**LINEAR FEATURE EXTRACTION IN SATELLITE IMAGERY**

**PROBLEM STATEMENT:** Road Network Extraction from Satellite Imagery

**DATASET DESCRIPTION:**

* Training Data:
  + 6226 RGB Satellite Images. 1024 x 1024 px.
  + 6226 Grayscale Mask Images. 1024 x 1024 px. White (255) represents road. Black (0) represents background
* Validation Data: 1234 Satellite images.
* Test Data: 1101 Satellite images

**INFORMATION GATHERING:**

* Since training set has images and masks available. Image segmentation is the approach for this kind of problem.
* Image segmentation in computer vision is all about dividing an image into meaningful parts.
* Segmentation Types:
  + Semantic Segmentation: Identifies each pixel's category (e.g., car, person, background) but doesn't distinguish individual objects within the same category. (Think coloring all cars in an image red, all people blue.)
  + Instance Segmentation: Goes beyond semantic segmentation by identifying and segmenting each individual object instance. (Think coloring each car in an image a different color, and each person a different color.)
* For this problem statement, semantic segmentation is an appropriate choice. This is because of the following reasons:
  + We only need to differentiate between two classes: Road & Background. It will assign a label to each pixel based on its features.
  + Semantic segmentation doesn't distinguish individual roads within the image (like highways vs local streets). This is ideal for our case where we just want to identify all road pixels collectively.
* Coming to methods for segmentation. There re two types of segmentation techniques:
  + Traditional Methods: Rely on analyzing individual pixel properties like color intensity or edges to group pixels together.
  + Deep Learning Methods: Used to identify complex patterns and segment images more accurately.
* Following are some of the methods for semantic segmentation:
  + Traditional Methods: Thresholding, Region-based Segmentation, Edge-based Segmentation, Clustering-based segmentation.
  + Deep Learning Methods: Fully Convolutional Networks (FCNs), U-Net, DeepLab etc.
* For our case we’ll be focusing on U-Net which is a good an established method for segmentation.
  + Given the complexities we might have in our satellite data like vegetation, buildings, diverse landscapes, U-Net is likely to perform great.
  + U-Net’s architecture is specifically designed to address challenges for complex images.
  + It uses skip connections to preserve spatial information throughout the network. This allows it to capture both high-level semantic information.
  + With its ability to handle complex features and maintain spatial information, U-Net has the potential to achieve a higher level of accuracy compared.

**IMPLEMETATION DETAILS:**

* Import Libraries:   
  Import the necessary libraries for handling data, images, model loading, training, evaluation, and plotting.
* Load Metadata:  
  Read the metadata.csv file which contains paths to the satellite images and the information on which split (train, valid, test) each image belongs to. Filter the metadata to get the train and validation set images.
* Load Class Dictionary:  
  Read the class\_dict.csv file to map pixel values to class names.
* Preprocess the Image:  
  Define a function to read and preprocess the images. This includes resizing the image to the input size expected by the model and normalizing the pixel values. Doing a resize from (1024, 1024) to (256, 256) because of computation resource constraints.
* Binarize the Masks:  
  Define a function to binarize the mask images at a threshold of 128. Greater than 128 should be set to 255 and less than 128 should be set to 0.
* Create Data Generators:  
  Create custom data generators to load the images and masks in batches to avoid memory issues during training.
* Define the U-Net Model:  
  Define the U-Net model architecture using Keras
* Compile the Model:  
  Compile the model with an appropriate optimizer (e.g., Adam), loss function (e.g., binary crossentropy), and metrics (e.g., accuracy). Binary Cross-Entropy (BCE) is used as a loss function in binary classification tasks, and it is particularly well-suited for semantic segmentation tasks with two classes and for our case it is “Road” or “Background”.
* Train the Model:  
  Train the model using the training and validation data generators. Use callbacks such as ModelCheckpoint and EarlyStopping to save the best model and stop training when performance stops improving.
* Evaluate the Model:  
  Evaluate the model on the validation set. Calculate performance metrics such as precision, recall, and F1 score using scikit-learn.
* Generate Masks for Test Images:  
  Load the test set images. Use the trained model to generate masks for these images. Resize the generated masks back to the original size of the images.
* Display an Image and its Generated Mask:  
  Define a function to display a test image and its corresponding generated mask side by side. Use Matplotlib to create a figure and plot the test image and mask in subplots for visual comparison.

**RESULTS**

* The model achieved a training accuracy more than 96%. The result graphs are attached.
* For more details checkout the IPYNB notebook.