Foundations of Deep Learning - Assignment 2 - Hurricane Harvey

Synthetic and real data for human aid and disaster relief for residential area flooding scenarios.

Pallav SAHU
pallav.sahu@student-cs.fr

Arindam Roy
arindam.roy@student-cs.fr

ABSTRACT

The purpose of the assignment was to implement and solve a remote sensing problem using recent deep learning techniques. In particular, the main objective of the challenge was to segment images acquired by a small UAV (s-UAV) in the area of Houston, Texas after Hurricane Harvey. There were twenty seven categories of segments such as roof, trees, pools etc.

For this assignment, this group of two students was formed and worked together in order to increase the performance of the baseline model. The task was to design and implement a deep learning model in order to perform the automatic segmentation of these images. The model was trained using the train images, which contained pixel-wise annotations. The performance of the submitted file was ranked accordingly using the dice score.

After implementing multiple architectures for the semantic segmentation challenge, this group was able to get a maximum score of 75.09, overperforming the baseline by 6.97 using DeepLabV3 with Resnet-101 backbone.

KEYWORDS

Semantic Segmentation; Computer Vision; Aerial Images; Pytorch; Neural Networks; U-net; Encoder-Decoder Architecture; Deep Learning; DeepLabV3

DATA DESCRIPTION

Table 1: Dataset Details

Detail	**Count**
Number of training images	299
Number of test images	75
Shape of images	~ 3000pi * 4000pi * 3 channels
Shape of masks	~ 3000pi * 4000pi * 1 channel
Image format	Tag Image File Format
Mask format	Portable Network Graphics Format

PRE-PROCESSING

There were multiple methods attempted for pre-processing the data. Following are the methods:

Final Image Resize

All of the fit models were fed in with the either of the image sizes 512*512, 512 * 384, 576*448. This was done. With the inter_nearest method available in the cv2 library, the images were resized to various different shapes. The objective of this activity was to train the model on different aspect ratios of the images which was lower than the high resolution provided, so that model computation can be efficient.

Augmentations

A combination of these operations were randomly repeated on the train and validation set images and masks.

a. **Horizontal flip**: The images were flipped on the horizontal axis for better model training.
b. **Vertical flip**: The images were flipped on the vertical axis for better model training.
c. **RandomBrightnessContras**t: Brightness and contrast of the images were varied for model training.

Custom patching

A custom patch was attempted where all images were deducted to the closest square from the horizontal axis in 512*512 pi. The same process was repeated on the horizontally flipped end. The final training images were reshaped to this size and the predicted masks were generated of the reduced size. This mask was needed to be resized to the original image size.

METHODOLOGY

This project group has tried a lot of different semantic segmentation architectures, out of which the below 3 performed the best. A brief description of each method follows:

DeepLab V3

The best performing model for this project team was DeepLab V3. DeepLab V3 consists of two main parts: an encoder network, which is based on a pre-trained CNN (such as ResNet or Xception), and a decoder network, which is added on top of the encoder to refine the segmentation results.

The encoder network extracts a feature map from the input image, which is then passed through a series of Atrous convolutions (also called dilated convolutions) to increase the field of view of the model and capture context at multiple scales. The Atrous convolutions are performed at different rates (or dilation rates) to increase the receptive field of the model.

The decoder network is then added on top of the encoder to refine the segmentation results. It typically consists of a series of up-sampling layers, which increase the resolution of the feature map, and convolutional layers, which fine-tune the segmentation results. Finally, a softmax layer is used to assign semantic labels to each pixel in the image.

Feature Pyramid Network

Feature Pyramid Network (FPN) is a type of architecture used in semantic image segmentation tasks. The main idea behind FPNs is to combine the high-resolution features from a shallow layer of a CNN with the semantic information from a deeper layer to improve the accuracy of the segmentation results. The architecture of FPN typically consists of three main components: a backbone network, a pyramid of feature maps, and a head network.

The backbone network is a pre-trained CNN, such as ResNet or VGG, that extracts feature from the input image. The feature maps from different layers of the backbone network, with varying resolutions, are then passed through a pyramid of feature maps. The pyramid of feature maps is a set of top-down and lateral connections that combine the features from different layers of the backbone network. The top-down connections are used to up-sample the features from deeper layers and provide high-resolution information. The lateral connections are used to combine the features from shallower layers and provide semantic information.

Finally, the head network is added on top of the pyramid of feature maps to predict the final segmentation results. It typically consists of a series of convolutional layers and a final softmax layer that assigns semantic labels to each pixel in the image. The FPN architecture has been demonstrated to improve the accuracy of semantic segmentation models, by combining the advantages of both high-resolution features and semantic information from different layers of a CNN, thus achieving state-of-the-art results in several benchmarks.

U-Nets

This project team also had significant success with U Net models. U-Net is a type of convolutional neural network architecture that is commonly used for image segmentation tasks. The architecture consists of a contracting path, where the spatial dimensions of the feature maps are reduced, and an expansive path, where the spatial dimensions are increased. The contracting path is composed of multiple layers of convolutional and max pooling operations, while the expansive path is composed of multiple layers of transposed convolutional (also known as "deconvolutional") layers. The name "U-Net" comes from the shape of the architecture, which resembles a "U" shape.

One of the key features of U-Net is the use of skip connections, which connect the output of a layer in the contracting path to the corresponding layer in the expansive path. This allows the network to preserve information from the contracting path as it increases the spatial dimensions of the feature maps in the expansive path.

U-Net was first introduced by Olaf Ronneberger, Philipp Fischer, and Thomas Brox in their 2015 paper "U-Net: Convolutional Networks for Biomedical Image Segmentation." The paper describes the application of U-Net to biomedical image segmentation tasks, such as cell segmentation in microscopy images, and demonstrates the effectiveness of the architecture on a variety of datasets.

In this project, we have utilised the library of “segmentation_models_pytorch” distributed under MIT licence. This library allows users to download pretrained segmentation models with multiple architectures and backbones. We have attempted to use multiple different backbones for the u-net architecture.

The models were trained using cross entropy loss, available in the pytorch framework. This loss was applied on the Intersection over Union score of the classes. Intersection over Union (IoU) loss is a loss function commonly used in object detection tasks in deep learning. It is used to evaluate the similarity between two sets of data, typically the predicted bounding box coordinates and the ground truth bounding box coordinates. The IoU loss is calculated as the ratio of the intersection of the predicted and ground truth bounding boxes to the union of the predicted and ground truth bounding boxes. A value of 1 indicates a perfect match, while a value of 0 indicates no overlap between the predicted and ground truth bounding boxes. The optimizer used for all models was

ADAMW optimizer, with minimum number of epochs being 15 and maximum number of epochs being 30. There was a weight decay applied to all optimizers.

The varying u-net backbones fit to the data have yielded in multiple different results. The following table provides results of the various trainings.

POST-PROCESSING

For the submission of predicted masks following steps were performed:

1. Final Prediction Masks

All of the final 75 test images were fed into the trained deep CNN model to output greyscale masks with 27 output classes. As, each output class contained logits for a label, we did “argmax” to bring the prediction to one layer for a pixel. The images were then resized to actual image size.

2. Final Mask Resize

As all of the fit U-net models were fed in images with the height of 512 pixels, the dimension of the output mask was changed to 512 * 384. We used cv2 library’s resize command using “inter_nearest” interpolation to bring the mask to target size of the respective images.

3. Submission TAR

For submission of the images to the geo-engine for generating the prediction scores, we created a function to write each prediction mask to a “.png” greyscale image with its name being identical to the test image name and then were tarred to output a final tar zip file.

RESULTS

For the results of training, we are displaying here the curves for model losses, accuracy and Iou as we train and validate (90-10 split of the 299 images with masks and keeping a threshold over classes in the training masks) over the epochs for our best performing model with DeepLabV3 architecture using Resnet-101 backbone for encoder with the input image and mask size as 512*512 pixels. We can observe that the optimal epoch is epoch-11 (10 in the graph due to zero-indexing) on the basis of train and validation evaluation metrics, so we the model with weights learned at this epoch for our predictions.

Figure 1: Cross Entropy loss vs Epochs

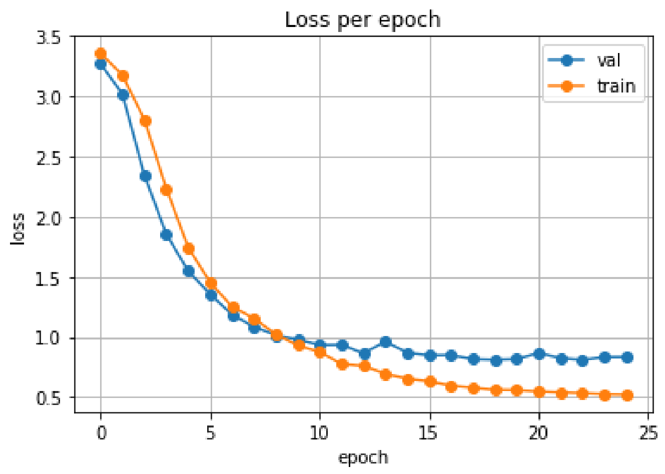


Figure 2: Pixel-wise Accuracy vs Epochs

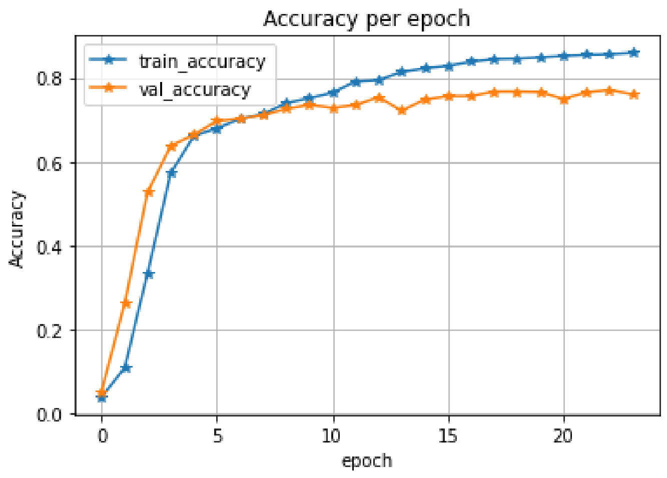
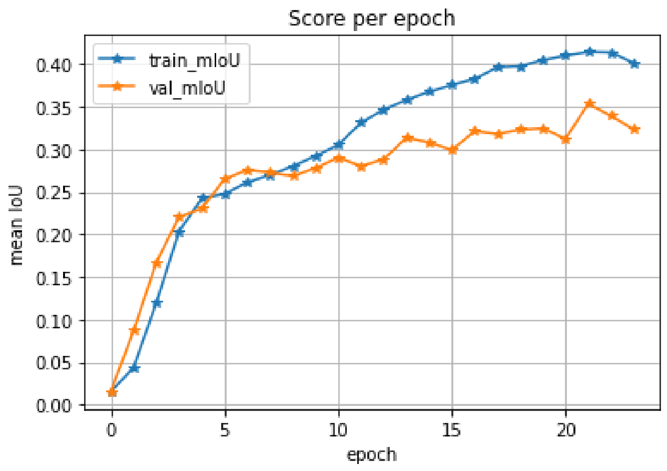


Figure 3: IoU coefficient vs Epochs



We tried various architectures with different mentioned parameters to train our model for predicting segmentation masks in the decreasing order of Final scores on the platform. Below are our 5-best performing models and respective parameters.

Model; Backbone	**Backbone Parameter Count**	**Input Image Resize**	**Transforms applied***	**Loss function used**	**Learning Rate; Weight Decay**	**Optimal Epoch**	**Geo-AI Platform Score**
DeepLabV3; Resnet-101; With sampled classes	42M	512*512	H, V, RBC	Unweighted Cross entropy	1e-4; 1e-4	21	75.09
DeepLabV3; Resnet-101	42M	512*512	H, V, RBC	Weighted Cross entropy	1e-4; 1e-4	11	74.02
FPN; Resnet-152	58M	576*484	H, V, RBC	Weighted Cross entropy	1e-4; 1e-4	6	73.97
PSP; Resnet-152	58M	512*512	H, V	Unweighted Cross entropy	1e-4; 1e-4	17	72.90
U-Net; Resnet-101	42M	512*512	H, V, RBC	Weighted Cross entropy	1e-4; 1e-4	11	71.37

* H: horizontal flipping, V: Vertical Flipping, RBC: Random Brightness Contrast || ** Optimizer: AdamW; Loss function: Cross Entropy Loss

CONCLUSIONS AND IMPROVEMENTS

Overall, it was a very interesting and challenging task of predicting grayscale masks for the high-resolution drone images. We could have tried further data augmentations with different image pixel sizes.

We noticed that the weighted cross entropy loss function gave similar results as unweighted, so the either the masks were not properly segmented, or the imbalance of the class was not reflected properly in the weights across our validation and training datasets.

We always tried with random shuffling on while loading the dataset, it might have been possible that if we turned the shuffle off completely, we could have received better accuracy and Iou, as we observed eventually that the drone was flying and capturing images in a particular sequence for most images.

In conclusion, we maybe could have improved the model performance by more specific custom loss function, a better image processing routine and using transfer learning method with a bit more efficiency.