoned Oil and Gas Wells in Canada and the United States*. Environmental Science & Technology, **55**, pp. 563–570, 2021.

[60] Chelsea Finn, Pieter Abbeel, and Sergey Levine. *Model-agnostic meta-learning for fast adaptation of deep networks*. International Conference on Machine Learning, pp. 1126–1135, 2017.

[61] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional networks for biomedical image segmentation*. International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 234–241, 2015.

[62] Ahmed Nassar, Karim Amer, Reda ElHakim, and Mohamed ElHelw. *A Deep CNN-Based Framework for Enhanced Aerial Imagery Registration With Applications to UAV Geolocalization*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2018.

[63] Thomas Blaschke, Stefan Lang, Eric Lorup, Josef Strobl, and Peter Zeil. *Object-oriented image processing in an integrated GIS/remote sensing environment and perspectives for environmental applications*. Environmental information for planning, politics and the public, **2**, pp. 555–570, 2000.

[64] Shuai Yang, Qihao Chen, Xiaohui Yuan, and Xiuguo Liu. *Adaptive coherency matrix estimation for polarimetric SAR imagery based on local heterogeneity coefficients*. IEEE Transactions on Geoscience and Remote Sensing, **54**, pp. 6732–6745, 2016.

[65] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. *Road Extraction by Deep Residual U-Net*. IEEE Geoscience and Remote Sensing Letters, **15**, pp. 749–753, 2018.

[66] Government of Alberta. *Landowner and indigenous community site nomination*. 2021.

[67] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. *The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition workshops, pp. 11–19, 2017.

[68] EarthLab. *Introduction to multispectral remote sensing data in Python*. 2018.

[69] Irem Ülkü and Erdem Akagündüz. *A Survey on Deep Learning-based Architectures for Semantic Segmentation on 2D images*. arXiv preprint arXiv:1912.10230, 2019.

[70] Irem Ulku, Panagiotis Barmpoutis, Tania Stathaki, and Erdem Akagunduz. *Comparison of single channel indices for U-Net based segmentation of vegetation in satellite images*. Twelfth International Conference on Machine Vision (ICMV 2019), pp. 338–345, 2020.

[71] Michael Wurm, Thomas Stark, Xiao Xiang Zhu, Matthias Weigand, and Hannes Taubenböck. *Semantic segmentation of slums in satellite images using transfer learning on fully convolutional neural networks*. ISPRS Journal of Photogrammetry and Remote Sensing, **150**, pp. 59–69, 2019.

[72] Jakaria Rabbi, Nilanjan Ray, Matthias Schubert, Subir Chowdhury, and Dennis Chao. *Small-Object Detection in Remote Sensing Images with End-to-End Edge-Enhanced GAN and Object Detector Network*. 2020.

[73] Statistics Canada Government of Canada. *Greenhouse detection with remote sensing and machine learning: Phase one*. 2021.

[74] Ling Tang and Sangeet Matthew. *Using Machine Learning and Deep learning with Imagery in ArcGIS*. .

[75] David Rolnick, Andreas Veit, Serge J. Belongie, and Nir Shavit. *Deep Learning is Robust to Massive Label Noise*. CoRR, **abs/1705.10694**, 2017.

[76] *Segmentation Models pytorch*. .

[77] *Footpath Dataset*. .

[78] *Building Dataset*. .

[79] *Kaggle Notebook*. .

[80] *kaggle Loss function*. .

[81] *IoU metric*. .

[82] *Planes Dataset*. .

[83] *Potvin*. .

[84] *notebook forest*. .

[85] *Segmentation metrics*. .

[86] *Data augmentation code*. .

[87] *Review DeepLabv3*. .