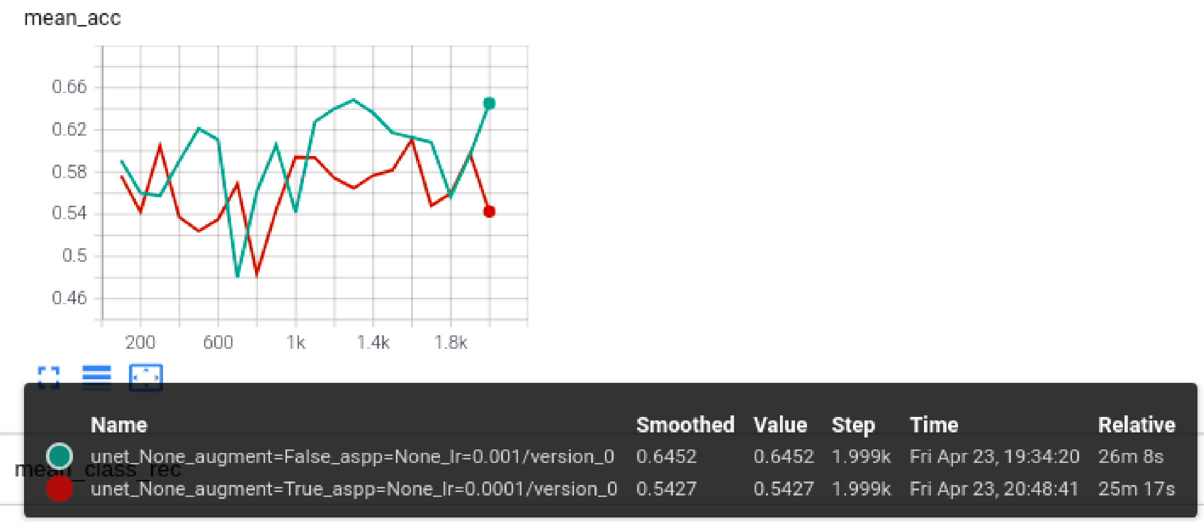
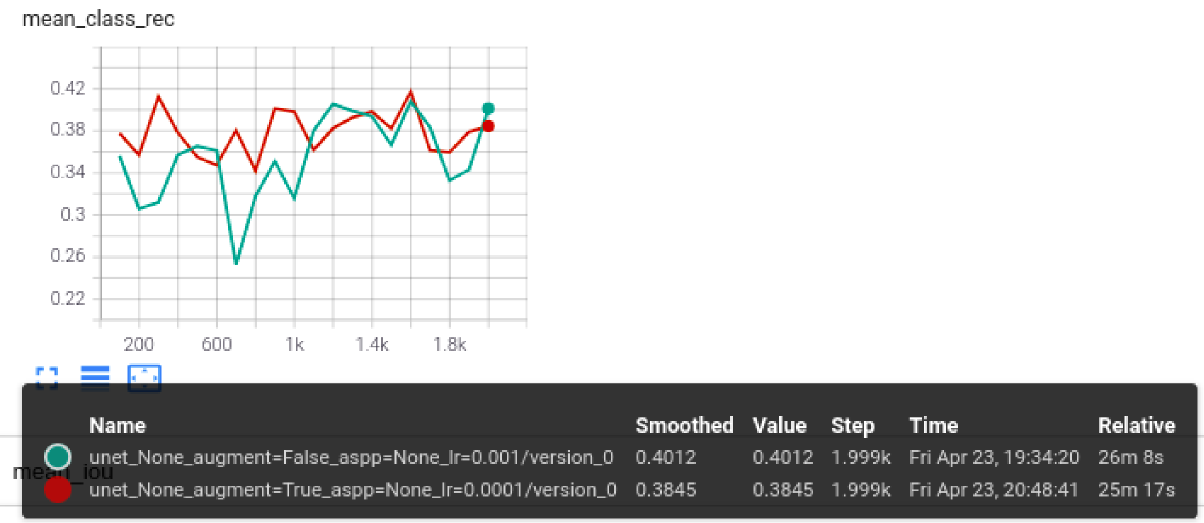
Task 1

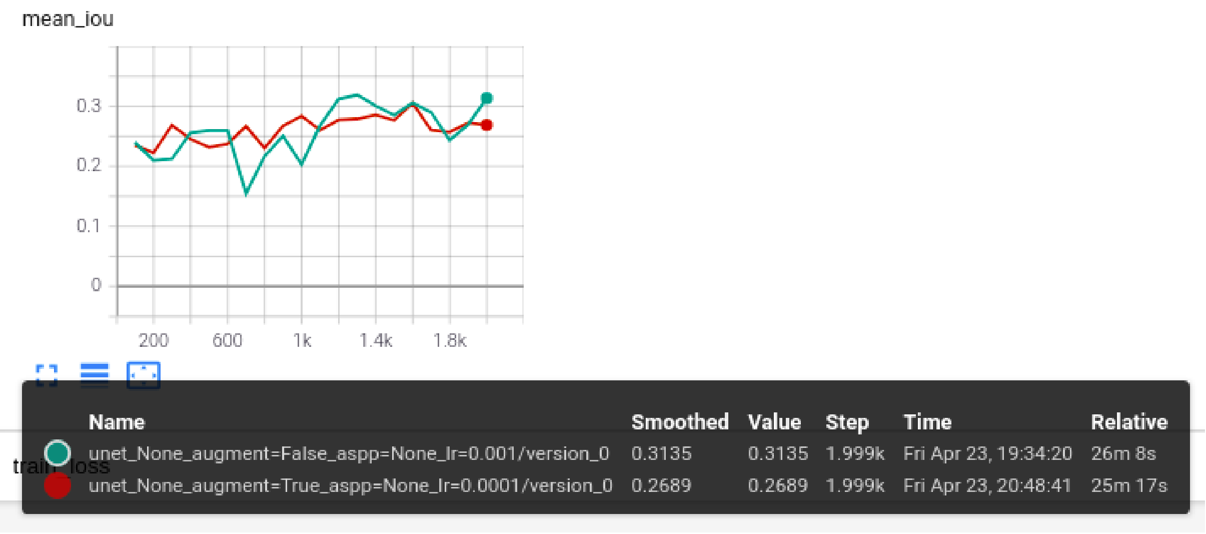
As can be seen from the train loss chart, no augmentation version converges faster to optimal loss value, but then there are no improvements for the whole training. On the other hand, the augmentation version converges slower, but it tends to get a lower loss if the number of training epochs increases. The only reason for that is using augmentation which helps to regularize our model.



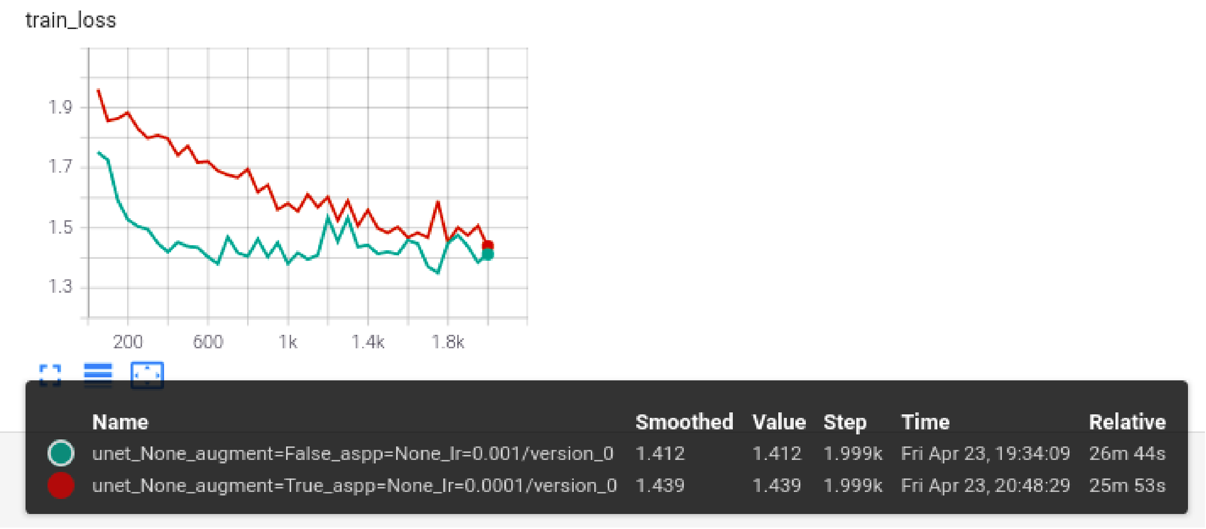
	Name	Smoothed	Value	Step	Time	Relative
●	unet_None_augment=False_aspp=None_lr=0.001/version_0	0.6452	0.6452	1.999k	Fri Apr 23, 19:34:20	26m 8s
●	unet_None_augment=True_aspp=None_lr=0.0001/version_0	0.5427	0.5427	1.999k	Fri Apr 23, 20:48:41	25m 17s



	Name	Smoothed	Value	Step	Time	Relative
●	unet_None_augment=False_aspp=None_lr=0.001/version_0	0.4012	0.4012	1.999k	Fri Apr 23, 19:34:20	26m 8s
●	unet_None_augment=True_aspp=None_lr=0.0001/version_0	0.3845	0.3845	1.999k	Fri Apr 23, 20:48:41	25m 17s



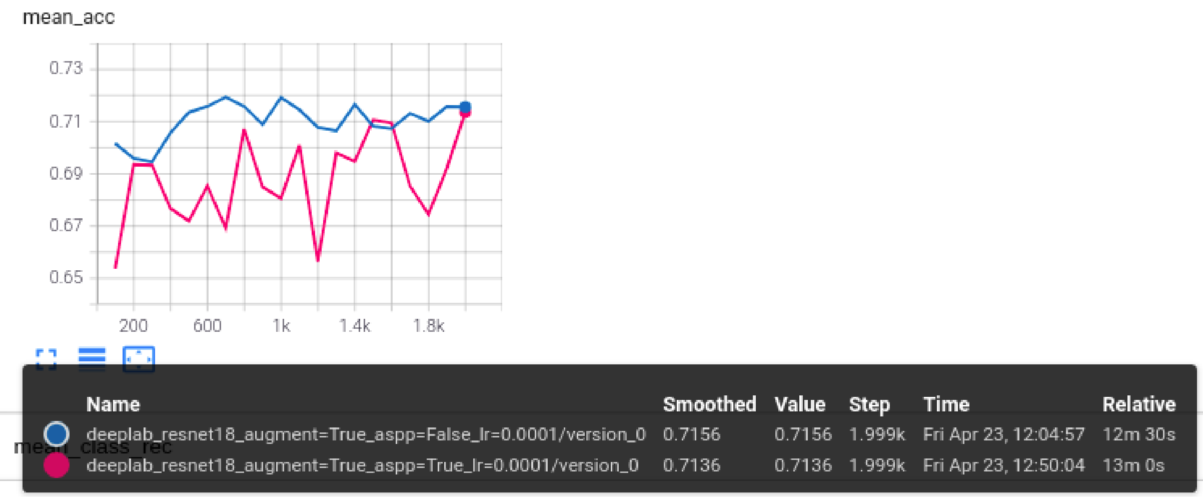
	Name	Smoothed	Value	Step	Time	Relative
●	unet_None_augment=False_aspp=None_lr=0.001/version_0	0.3135	0.3135	1.999k	Fri Apr 23, 19:34:20	26m 8s
●	unet_None_augment=True_aspp=None_lr=0.0001/version_0	0.2689	0.2689	1.999k	Fri Apr 23, 20:48:41	25m 17s



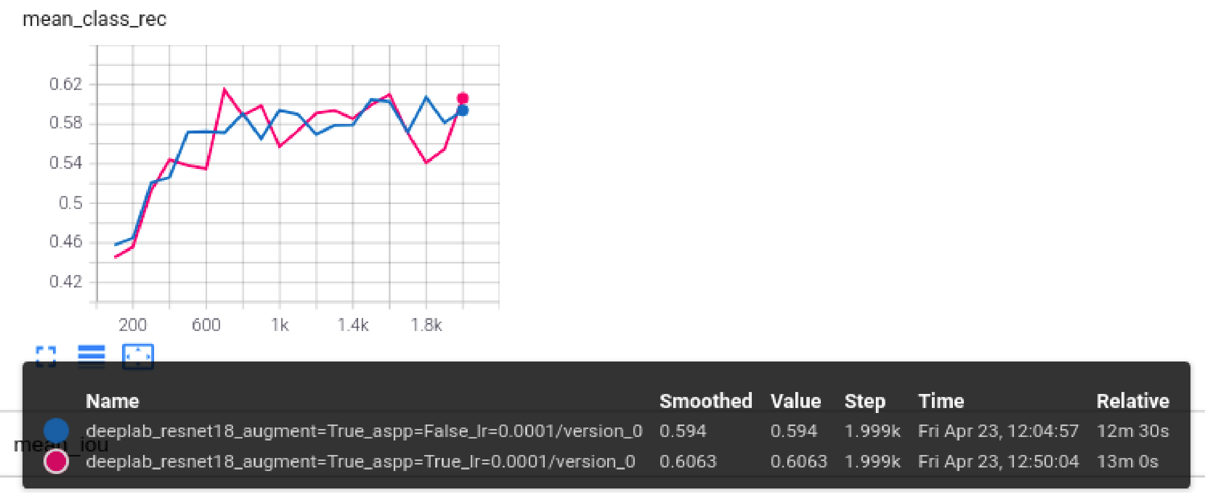
	Name	Smoothed	Value	Step	Time	Relative
●	unet_None_augment=False_aspp=None_lr=0.001/version_0	1.412	1.412	1.999k	Fri Apr 23, 19:34:09	26m 44s
●	unet_None_augment=True_aspp=None_lr=0.0001/version_0	1.439	1.439	1.999k	Fri Apr 23, 20:48:29	25m 53s

Task 2

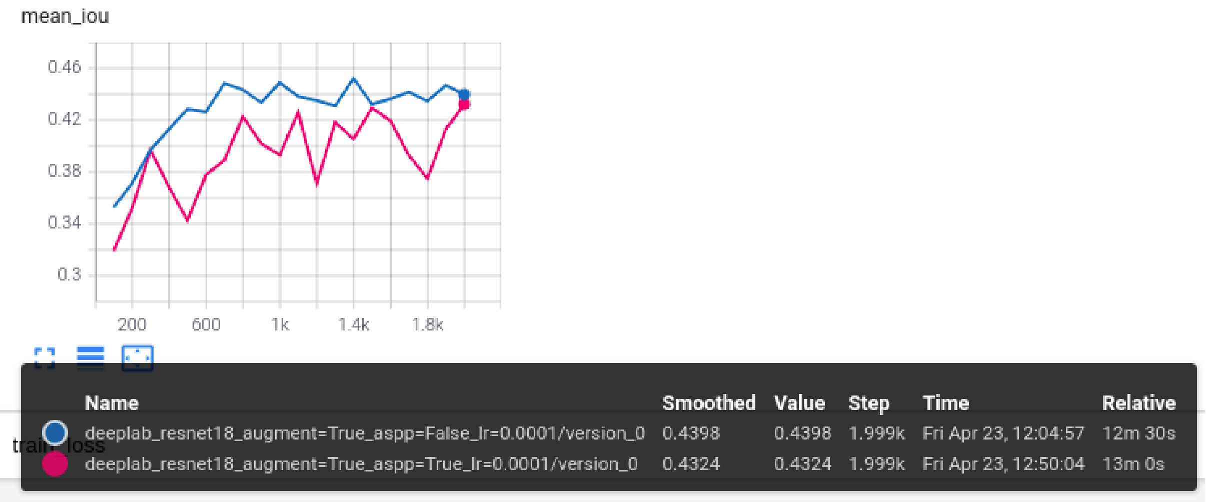
The best model performances look the same. However, no ASPP version seems to perform better in terms of learning stability.



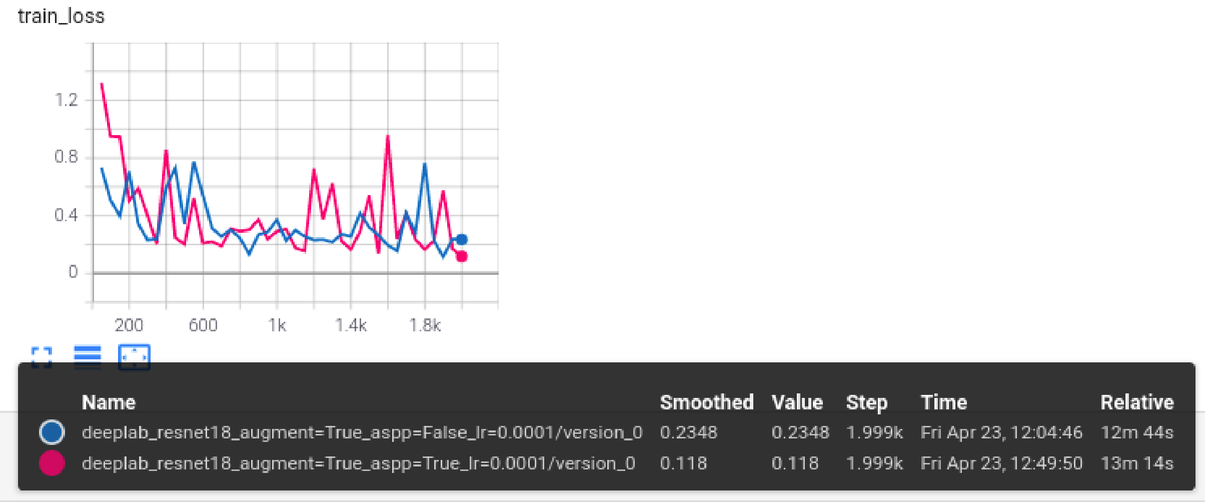
	Name	Smoothed	Value	Step	Time	Relative
●	deeplab_resnet18_augment=True_aspp=False_lr=0.0001/version_0	0.7156	0.7156	1.999k	Fri Apr 23, 12:04:57	12m 30s
●	deeplab_resnet18_augment=True_aspp=True_lr=0.0001/version_0	0.7136	0.7136	1.999k	Fri Apr 23, 12:50:04	13m 0s



	Name	Smoothed	Value	Step	Time	Relative
●	deeplab_resnet18_augment=True_aspp=False_lr=0.0001/version_0	0.594	0.594	1.999k	Fri Apr 23, 12:04:57	12m 30s
●	deeplab_resnet18_augment=True_aspp=True_lr=0.0001/version_0	0.6063	0.6063	1.999k	Fri Apr 23, 12:50:04	13m 0s



	Name	Smoothed	Value	Step	Time	Relative
●	deeplab_resnet18_augment=True_aspp=False_lr=0.0001/version_0	0.4398	0.4398	1.999k	Fri Apr 23, 12:04:57	12m 30s
●	deeplab_resnet18_augment=True_aspp=True_lr=0.0001/version_0	0.4324	0.4324	1.999k	Fri Apr 23, 12:50:04	13m 0s



	Name	Smoothed	Value	Step	Time	Relative
●	deeplab_resnet18_augment=True_aspp=False_lr=0.0001/version_0	0.2348	0.2348	1.999k	Fri Apr 23, 12:04:46	12m 44s
●	deeplab_resnet18_augment=True_aspp=True_lr=0.0001/version_0	0.118	0.118	1.999k	Fri Apr 23, 12:49:50	13m 14s

Task 3

- The most efficient model in terms of training time is the DeepLab with ResNet-18 backbone.

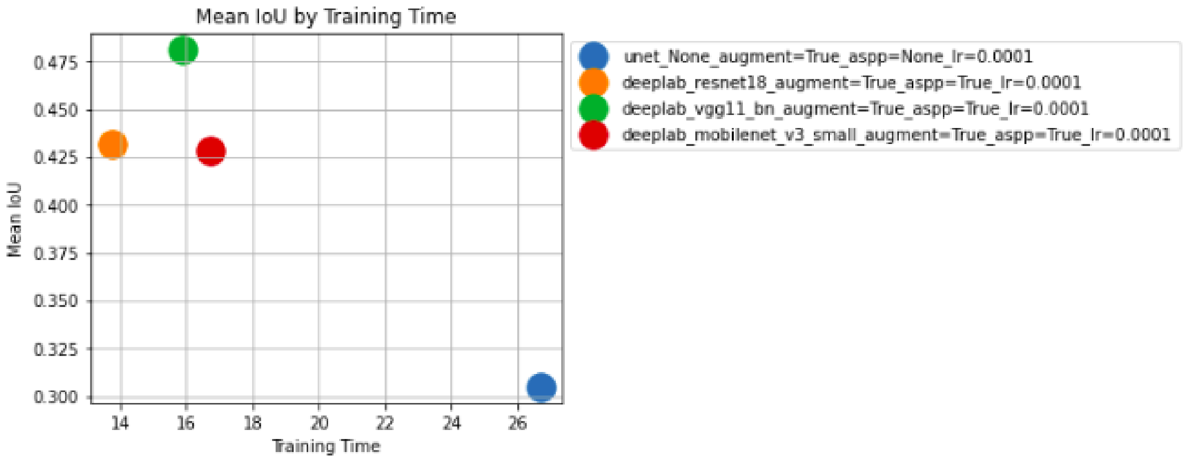
- The most efficient model in terms of inference time is the DeepLab with a MobileNet backbone.
- The most efficient model in terms of training time is the DeepLab with a MobileNet backbone.

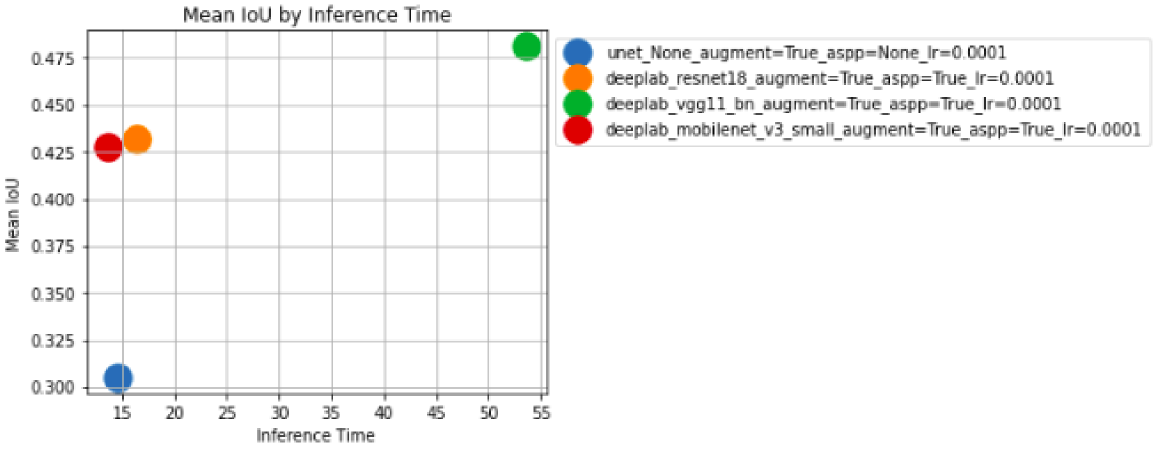
The DeepLab with the VGG11-BN shows the best performance in mean IoU and has a solid gap. Also, its training time is enough low. The only lack of this model is size.

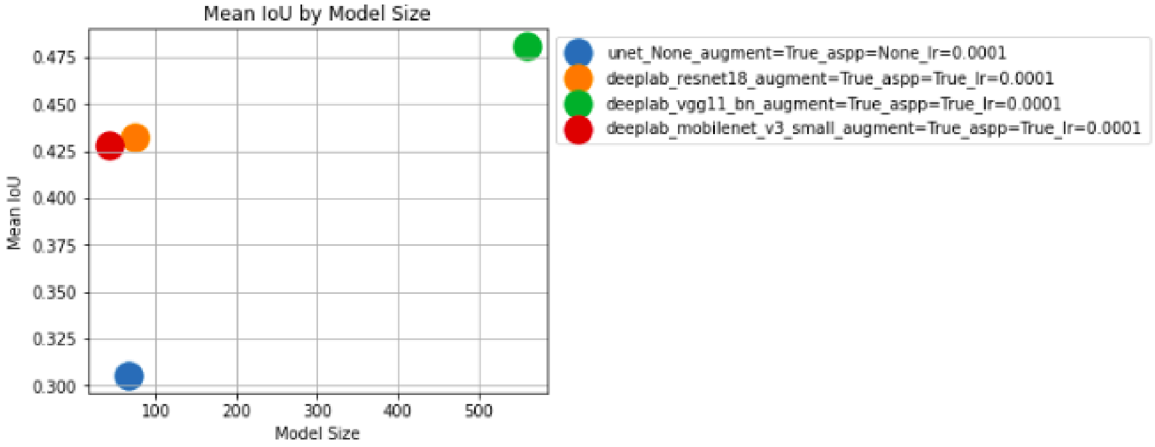
The UNet training is almost two times longer than others. That’s a big issue if the running time is limited. And the model shows the worse performance in mean IoU.

The DeepLab with ResNet-18 and MobileNet backbones look similar both for good performance in mean IoU and training/inference time and model size. These models are the golden mean in terms of quality and speed performance.

The general improvement that could be applied to any model is the following. We are interested in improving quality metric and decreasing the running time and model size.



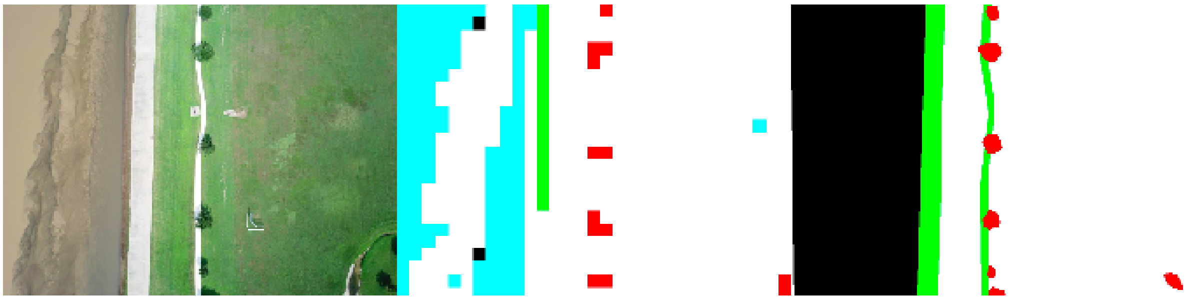




Task 4

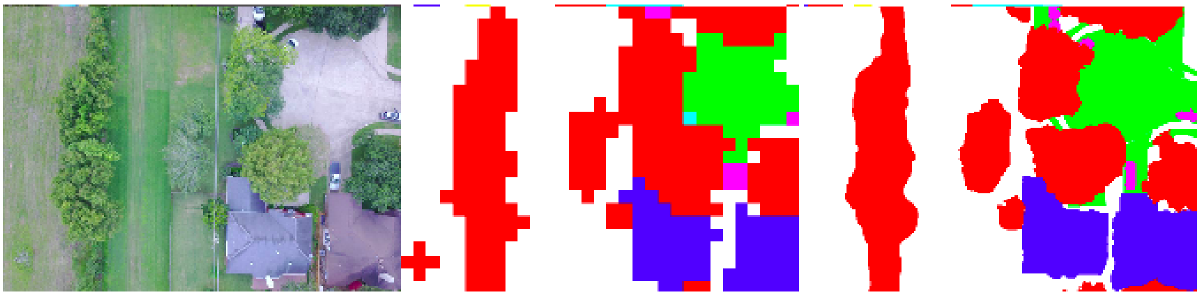
Background

Bad

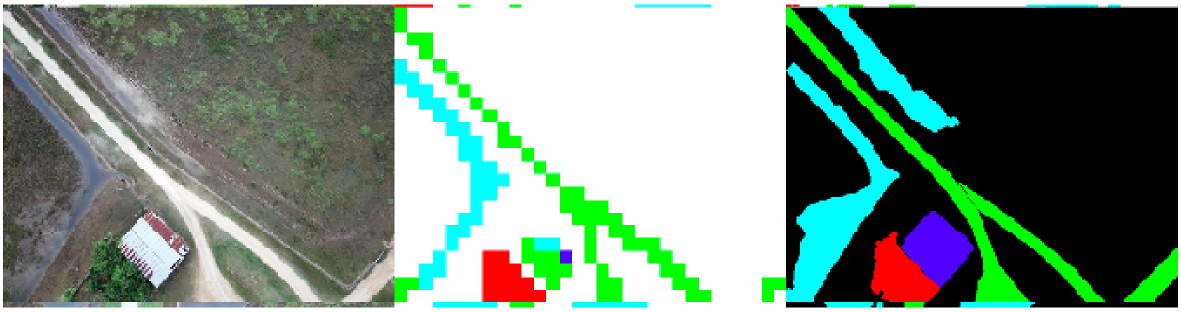


Building

Good

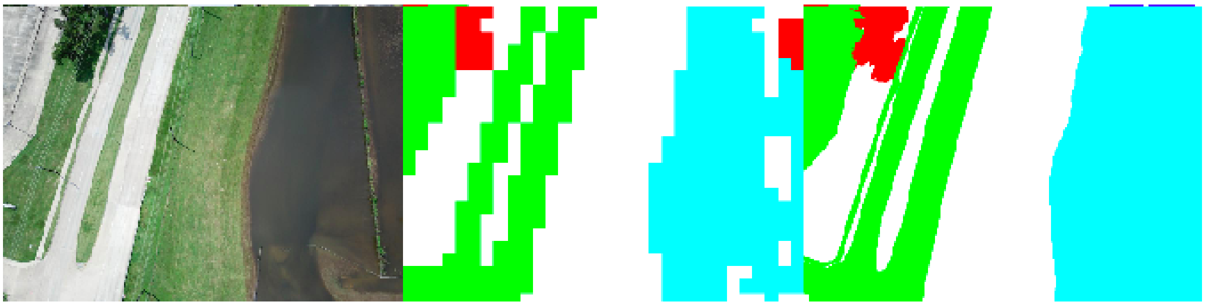


Bad

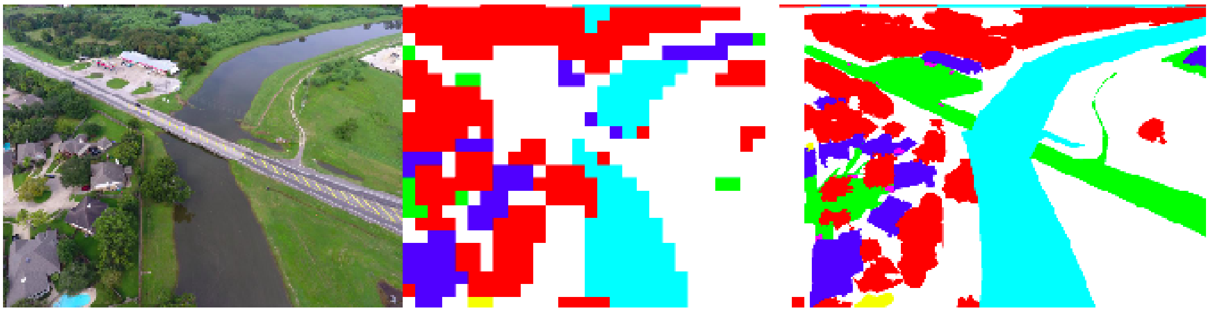


Road

Good

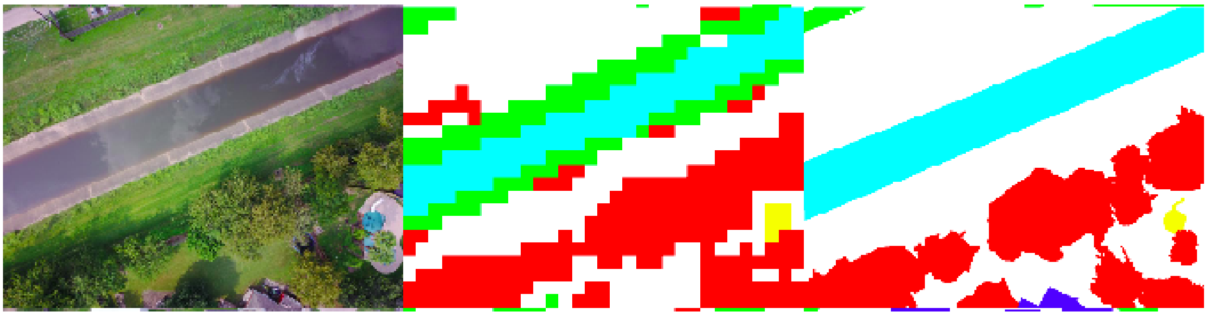


Bad

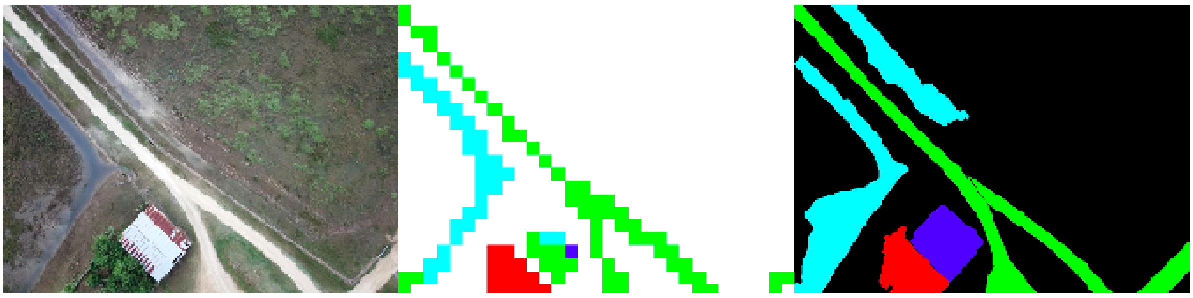


Water

Good

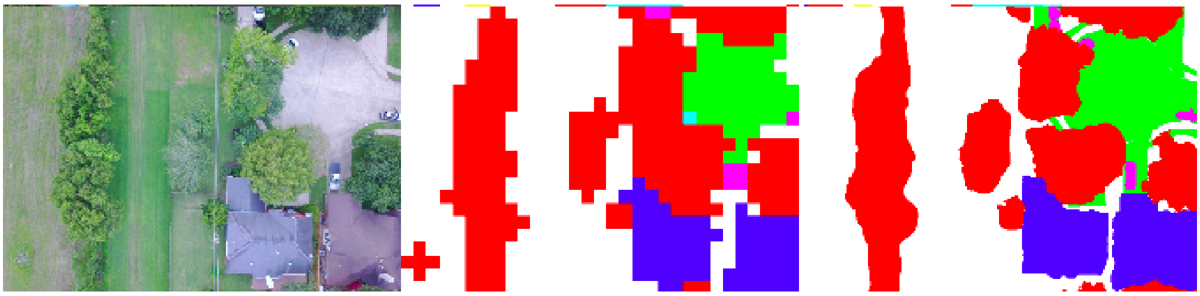


Bad

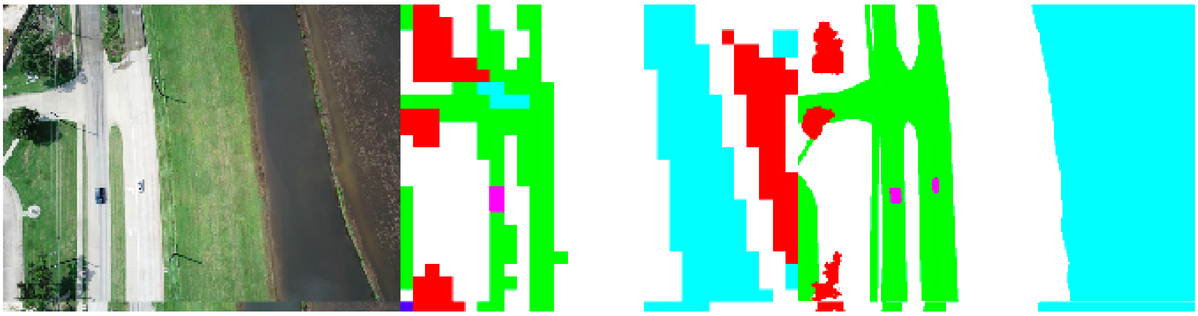


Tree

Good

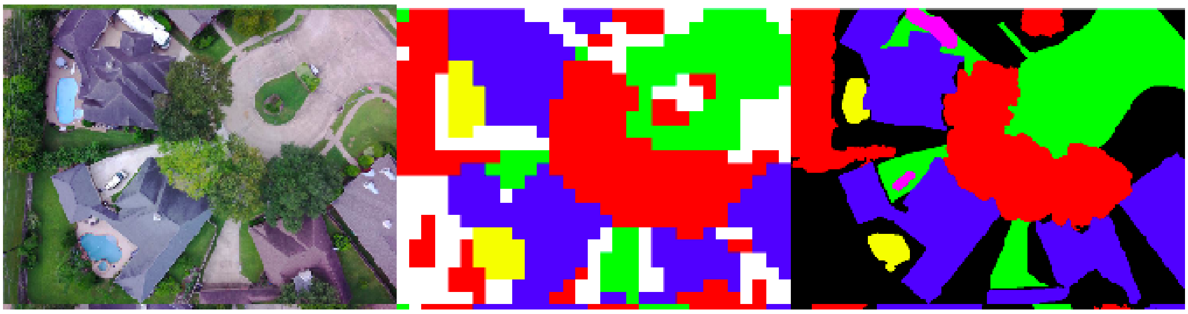


Bad

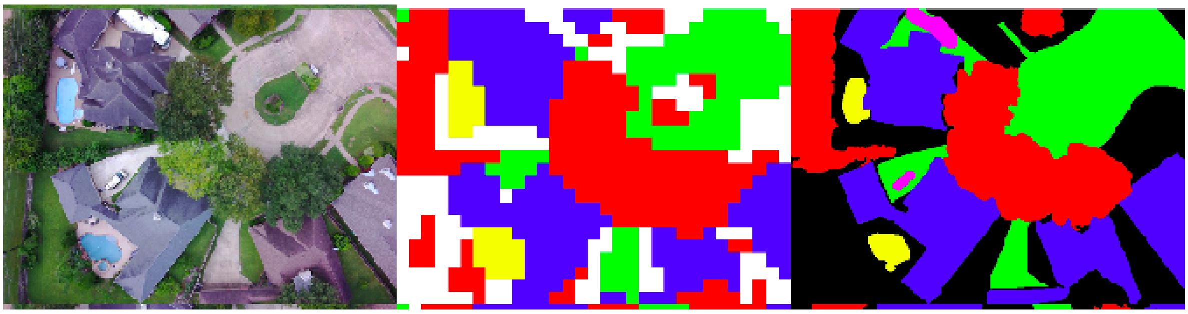


Vehicle

Good

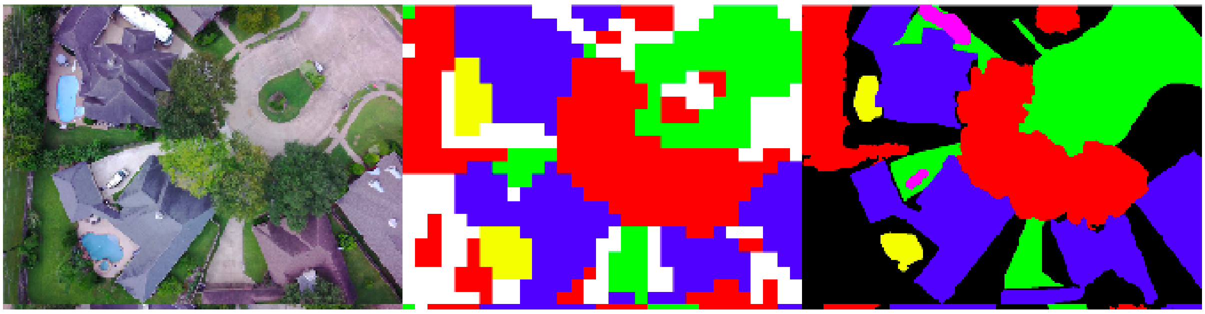


Bad

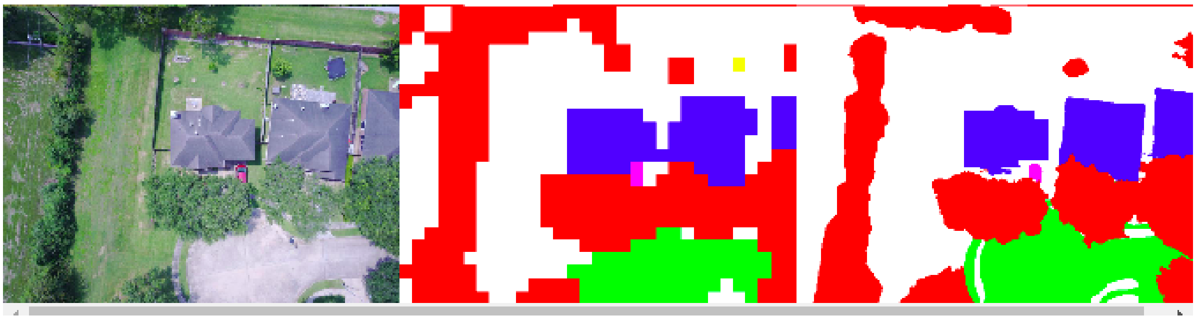


Pool

Good

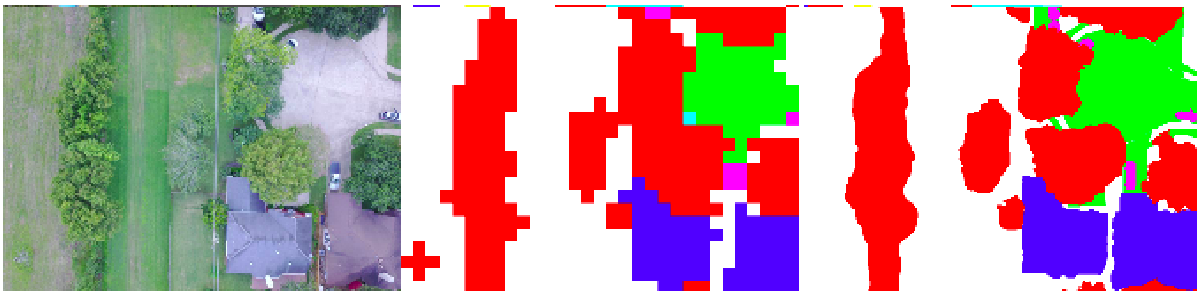


Bad



Grass

Good



Bad