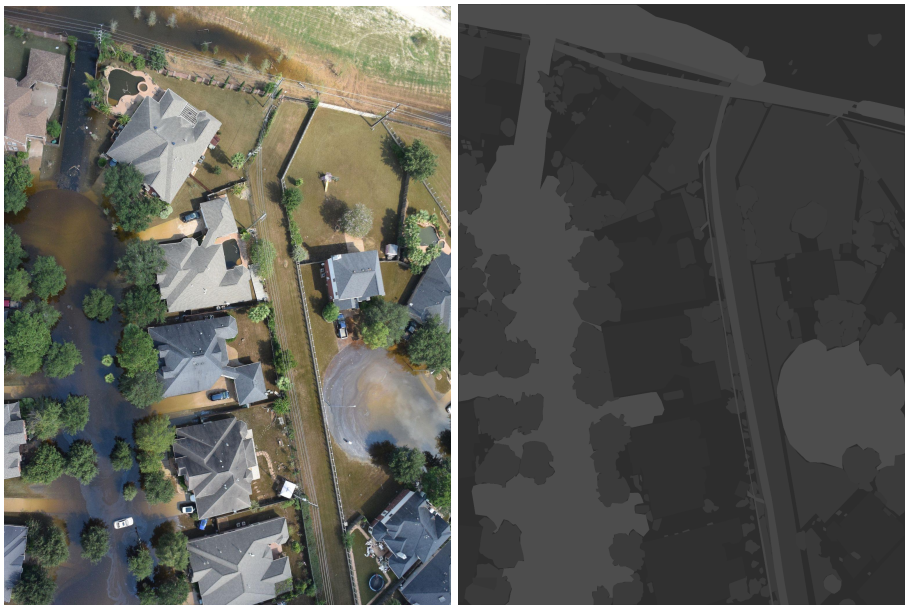
Foundations of Deep Learning
Project Report
Segment features around residential buildings in UAV images of flooded areas taken in Houston after Hurricane Harvey

Team $\hat{\sigma}$*
Sarah Abdelbar
Josh Hiepler
Daniel Roca

I. Introduction

The aim of this project was to implement and solve a remote sensing task using recent deep learning techniques. In particular, the main objective of this challenge was the segmentation of images acquired by a small UAV (sUAV) in Houston, Texas. In total there are 25 object categories:

0: Background	5: Grass	10: Window	15: Under Construction/ In Progress Status	20: Industrial Site
1: Property Roof	6: Trees/Shrubs	11: Satellite Antenna	16: Power Lines & Cables	21: Dense Vegetation/Forest
2: Secondary Structure	7: Solar Panels	12: Garbage Bins	17: Water Tank/Oil Tank	22: Water Body
3: Swimming Pool	8: Chimney	13: Trampoline	18: Parking Area - Commercial	23: Flooded
4: Vehicle	9: Street Light	14: Road/ Highway	19: Sports Complex/ Arena	24: Boat



Training image and corresponding segmentation mask

The above pictures show an example of the original image and the mask that is used to identify the segmented objects.

The task was to design and implement a deep learning model to perform the automatic segmentation of such images with their masks. The model can be trained using the train images which contain pixel-wise annotations represented in the image mask. The following section will explain the deep learning algorithm that was used for the segmentation.

II. Methodology

To implement the segmentation model, a series of steps were designed using PyTorch to extract the data, create the model, train, and predict.

1. Data Extraction
   The wget package was used to extract the train images, train masks and test images to the python environment to be able to use them as training and prediction input. The original images had a size of 3000x4000 or 3072x4096 pixels respectively.

2. Data Augmentation
    - Patchify Library
        - To increase the training set and allow the model to generalize better, the Patchify package was used to split the larger original images to 512x512 patches. Since we resized the original images to 1024x1536 this allowed us to create 6 training images + a resized original image from each source image.
    - Mirroring & Rotation
        - To further expand our available training data we then applied a series of transformations to each image, including mirroring and 90 degree turns to either side, increasing our training data by another factor of 4.

    Overall, our data augmentation allowed us to turn our initial 261 training images and masks into a total of 5220 images. We then converted the training data to tensors of the size 512x512.

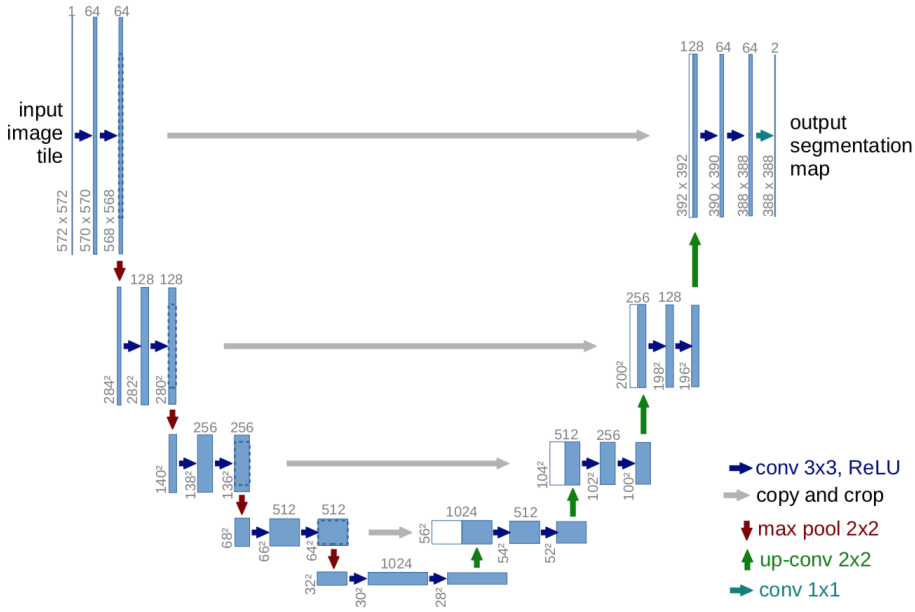
    The training data was split into train and validation sets in a ratio of 60/20 to assess the performance of the neural network and to review the losses to avoid overfitting the model with the training data.

    The test data was also normalized and converted to 512x512 so that the prediction task can be handled appropriately with the trained model.

3. Model creation

    As this is an image segmentation task using convolutional neural networks, we chose a U-Net architecture which is one of the most popular architectures for semantic segmentation (Ronneberger et al., 2015). To improve accuracy and reduce the required training time (Cao & Zhang, 2020), we added a popular modification in the form of a pre-trained ResNet18 encoder (https://pytorch.org/vision/stable/models.html). Our model was heavily based on an existing architecture by Naoto Usuyama (https://github.com/usuyama/pytorch-unet).

The ResNet is composed of five convolutional networks, followed by an average pooling layer and finishes with a softmax function. After that, the architecture is followed by the other half of a U-Net architecture. The following image shows the base implementation of the U-Net, where in our implementation, the downsampling portion on the left side is replaced by the ResNet architecture.



U-net layer composition. Source: [Towards Data Science]

The first convolutional modules encode the image, while the last ones after the bridge increase again in size, producing a segmentation map with the size of the original input that aims to segment the image into 25 object classes.

4. Training
   The model was trained on Colab Pro using a NVIDIA P100 GPU with 16GB VRAM to reduce processing time. We used the following components for Training:

Parameter	**Value**
Optimizer	Adam
Loss Function	Cross Entropy Loss
Learning Rate	0.0001
Epochs	20
Batch Size	3

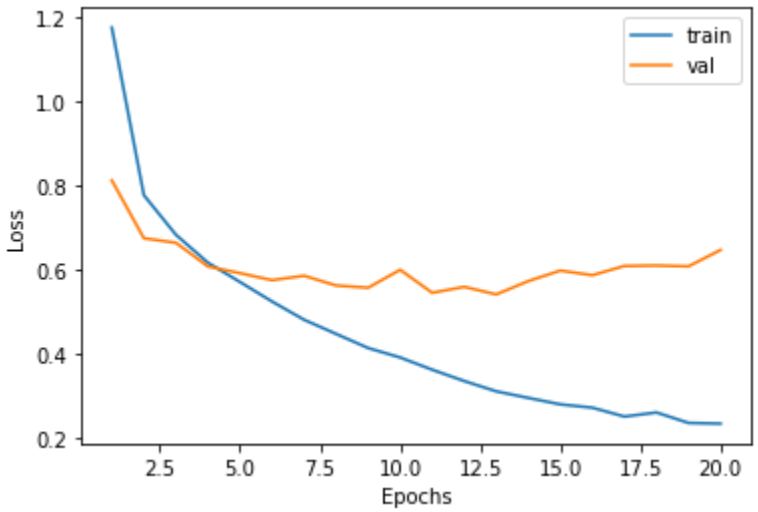
5. Prediction
   The processed test images were used as inputs for the trained model to obtain the predicted masks. These one-hot encoded masks had dimensions of 25x512x512. We then used an argmax function to obtain a mask array of dimension 512x512 with each pixel being assigned a class represented by an integer, and converted the arrays into png images. With the help of the provided prepare_submission.py file, the submission file for Kaggle was then created from our predicted masks.

III. Results

After training our model achieved a prediction score of 0.76640 on 20% of the provided test data, putting us at 4th place (at the time of writing). Since we do not have any scores for the remaining 80%, this score is preliminary and subject to change.

IV. Discussion

Considering the score achieved by our model we can observe that our combined ResNet-UNet implementation performs very well on the provided dataset. Looking back at the steps taken to improve our model, it becomes clear that data augmentation and the inclusion of a pre-trained ResNet18 architecture represented two of the most effective steps to improve performance. Even though we attempted to prevent overfitting by providing a broad and varied training set, our loss curve shows that our validation accuracy quickly reached a minimum, while training accuracy kept increasing, implying that the models already began to overfit to the training data and that further epochs would not be likely to further improve performance. As a result, we suggest that performance can only be improved by taking further steps in the areas of data augmentation and improvements of the model's architecture.



Training and validation loss curves

References:

Cao, K., & Zhang, X. (2020). An Improved Res-UNet Model for Tree Species Classification Using Airborne High-Resolution Images. *Remote Sensing*, *12*(7), 1128. https://doi.org/10.3390/rs12071128

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In N. Navab, J. Hornegger, W. M. Wells, & A. F. Frangi (Eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (pp. 234–241). Springer International Publishing. https://doi.org/10.1007/978-3-319-24574-4_28