LoveDA Semantic Segmentation Challenge

Chavda JayrajSinh (2021CSB1078)

Imrozepal Singh (2021CSB1096)

Motivation

The Land-cOVEr Domain Adaptive semantic segmentation (LoveDA) dataset was created to provide land-cover semantic segmentation and unsupervised domain adaptation (UDA) tasks. Exploring the use of deep transfer learning methods on this dataset will be a meaningful way to promote city-level or national-level land-cover mapping.

The initial version of the dataset was created by Junjue Wang, Zhuo Zheng, Ailong Ma, Xiaoyan Lu, and Yanfei Zhong, most of whom were researchers at the Wuhan University.

Objective

- Given:

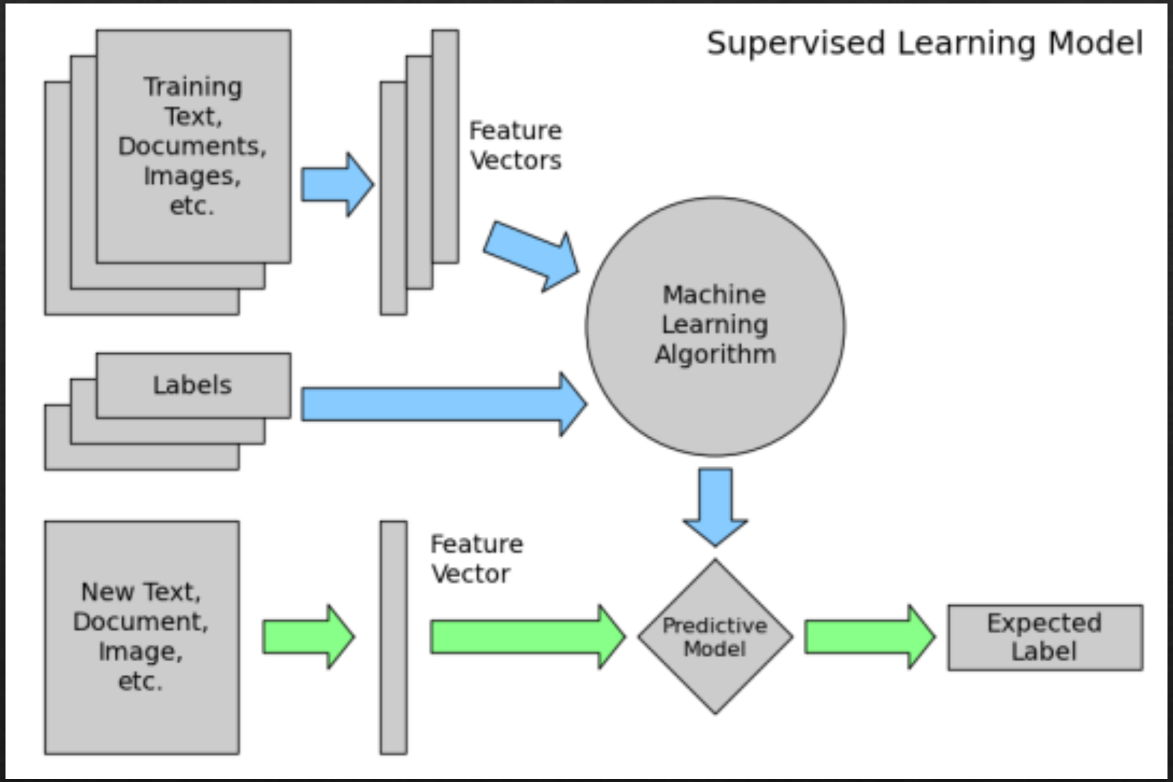
1. A set of input images and corresponding semantic masks representing land-cover types: background – 1, building – 2, road – 3, water – 4, barren – 5,forest – 6, agriculture – 7. And the no-data regions were assigned 0.
2. A set of test images

- Aim:

  To return the set of corresponding semantic masks.

Machine Learning

- Instead of relying on predefined rules, machine learning enables systems to automatically learn and improve from data.
- A model is randomly initialised and trained on a dataset, which consists of input data and corresponding outputs.
- The training process involves an optimization algorithm that adjusts the model's parameters to minimize a given loss function which is a measure of how poorly the model performed.


Supervised Learning Model
Training Text, Documents, Images, etc.
Feature Vectors
Machine Learning Algorithm
Labels
New Text, Document, Image, etc.
Feature Vector
Predictive Model
Expected Label

Usage of Machine Learning in given problem

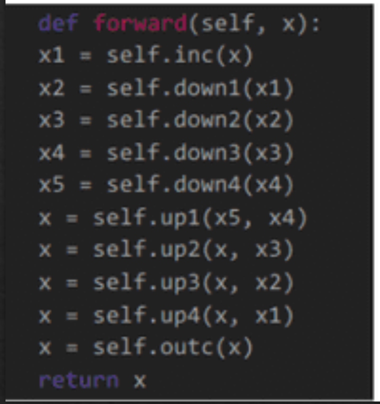
- The set of train images and corresponding masks are given as input to train the model.
- After sufficiently trained, the model can be used to make predictions on test images.
- In given problem, specifically, convolutional neural network (CNN) has been utilised.
- A Convolutional Neural Network (CNN) is a specialized type of deep neural network designed to process and analyse visual data, such as images or videos. CNNs are particularly effective for tasks like image classification, object detection, and image segmentation.

CNN

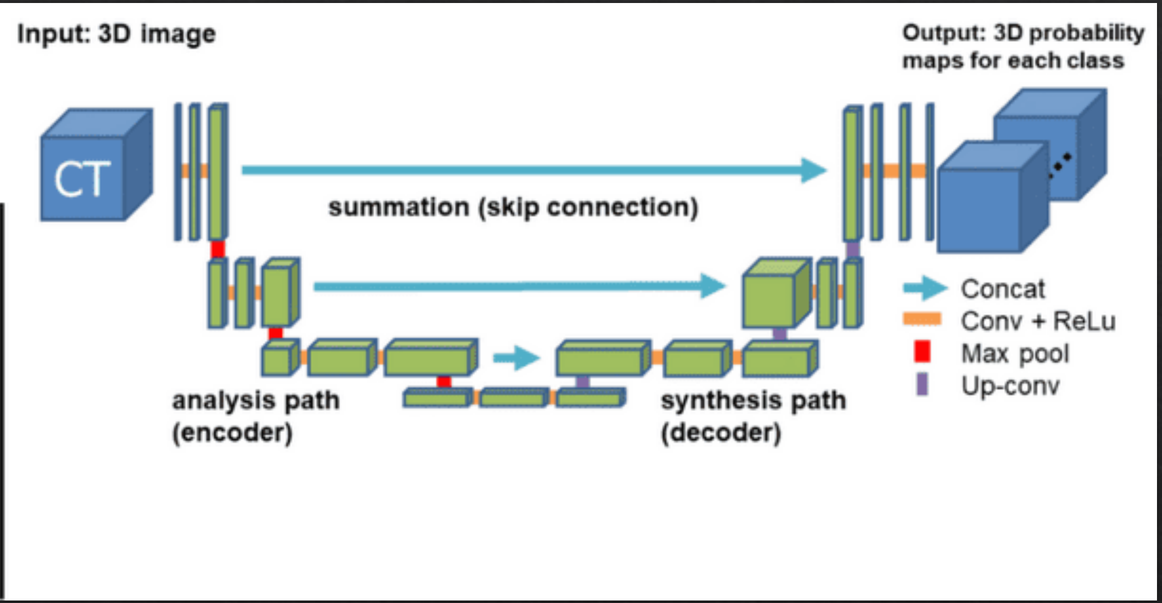
- CNNs incorporate convolutional layers, which perform convolution operations on the input data.
- Convolution involves sliding a small filter or kernel over the input image and computing the dot product between the filter and the local receptive field at each position
- The fundamental difference between a densely connected layer and a convolution layer is that dense layers learn global patterns whereas the latter learns local patterns.
- Local patterns are more important in given problem, hence CNN is more appropriate.

U-Net CNN

•U-Net has a U-shaped architecture.
•It consists of an encoder and a decoder, connected by a bottleneck layer.
•The encoder captures context and extracts high-level features
• The decoder generates the segmentation map.
•The skip connections allow information to flow between corresponding layers of the encoder and decoder.

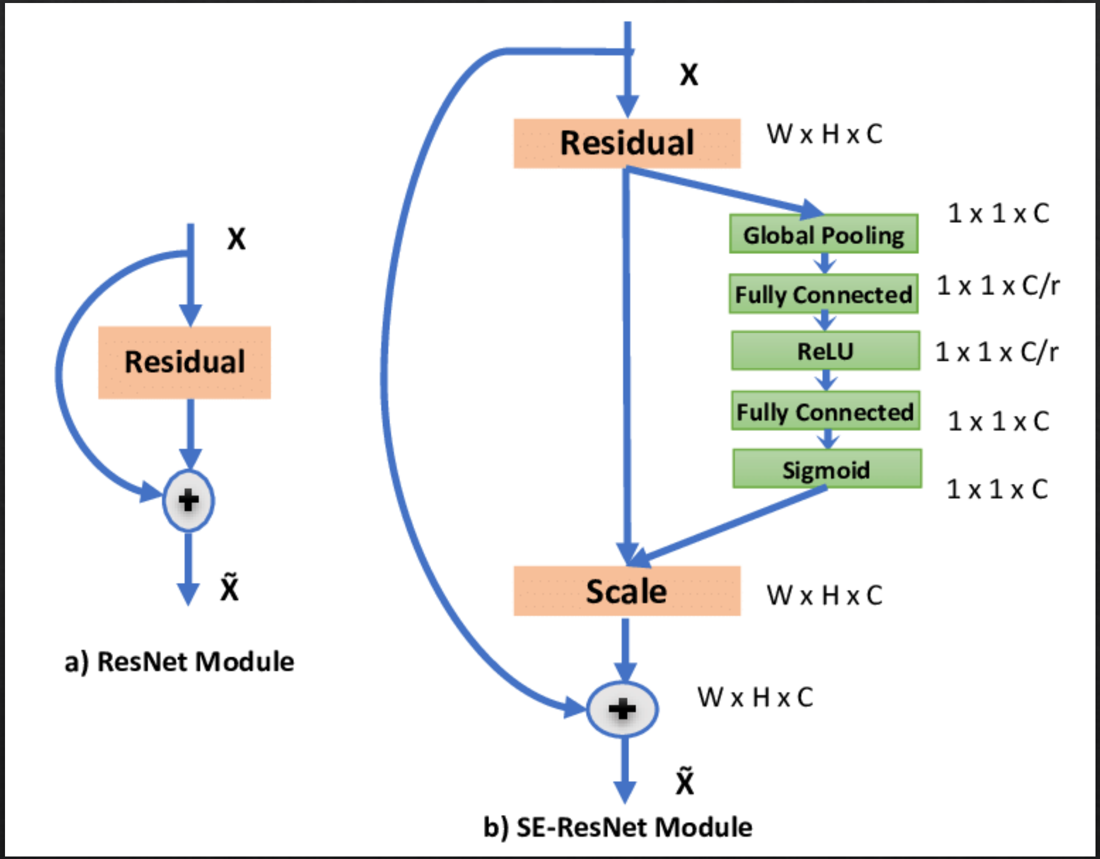


```
def forward(self, x):
    x1 = self.inc(x)
    x2 = self.down1(x1)
    x3 = self.down2(x2)
    x4 = self.down3(x3)
    x5 = self.down4(x4)
    x = self.up1(x5, x4)
    x = self.up2(x, x3)
    x = self.up3(x, x2)
    x = self.up4(x, x1)
    x = self.outc(x)
    return x
```



ResNet CNN

- ResNet (Residual Network) is a CNN architecture designed to address the challenge of training very deep networks where adding more layers can lead to performance degradation.
- It introduces residual connections, allowing information to bypass one or more layers.
- Residual connections perform identity mapping, preserving important features and gradients for better information flow during training.



U-net model

Model-Parameters

Total params: 1,941,224

Trainable params: 1,941,224

Non-trainable params: 0

This was the initial model but was not converging quickly
The reference video confirms that initial convergence is slow but later the results obtained form u-net are same as resNet

But it must be noted that unet has lesser parameters

ResNet

Details about Model parameters

Back-Bone: 'resnet34'

encoder_weights='imagenet'

Total params: 24,457,169

Trainable params: 24,439,819

Non-trainable params: 17,350

For Detailed Model summary refer code file

Pre Processing

We had a huge dataset so it was not possible to store the entire dataset on one numpy array as ram was limited.

To, Tackle this we wrote a custom ImageDataGenerator which inherits from keras sequence

The image generator loads only the images required in one batch and feed it to model.

We first get the array of file_names of images and corresponding mask and shuffle

It so that our model do not have a bias to order in which images are feed

The we can sample the dataset by slicing the file array

We also have provided a argument for image augmentation to avoid “OVERFITTING”

Pre-Processing(contd.)

- Each image is of Size 1024×1024
- We make patches of the image of size 256×256
- This is done to limit the GPU usage
- We convert the corresponding mask into categorical(one hot encoded)

Training

HyperParameters:

Metrics used are accuracy and Jacard coefficient

Optimizer used for training = adam

loss= categorical_crossentropy

batch_size=64

epochs=160

Accuracy after 160 epochs

```
loss: 0.5164 - accuracy: 0.8057 - jacard_coef: 0.5690 - val_loss: 0.9820 - val_accuracy: 0.6576 - val_jacard_coef: 0.3995
```

Training Data

Loss on training data: 0.5164

Accuracy on training data: 0.8057

Jacard coefficient for training data: 0.5690

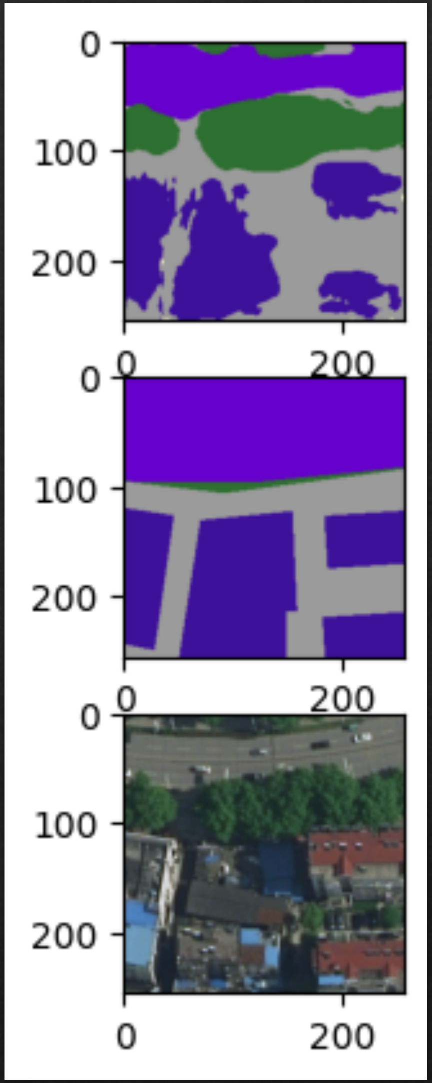
Validation Data

Loss on training data: 0.5164

Accuracy on training data: 0.8057

Jacard coefficient for training data: 0.5690

Outputs

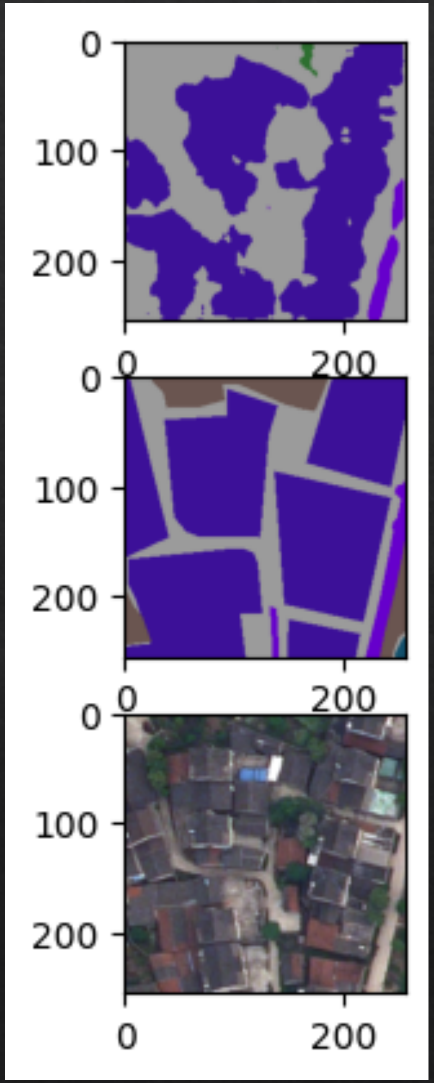


Predicted Mask

True Mask

Input Image

1



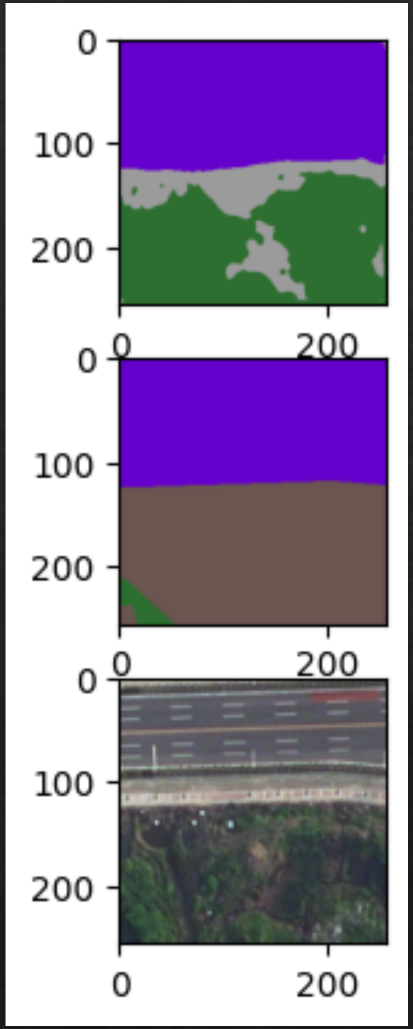
Predicted Mask

True Mask

Input Image

2

Outputs (continued..)

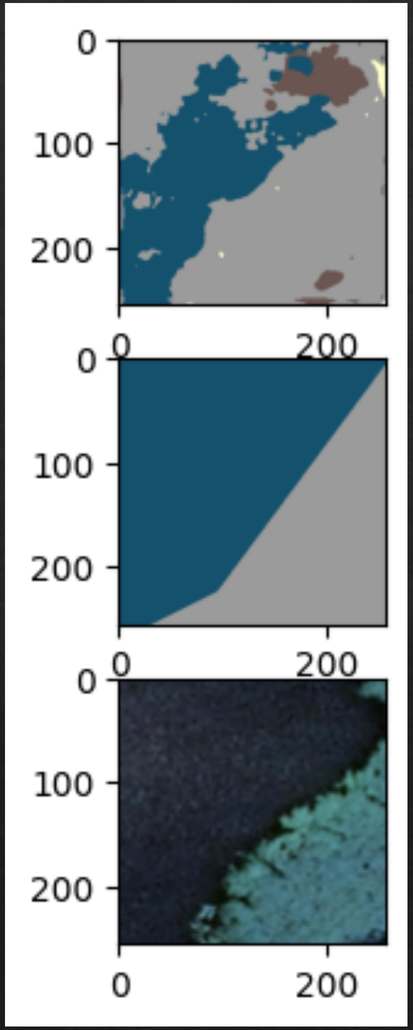


Predicted Mask

True Mask

Input Image

3



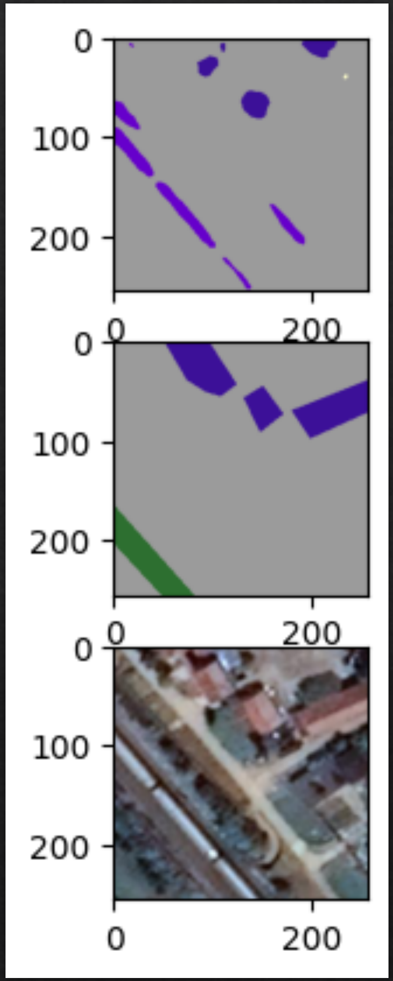
Predicted Mask

True Mask

Input Image

4

Outputs (continued..)



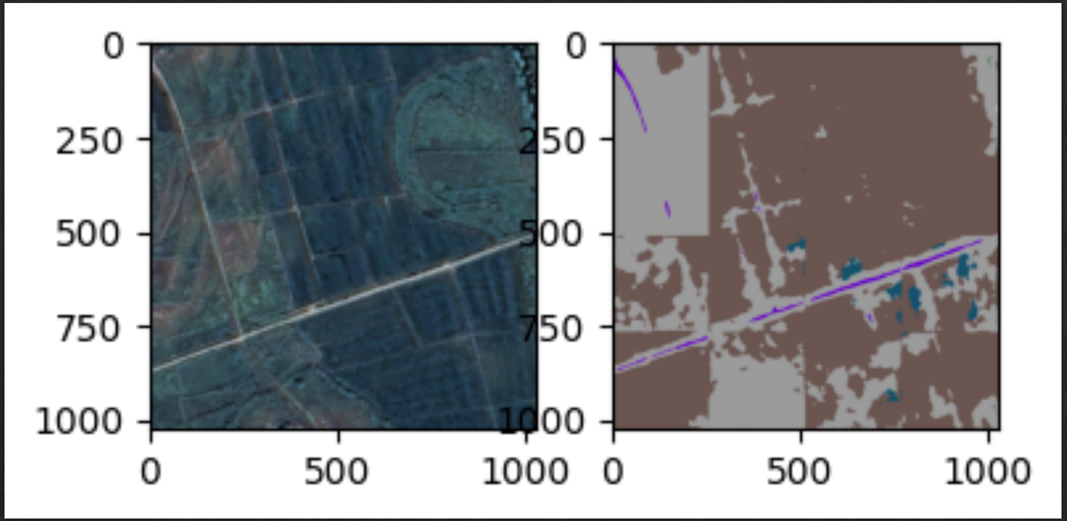
Predicted Mask

True Mask

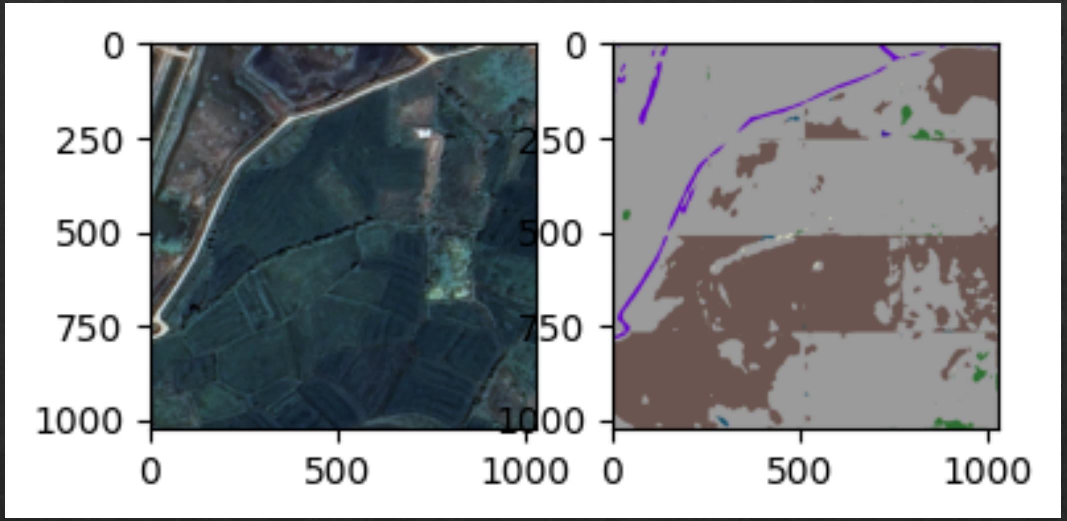
Input Image

Outputs (Random Images from dataset)

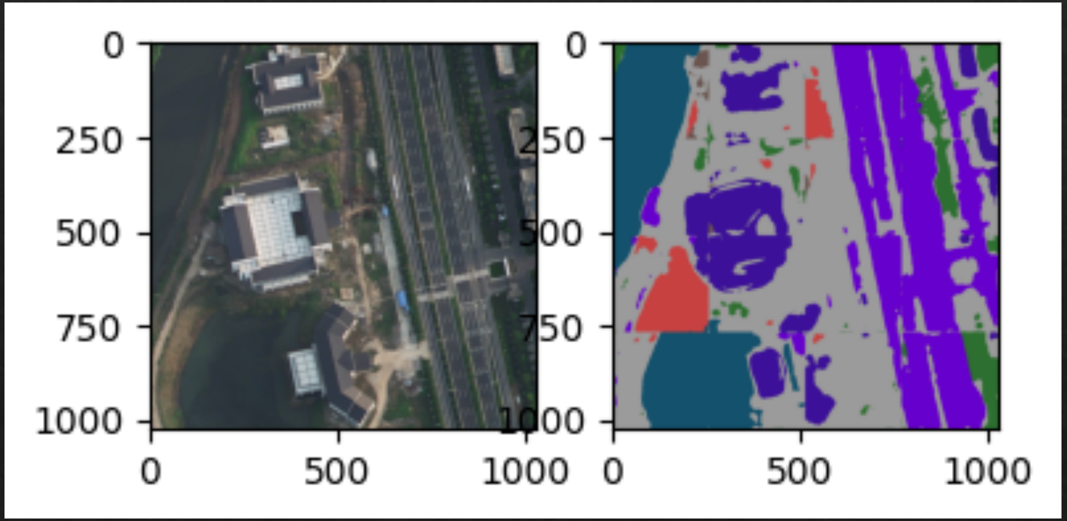
f1



f2



f3



For f1 ,f2 ,f3
(Input Test Image , Output mask prediction)