Homework Semantic Segmentation

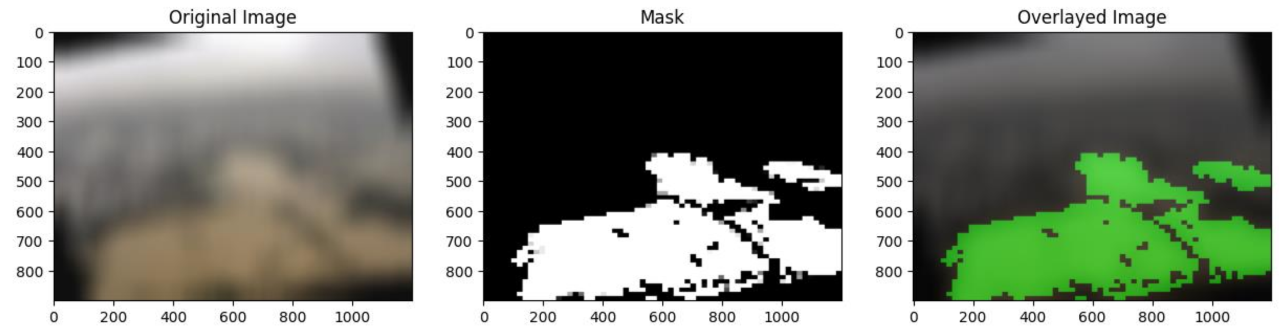
Dmytro Strus

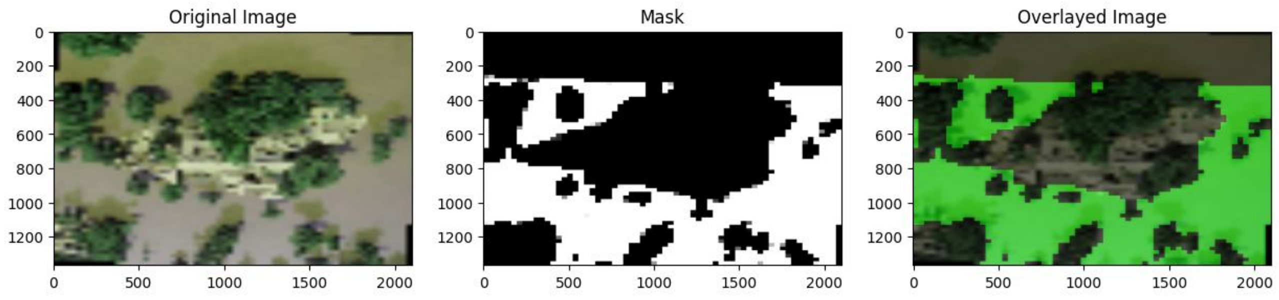
Data pre-processing and processing:

I’ve divided initial dataset into training and validation sets using an 80/20 split. The images were sorted firstly, were paired with image-ground truth mask and shuffled in training and validation folders. This 80/20 split is a common approach that balances the need for enough training data with the requirement for a robust validation set.

I’ve created an original dataset and augmented. For augmentation I’ve chosen horizontal and vertical flips, random rotation for 30 degrees, some blur and contrast changing because for watered territories it important to show different flooded ways from different perspectives and have different shades of polluted water in order to have more information about water flooding.

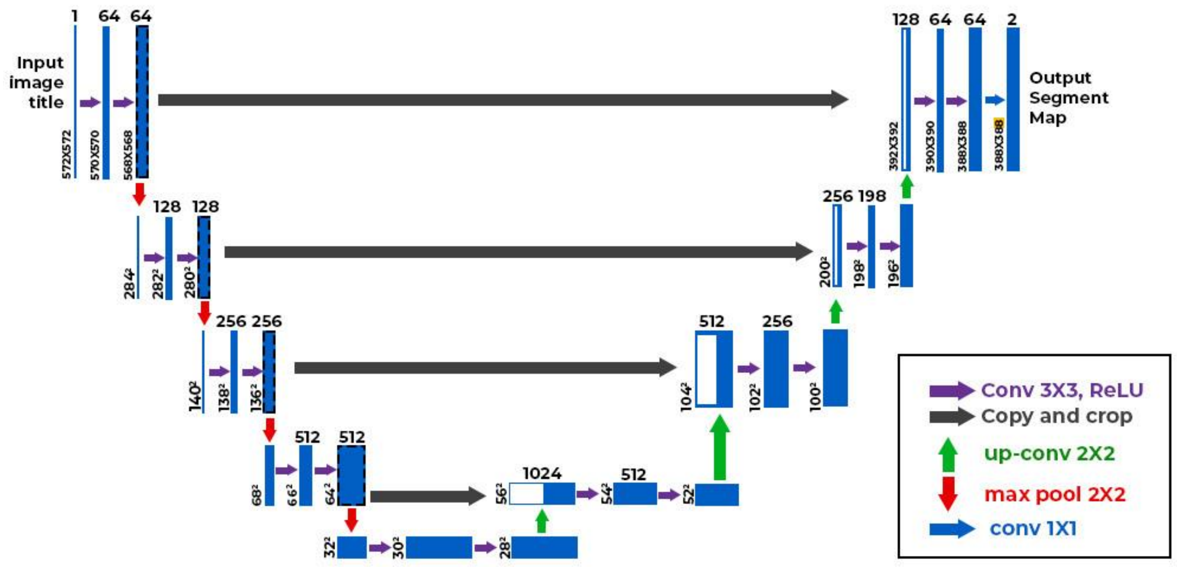
Augmented visualization dataset





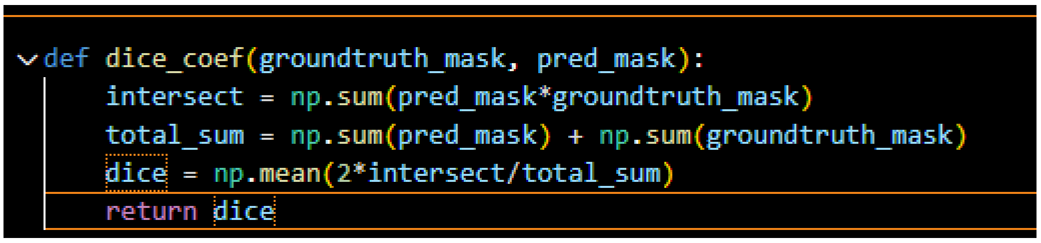
Experiment log:

I've tried different models - FCNN, SEGNET, U-NET. First two showed bad results and I've decided to concentrate on last model U-net which has the primary purpose of this architecture was to address the challenge of limited annotated data in the medical field. This network was designed to effectively leverage a smaller amount of data while maintaining speed and accuracy.



In our case, where there was 290 photos it was a good choice of a model.

The model haven't got KPI of 0.9 using dice metric



```
def dice_coef(groundtruth_mask, pred_mask):
    intersect = np.sum(pred_mask*groundtruth_mask)
    total_sum = np.sum(pred_mask) + np.sum(groundtruth_mask)
    dice = np.mean(2*intersect/total_sum)
    return dice
```

There have been many attempts to obtain a better KPI. From increasing and decreasing the size of the dataset itself and the size of the batch, to adding and decreasing convolutional layers, adding batch norms, and decreasing the size of the convolution but it didn't help much and the best dice metric was 0.82, ranging from 0.7 to 0.8+ with all these different changes. Number of epochs also didn't improve the situation much, as the model hit the 0.29 validation loss limit. After 35+ epochs it becomes overfitted and worse validation loss (bigger), and I guess it happens due to lack of more images and a lot of epochs.

Augmentation for training dataset didn't improved performance of the model prediction and I have a suggestion why: our validation set isn't blurred or doesn't have inaccurate snapped images and our model were trained for different contrasted, blurred, rotated and flipped images which can be in real world but not in our validation dataset.

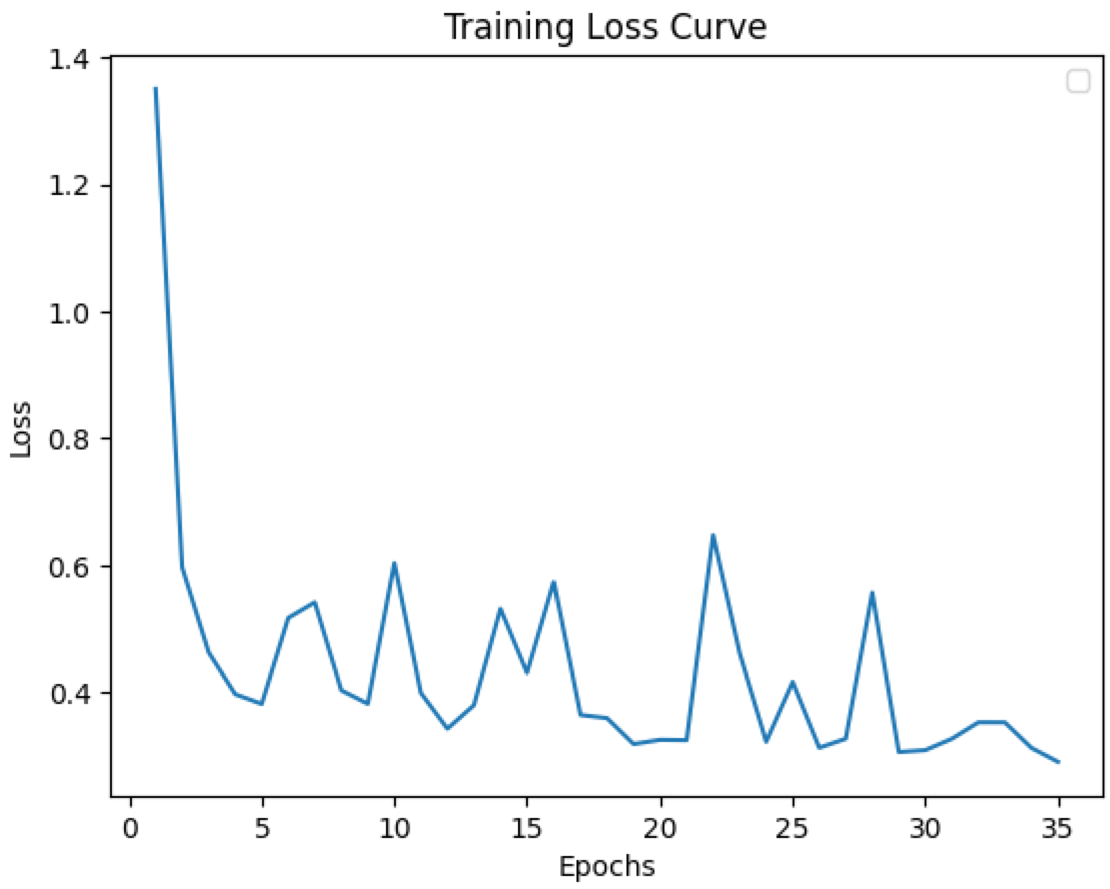
I've tried to change size of pictures in dataset 128x128 to 256x256 and to 64x64 and comes to result that the best will be 64x64 because increase in efficiency is too small taking into account increase in time for training model which increases in times.

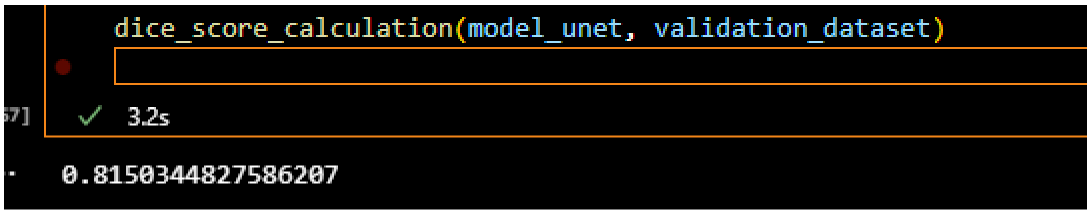
Changing size of filter for convolution from 64 to 128 and 256 made my GPU abort and the training suddenly stopped, so the best was size of 64. I also tested 32 filter block but it seemed worse.

I decided to put batch norm which helped to improve the situation a little.

The final U-net model had 4 conv blocks with batch-norming after convolution and relu activation. Filter block was 64 and size of images in dataset also 64.

For simple dataset





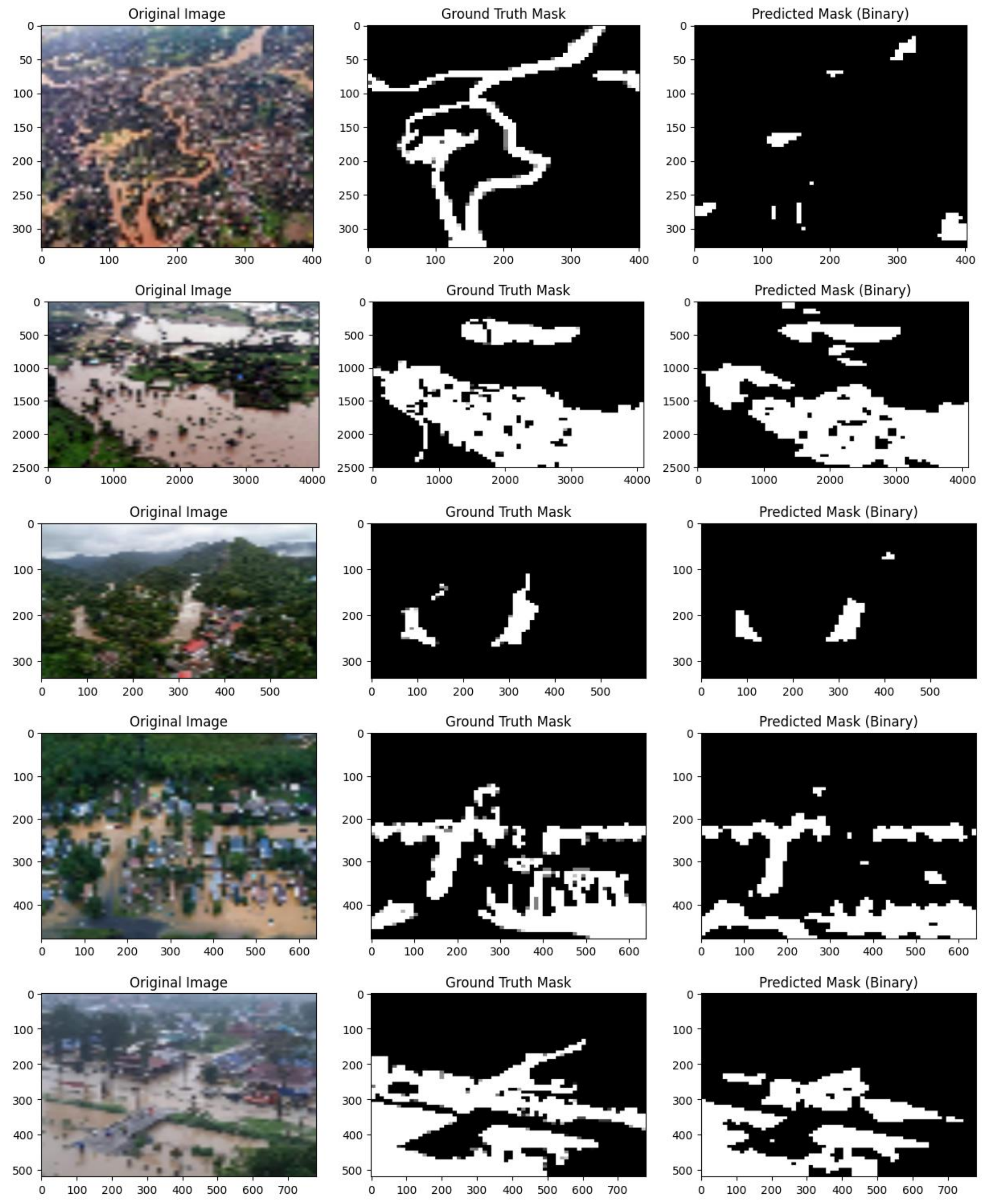
```
dice_score_calculation(model_unet, validation_dataset)

✓ 3.2s

0.8150344827586207
```

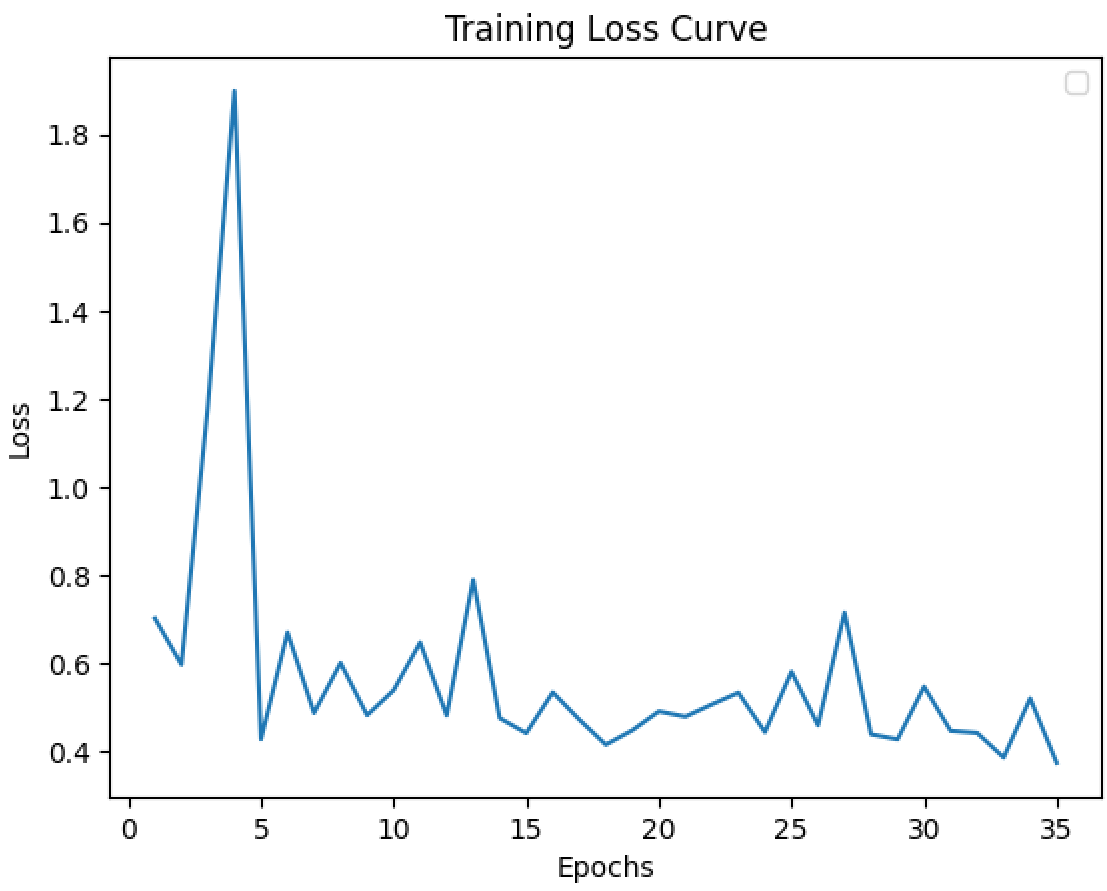
Not a good result but what we have…

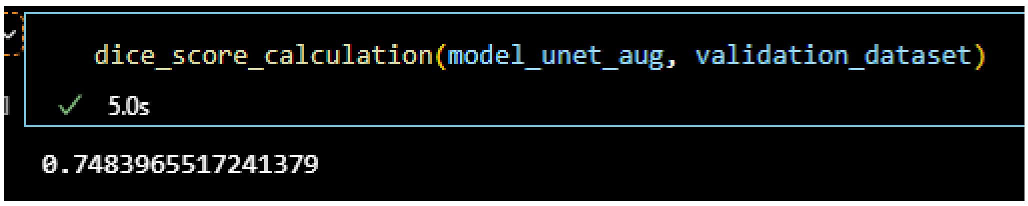
Predictions



Can see how model missclasifies small and detailed objects to water or can’t detect stream of water with small objects and I think the main reason here is the poor quality of images in dataset. Sometimes it made wrongs prediction based on color and contrast – like sky was classify as flooded territory.

For augmented dataset

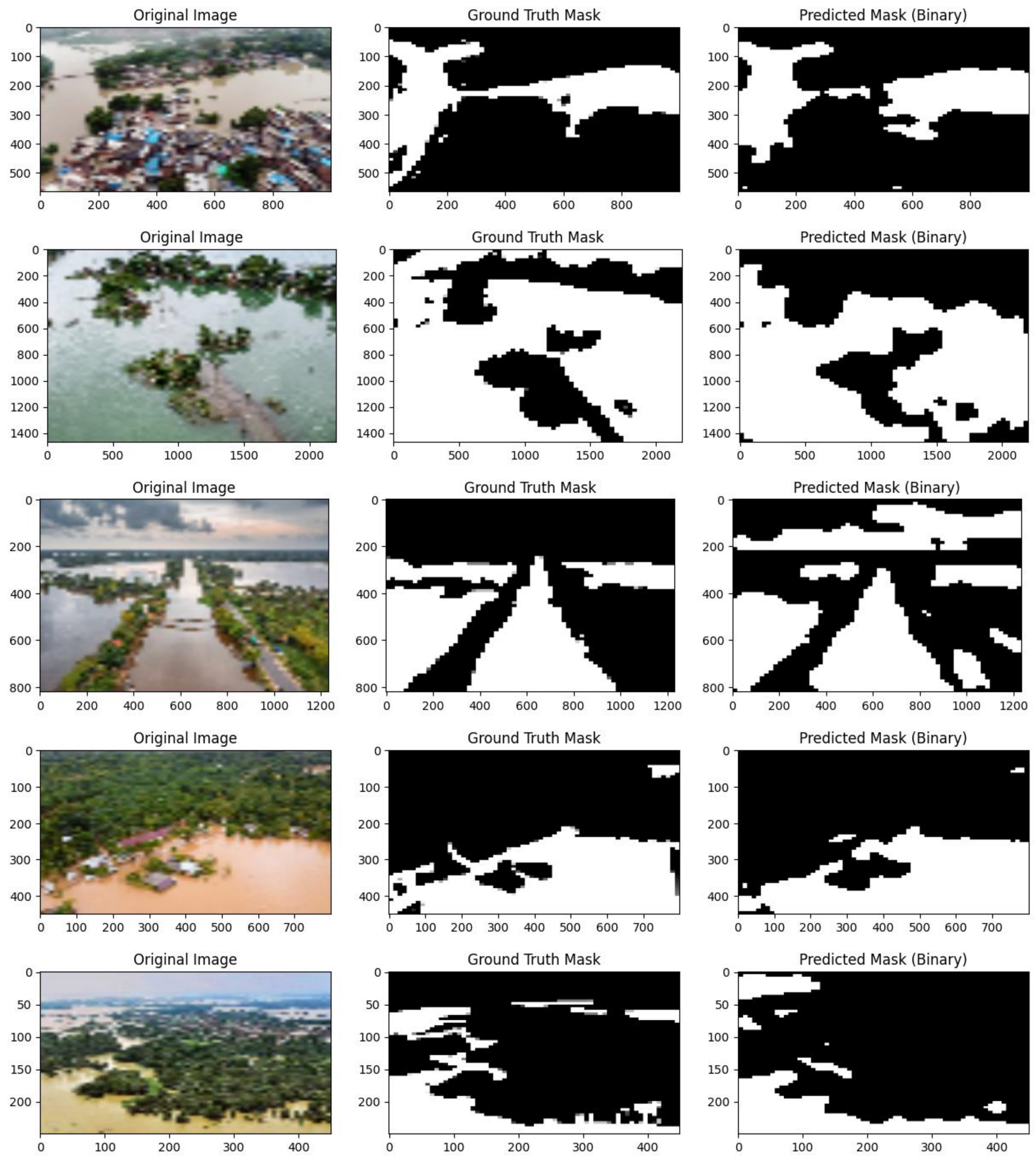




```
dice_score_calculation(model_unet_aug, validation_dataset)
✓ 5.0s
0.7483965517241379
```

Even worse results as in our simple dataset and I was guessing why already above.

Predictions



Same problem as in simple dataset and here we clearly can see how in third image sky misclassified as water.

In conclusion the most failure was the I can’t reach 0.9 KPI dice metric however I’ve tried different configurations and datasets modifications.

Future work:

Firstly, I would expand the number of photos in both the training and validation sets. While 300 photos might be sufficient for training (considering the Central Limit Theorem), increasing the dataset with more images would likely improving segmentation outcomes.

Also, I was planning to conduct a series of experiments with parameters like batch size, learning rate, and image size. With CUDA technology which is boosting efficiency a lot my PC has hardware limitations restrict computations. With a better GPU and more RAM, I could reduce the learning rate to enhance training accuracy, and increase batch sizes for smoother convergence. Of course, this would affect training time but also will lead to another and maybe better results.

Important note that it's common in deep learning to fine-tune pre-trained models for specific segmentation or classification tasks and it will be nice to take pre-trained model and retrain for our case

All code is available in my colab