Abstract

An abstract sums up your work in very few sentences:

(i) state the problem you are addressing;

(ii) say why it’s an interesting problem, and which issues are hard to tackle;

(iii) give your approach towards solving the problem;

(iv) say Why and how well your approach solves the problem.

1. Introduction

Your introduction briefly explains the problem you address, and what you’ve achieved towards solving the problem. It’s an edited and updated version of your introduction and objective from your topic proposal

Aerial Drone, or unmanned aerial vehicle (UAV) has recently gained an increased popularity in civil use. While current manufacturers incorporating automation to the aerial drones, they are at a relatively lower level 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 automizing 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](https://www.tugraz.at/index.php?id=22387)” 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](https://arxiv.org/abs/1505.04597)” 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](https://journals.sagepub.com/doi/10.1177/0040517520928604)” 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](https://arxiv.org/abs/1801.04381)” 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 mean intersection over union (mIoU) scores. To further test the model, self-generated evaluation set and test set were created using images on unsplash.com, with further preprocessing and tagging using [Semantic Segmentation Editor](https://github.com/Hitachi-Automotive-And-Industry-Lab/semantic-segmentation-editor) contributed by Hitachi Automotive and Industry Lab. Generally, the model yields good result and further reflections were made based on practical and theoretical reasons.

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

1. Dataset

Explain what dataset you will use, and give a short description about the dataset.

1. Dataset Features
2. Dataset Preprocessing
3. Methods

The solution section covers all of your model design, algorithms, formulas, findings etc. It explains in detail each contribution, if possible with figures/schematics.

1. U-Net

The reason to choose U-Net based model is that it claims to have a high performance upon limited dataset. Also, our dataset, the aerial drone images are shot from a bird’s eye view, which is different from what’s usually seen. While U-Net is originally applied on the biomedical image shot from 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.

In the U-Net model, a u-shaped architecture is used [Figure 1.], with the left side using double convolutional layers of 3x3 kernel size and consecutive max pooling layers to extract features of the images, and the right side using 2D transposed convolution to consecutive to identical double convolutional layers to increase the resolution of the output. To ensure the accuracy of the up-convolution, same level of output from previous down sampling were copied and concatenated to the up-convolution layers. As the U-Net in the demonstrated figure is using zero padding to the double convolutional layers, the image size actually shrinks from 572x572 to 388x388. Meanwhile, in the coping process of up-sampling, cropping the output from previous down sampling to the same size of the up-conv output is involved. Since we do not want a loss of border pixels for our model, paddings in the double-conv layers and optional padding in the copy layer are implemented to maintain the shape of the output and avoids further reshaping on the validation set.

That means, if we input an image tensor with shape: *batch\_size \* color\_channel \* height \* weight*

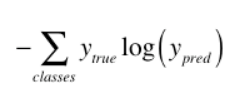
The U-Net model outputs a tensor with shape: *batch\_size \* classes \* height \* weight*, with each class a separate channel.



Fig. 1: U-net architecture

1. First & Second Attempts

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



We then train our self-implemented U-Net on our semantic drone dataset, with first attempts based on a small set of 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, as will be introduced in Section IV) is actually oscillating badly [Figure 2.], yielding unsatisfactory results.

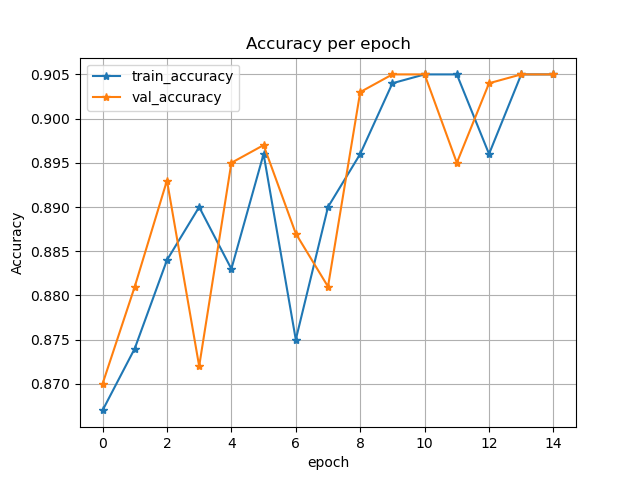


Fig. 2: Pixel accuracy of the very first attempt, with no weight decay or learning rate scheduler.

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 in later epochs [Figure 3.]

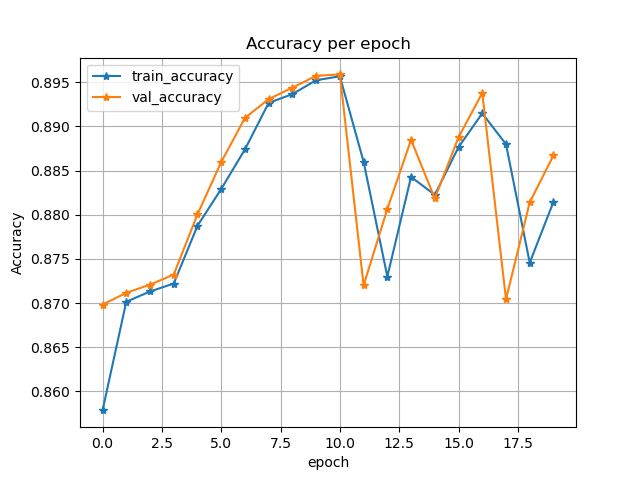


Fig. 3: Pixel accuracy of the attempt with weight decay = 1e-04, AdamW, and batch size 2.

From that 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 is proposed by [Leslie N. Smith and Nicholay Topin](https://arxiv.org/abs/1708.07120), which adjusts the learning rate from an initial learning rate to some maximum learning rate and then from that maximum learning rate to some minimum learning rate much lower than the initial learning rate, a sound choice for our need. This gives a much faster convergence rate and more stable accuracy in later epochs [Figure 4.]

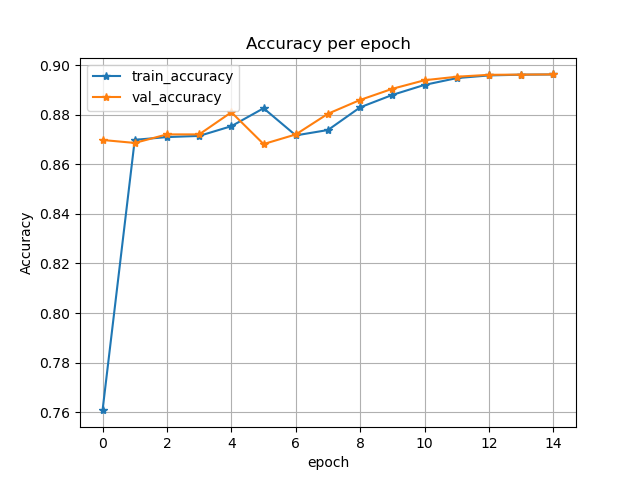


Fig. 5: Pixel accuracy of the attempt with learning rate scheduler OneCycleLR.

However, when we extend the model to the full validation set using the same parameters, the validation accuracy is not ideal at all, yielding accuracy of only 0.7 in 15 epochs, with slope of convergence no further than the boundary of 0.75 [Figure 6.]. After another few tweaking of the parameters, the model performance remains identical. Thus, we conclude that the U-Net structure would need some major improvement if we wish to reach a relatively high performance. And that is the reason we choose an improved version of U-Net, the Mobile-Unet.

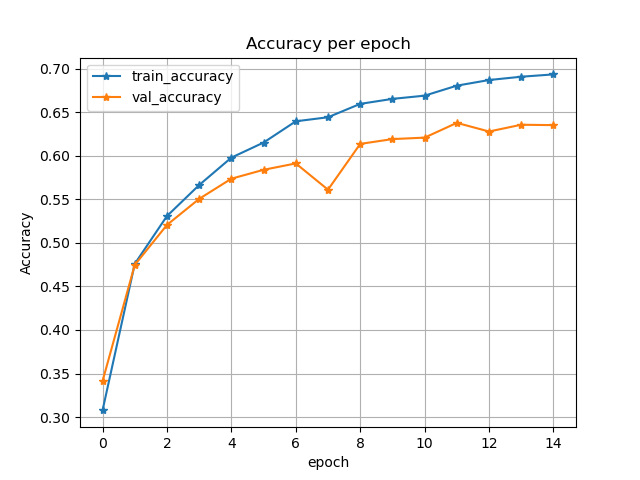


Fig. 6: Pixel accuracy of the attempt with finetuned parameters on the whole validation set.

1. Mobile-Unet
2. Third Attempts

After examine the advancements in Mobile-Unet, we run our training on the Mobile-Unet with encoder depth 5. While keeping the other hyperparameters the same, it yields a much better result [Figure 7.]; however, we still wish to close the gap between the validation accuracy and train accuracy.

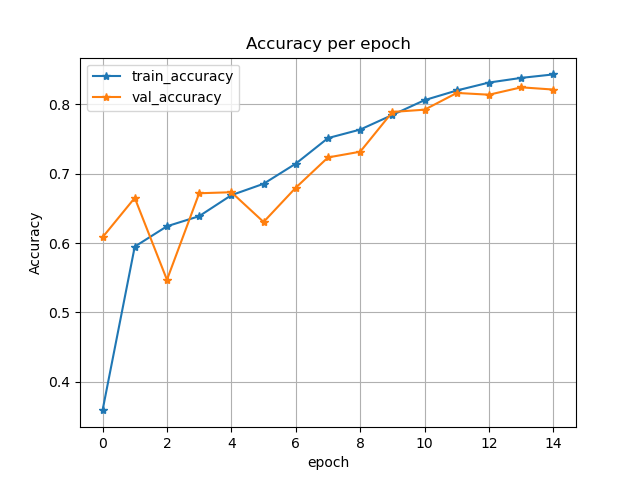


Fig. 7: Accuracy per epoch based on Mobile-Unet, with 15 epochs.

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 and 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.

1. Results and Discussion

The results section details your metrics and experiments for the assessment of your solution. It allows you to compare your idea with other approaches you've tested.

1. Evaluation Metrics
   1. Pixel Accuracy
   2. Mean Intersection Over Union (mIoU)
2. Model Result Comparison

For the self-implemented U-Net and the Mobile-Unet, we took one representing example to visualize the result [Figure 8.]. We see that our U-Net could roughly identify correctly the location of the objects, but the boundaries are far worse then Mobile-Unet. From previous examination, we conclude that it is because Mobile-Unet uses 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 (3x3). 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.

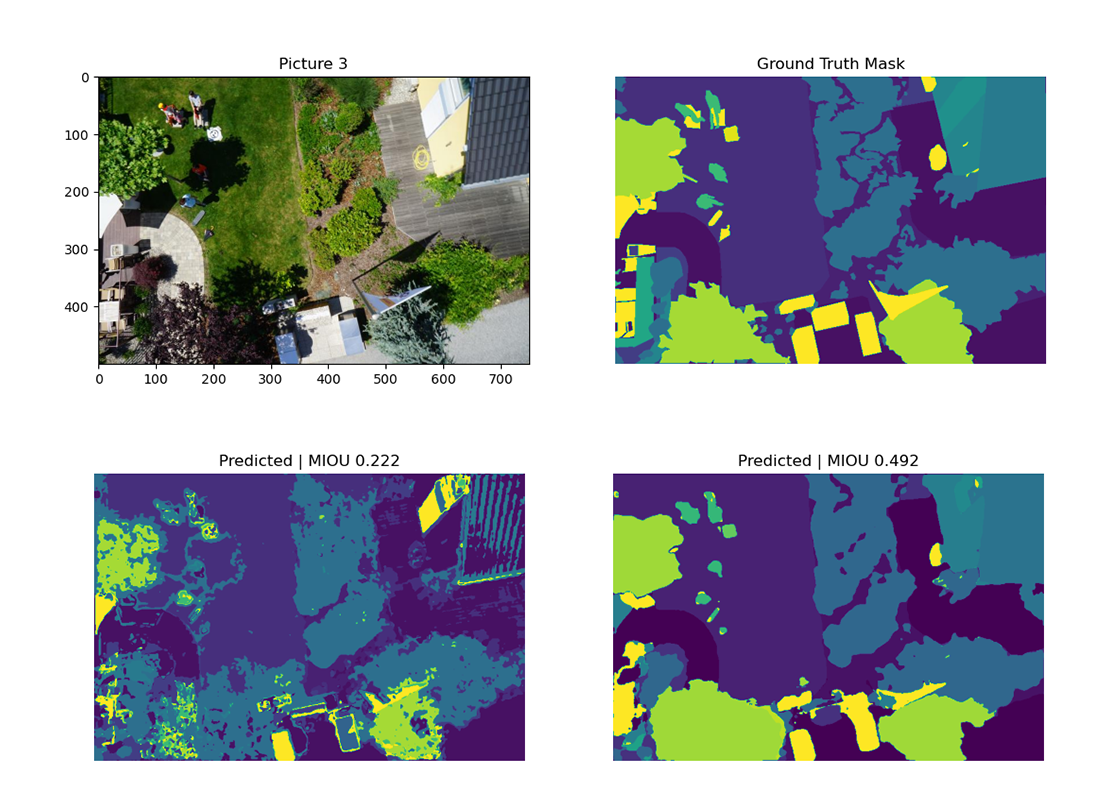


Fig. 8: A comparison of the model performance. Top left: original image. Top right: ground truth mask. Bottom left: prediction result of our U-Net. Bottom right: prediction result of Mobile-Unet.

1. Self-generated Evaluation Set
   1. Process
   2. Output Discussion
2. Future Works

References