An efficient CNN architecture for land use segmentation of high- resolution imagery using Genetic algorithm and compare with other optimizers (Adam, GD, SGD)

Aayushman 20004

We perform semantic by updating the weights by using the Genetic algorithm. After the best result is generated we compare it with the existing state-of-the-art optimizers such as Adam and SGD, GD. The dataset was created from high-resolution, true-color satellite imagery of Pleiades-1A acquired on March 15, 2017. The CNN architecture used here is UNet for multiclass segmentation into 6 unique classes were used to categorize the images, namely Vegetation, built-up,informal settlements,impervious surfaces (roads/highways, streets, parking lots, road-like areas around buildings, etc.), barren and water

Introduction: Dataset and Background

For this assignment we are using Manually Annoted High Resolution Satelite Image Dataset of Mumbai for Semantic Segmentation. The dataset is divided into two groups, each including satelite images and their corresponding semantic segmentation masks. The images in each group are of two different sizes. The first group comprises 110 satellite images of size 600×600 pixels and manually labelled semantic segmentation masks. Figure 1 depicts some sample images and their labelled semantic segmentation masks from the first group. The second group contains three sets: training, validation, and testing. Each set consists of images of size 120×120 pixels which are derived and processed from the first group. Researchers can employ it to train and evaluate machine learning models. The first group is provided in the Main_Dataset directory, and the second group is provided in the Prepared_Dataset directory. All the image-mask pairs were named with the same file names, for example, tile_5.37.tif and tile_5.37.png. For the sample images, directory tree of the dataset, see Fig 1 and Fig 2 respectively.

For semantic image segmentation, we are implementing a UNet-based deep learning model for image segmentation. The UNet architecture is a popular approach for image segmentation that leverages a convolutional neural network (CNN) to predict pixel-level segmentation masks. The model is trained using a custom dataset consisting of image and corresponding mask pairs. Finally, it creates an optimizer and a learning rate scheduler to train the model. Optimizers such as Adam, Stochastic Gradient Descent and Genetic Algorithm for updating weights of the neural network.

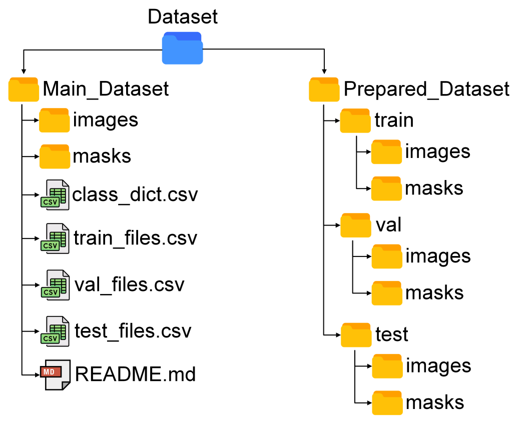

Fig. 1. The directory structure of the dataset.

Methodology

Here is a step-by-step description of our methodology:

Step 1: Importing Required Libraries: The necessary libraries required for the implementation of the code are imported. These include Pandas, Matplotlib, Pillow (PIL), NumPy, PyTorch, and various modules from the PyTorch library.

Step 2: Mounting the Google Drive: The Google Drive is mounted on the Google Colaboratory platform, which allows the user to access the datasets stored in the drive. You can skip this step if not using Colaboratory.

Step 3: Reading the Dataset: The path to the dataset is defined, and the CSV files containing the data are read into Pandas data frames.

Step 4: Preparing the Dataset: The paths to the training, testing, and validation data directories are defined. A function is defined to retrieve the file paths for all the images and masks in a directory. These file paths are used to create a custom dataset class that loads the images and masks from the file paths.

Step 5: Creating the Data Loaders: The data loaders are created using the custom dataset class defined earlier. The data loader for the training set is created with a batch size of 8, and the data loader for the validation set is created with a batch size of 8 as well.

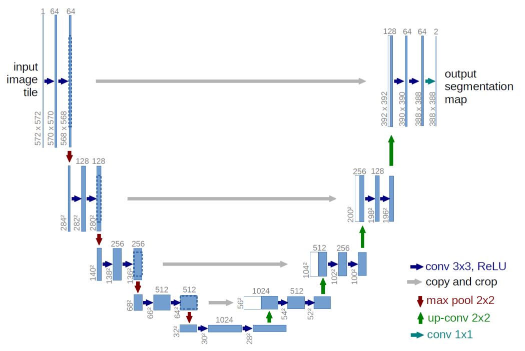

Fig. 2. UNet Architecture.

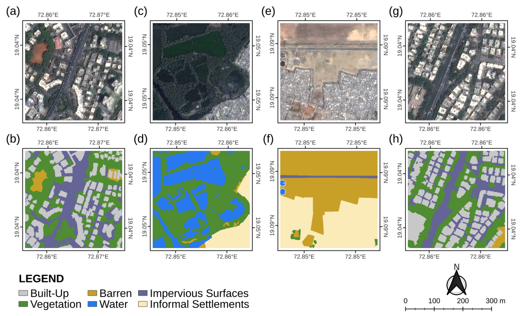

Fig. 3. Original satellite images (a, c, e, g) and their corresponding segmentation masks (b, d, f, h).

Step 6: Defining the Neural Network: The neural network for the image segmentation task is defined using PyTorch. The network architecture is a U-Net, which consists of a series of downsampling and upsampling layers that are connected by skip connections.

Step 7: Defining the Loss Function and Optimizer: The loss function for the network is defined as the Cross-Entropy loss. The optimizer used for training the network is the Adam optimizer.

Step 8: Training the Network: The network is trained using the training data loader and the validation data loader. The number of epochs is set to 20, and the learning rate is initially set to 0.01. A scheduler is defined to decrease the learning rate by a factor of 0.1 after every 10 epochs.

Step 9: Saving the Trained Model: The trained model is saved to the Google Drive or on local device so that it can be used for prediction or further training in the future.

Results and Discussion

1. images for train_dataloader increases a loss after training for 100 epochs keeping lr = 0.001 constant where lr is learning rate, momentum = 0.9 for optimizer sgd.

Number of Images	Running Loss	Validation Loss
24	5.3940	1.8540
240	43.0305	1.4816
1028	179.2106	1.4615
3072	nan	nan

I was not able to train 3072 training images.

2. Changing learningrate for training 108 images as constant after 100 epochs of training.

learning_rate	Running Loss	Validation Loss
0.0001	22.0695	1.5874
0.001	21.6433	1.5656
0.01	22.4769	1.5388

3. For genetic algorithm implementation, I tried using PyGAD module but still not able to improve on the results using algorithm. I got -439.85 fitness = -loss_function.value()

4. The U-Net based architecture is used for image segmentation, it consists of a contracting path and an expansive path. The contracting path implemented using a series of double convolutional layers, defined by a DoubleConv class, and maxpooling layers. The expansive path is implemented using transposed convolutional layers for upsampling and double convolutional layers for processing the upsampled feature maps. Skip connections from the contracting path are concatenated layer with the upsampled feature maps along the channel dimension. A final 1x1 convolutional layer is used to map the final feature map to the desired number of output channels.

5. Our dataloader tensor shape is [batch_size, height, width, no_channel] and we use permute method from tensor to convert it into [batch_size, no_channel, height, width], no_channel = 3 and height = width = 120

Conclusions

Fig. 4 is the model trained on 512 train_dataloader images on goolge colaboratory and the whole model on Institute GPU but was able to use any plotting function to show the ouput of trained model on GPU, so Fig. 4 is output for 512 images trained on adam optimizer on google colaboratory.
Output images from model and target for val_dataloader the boundary is not smooth for output images from the figure

Bibliography

1. Dabra, Ayush. *Manually Annotated High Resolution Satellite Image Dataset of Mumbai for Semantic Segmentation.* https://data.mendeley.com/datasets/xj2v49zt26/1 Mendeley, 2023.
2. Competition_Notebook *PyTorch UNet from Scratch* https://www.kaggle.com/code/seraphwedd18/pytorch-unet-from-scratch Kaggle, Internet
3. Gad, Fawzy, Ahmed . *PyGAD: An Intuitive Genetic Algorithm Python Library.* arXiv: 2106.06158 [cs.NE], 2021.

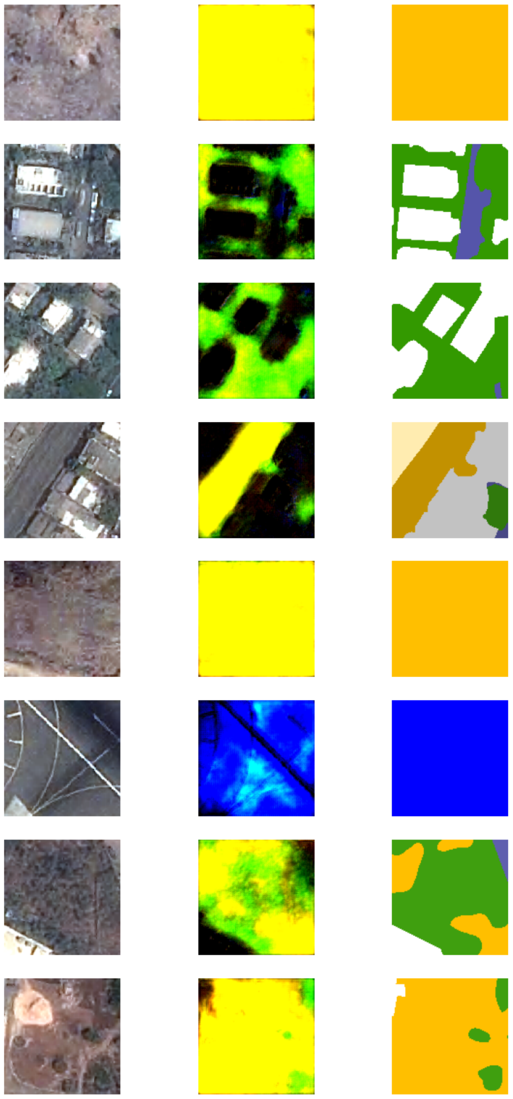

Fig. 4. Ouput after training on 512 images.