**UAVs’ Landslide Detection using UNET**

**Ahsan Bilal 341480, Muhammad Haseeb 345783, Uzair Shabbir 336729,**

School of Electrical Engineering and Computer Science, NUST

**ABSTRACT** Landslide detection is a critical task for safeguarding communities and infrastructure in regions prone to geological hazards. In this project, we employ the U-Net architecture, originally designed for tumor detection, as a powerful tool for semantic segmentation in landslide-prone areas. The dataset is curated using Labelme, providing annotated images with nine distinct classes including 'background,' 'land,' 'road,' 'road\_affected,' 'house\_affected,' 'water,' 'landslide,' and 'house.' The U-Net model is trained to accurately delineate and classify these classes, contributing to an advanced understanding of the landscape and potential landslide risks. Leveraging a 23-layer convolutional network, the model integrates skip connections and transposed convolutions to preserve spatial information during both downsampling and upsampling, addressing the challenges posed by information loss in traditional architectures. The project aims to enhance the accuracy of landslide detection, providing a valuable resource for disaster preparedness and mitigation efforts.

# I. INTRODUCTION

Geological hazards, such as landslides, pose significant threats to communities and infrastructure in vulnerable regions. Accurate and timely detection of potential landslide areas is crucial for effective disaster preparedness and mitigation. In this project, we focus on employing state-of-the-art deep learning techniques for landslide detection, utilizing the U-Net architecture. Originally designed for medical image segmentation, U-Net has proven to be highly effective in various semantic segmentation tasks, making it a promising candidate for geological hazard analysis.

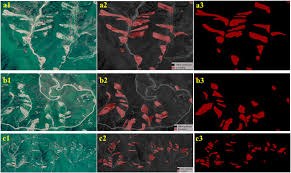
The dataset is constructed using Labelme, a tool for annotating images, providing a diverse set of images with annotations for nine distinct classes. These classes include 'background,' 'land,' 'road,' 'road\_affected,' 'house\_affected,' 'water,' 'landslide,' and 'house.' Each class represents a specific feature in the landscape, contributing to a comprehensive understanding of the terrain. The U-Net model is trained on this dataset, leveraging its 23 convolutional layers and unique architecture to capture intricate spatial information crucial for accurate segmentation.

The U-Net architecture incorporates skip connections and transposed convolutions in the decoding path, mitigating information loss during both downsampling and upsampling. These innovations enhance the model's ability to recognize

This project's objective is to develop a robust and accurate landslide detection model, contributing to the field of disaster management and environmental monitoring. By combining advanced deep learning techniques with a carefully curated dataset, we aim to provide a valuable tool for identifying potential landslide risks and informing proactive measures to mitigate their impact.

# II. Preprocessing

**Annotation**



**[[1]](#footnote-1)**

**Image Resizing:**

All input images are resized to a standardized resolution to ensure uniformity during training and inference. This resizing helps in efficient memory utilization and accelerates the training process.

**Normalization:**

Pixel values in the images are normalized to a specific range (e.g., [0, 1]) to facilitate convergence during training. Normalization helps in stabilizing the learning process and improving the model's performance.

**Data Augmentation:**

To increase the diversity of the training dataset and enhance the model's generalization ability, data augmentation techniques such as rotation, flipping, and zooming are applied. This ensures that the model learns to handle variations in the input data.

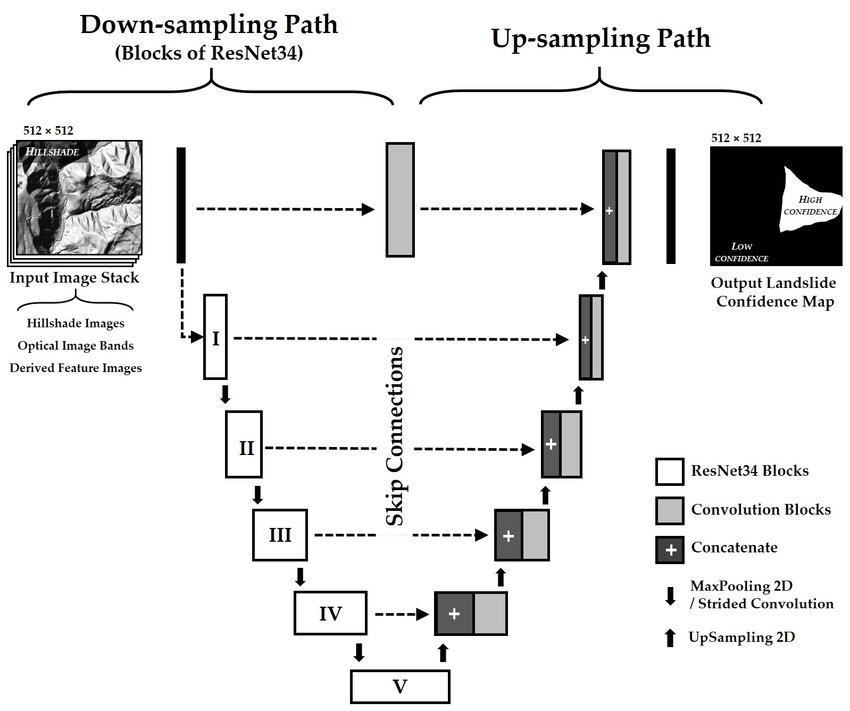
**Label Encoding:**

Semantic segmentation involves classifying each pixel in an image. Therefore, the labels generated using Labelme are encoded to match the number of output classes in the model. This encoding facilitates the training process by converting categorical labels into a format compatible with the model's output layer.

**Handling Class Imbalance:**

If there is a significant class imbalance in the dataset, techniques such as class weighting or oversampling/undersampling may be applied to ensure that the model does not favor the majority class and can effectively learn from minority classes.

# III. Model Building



**UNET**

U-Net, named for its U-shape, was originally created in 2015 for tumor detection, but in the years since has become a very popular choice for other semantic segmentation tasks.

U-Net builds on a previous architecture called the Fully Convolutional Network, or FCN, which replaces the dense layers found in a typical CNN with a transposed convolution layer that upsamples the feature map back to the size of the original input image, while preserving the spatial information. This is necessary because the dense layers destroy spatial information (the "where" of the image), which is an essential part of image segmentation tasks. An added bonus of using transpose convolutions is that the input size no longer needs to be fixed, as it does when dense layers are used.

Unfortunately, the final feature layer of the FCN suffers from information loss due to downsampling too much. It then becomes difficult to upsample after so much information has been lost, causing an output that looks rough.

U-Net improves on the FCN, using a somewhat similar design, but differing in some important ways. Instead of one transposed convolution at the end of the network, it uses a matching number of convolutions for downsampling the input image to a feature map, and transposed convolutions for upsampling those maps back up to the original input image size. It also adds skip connections, to retain information that would otherwise become lost during encoding. Skip connections send information to every upsampling layer in the decoder from the corresponding downsampling layer in the encoder, capturing finer information while also keeping computation low. These help prevent information loss, as well as model overfitting.

**Encoder:**

The encoder comprises a stack of various convolutional blocks, with each conv\_block() consisting of two Conv2D layers with ReLU activations. Some conv\_blocks apply Dropout and MaxPooling2D, specifically in the last two blocks of the downsampling.

The function returns two tensors:

next\_layer: This tensor proceeds to the next block.

skip\_connection: This tensor feeds into the corresponding decoding block.

If max\_pooling=True, next\_layer is the output of the MaxPooling2D layer, while skip\_connection is the output of the preceding layer (Conv2D or Dropout, depending on the case). If max\_pooling is False, both results are identical.

**Decoder:**

In the decoder, or upsampling block, the objective is to upsample the features back to the original image size. This involves taking the output of the corresponding encoder block and concatenating it before feeding it to the next decoder block. The upsampling block consists of two new components: up and merge. The up operation is performed by the Conv2DTranspose layer, which essentially acts as the inverse of the Conv2D layer. Additionally, skip connections are established through the merge operation.

E

**First Half of the Model:**

Commence with a convolutional block that accepts the model inputs and specifies the number of filters.

Subsequently, link the initial output element of each block to the subsequent convolutional block's input.

Increment the number of filters twofold at each progressive step.

Introduce a dropout of 0.3 starting from conv\_block4.

For the concluding conv\_block, maintain a dropout of 0.3 and deactivate max pooling. (At this stage, set n\_filters=n\_filters\*16)

**Second Half of the Model:**

Proceed by halving the number of filters at each stage.

Utilize cblock5 as the expansive input and cblock4 as the contractive input, employing n\_filters \* 8. This constitutes the bottleneck layer.

Connect the previous block's output as the expansive input and the corresponding contractive block output.

It is imperative to utilize the second element of the contractive block before the max pooling layer.

Introduce conv9 as a Conv2D layer featuring ReLU activation, He normal initializer, and "same" padding.

Conclusively, implement conv10 as a Conv2D layer with the number of classes as filters, a kernel size of 1, and "same" padding. The output of conv10 serves as the model's output.

# IV. Hyperparameters

**Learning Rate:**

The learning rate determines the step size during the optimization process. A suitable learning rate is crucial for model convergence. Commonly used initial learning rates range from 0.1 to 0.0001, and learning rate schedules may be employed to adaptively adjust the learning rate during training.

**Batch Size:**

The batch size represents the number of samples used in each iteration during training. A balance must be struck to ensure that the batch size is large enough for computational efficiency but not so large that it exhausts memory resources. Typical values range from 8 to 64.

**Number of Epochs:**

An epoch is one complete pass through the entire training dataset. The number of epochs determines how many times the model sees the entire dataset during training. This parameter is chosen based on the convergence behavior observed during training, often through monitoring validation performance.

**Optimizer:**

The choice of optimizer affects how the model parameters are updated during training. Common optimizers include Adam, SGD (Stochastic Gradient Descent), and RMSprop. The selection depends on the specific characteristics of the dataset and the problem at hand.

**Loss Function:**

For semantic segmentation tasks, a suitable loss function is crucial. Cross-entropy loss is commonly used for multi-class segmentation problems. Other variations, such as Dice loss or focal loss, may be considered based on the specific requirements of the project.

In semantic segmentation, you need as many masks as you have object classes. In the dataset you're using, each pixel in every mask has been assigned a single integer probability that it belongs to a certain class, from 0 to num\_classes-1. The correct class is the layer with the higher probability.

This is different from categorical crossentropy, where the labels should be one-hot encoded (just 0s and 1s). Here, you'll use sparse categorical crossentropy as your loss function, to perform pixel-wise multiclass prediction. Sparse categorical crossentropy is more efficient than other loss functions when you're dealing with lots of classes.

**Model Architecture and Depth:**

The U-Net architecture's depth, including the number of convolutional layers and the configuration of the contracting and expanding paths, plays a significant role. The choice of this architecture is influenced by the complexity of the segmentation task and the available computational resources.

**Skip Connection Strategy:**

The strategy for implementing skip connections, which preserve spatial information, can impact model performance. Ensuring effective information flow between encoder and decoder through skip connections is essential for maintaining fine details in the segmentation results.

**Dropout and Regularization:**

Techniques such as dropout and regularization may be employed to prevent overfitting, especially when dealing with limited training data. These hyperparameters control the degree of regularization applied to the model.

**Transposed Convolution Parameters:**

Hyperparameters related to transposed convolutions, such as kernel size and stride, influence the upsampling process in the decoding path. Tuning these parameters ensures that the model effectively reconstructs spatial details during upsampling.

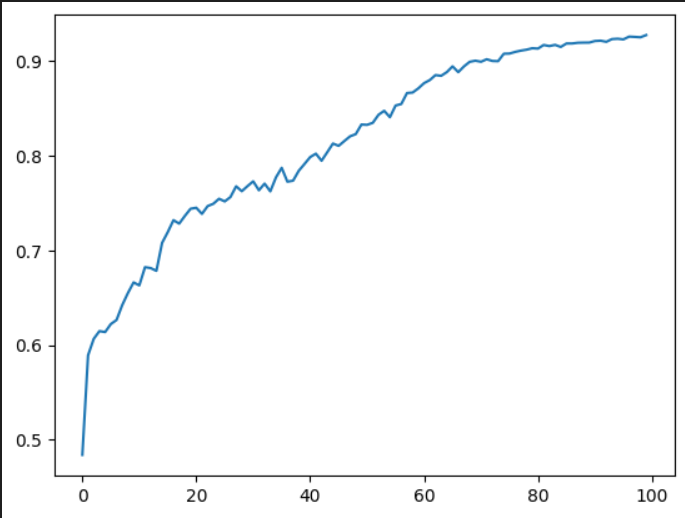
**Crop Function Parameters:**

Parameters related to the crop function, which handles skip connections between contracting and expanding paths, need to be configured appropriately to avoid inconsistencies in feature maps.

**Input Size:**

The input size of the images fed into the model needs to be chosen based on the available computational resources and the desired trade-off between spatial resolution and memory requirements.

**Pix2Pix Model Accuracy**



**FIGURE 1.Accuracy curve of UNET Modle**

# VII. Project Prediction & Streamlit UI

A diagram of a network

Description automatically generated

# VIII. CONCLUSION

In conclusion, this project demonstrates the efficacy of the U-Net architecture for landslide detection in diverse landscapes. The incorporation of skip connections and transposed convolutions addresses the challenges posed by information loss during image processing, resulting in a model capable of accurate semantic segmentation. The dataset, annotated using Labelme, facilitates the training of the model across nine distinct classes, offering a comprehensive understanding of the terrain.

The trained U-Net model not only showcases its adaptability beyond its original medical imaging applications but also highlights its potential as a versatile tool for geological hazard analysis. The ability to accurately delineate and classify features in landslide-prone areas contributes to improved disaster preparedness and response strategies.

Moving forward, this work opens avenues for further research in the application of deep learning models to environmental monitoring and disaster management. The insights gained from this project can be instrumental in developing advanced systems for early detection and prediction of geological hazards, ultimately fostering safer and more resilient communities in vulnerable regions.

9

1. [Frontier](https://www.frontiersin.org/articles/10.3389/feart.2023.1248340/full) [↑](#footnote-ref-1)