# 1 Proposed Methodology

Our methodology adopts a transfer learning approach, leveraging a powerful pre-trained Vision Transformer (ViT) as the encoder for a U-Net segmentation model. The framework is bifurcated into two key stages: (1) The use of a ViT encoder pre-trained via the self-supervised Masked Autoencoder (MAE) technique, which learns rich features without manual labels. (2) A downstream fine-tuning stage, where this encoder is integrated into a U-Net architecture and trained on a small, labeled satellite image dataset for the 6-class segmentation task.

## 1.1 Model Architecture

We utilize the U-Net model from the `segmentation_models_pytorch` (smp) library, which allows for a modular combination of various encoders and decoders.

- **Encoder:** We set the `encoder_name` to 'timm-vit_base_patch16_224'. This specifies a base-sized Vision Transformer with a 16x16 patch size, sourced from the PyTorch Image Models (`timm`) library.
- **Pre-trained Weights:** We initialize the encoder with 'mae_in1k' weights by setting `encoder_weights='m` This leverages the weights from a model pre-trained on the ImageNet-1K dataset using the self-supervised MAE strategy. This pre-training is ideal for our few-shot scenario, as the model has learned robust visual representations without relying on supervised labels.
- **Input/Output:** The model is configured to accept 3-channel (RGB) images (`in_channels=3`) and output a segmentation map for 6 distinct classes (`classes=6`).

The standard U-Net decoder then upsamples the contextual features from the ViT encoder to reconstruct the full-resolution segmentation map.

## 1.2 Dataset and Preprocessing

The model is trained on a custom satellite image dataset, managed by a bespoke `SatelliteDataset` class in PyTorch.

### 1.2.1 Data Loading

Input images are loaded using OpenCV, converted from BGR to RGB color space, and normalized to a [0, 1] floating-point range. The corresponding ground truth masks are loaded as grayscale images.

### 1.2.2 Image Augmentation

To improve model robustness and mitigate overfitting on the limited dataset, we apply data augmentation during training using the `albumentations` library. The applied transformations include:

- Resizing all images and masks to $224 \times 224$ pixels, matching the ViT's expected input dimensions.
- `HorizontalFlip` (p=0.5)
- `VerticalFlip` (p=0.5)

### 1.2.3 Normalization

Following augmentation, both training and validation images are normalized using the standard ImageNet mean ([0.485, 0.456, 0.406]) and standard deviation ([0.229, 0.224, 0.225]). Ground truth masks are converted to `torch.long` tensors for compatibility with the loss function.

## 1.3 Training and Implementation Details

The model is implemented in PyTorch and trained for 20 epochs with a batch size of 32.

- **Optimizer:** We use the `AdamW` optimizer, which is particularly effective for Transformer-based models, with a learning rate of $1 \times 10^{-4}$.
- **Loss Function:** As this is a multi-class segmentation task, we employ the `nn.CrossEntropyLoss` function.
- **Mixed Precision:** To accelerate training, we utilize Automatic Mixed Precision (AMP). This is implemented using PyTorch's `autocast` context manager and `GradScaler` for the backward pass, allowing for faster computation with 16-bit floats while maintaining numerical stability.

During training, the model's weights are saved after each epoch if and only if the validation Dice score improves upon the best score recorded so far. This ensures that the final saved model is the one that achieved the best performance on the unseen validation data.

## 1.4 Evaluation Metrics

To quantitatively assess the model's performance, we evaluate it on the validation set after each training epoch. We use three standard segmentation metrics, all sourced from the `segmentation_models_py` library:

- **Pixel Accuracy:** The percentage of pixels in the image that were correctly classified.
- **Dice Score (F1 Score):** A measure of overlap between the predicted and ground truth masks, calculated as $2 \times \frac{|A \cap B|}{|A|+|B|}$.
- **Intersection-over-Union (IoU):** Also known as the Jaccard index, this metric measures $\frac{|A \cap B|}{|A \cup B|}$.

The final model's performance is reported based on these metrics evaluated on the held-out test set.