Semantic Segmentation on Aerial Drone Images using U-Net

Star Chen, Muyang Xu, William Zhang

May 14 2021

Abstract

Our project addresses the developing automation in aerial drone piloting and image capturing. While current automation is not yet ripe, with the manual operation still needed, we decided to run image semantic segmentation on aerial drone-captured images to automate and refine the object detection in drone operation. This project first uses U-Net and then advances to Mobile-Unet as the primary machine learning approach towards solving the problem. Our model yields satisfactory results with the implementation of the high-performance model (in both computational time and accuracy) and by fine-tuning the model and self-creating testing datasets. The full implementation (based on PyTorch) and the trained networks are available at repository, please refer to its README.md file for details.
`https://github.com/MstXy/ML_final_project`

1 Introduction

Aerial Drone or uncrewed aerial vehicle (UAV) has recently gained increased popularity in civil use. While current manufacturers incorporate automation to aerial drones, they are relatively lower and are not yet ripe. To further simplify the drone operation and improve customers' experience, we deem that further automation in drone piloting and image capturing is needed. While image semantic segmentation is relatively ripe, most are implemented for daily scenes shot in usual view directions, and few are on the top-down birds-eye view images.

Therefore, we intend to implement the image semantic segmentation on images shot by aerial drone, where given images, the algorithm is able to identify different objects in those images and apply colored masks to objects. Built on such object detection, previously proposed improvements in drone automation could be achieved by atomizing the process of identifying and avoiding obstacles and identifying key objects, and locking camera views during flight.

In this project, we implemented image semantic segmentation on the "Semantic Drone Dataset" found on the Institute of Computer Graphics and Vision.

We first get a preliminary result using a self-implemented U-Net proposed in "U-Net: Convolutional Networks for Biomedical Image Segmentation"[8] with modifications on the padding settings of the double convolutional layers and up-convolutional layers to preserve the shape of the images. Later, we advanced to using Mobile-Unet proposed in "Mobile-Unet: An efficient convolutional neural network for fabric defect detection"[5] for a better prediction result. Comparisons between the two models' performances were made based on their model structures, and we conclude that ResNet based Mobile-Unet outperformed the VGGNet based U-Net both in computation speed and model accuracy due to the mobilev2 encoding originated in Sandler et al.'s "MobileNetV2: Inverted Residuals and Linear Bottlenecks"[4] and the skip connections in the inverted residual blocks in the Mobile-Unet model.

Our model evaluation is based on self-implemented pixel accuracy and the mean score of intersection over union (mIoU). To further test the model, a self-generated evaluation set and test set were created using images on unsplash.com, with further preprocessing and tagging using Semantic Segmentation Editor contributed by Hitachi Automotive and Industry Lab. Generally, the model yields good results, and further reflections were made based on practical and theoretical reasons.

The general structure of the paper is as follows. In Section 2, we will be briefly introducing the dataset and our preprocessing strategies. Section 3 will be about the model, evaluation metrics we used, and the training process. In Section 4, we will analyze our experimental results, reflections, and future works will be introduced in Section 5.

2 Dataset

2.1 Dataset Features

This project uses the "Semantic Drone Dataset" found on the Institute of Computer Graphics and Vision for the dataset. The dataset consists of 400 images of different houses and streets from nadir (bird's eye) view acquired at an altitude of 5 to 30 meters above the ground [3]. It is created initially to understand urban scenes for increasing the safety of autonomous drone flight and landing procedures [3]. Since our project intends to do object detection for drone images in civil use, mainly in neighborhoods, this dataset is ideal for achieving such a task. The dataset is detailed into three parts [2]: Original full-color images of high resolution (6000×4000 pixels), RGB masks for each of the images, and a CSV file containing the labels' information each of the RGB colors represents.

2.2 Dataset Processing

In terms of pre-processing stage, an image mask must contain the class's code for each object, and each pixel in the image should be represented with a class number in the mask. So, our data pre-processing would mainly involve the transformation of the RGB masks into gray-scale masks. In order to better hold the training process and let the data fit our model, we resize the original images as well as the gray-scale masks into lower resolution (768×512 pixels). Furthermore, we manually split the 400-image dataset with the ratio being $0.75 : 0.15 : 0.1$ ($train : val : test$). We also normalize the data by setting the RGB mean value as $[0.485, 0.456, 0.406]$ and the standard deviation as $[0.229, 0.224, 0.225]$, which is calculated based on millions of images from the ImageNet dataset.

3 Solution

3.1 U-Net

The reason to choose U-Net based model is that it claims to have a high performance upon a limited dataset. Also, in our dataset, the aerial drone images are shot from a bird's eye view, which is different from what is usually seen. While U-Net is applied initially on the biomedical image shot from a microscope, we found that an analogy could be drawn: both image sets are a top-down view of the objects, which might make U-Net more useful in the aerial image segmentation task.

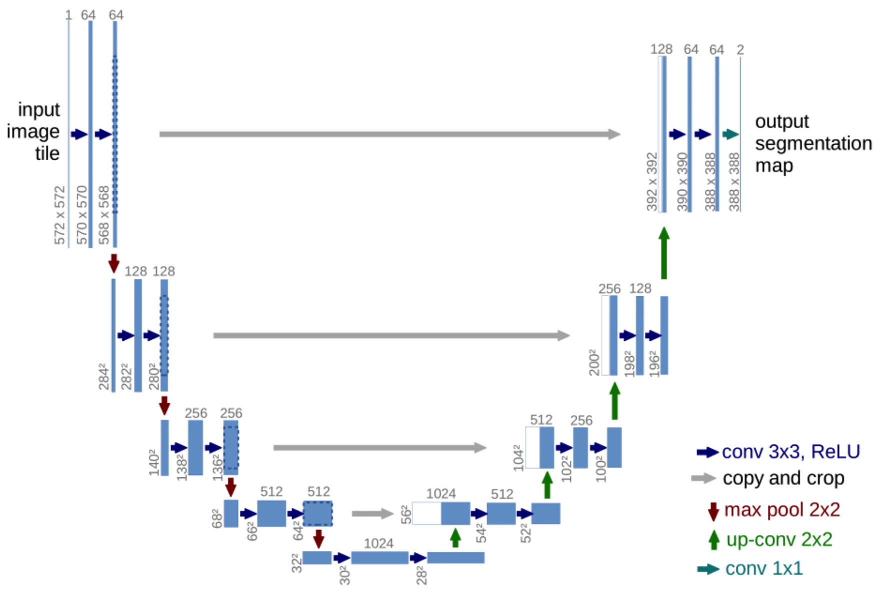


Figure 1: U-Net architecture (Olaf Ronneberger, Philipp Fischer, and Thomas Brox 2015)

In the U-Net model, a u-shaped architecture is used [Fig. 1], with the left side using double convolutional layers of 3×3 kernel size and consecutive max-pooling layers to extract features of the images. The right side using 2D transposed convolution to consecutive to identical double convolutional layers to increase the resolution of the output. To ensure the up-convolution accuracy, the same level of output from the previous downsampling was copied and concatenated to the up-convolution layers. As the U-Net in the demonstrated figure uses zero paddings to the double convolutional layers, the image size shrinks from 572×572 to 388×388. Meanwhile, in the coping process of up-sampling, cropping the output from previous down sampling to the exact size of the up convolution output is involved. Since we do not want a loss of border pixels for our model, paddings in the double-convolutional layers and optional padding in the copy layer are implemented to maintain the output's shape and avoid further reshaping validation set.

That means, if we input an image tensor with shape: $batch\ size \times color\ channel \times height \times weight$, then U-Net model will output a tensor with shape: $batch\ size \times classes \times height \times weight$, with each class a separate channel.

3.2 First and Second Attempts

As image semantic segmentation is essentially classifying every pixel, the loss function we use is Cross-Entropy Loss. It is performed pixel-wisely, which examines each pixel individually, comparing the class predictions (depth-wise pixel vector) to our one-hot encoded target vector:

$$-\sum_{classes} y_{true} \log(y_{pred})$$

We then train our self-implemented U-Net on our semantic drone dataset, with first attempts based on a small set of the validation set. The initial parameters are set as such: $batch\ size = 1, epoch = 15, optimizer = Adam$ with no weight decay. The resulting accuracy (pixel accuracy will be introduced in Section 4) is oscillating badly [Fig. 2], yielding unsatisfactory results.

Then, we decided to add regularization terms, increase the batch size and change the optimizer to more advanced AdamW, which yields a slightly better result at the earlier half of the epochs and still oscillating accuracy at later epochs [Fig. 3]

We concluded that the learning rate is too low at the beginning, causing updates not aggressive enough and too high at the end, causing the model's failure to converge. Thus, we introduced the OneCycleLR for the learning rate scheduler, which Leslie N. Smith and Nicholay Topin propose. OneCycleLR adjusts the learning rate from an initial learning rate to some maximum learning rate. Then, from that maximum learning rate down to some minimum learning rate much lower than the initial learning rate, a sound choice for our need [3]. This gives a much faster convergence rate and more stable accuracy in later epochs [Fig. 4]

However, when we extend the model to the full validation set using the same parameters, the validation accuracy is not ideal at all, yielding an accuracy of

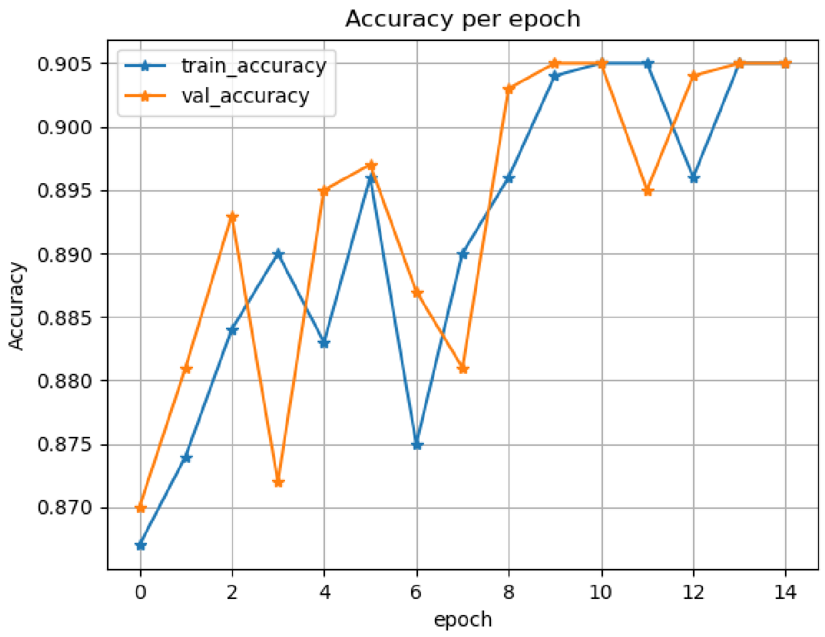


Figure 2: Pixel accuracy of the very first attempt, with no weight decay or learning rate scheduler.

only 0.7 in 15 epochs, with a slope of convergence no further than the boundary of 0.75 [Fig. 5]. After another few tweaking of the parameters, the model performance remains identical. Thus, we conclude that the U-Net structure would need some significant improvement if we wish to reach relatively high performance. Furthermore, that is the reason we choose an improved version of U-Net, the Mobile-Unet.

3.3 Mobile-Unet

In order to improve the overall performance of our model, we determine to utilize Mobile-Unet structure for the training process. To evaluate the improvement in more detail, we would like to analyze into two aspects: computational efficiency and prediction accuracy.

In terms of computational efficiency, we are able to visually observe the gap of time consumption between the simple structure U-Net and Mobile-Unet. [Fig. 6]

Compared with the traditional simple U-Net structure, Mobile-Unet also has two stages—the pre-training stage as “encoder” and the semantic segmentation stage as “decoder.” However, Mobile-Unet replaces the normal convolution with

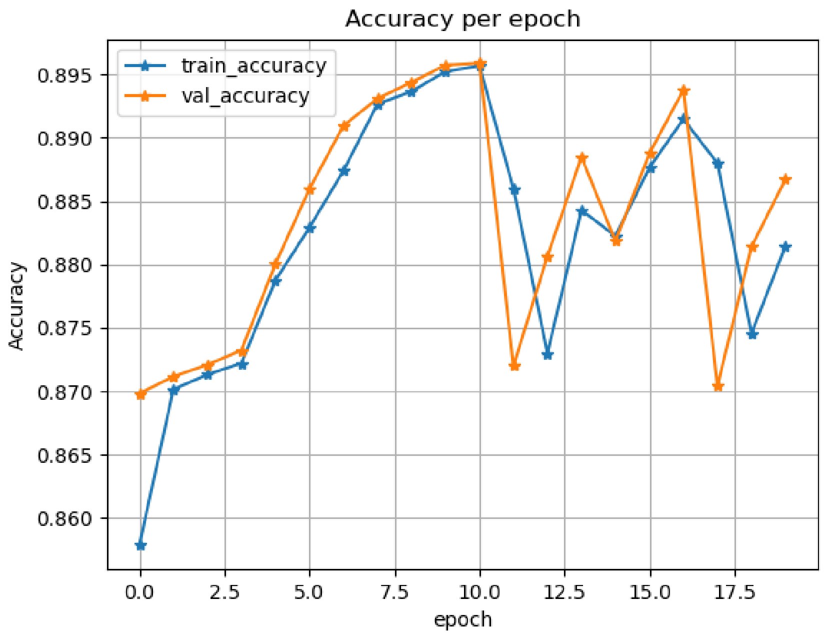


Figure 3: Pixel accuracy of the attempt with weight decay = 1e-04, AdamW, and batch size 2.

the MobileNetV2, which uses depth-wise separable convolution in the "encoder" stage. Depth-wise separable convolution [(b) in Fig. 7] is critical for efficient neural network architecture. It splits the original full convolution into two separate layers. The first layer is depth-wise convolution, and the second layer is a 1×1 convolution [4]. The depth-wise convolution performs lightweight filtering, and each filter matches to one input channel [4]. In other words, we separate the training sample into several sub-operating spaces concerning different channels. Therefore, the depth-wise convolution substantially reduces the complexity of computation. The 1×1 convolution is aiming at the linear projection, which deals with the channel-wise pooling. It transforms the multi-channel space into a one-channel output space, thus reducing the dimensionality. Generally, due to the depth-wise separable convolution structure, our training requires fewer parameters to be adjusted and reduces the possible over-fitting problem; meanwhile, the computation of our training is much cheaper, which is much more suitable for the real-time fickle application [5]. Moreover, since the depth-wise separable convolution, namely, splits the whole sample into several subspaces, the ReLU activation function receives lower-dimensional functional spaces for each time. Meanwhile, ReLU is capable of effectively transmitting and preserving the input manifold, but only if the input manifold lies in a lower-

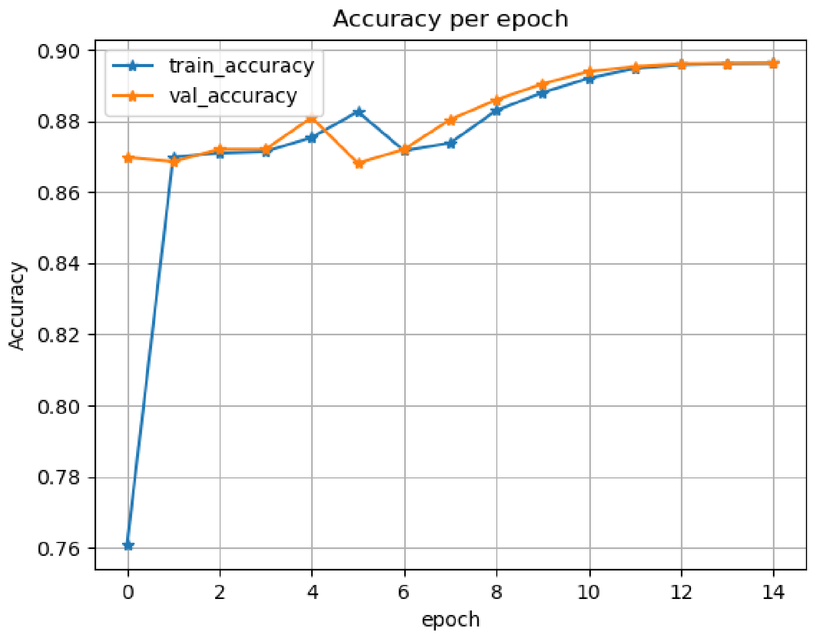


Figure 4: Pixel accuracy of the attempt with learning rate scheduler OneCycleLR.

dimensional subspace of the input space [4]. Therefore, we believe it must be another significant factor to speed up our training to some extent.

In terms of prediction accuracy, we probe into the inverted residual blocks of MobileNetV2. There are two types of inverted residual blocks: $stride = 1$ block and $stride = 2$ block [5].

3.4 Third Attempts

After examining the advancements in Mobile-Unet, we run our training on the Mobile-Unet with encoder depth 5. While keeping the other hyper-parameters the same, it yields a much better result [Fig. 8]; however, we still wish to close the gap between the validation accuracy and train accuracy.

Given that we conclude it might be the model overfitting the training set, we added further transformations to the training and validation set, with vertical and horizontal flip, Grid Distortion, Random Brightness and Contrast Gaussian Noise. However, with the test-val accuracy gap being closed, we sacrificed the overall accuracy, from originally 0.83 to now 0.76; therefore, we concluded that we stick to no distortion being conducted. The final model is then trained on an epoch of 30, with the final training accuracy reaching 0.885 and validation accuracy 0.846.

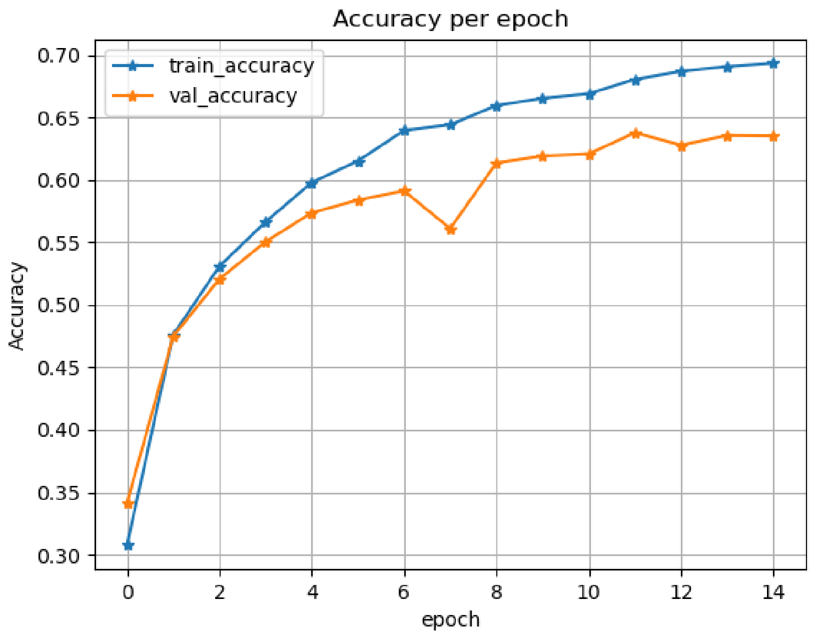
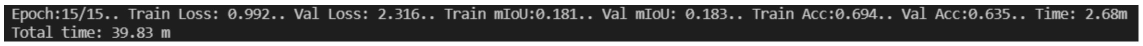


Figure 5: Pixel accuracy of the attempt with fine-tuned parameters on the whole validation set.

4 Results and Discussion

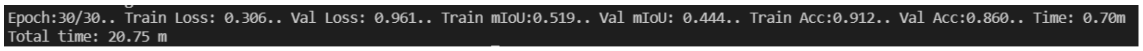
4.1 Evaluation Metrics

We have chosen two specific evaluation matrices to evaluate our model to judge the training performance: pixel accuracy and mean intersection over union. According to the combination of ground truth label and our prediction output, each pixel can be divided into four types: true positive (TP), false positive (FP), true negative (TN), and false negative (FN).



```
Epoch:15/15.. Train Loss: 0.992.. Val Loss: 2.316.. Train mIoU:0.181.. Val mIoU: 0.183.. Train Acc:0.694.. Val Acc:0.635.. Time: 2.68m
Total time: 39.83 m
```

(a) U-Net



```
Epoch:30/30.. Train Loss: 0.306.. Val Loss: 0.961.. Train mIoU:0.519.. Val mIoU: 0.444.. Train Acc:0.912.. Val Acc:0.860.. Time: 0.70m
Total time: 20.75 m
```

(b) Mobile-Unet

Figure 6: Training execution time.

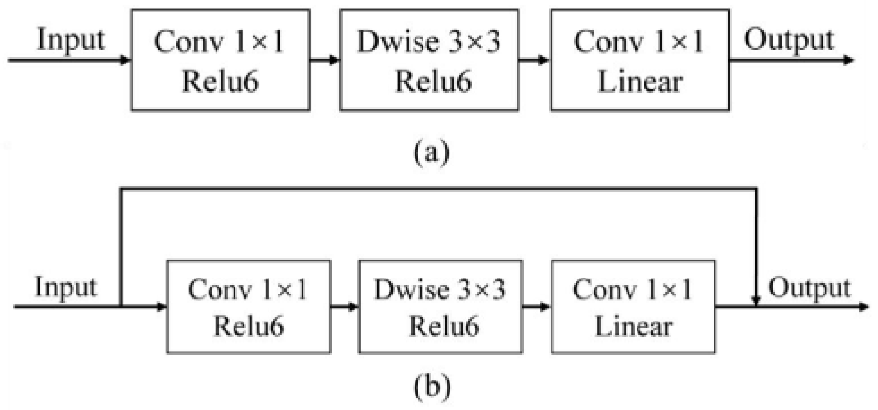


Figure 7: Inverted residual blocks (a) stride=2 block; (b) stride=1 block (Matthias Rätsch et.al 2020)

4.1.1 Pixel Accuracy

The pixel accuracy is to calculate the proportion of the pixels correctly predicted. The formula is given:

$$\frac{TP + TN}{TN + FN + TP + FP}$$

Our objectivity is to effectively improve the accuracy of both our training and testing dataset.

4.1.2 Mean Intersection Over Union (mIoU)

The intersection over union is to calculate the proportion of pixel areas which our output mask overlaps with the ground truth label mask for each class. Then we calculate the mean of all classes. The higher mIoU we obtain, the more accurately our prediction fits the targe. The formula is given:

$$IoU = \frac{TP}{FN + TP + FP}$$

Besides, we have also tried several attempts to speed up the evaluation computing process. For example, we take a crack at utilizing a one-hot matrix to do the IoU calculation. We may first transform the output and target mask into the format of the one-hot matrices. The multiplying output matrix with the target matrix directly represents a total of pixels in the intersection part. However, we encountered coding errors when we use the PyTorch and TensorFlow at the same time. Therefore, it motivates us to reflect on other means to carry on the optimization process and figure out the potential problems between PyTorch and TensorFlow.

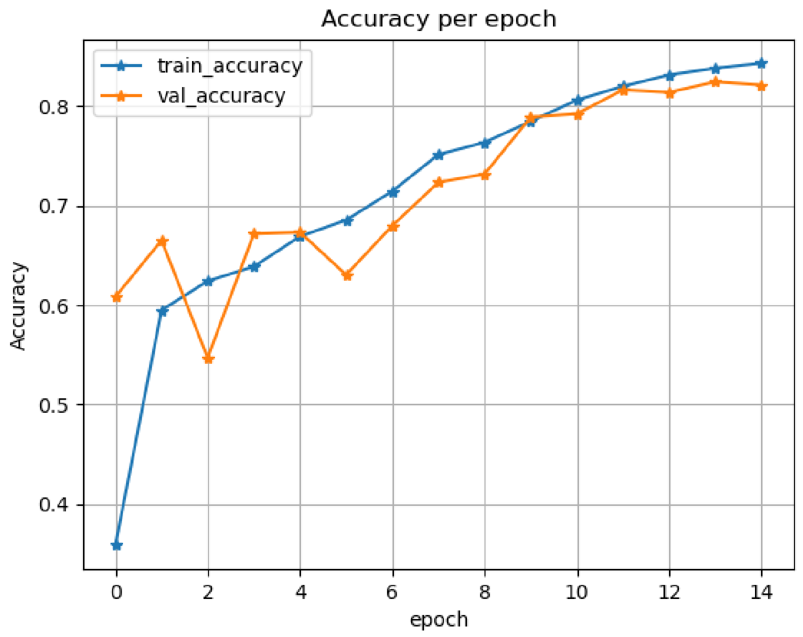


Figure 8: Accuracy per epoch based on Mobile-Unet, with 15 epochs.

4.2 Model Result Comparison

For the self-implemented U-Net and the Mobile-Unet, we took one representing example to visualize the result [Fig. 9]. We see that our U-Net could roughly identify the location of the objects correctly, but the boundaries are far worse than Mobile-Unet. From the previous examination, we conclude that Mobile-Unet uses a skip connection similar to that of ResNet in the up-sampling process, which yields a much better result than our VGG-based U-Net with up-Conv layers and convolutional layers of small kernel size (3×3). Moreover, for each of the epochs, our U-Net would take an average of 2.7 minutes, while Mobile-Unet takes a far less average of 0.7 minutes. We contribute this outcome to the encoding layers using MobilenetV2, which decreases the parameters needed in a training process.

4.3 Self-generated Evaluation Set

In order to test the generality of our approach, we create our testing data with other drone footage found on the Internet and then apply our model to the dataset to get predicted segmentation results. We also attempt to use the Semantic Segmentation Editor on GitHub [1] to manually apply the RGB masks on our self-created dataset to calculate accuracy. However, since this process

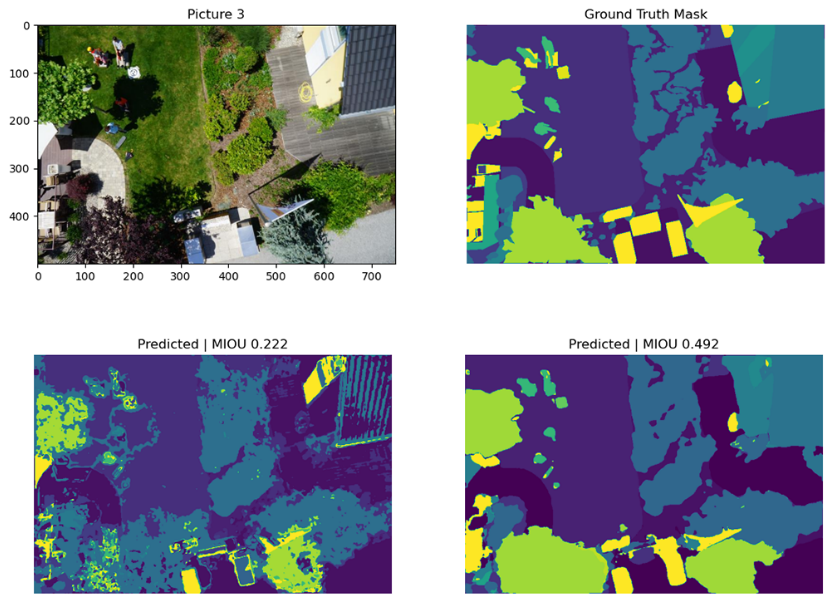


Figure 9: A comparison of the model performance. Top left: original image. Top right: ground truth mask. Bottom left: prediction result of our U-Net. Bottom right: prediction result of Mobile-Unet.

is time-consuming, we cannot massively create RGB masks for the dataset. Nevertheless, through the limited masked data we create, we still draw several conclusions on our model.

Figure 10 shows that the image segmentation is conducted well through our model. In addition, classes such as roads, trees, vegetation, and roofs are predicted accurately. However, some classes are predicted poorly. We classify the poorly predicted classes into two categories. The first category has limited samples in our trained dataset, such as the pool class. The other category is mainly represented by the obstacle class, which has too many samples in the dataset, and each sample is broadly different from one another. In other words, the obstacle class does not have a precise definition. As a result, classes in the first category are easily predicted as obstacle class objects. From Figure 10, we can see that the pool is falsely masked with yellow, representing the obstacle class. What is more, the angle, altitude, and color of the images may also influence the prediction results.

Furthermore, We have also attached some prediction results, which indicates our model still have a great potential for the further improvement.[Fig. 11]

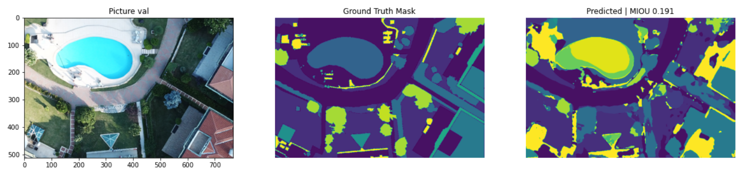


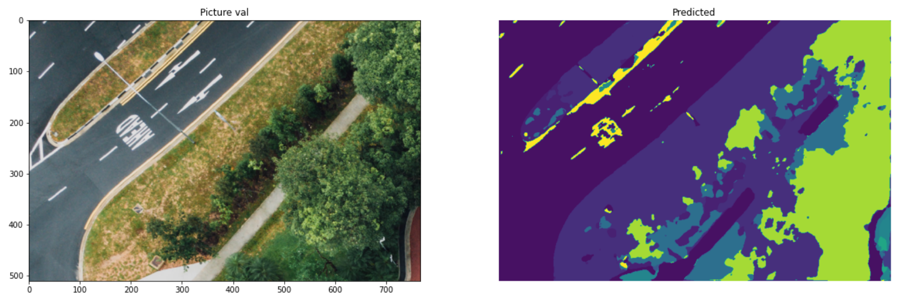
Figure 10: Prediction of self-created data. Left: original image. Middle: self-created ground truth mask. Right: prediction result.

5 Future Works

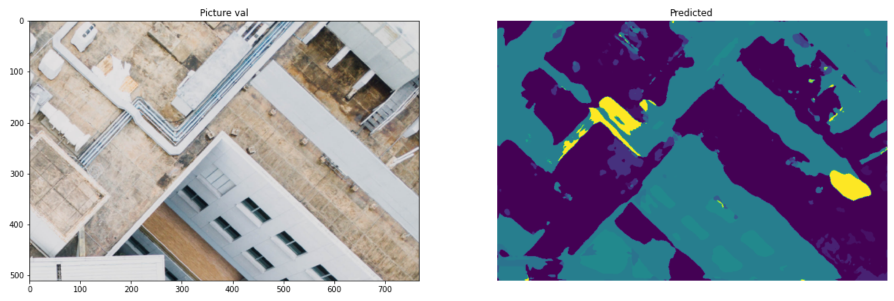
Recalling on all previous progresses, we have several anticipations of our future works.

First of all, we attempt to probe into the connections between image segmentation and video segmentation. Our purpose is to boost the real-time automation in drone piloting and image capturing; an extension from our image model to the video model is necessary. Since a video can be regarded as an aggregation of abundant consecutive pictures, the synergistic effect of those pictures' segmentation may contribute to a video segmentation correspondingly as a whole. It is also essential for us to figure out a more efficient algorithm for the training process, thus guaranteeing that our method is cost-effective and less time-consuming. That being said, our final approach is using Mobile-Unet, which utilizes the MobileNetV2 encoding structure; since MobileNetV3 has been published by Howard et al. [6] and has reached more state-of-the-art results, integration of that algorithm into our model is possible. Moreover, the mIoU could be based on transforming the output and label mask into the one-hot matrices, with further multiplication to get the result of the intersection area.

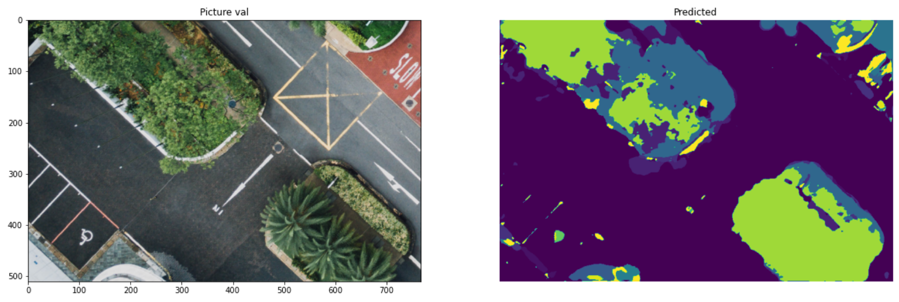
Additionally, our dataset only contains 24 classes, which is much simpler than a real-time application consisting of multiple classification objects. To generalize our model, we may add more contents images to imitate the diversity of real-time scenarios and then augment our dataset from different angles, such as adding more features and considering objects with messier neighboring boundaries. Furthermore, in the end, integration of the Pyramid Scene Parsing Network [7] with our current model, which includes more context of the scene to facilitate the segmentation, could be experimented with to boost our model accuracy in real applications further.



(a) Template 1



(b) Template 2



(c) Template 3

Figure 11: Prediction Results

References

[1] Evaluation Data Generator: Semantic Segmentation Editor https://github.com/Hitachi-Automotive-And-Industry-Lab/semantic-segmentation-editor.

[2] Semantic Drone Dataset https://www.tugraz.at/index.php?id=22387 *Alternative Dataset Resource: Aerial Semantic Segmentation Drone Dataset* https://www.kaggle.com/bulentsiyah/semantic-drone-dataset.

[3] Leslie N. Smith and Nicholay Topin. “Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates” `arXiv:1708.07120v3`, 2018.

[4] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. “MobileNetV2: Inverted Residuals and Linear Bottlenecks.” In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* `arXiv:1801.04381v4`, 2019.

[5] Junfeng Jing, Zhen Wang, Matthias Rätsch and Huanhuan Zhang. “Mobile-Unet: An efficient convolutional neural network for fabric defect detection.” In *Textile Research Journal* `DOI:10.1177/0040517520928604`, 2020.

[6] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam “Searching for MobileNetV3.” `arXiv:1905.02244v5`, 2019.

[7] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia “Pyramid Scene Parsing Network.” `arXiv:1612.01105v2`, 2017.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox “U-Net: Convolutional Networks for Biomedical Image Segmentation.” `arXiv:1505.04597v1`, 2015.