Foundation of Deep Learning - Project Report

HONG Xiaoyan, BROWN Chara Vega, JOUAIDI Marouan

January 2023

1 Introduction

1.1 Background

The project aimed to use advanced deep learning methods for semantic segmentation to identify and locate residential properties and neighborhood assets after Hurricane Harvey, one of the most costly natural disasters in US history, by segmenting pictures taken by a drone in Houston, Texas. The goal was to identify and separate 26 different types of objects in the images using a competitive and collaborative deep learning approach.

1.2 Datasets

Provided to us was a data set containing 374 UAV images of the neighborhoods in Houston, and for each picture, a dense segmentation mask has been created to separate each class of object in the capture.

There has also been a second synthetic data set containing 2093 images and 156,933 annotations that need to be converted into masks, which can be used to do extra pre-training for the model. Considering the time and GPU resource limitation, we didn't use it this time.

2 Methodology

A set of steps were put in place using Python to carry out data pre-processing, model training and submission adjustment. The model is constructed with PyTorch and based on U-Net or DeepLabV3+ model.

2.1 Data pre-processing

2.1.1 Obtaining the data

The gsutil tool was utilized to download the train images, train masks and test images, and then we import them into the python environment for use as input for training and prediction. The original images have different shapes, such as 3000x4000 or 3072x4096 pixels.

2.1.2 Data augmentation

In order to reduce over-fitting, improve generalization and robustness of our model, we used data augmentation techniques to increase the size of the training and validation dataset, for example:

RGB shift: This method randomly shifts the values of the red, green and blue channels of an image.

Gaussian noise: This method adds random noise to an image, simulating sensor noise or other sources of image degradation.

Gaussian blur: This method blurs an image using a Gaussian filter. This can simulate the effect of out-of-focus images or images captured through a haze.

Brightness and contrast: This method adjusts the brightness and contrast of an image. This can simulate different lighting conditions or camera settings.

Random crop: This method randomly crops an image to a specified size. This can simulate different object scales or view points.

Flip: This method flips an image horizontally or vertically. This can simulate different view angles.

Rotation: This method rotates an image by a random angle.

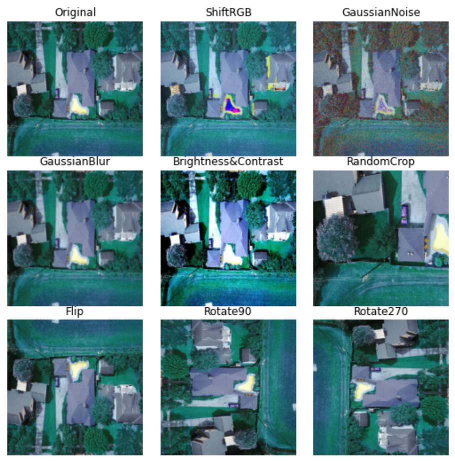

Figure 1: The visualization of data augmentation methods

The techniques above allowed us to use more images as input for training our model. We also made sure that all the images were the same size (512x512) so that the model could process them efficiently. We also split the training data into two sets, one for training and one for evaluating the performance of the model, to test if the model is over-fitting. Lastly, we made sure that the test data was also the correct size and had been also normalized so that the model could make accurate predictions with it.

2.2 Model Selection

2.2.1 U-Net & ResNet18

We decided to use a U-Net architecture with a ResNet18(pretrained on ImageNet) encoder for image segmentation at first. Compared to CNN which is suitable for classification tasks, the U-Net architecture is known for its ability to perform pixel-to-pixel predictions and its U-shape structure that allows for efficient information flow between different levels of resolution. The ResNet18 encoder, which is a version of the ResNet architecture with 18 layers, is added to the model to improve its accuracy and efficiency. The ResNet18 encoder extracts features from the input image, which are then used by the decoder part of the U-Net to produce the segmentation map. In this approach, the ResNet18 encoder is used to replace the downsampling part of the U-Net architecture. The following image illustrates the basic structure of U-Net.

The initial layers of the convolutional networks process the input image and extract features from it. The later layers after the "bridge" section of the architecture, increase the size of the feature maps and produce a segmentation map of the same size as the original image. This map segments the image into 26 different classes of objects. However, this model did not yield satisfactory results, even after hyperparameter tuning, it can only reach the baseline score. Therefore, we decided to explore other type of semantic segmentation architectures.

2.2.2 Deeplabv3+ & MobileNetV2

Although U-Net is good at handling pixel to pixel prediction tasks. The poor quality of the mask

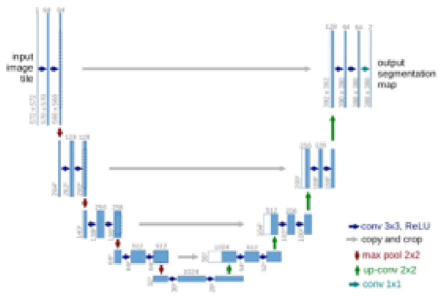

Figure 2: Unet layer composition

determines that we need to use a deeper neural network. Therefore, we turn to another robust model DeepLabv3+.

The DeepLab architecture, specifically its extension, DeepLabv3 and DeepLabv3+. These architectures are used for semantic segmentation and are built on top of a CNN architecture that uses atrous convolutions to increase the resolution of the output feature maps. DeepLabv3 uses a pre-trained ResNet as an encoder and an Atrous Spatial Pyramid Pooling (ASPP) module to capture multi-scale context information, followed by a decoder module to produce the final segmentation map.

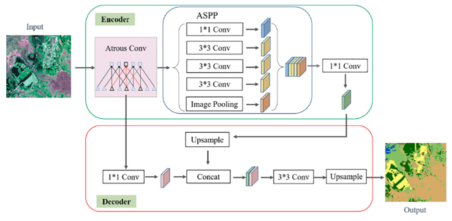

Figure 3: Illustration of DeepLabV3+ Model

To enhance the performance of our model and save computing resources. We used the pretrained MobileNetV2, which is a lightweight efficient architecture that is designed for mobile and embedded devices, as the backbone for our DeepLabV3+ model.

3 Results

After the model training, we get the masks with uniform format, we need to resize them back to the original size before submitting to the platform, then we are able to acquire the score of the model. Here is the hyper-parameters for the best result of each model we tried:

Hyper-parameters	UNet	DeepLabV3+
Training epochs	20	100
Training batches	3	8(frozen),4(unfrozen)
Learning rate	0.0001	adaptive
Weight decay	None	1e-4

3.1 U-Net & ResNet18

For Unet, we didn't set weight decay rate, though we saw over-fitting problem from the line graph of training and validation losses. The reason is that it is already not easy for the model to learn as the quality of the training masks is poor, and the U-Net model we applied here has rather shallow layers, so we didn't add the weight decay here.

With the hyperparameters above, we got the score around 62. Increase the data augmentation and batch size can improve the result of the model to around 68. But we want to save the limited GPU resources, thus switch to more effective model directly.

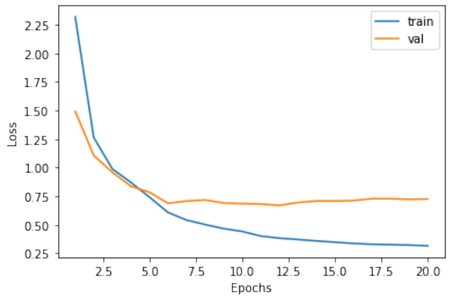

Figure 4: The trend of training and validation losses by epochs - U-Net

3.2 DeepLabV3+ & MobileNetV2

The model of DeepLabV3+ performs better compared to the U-Net model, thus proved that we need deeper achitecture to compensate for the poor quality of the masks. The model we used also have comprehensive hyperparameter setting, including much larger learning epochs, different batches for frozen and unfrozen layers, adaptive learning rate.

The reason to distinguish frozen and unfrozen phase is that it allows the model to continue to use general features while updating the task-specific features. Adaptive learning rate is also a useful trick. As we learn more, the learning rate will decrease like human learning. Also too large or too small learning rate will make the model converge to a sub-optimal solution too fast or difficult to converge within the training epochs.

We got 75.53 for the above parameters. Confined to the GPU and time limit, we didn't have the chance to further tune this model. We will discuss more possibilities in the following further thoughts part.

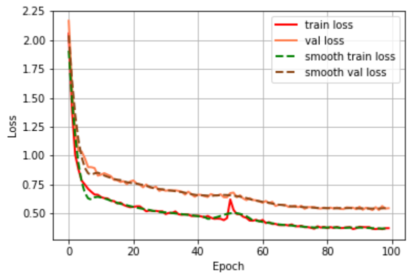

Figure 5: The trend of training and validation losses by epochs - DeepLabV3+

4 Further thoughts

4.1 Imbalanced variables

As we can see from figure 6, the distribution of the 26 classes is not balanced, the "Sports Complex / Arena class" only has 2 training images. To resolve this issue, either we can delete the least classes simply or use the Generative Adversarial Network(GAN) and the synthetic dataset to generate more images for the minor classes.

4.2 Model tuning

As mentioned above, if the GPU resources allow, we can try larger batch size to reflect the whole dataset better and do more the data augmentation. We can also try other backbone architecture like ResNet101(with U-Net), Xception(with DeepLabV3+), try more powerful semantic segmentation models to explore different possibilites, such as YOlOv5.

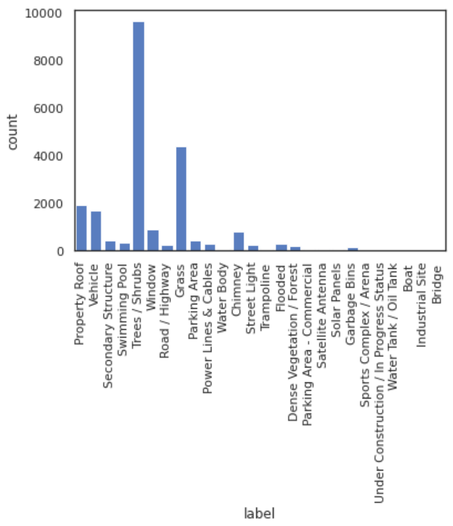

Figure 6: The count number of different classes

5 References

[1] Sandler, H. M., Zhu, A., Zhmoginov, M., & Chen, L. (n.d.). Mobilenetv2: Inverted residuals and linear bottlenecks. . In Proceedings of the IEEE conference on computer vision and pattrern recognition, 4510-4520.

[2] Chen, L.-C., Papandreou, G., Schroff, F., & Adam, H. (n.d.). Rethinkking Atrous Convolution for Sematic Image Segmentation. arXiv preprint arXiv:1706.05587.

[3] Sampath, V., Maurtua, I., Aguilar Martín, J.J. et al. A survey on generative adversarial networks for imbalance problems in computer vision tasks. J Big Data 8, 27 (2021). https://doi.org/10.1186/s40537-021-00414-0

[4] usuyama/pytorch-unet: https://github.com/usuyama/pytorch-unet

[5] bubbliiiing/deeplabv3-plus-pytorch: https://github.com/bubbliiiing/deeplabv3-plus-pytorch