Deep Learning Methods for Multiclass Semantic Segmentation of High-Resolution Remote Sensing Images

28-11-2021 to 29-11-2021

Table of Contents

Objective

Goals

1. Simulation and Performance Analysis of existing methods for Multiclass semantic Segmentation of High-Resolution Remote Sensing Images.
2. Design and implementation of Robust Deep Learning Model for Multiclass semantic Segmentation of High-Resolution Remote Sensing Images.

Drive Link

This link consists of all the code and the latest version of the report.

1. **Benchmark Models:**
    - **Standard U-Net** implementation code
    - **U-Net with ResNet** backbone implementation code
    - **U-Net with DeepLabV3+** backbone implementation code
2. **Proposed Models:**
    - **Data Augmentation** implementation code
    - **Transfer Learning** on an **InceptionResNetV2 encoder** based **UNet CNN model**

https://drive.google.com/drive/folders/1DV0jbjHoMJS_gYCn2qZjH5herMHUzfnq?usp=sharing

Github repository

https://github.com/nive927/Dubai-Satellite-Imagery-Multiclass-Segmentation

Brief Introduction

Semantic segmentation is the task of clustering parts of an image together that belong to the same object class. It is a form of pixel-level prediction because each pixel in an image is classified according to a category. When there are 2 classes it is a binary semantic segmentation problem. AI in Remote Sensing involves the analysis of High-Resolution satellite or Aerial Images.

In this work, a satellite imaging dataset consisting of aerial satellite imagery of Dubai was obtained by MBRSC satellites and annotated with pixel-wise semantic segmentation into 6 classes is used. The images were segmented by the trainees of the Roia Foundation in Syria. This semantic segmentation dataset is dedicated to the public domain by Humans in the Loop under CC0 1.0 license. The dataset consists of 8 large imaging tiles obtained through remote sensing. Each tile is broken down into 9 similar sized image patches, amounting to a total of 72 different images. This is a significantly small dataset for training and testing deep learning models and data augmentation techniques are important to achieve good model performance

The project implements the multiclass semantic segmentation in two phases:

- The study and simulation of benchmark models: **U-Net, U-Net with ResNet and U-Net with DeepLabV3+** (can be extended to Inception, VGG16, etc)
- A new model that implements **Transfer Learning on a U-Net CNN with IncpetionResNetV2** after data augmentation to artificially expand the training data

The aforementioned benchmark models were chosen due to their established performance with small datasets. Both U-Net and DeepLab architectures generalize well, even when trained with a substantially small number of training samples. Furthermore, the use of ResNet as a feature extraction unit was used to leverage pre-learned weights to drive the model towards convergence faster. Due to the small size of the dataset, we need to patch and re-assemble for the benchmark models and apply augmentation for the proposed architecture. The model has achieved ~81% dice coefficient and ~86% accuracy on the validation set.

Benchmark Models

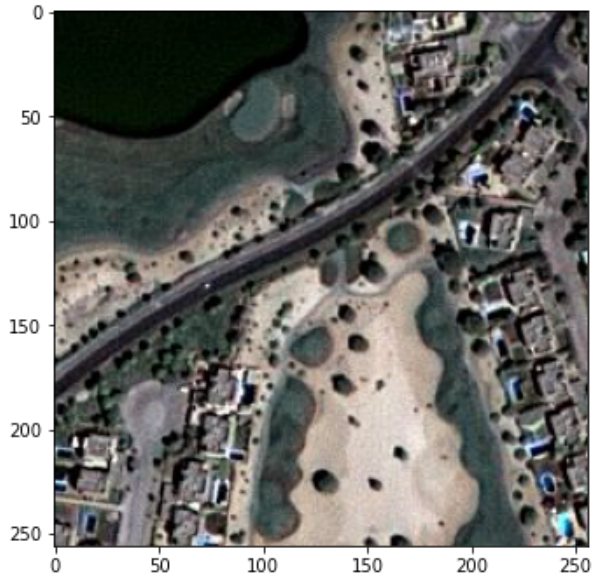
Common Preprocessing

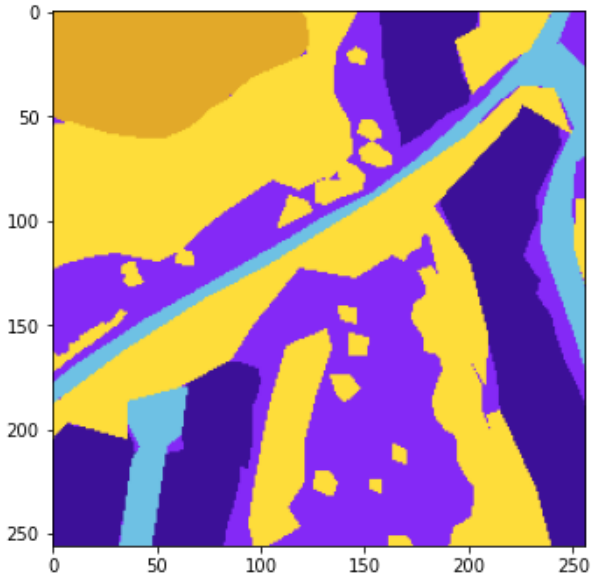
Creating image patches using Patchify

Using patchify, patches were created for the .**jpg images** and **.png masks**:

- **Tile 1:** 797 x 644 --> 768 x 512 --> 6
- **Tile 2:** 509 x 544 --> 512 x 256 --> 2
- **Tile 3:** 682 x 658 --> 512 x 512 --> 4
- **Tile 4:** 1099 x 846 --> 1024 x 768 --> 12
- **Tile 5:** 1126 x 1058 --> 1024 x 1024 --> 16
- **Tile 6:** 859 x 838 --> 768 x 768 --> 9
- **Tile 7:** 1817 x 2061 --> 1792 x 2048 --> 56
- **Tile 8:** 2149 x 1479 --> 1280 x 2048 --> 40

Totally, **9 images in each folder * (145 patches) = 1305** Total patches: **1305 patches of size 256x256.**





Hexadecimal to RGB

To convert the hex (Hexadecimel --> base 16) array to RGB, the number is divided by sixteen (integer division; ignoring any remainder) giving the first hexadecimal digit (between 0 and F, where the letters A to F represent the numbers 10 to 15). The remainder gives the second hexadecimal digit.

0-9 --> 0-9

10-15 --> A-F

Example:

RGB --> R=201, G=, B=

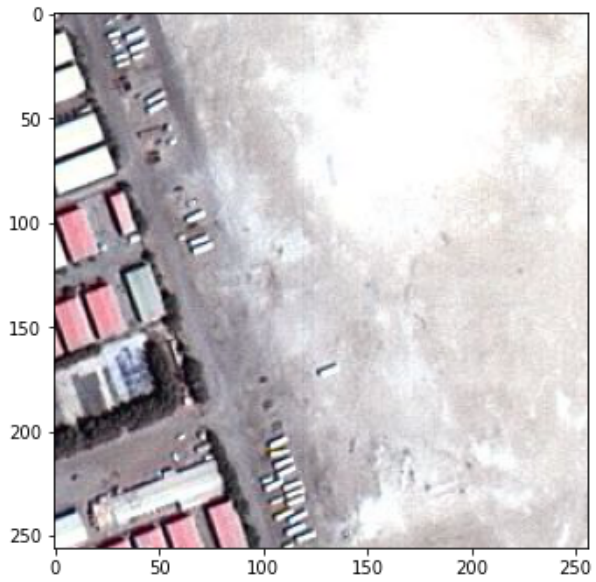
R = 201/16 = 12 with remainder of 9. So hex code for R is C9 (remember C=12)

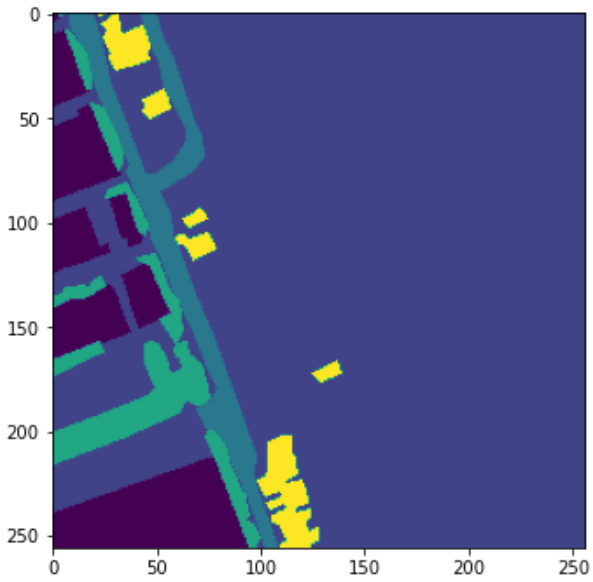
Calculating RGB from HEX: #3C1098

3C = 316 + 12 = 60

10 = 116 + 0 = 16

98 = 9*16 + 8 = 15

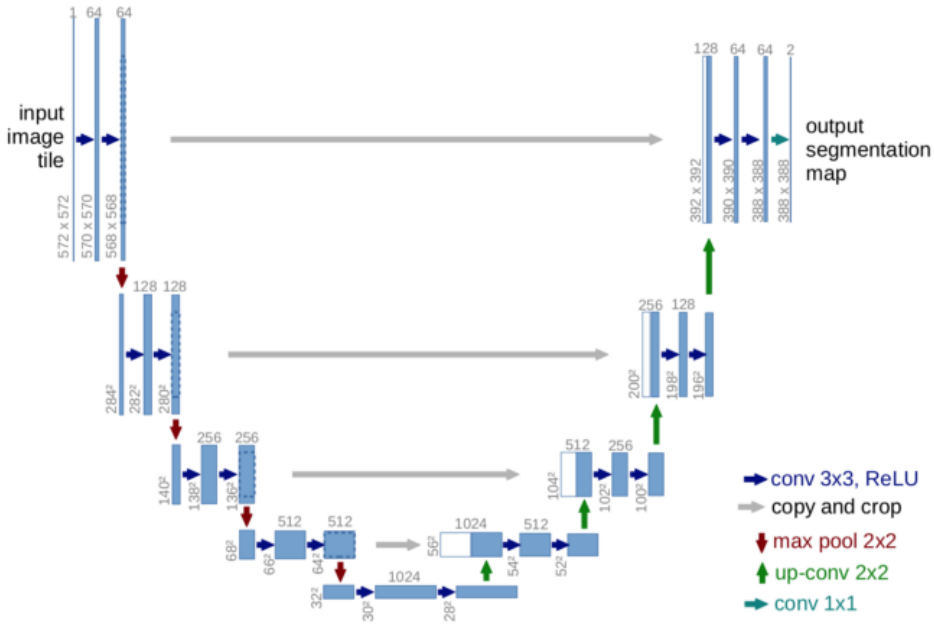




Standard U-Net

The U-Net is among the most well-known segmentation architectures, evolved from the traditional convolutional neural network, was first designed and applied in 2015 to process biomedical images. The network architecture is depicted in Figure-1

The architecture follows an encoder-decoder structure, with a characteristic U-shape, where the encoder deals with extracting image features through convolutions and down-sampling. The decoder up-samples the learnt features back to the original size of the input to localize and categorize each pixel of the input into a semantic class. The U-shape arises in the branch connecting convolution blocks at the bottom of the U-Net, which performs convolutions before starting to up-sample.



U-Net with ResNet backbone

The ResNet is a Convolutional Neural Network (CNN) architecture, made up of a series of residual blocks. It is known for its fast and representative feature extraction capabilities from images through convolution operations. The Residual blocks attribute the model with a capability to learn both complex as well as simplex relationships between subsequent blocks of layers. When the relationships are simple, the residual identity connection across the blocks allow the model to break down the order of relation polynomial between the subsequent blocks to better represent a simple relationship.

Since the encoder of a U-Net is essentially a feature-extractor, the pre-trained ResNet architecture can be loaded as the encoder architecture to down-sample the image and extract representation vectors, Through transfer learning, the encoder can be further trained to adapt to the specific use-case.

U-Net with DeepLabV3+ backbone

DeepLabv3+ (ref) is the recent version of the state-of-art class of deep-learning architectures for semantic image segmentation called DeepLab, open-sourced by Google Inc. in 2016. This architecture also follows an encoder-decoder structure.

It is known for its Atrous Spatial Pyramid Pooling (ASPP) scheme. Here, parallel dilated convolutions with different rates are applied in the input feature map, which is then fused together. As objects of the same class can have different sizes in the image, ASPP helps to account for different object sizes.

The architecture also leverages the use of Dilated Convolutions, a type of convolution that inflate the kernels by inserting holes between the kernel elements. An additional parameter called dilation rate, akin to dropout rate indicates how much the kernel is widened. Dilated convolution helps expand the area of the input image covered without pooling. This allows the model to cover more information from the output obtained with every convolution operation.

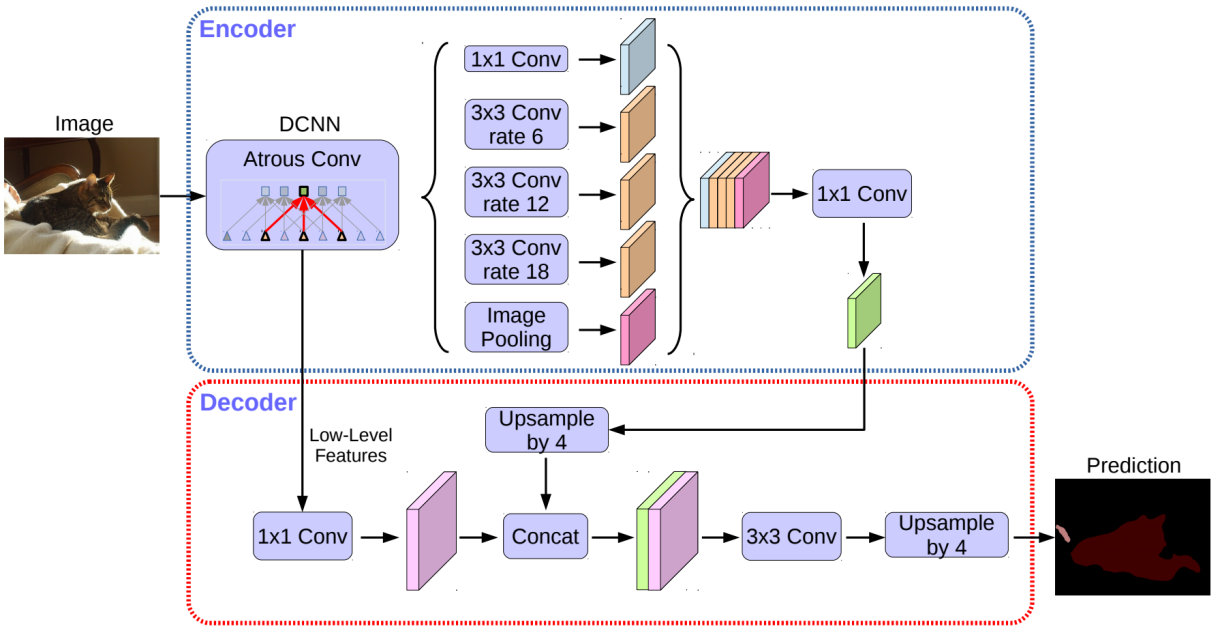


Figure-2: DeepLabV3+ Architecture

Proposed Model

Transfer learning on InceptionResNetV2 encoder based U-Net CNN Model

For the multiclass semantic segmentation on Dubai's Satellite Imagery Dataset, this work proposes an approach that implements transfer learning on an InceptionResNetV2 encoder based U-Net CNN model. Prior to training the model, the dataset was artificially expanded by augmenting the training dataset. This also prevents overfitting. The model achieved ~76% dice coefficient and ~83% accuracy on the validation set after running 20/50 epochs.

Data Augmentation

The small dataset consisting of 72 images (8 Tiles with 9 images each) is augmented to increase the number of images for both the images and corresponding masks:

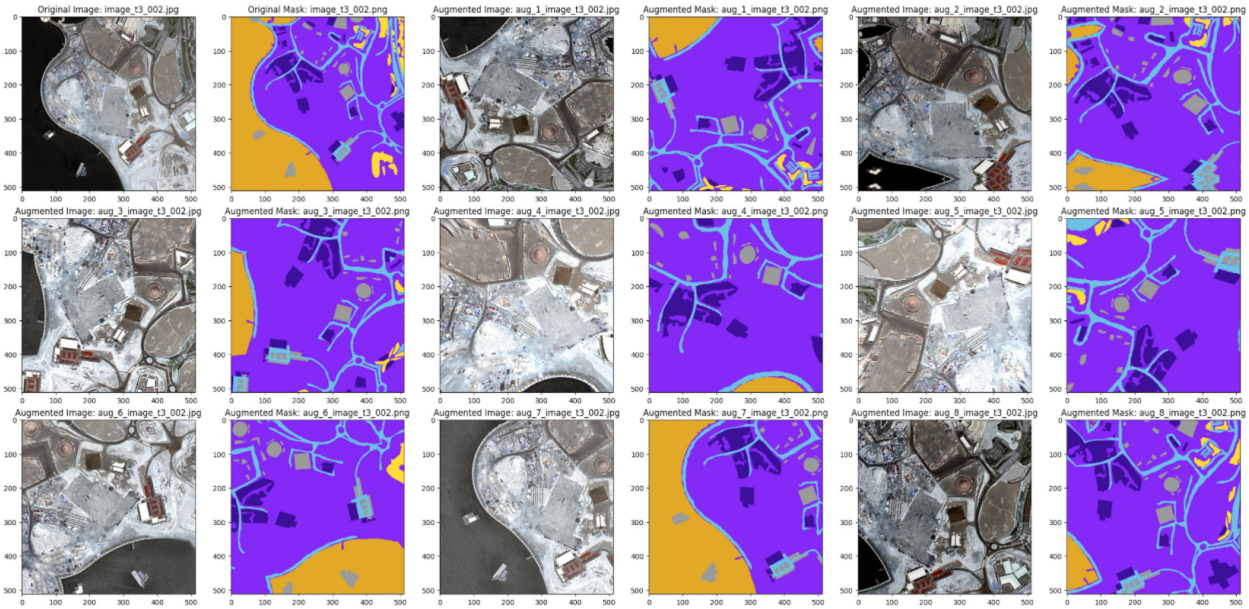
- Original Dataset: 72 images (8 Tiles with 9 images each)
- This is split into Training set: 56 (8 Tiles, first 7 images) Validation set: 16 (8 Tiles, last 2 images) which is ~78% for training and the remaining for validation
- Subsequently, the 56 images in the training set are augmented: Each image is augmented 8 times (56 * 8 = 448) and the original 56 are also retained, Total Augmented Training set: 484 + 56 = 504

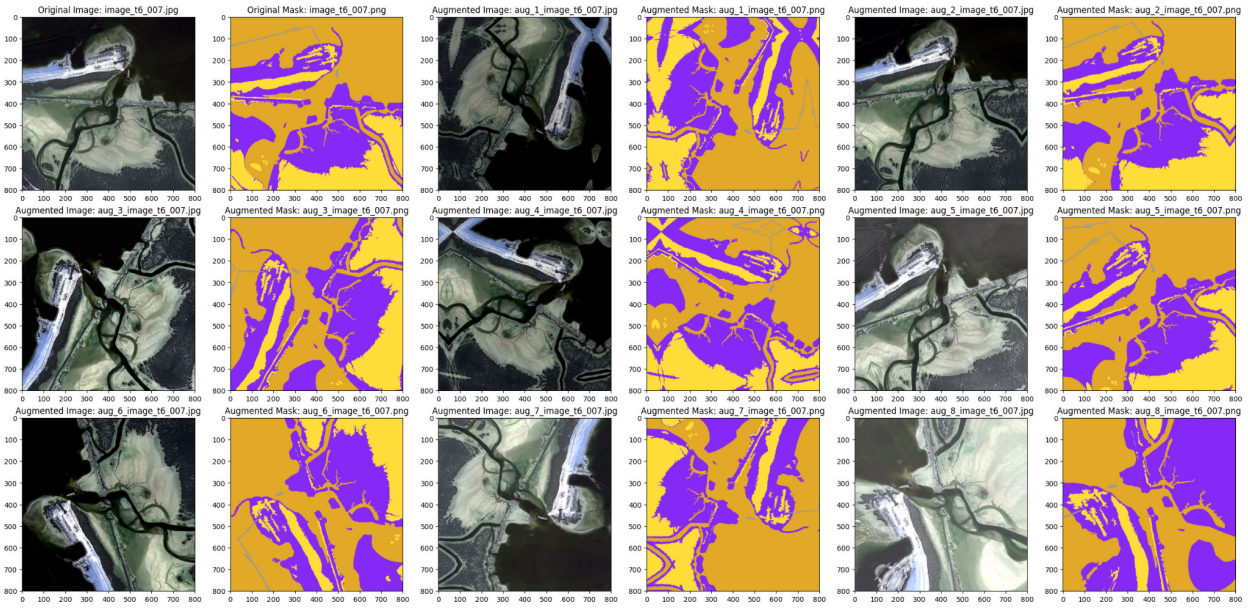
Albumentations is a Python library for fast and flexible image augmentations. Albumentations efficiently implements a rich variety of image transform operations that are optimized for performance and does so while providing a concise, yet powerful image augmentation interfaces for different computer vision tasks, including object classification, segmentation, and detection.

In this work, data augmentation is done by the following techniques:

1. Random Cropping
2. Horizontal Flipping
3. Vertical Flipping
4. Rotation

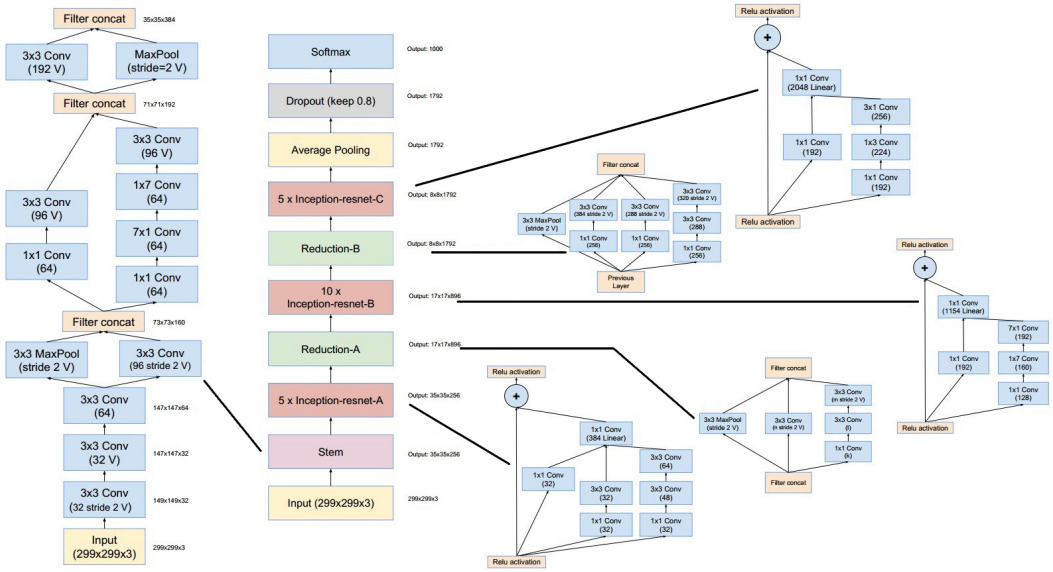
5. Random Brightness & Contrast
6. Contrast Limited Adaptive Histogram Equalization (CLAHE)
7. Grid Distortion
8. Optical Distortion



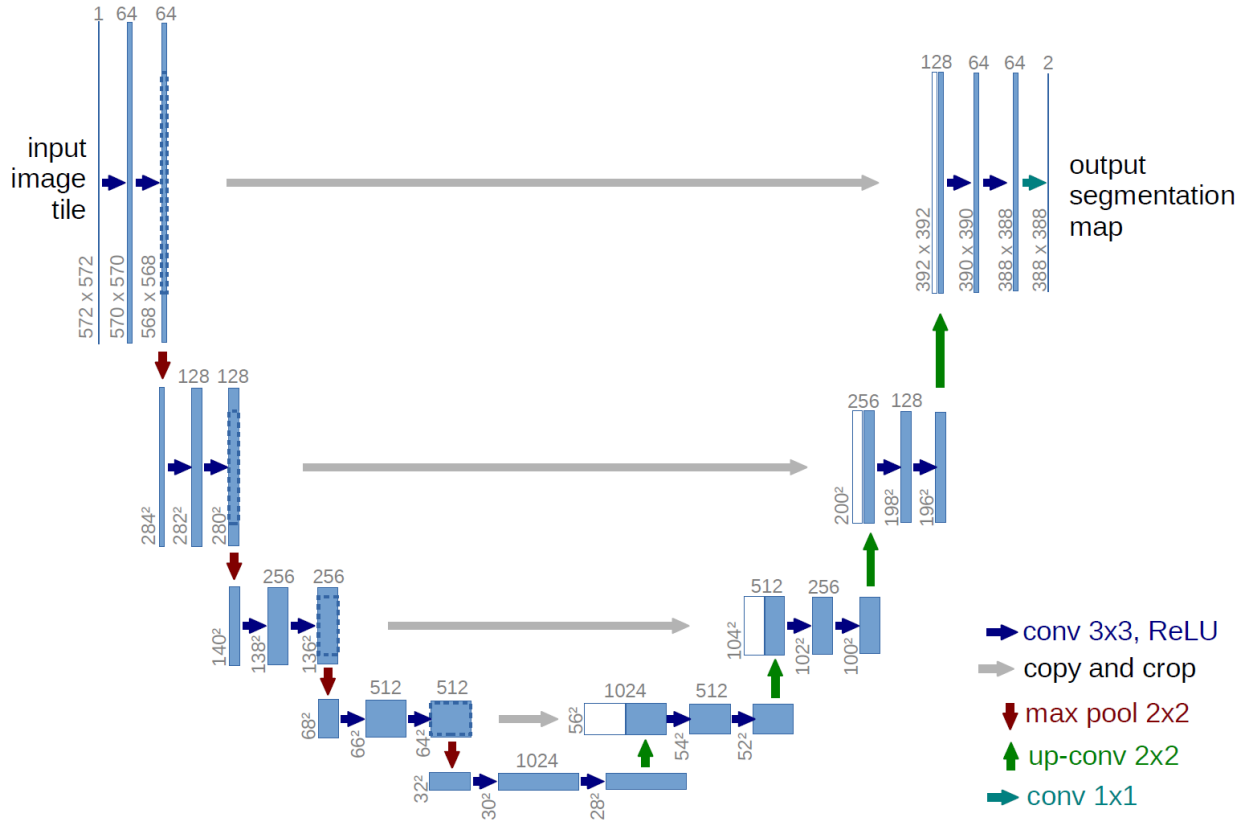


Architecture

InceptionResNetV2



U-Net



InceptionResNetV2 model pre-trained on the ImageNet dataset has been used as an encoder network. A decoder network has been extended from the last layer of the pre-trained model, and it is concatenated to the consecutive layers.

Semantic Annotation & Preprocessing

Name	R	G	B	Color
Building	60	16	152	
Land	132	41	246	
Road	110	193	228	
Vegetation	254	221	58	
Water	226	169	41	
Unlabeled	155	155	155	

- Label and code conversion dictionaries for the 6 classes
- One Hot Encoding of RGB Labels
- Decoding Encoded Predictions

Hyper-Parameters

HYPER-PARAMETERS	
Batch Size	16
Steps per Epoch	32
Validation Steps	4
Input Shape	(512, 512, 3)
Initial Learning Rate (with Exponential Decay LearningRateScheduler Callback)	0.0001
Number of Epoch (with ModelCheckpoint & EarlyStopping callback)	50 (total)

Training and Implementation Details (Software and Library: Python Keras/Tensorflow/Pytorch)

Software

- Python 3.* as the primary programming language
- Jupyter Notebook for interactive running and markdown
- IPython for interactive programming

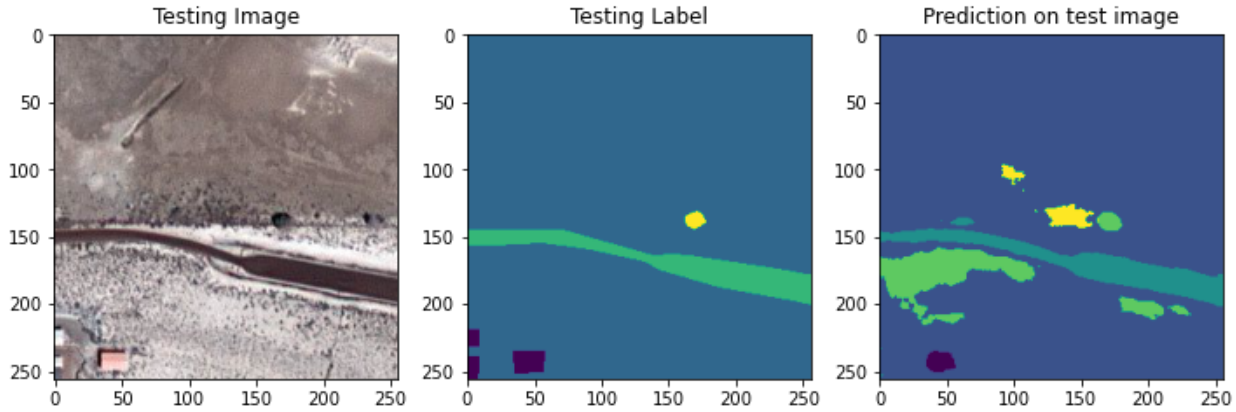
Libraries for Statistical and Deep Learning

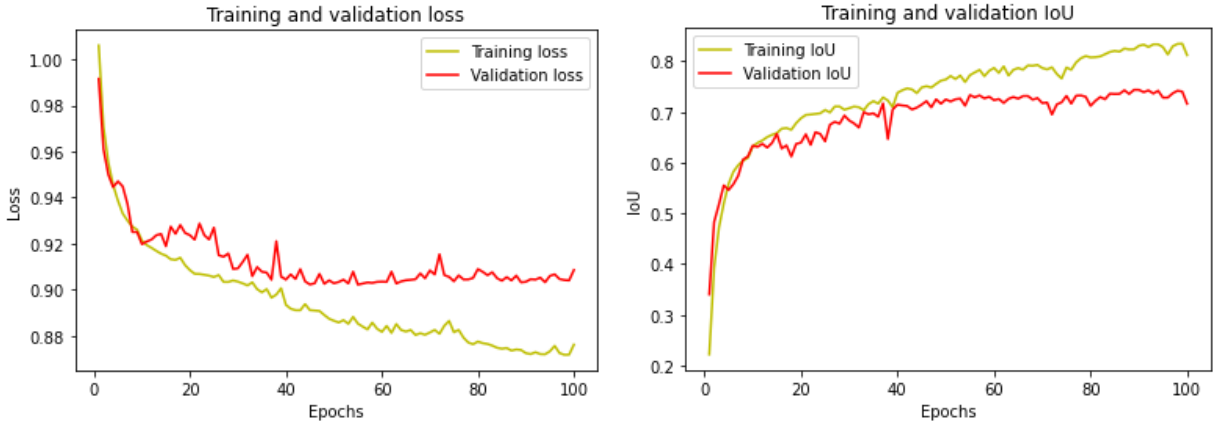
- NumPy for vectorized implementation and fast arrays
- Pandas for dealing with tabular and high dimensional data
- Matplotlib for plotting and visualizations
- OpenCV for image manipulations
- Tensorflow for Deep Learning
- Keras models for Computer Vision
- Patchify for creating patches of the images by cropping to re-piecing instead of resizing all the different sizes of the images
- Segmentation-models for the pre-trained backbones used along with a U-Net for benchmark implementations
- Albumentations for data augmentation
- Keract for activations

Simulation Results and Discussion

Standard U-Net

Simulation Results





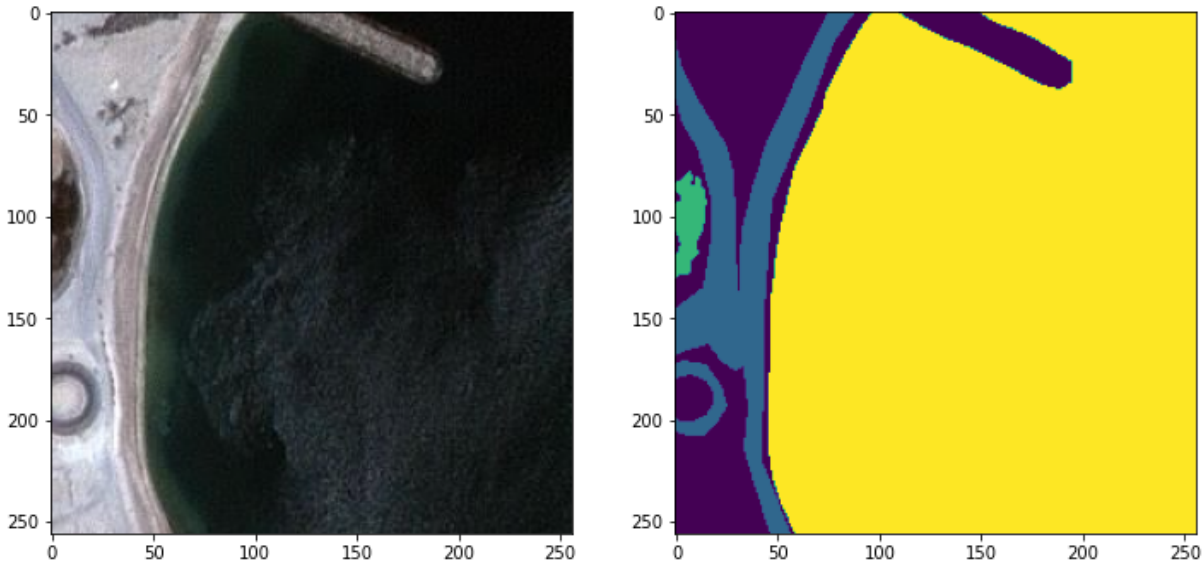
Inference

With Minmaxscaler and weights of [0.1666, 0.1666, 0.1666, 0.1666, 0.1666, 0.1666] in Dice loss:

- With focal loss only, after 100 epochs val jacard is: 0.62 (Mean IoU: 0.6)
- With dice loss only, after 100 epochs val jacard is: 0.74 (Reached 0.7 in 40 epochs)
- With dice + 5 focal, after 100 epochs val jacard is: 0.711 (Mean IoU: 0.611)
- With dice + 1 focal, after 100 epochs val jacard is: 0.75 (Mean IoU: 0.62)
- Using categorical cross-entropy as loss: 0.71

U-Net with ResNet backbone

Simulation Results



A sample Validation Input and its corresponding Prediction Mask

Inference

The model was trained by minimizing the *Categorical Cross-Entropy* loss and the Adam optimizer with an initial learning rate of 0.001

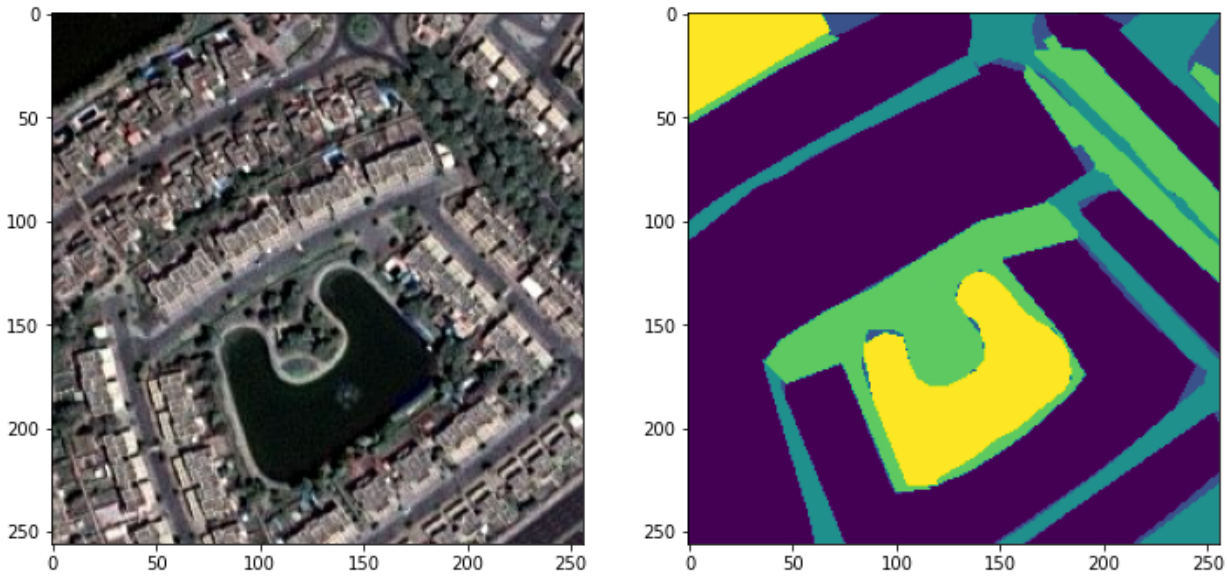
At the end of 10 epochs, the model reached

- Loss: 0.5284
- Pixel Accuracy: 81.87%
- Jaccard Coefficient: 0.732

(Metrics evaluated on the validation set)

U-Net with DeepLabV3+

Simulation Results



A sample Validation Input and its corresponding Prediction Mask

Inference

The model was trained by minimizing the *Categorical Cross-Entropy* loss and the Adam optimizer with an initial learning rate of 0.001

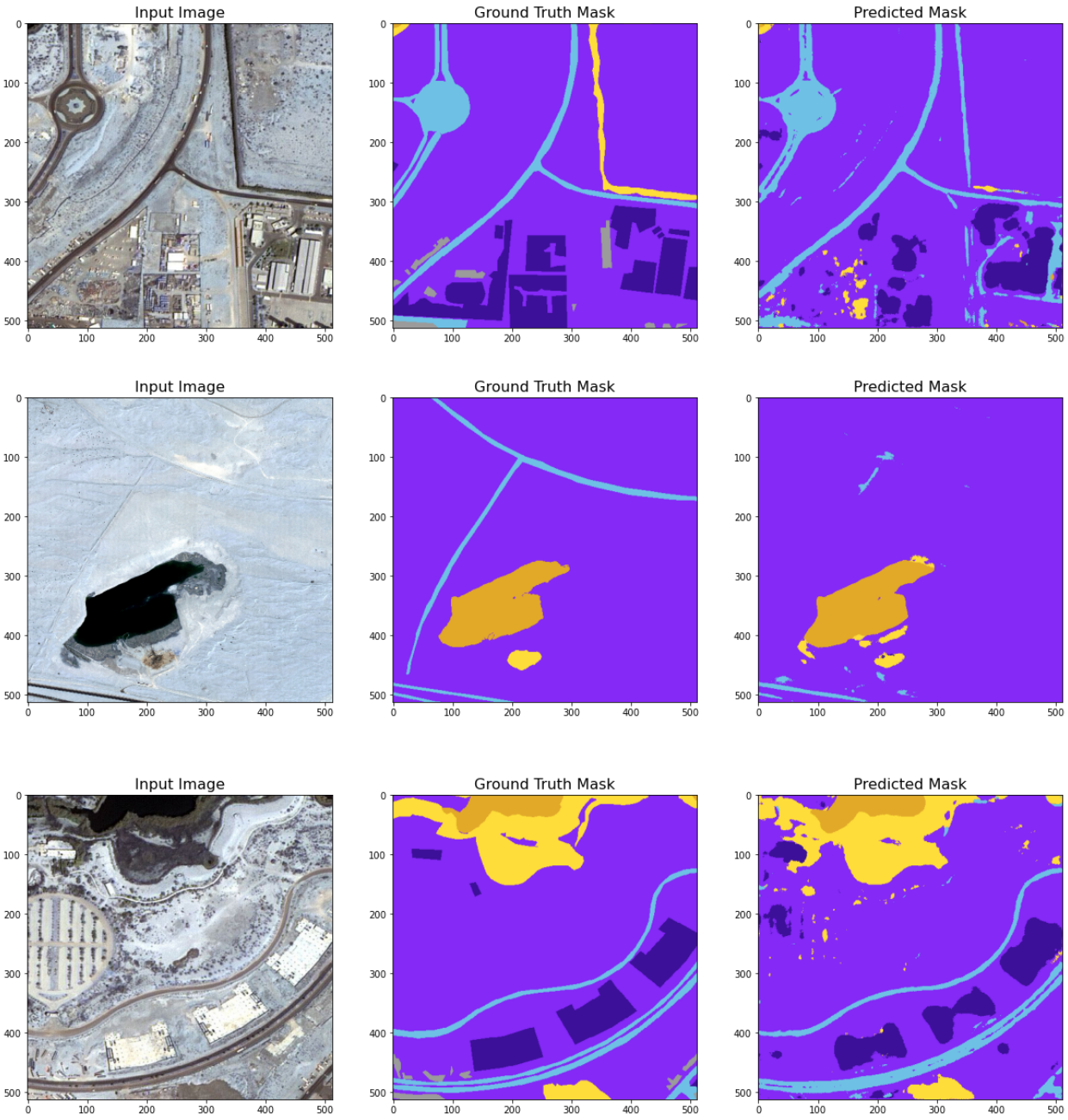
At the end of 10 epochs, the model reached

- Loss: 0.5556
- Pixel Accuracy: 81.17%
- Jaccard Coefficient: 0.658

(Metrics evaluated on the validation set)

Proposed Transfer Learning using InceptionResNetV2 encoder with U-Net CNN model

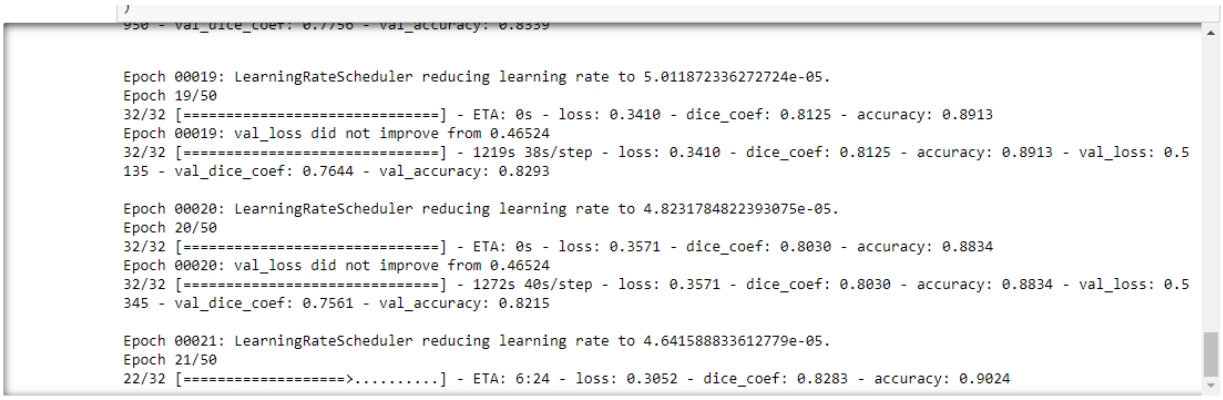
Simulation Results



Inference

It achieves a dice_coef ~76% and accuracy ~83% at 20/50 epochs.

Since the model has been running for 8+ hrs, the notebook was duplicated to summarize the intermediate results for the sake of the competition. Post the complete training, the final notebook will be uploaded to the drive and Github.



Conclusion

In conclusion, we see that the proposed model that was adapted for this particular application archives good results determined by ~0.78% dice coefficient and ~84% accuracy on the validation data at 22/50 epoch after 8+ hrs of training.

```
Epoch 00022: LearningRateScheduler reducing learning rate to 4.46683592150963
14e-05.
Epoch 22/50
32/32 [==============================] - ETA: 0s - loss: 0.3267 - dice_coef:
0.8189 - accuracy: 0.8938
Epoch 00022: val_loss did not improve from 0.46524
32/32 [==============================] - 1269s 40s/step - loss: 0.3267 - dice
_coef: 0.8189 - accuracy: 0.8938 - val_loss: 0.5091 - val_dice_coef: 0.7790 -
val_accuracy: 0.8343
```

References

[1] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning,” arXiv.org, 23-Aug-2016. [Online]. Available: https://arxiv.org/abs/1602.07261.

[2] Ulmas, Priit and Innar Liiv. “Segmentation of Satellite Imagery using U-Net Models for Land Cover Classification.” ArXiv abs/2003.02899 (2020) [Online]: Available: https://arxiv.org/pdf/2003.02899.pdf

[3]Study of the different inception networks https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

[4] Dataset- https://humansintheloop.org/resources/datasets/semantic-segmentation-dataset/

[5] https://arxiv.org/pdf/1602.07261v2.pdf - InceptionResnNetV2

[6] https://arxiv.org/pdf/1505.04597.pdf - U-Net

[7] Keras Models - https://keras.io/api/applications/

[8] Resnet - https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

[9] Chen, Liang-Chieh et al. “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation.” ArXiv abs/1802.02611 (2018): [Online] Available: https://arxiv.org/pdf/1802.02611v3.pdf