Hurricane Harvey Dataset Challenge - Foundations of Deep Learning project

Łukasz Pszenny, Abhishek Tiwari

1 Introduction

This challenge focuses on using machine vision models to identify residential property features and neighborhood assets after a major flooding event like Hurricane Harvey. We will develop an accurate dense segmentation model for aerial post-flood imagery in the Houston area using high resolution image data. The goal is to accurately identify and segment assets that may or may not be damaged by the storm, and to use machine learning and Earth Observation data to design effective Humanitarian Assistance and Disaster Relief interventions. However, the scarcity of historical data has been a hurdle in the development of practical machine learning-based methods. This challenge aims to address this challenge and improve technology for disaster response and management.

2 Dataset and Task

The dataset was first introduced in following paper (Rahnemoonfar et al., 2020). The Harvey dataset consists of video and imagery taken from several flights conducted between August 30 - September 04, 2017, at Ford Bend County in Texas and other directly impacted areas by small UAV platform, DJI Mavic Pro quadcopters, after Hurricane Harvey. The dataset is unique as it contains imagery taken during the response phase by emergency responders and is the only known database of sUAV imagery for disasters. The images are very high in resolution and demonstrate post-flooded damages to affected areas. The dataset is prepared for semantic segmentation and visual question answering. For this challenge, we have considered only imagery.

The raw data file contains 374 images in .tif format. Of those 374 images, 299 have their corresponding masks, while 75 are the test set. When looking at the resolutions, we found they had two different sizes, as follows:

Resolution	Count of images
4592x3072	111
4000x3000	263

As mentioned on the website, the data contains 27 classes, with 22,419 annotation vectors. The frequency of these annotations are shown in Figure 1. Trees/shrubs have the highest annotation vectors in the dataset of 9644, while sports complex/arena is 2.

Along with the Harvey Hurricane dataset, there was also the Hurricane Harvey Synthetic dataset, which contains 2093 images, with 156,933 annotation vectors. Here, the images are in 5 different qualities - High, Medium, Low, Very Low, and Unrealistic.

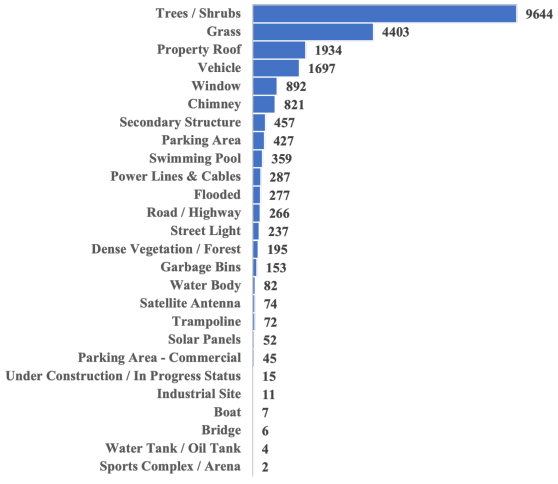

Figure 1: Frequency of classes in the challenge dataset

3 Model description

In this study, we evaluate the performance of several state-of-the-art models for image semantic segmentation which we tried to optimize to obtain highest possible Multiclass Dice Score on unseen test set. In this section we present used models:

3.1 DeepLabv3+

Deeplabv3+ is a model for semantic image segmentation, which uses atrous convolutions and a simple yet effective decoder module to improve segmentation accuracy. It also employs a feature pyramid network (FPN) to incorporate both high-resolution and semantically strong features, which helps to improve the segmentation results in small objects and objects with fine details. It addresses the limitations of previous models and it is designed to work with large images and handle multiple objects.

3.2 U-Net

U-Net is a convolutional neural network (CNN) architecture for semantic image segmentation. It is composed of an encoder and a decoder. The encoder is made up of a series of convolutional and pooling layers that reduce the spatial resolution of the input image while increasing the number of feature maps. This allows the network to learn more abstract and high-level features of the image. The decoder, on the other hand, is made up of a series of upsampling layers that increase the spatial resolution of the feature maps while decreasing the number of feature maps. This allows the network to refine the segmentation by combining the high-level features learned in the encoder with the spatial information from the input image. Additionally, U-Net uses skip connections between the encoder and decoder to concatenate the feature maps from the corresponding levels of the encoder to the decoder, which helps to improve the segmentation results.

3.3 SegFormer

Transformers are increasingly being applied to problems in computer vision based on their great performance in natural langauge processing tasks. Vision transformer, proposed for image classification task, performed well on ImageNet and showed promise for applying transformers to vision tasks. Another model, SETR was proposed for semantic segmentation which used ViT as a backbone. However, these models are computationally expensive. SegFormer is a recently proposed model (Xie et al., 2021) which addresses some of the limitations of past transformer-based architectures. SegFormer uses a hierarchial encoder to generate multi-scale features as opposed to the single-scale features generated by ViT or SETR. The encoder is hence, able to produce both highly local and non-local attentions, and has a larger effective receptive field than convolution models such as DeepLabv3 at their deepest stage. It uses a lightweight decoder which makes it computationally less expensive. Further, it does not use positional encodings which allows it maintain performance even when the test images may vary in resolution from the training images.

4 Training criterion

4.1 Losses

In this section, we present the loss functions that were utilized during the training of our models. Our objective was to design a model that maximizes the multiclass Dice coefficient, so we initially trained our models using the Dice loss[1]. However, upon examining the gradient forms of this function, we found that the model was susceptible to issues such as exploding or vanishing gradients. We then experimented with more robust loss functions, which are also commonly used in semantic segmentation tasks, such as the multiclass cross-entropy loss [2]. Despite the drawbacks of cross-entropy, such as poor performance on imbalanced datasets (which is the case in our study), we attempted to mitigate this by using class weights and also training with Focal Loss [3] which is a variation of cross-entropy. Focal loss applies a modulating term to the cross entropy loss in order to focus learning on hard misclassified examples (Lin et al., 2017). As proposed in the original paper we used parameters $\gamma = 2$ and $\alpha = 0.25$. However we are aware these parameters could be further optimized.

4.2 Weightning losses

To tackle class imbalance problem we tried to adjust weights of specific class impact on loss function by following formula:

$$w_c = \frac{N}{c \cdot n_c}$$

where N is a total number of pixels in dataset, c is the number of classes, n_c is the number of pixels belonging to class c

[1]We implemented dice loss on our own based on implementation provided under this link https://gita.utia.cas.cz/schier/pytorch-3dunet/-/blob/97e0311e84eb54c4b3858ef2be3deaf97b0d1d82/pytorch3dunet/unet3d/losses.py

[2]We used implementation from https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

[3]We implemented Focal Loss based on https://github.com/zhezh/focalloss/blob/master/focalloss.py

5 Data preprocessing

The SegFormer model requires an input image of 512x512 in dimension. To meet this requirement, all images were resized to these dimensions. Furthermore, we performed experiments on normalizing the range of pixel intensity values. The models were trained on both normalized images that were computed using the mean and standard deviation of ImageNet-1K, as well as on raw data. The findings of these experiments are discussed in the "Experiments and Results" section. The experiments were conducted on a data split into two sets: validation (consisting of 20% of images) which was not used to update the model parameters, and training (consisting of 80%) which was used to update the model parameters each epoch.

6 Experiments and results

Our approach consisted of four main stages:

1. Evaluation of-the-shelf performance of models described in previous section
2. Optimizing performance of the few best performing models
3. Identification of models drawback and proposing solution
4. Training the best performing model on the whole dataset and evaluation

All stages consist of deadened ideas that didn't result in increase in performance, or very poor increase. We present them in "Deadends" section.

In all experiments involving the Segformer, we utilized an initial learning rate of 5e-5, with a learning rate scheduler on a plateau, as proposed in (Gupta and Mishra). The scheduler reduces the learning rate by 0.1 when there is no improvement in validation metric for 10 epochs. The best model, based on the dice coefficient on validation data, was saved in .pt format at the end of each epoch so that predictions could be made using it. The models were trained for either 60 or 30 epochs, depending on the situation with batch size 4. All training data metrics across epochs using SegFormer architecture, can be found under this link [4]

[4] https://wandb.ai/pszenny_t/FloodNet_SegFormer?workspace=user-lpszenny (This link is a a fullfillment of "1 figure with qualitative results" requirement)

6.1 SegFormer tuning

We started optimization of b0-SegFormer model by considering different losses described in section "Losses". For Dice loss and Cross entropy loss we experimented with applying weights to tackle class imbalance problem. This resulted in training 10 models. All training information can be found in table 1. We can see that whenever normalization was used the model obtained slightly better test score and training Dice Score, this is however not true for validation Dice score. We can also notice that test score doesn't vary significantly (it is in range from 74 to 76 - not considering training with weighted Cross Entropy loss) even though the validation metric vary from 73 to 81.

6.2 Interpolation of model output to real image

The output of our transformer model is a 128x128 mask with pixel classes, which needs to be enlarged to match the original mask size. There are various ways to achieve this, so we experimented with different methods and evaluated their impact on the validation Dice score. The methods we considered were 'nearest', 'bilinear', 'bicubic', 'area', and 'nearest-exact'. We found that the mean Dice score on the validation dataset varied by 0.4 depending on the method used. However, we do not consider this difference to be significant as the standard deviation of the Dice score per image is 13.4. Ultimately, we decided to use bilinear interpolation for all experiments.

6.3 Analysis of predictions

In figure 2 we gathered predictions made on validation and training set for randomly choosen images. We can see that models learns to classify regions quite well, despite 2^{nd} example where he missclassified huge area of Forest class. First thing worth noticing it is the fact that it has problems with correct classifying pixels on borders of classes which can be seen by red thin areas around objects in 3^{rd} column. Secondly it struggles with correct classifying small objects like cars which can be seen on 5^{th} image in 3^{rd} column.

6.4 Proposing own decoder architecture

As we noted in the previous subsection, the model has a problem identifying objects of small size and correctly identifying the class of pixels at the boundaries of two different classes. We hypoth-

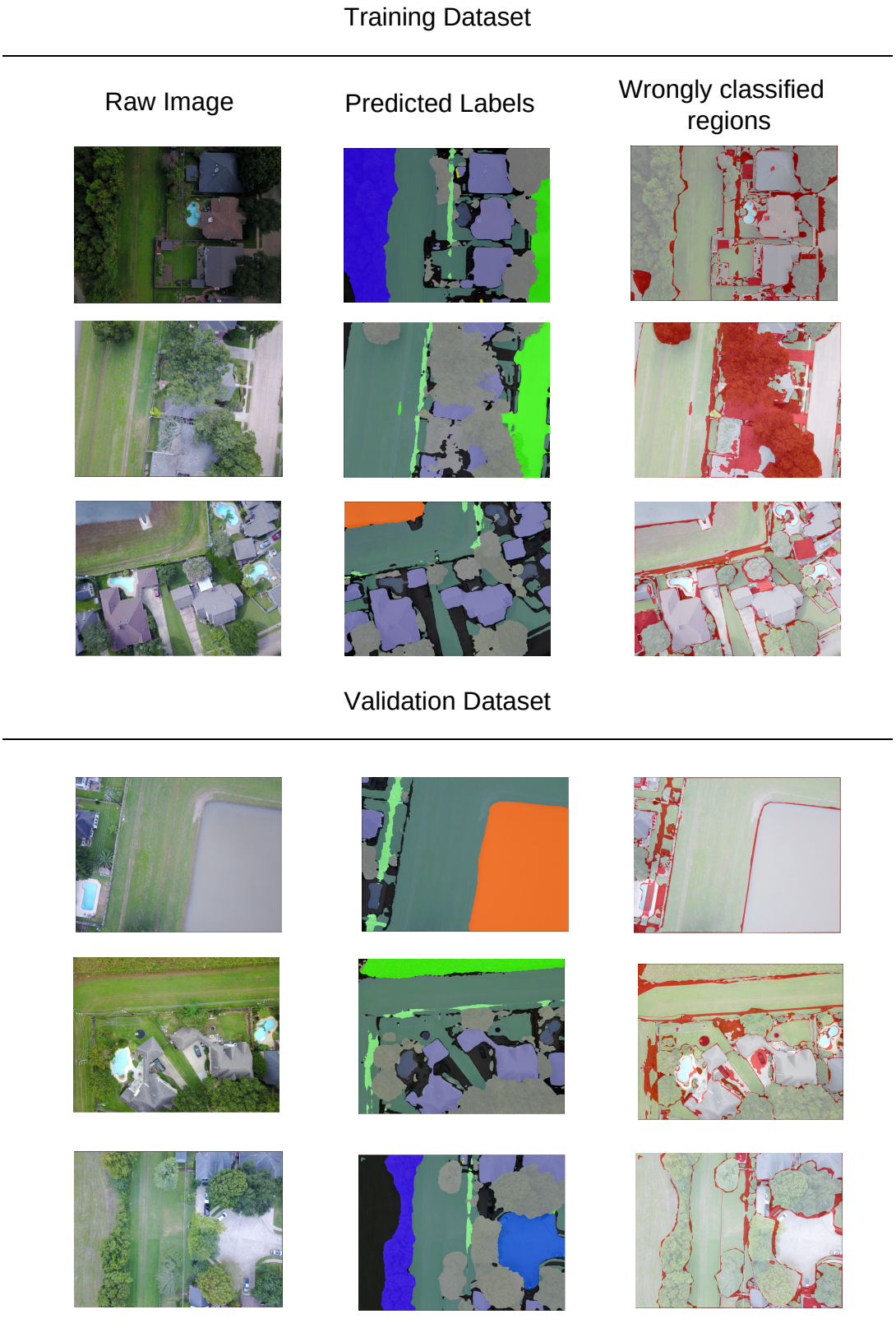

Figure 2: Predictions using b0-SegFormer model trained with Focal Loss and data normalization on randomly choosen photos from training set and validation set. Dice Score from top to bottom are following: 89,86 ; 60,53 ; 87,72 ; 94,04 ; 88,12 ; 91,66. Image file numers from top to bottom : 6802, 6421, 6467, 6462, 10817, 9723

esize that the reason for this is the loss of information through the image resolution that is in the model output of 128x128 pixels and the naive UpSampling applied to the output. After in-depth research , inspired by (Kumar et al., 2022) we decided to propose a new decoder architecture. SegFormer encoder generates 4 embeddings of different sizes encoding local and global information. The proposed decoder consists of 5 stages. As input, the decoder receives the deepest hidden state (in the case of the b0 model with sizes [256,16,16]). In each stage using Bilinear UpSampling, the resolution is doubled. Then using 3x3 convolution with GELU activation function, the number of channels is reduced twice. This produces a tensor with a dimensions of the next deepest unused hidden state. This vector is concatenated across channel dimension with the newly created vector and passed to the next stage. After 4th stage there is no unused hidden states left but we still need to pass it through 5th stage to upsample and reduce channel number. Finally, a 1x1 convolution is applied reducing the number of channels to the number of classes. The architecture is shown in the image 3, the encoder is a b0-SegFormer encoder with weight initialized from full SegFormer model trained with Focal loss and normalization. We hoped to perform intelligent UpSampling which incorporates powerful embeddings generated by transformer so that the input size is equal to the output size. Experiments showed significantly higher dice coefficent on the validation dataset. Ultimately, however, the test score did not improve (75,24 test score, while validation Dice coefficient was equal to $\approx$85). We trained the model for 30 epochs and best validation score was obtained in epoch 19.

6.5 Impact of model size

It is well entrenched belief that bigger models perform better. We decided to validate this belief on this task by changing our SegFormer model from version b0 to version b4 (the biggest we could fit into our memory). Version b0 has 3,7M parameters while b4 has 62.6M parameters. The rest of hyperparameters (loss, whether to apply weights and normalization) stayed the same as the best performing model on test score from table 1. Indeed we obtained the highest test score even though metrics during training didn't indicate so. This questions our training scheme - namely dice coefficient computation (test score can be computed differently as we assumed - this stands in line with fact the we found two formulas of computing dice score in the internet) or may be associated with significant difference between photos provided in test and training set. It is worth mentioning that improvement on test score wasn't big as compared to experiment witch b0-SegFormer.

6.6 Final model trained on all labeled data

We finally trained best performing model out of b0-Segfomer models on all provided data without validation step, namely - with Focal Loss and normalization applied to data. We decided to train for 40 epochs as on 80% of data model achieved best performance after 31 epoch, so to incorporate bigger size of dataset we decided to increase epoch number a little. We obtained test score: 76,99. This was the highest test score we achieved in our experiments.

6.7 Deadends

1. In the initial phase of our study, we evaluated the performance of various neural network architectures on a specific dataset. Among the models analyzed, U-Net and DeepLabv3+ were identified as potential candidates for further investigation. Upon analyzing the results of U-Net, we determined that its performance on the validation set was inferior to that of DeepLabv3+.

2. As a result, we elected to only utilize DeepLabv3+ for testing. The results yielded a Dice Score of 61.91 on the test set. Given the superior performance of the SegFormer transformer that were concurrently evaluated, we elected to proceed with further development and implementation of that model.

7 Further Ideas

1. We never used synthetic data to pretrain our model and fine-tune it on real image data because of computing power limitations. We hypothesize that by seeing more images the model can learn to generelize better.

2. Once again because of computing power limitation we never tried an idea of splitting images in patches, training model on patched data and evaluation on composed predictions on each patch. On one hand this approach limits loss of close neighbour information around

Normalization	Loss	Using weights	Best epoch	training Dice Score	validation Dice score	test score
True	Dice	True	1	85.18	74.97	76.24
True	Dice	False	44	84.97	80.54	74.43
True	Cross Entropy	True	4	9.67	7.91	8.97
True	Cross Entropy	False	39	90.26	77.56	75.19
True	Focal	False	31	90.52	75.05	**76.54**
False	Dice	True	46	74.90	**81.51**	74.23
False	Dice	False	58	82.77	80.80	74.40
False	Cross Entropy	True	2	6.48	4.90	15.29
False	Cross Entropy	False	47	86.57	75.60	74.97
False	Focal	False	17	87.71	73.02	74.92

Table 1: SegFormer b0 performance with different loss, image normalization using ImageNet mean and standard deviation and adjusting weights of classes on loss function

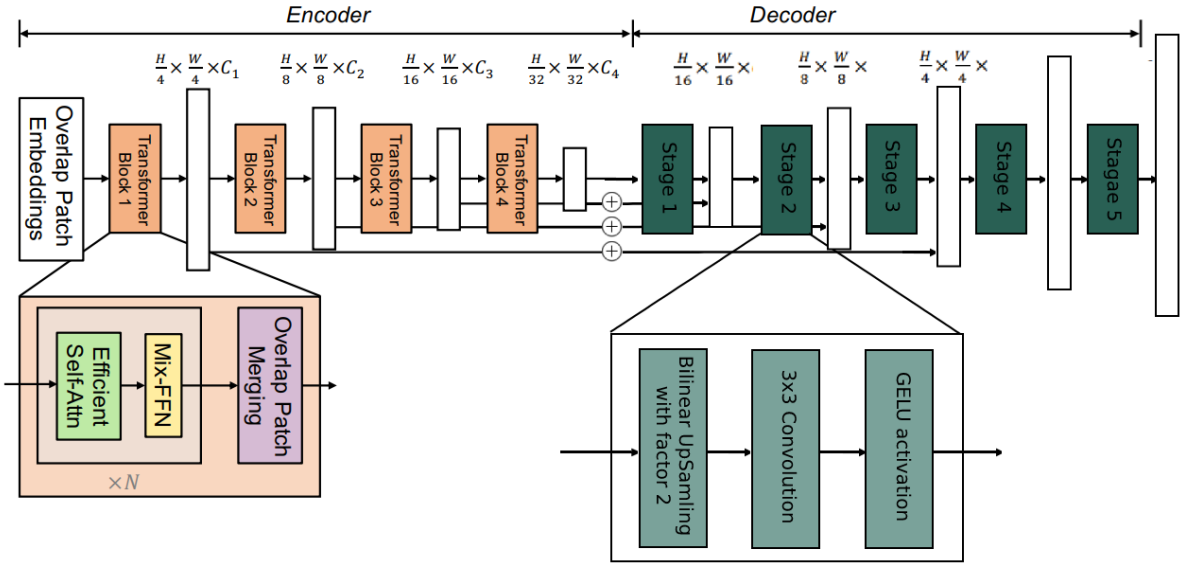

Figure 3: Decoder architecture proposed as solution to low resolution output, plus in circle denotes concatenation, convolution layer downsizes channel number by factor 2 while Bilinear Upsampling increases resolution by factor 2. Encoder architecture is an architecture of encoder in SegFormer

pixels in input coming from resizing and the final representation 128x128 pixels can better incorporate information about image but on the other model losses ability to look at long range dependencies (it has no access to information which is not in the patch). We believe this approach should be examined in future

References

Kushagra Gupta and Priya Mishra. Post-disaster segmentation using floodnet. *studies*, 13:8.

Satyawant Kumar, Abhishek Kumar, and Dong-Gyu Lee. 2022. Semantic Segmentation of UAV Images Based on Transformer Framework with Context Information. *Mathematics*, 10(24):1–17.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection.

Maryam Rahnemoonfar, Tashnim Chowdhury, Argho Sarkar, Debvrat Varshney, Masoud Yari, and Robin R. Murphy. 2020. Floodnet: A high resolution aerial imagery dataset for post flood scene understanding. *CoRR*, abs/2012.02951.

Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. 2021. Segformer: Simple and efficient design for semantic segmentation with transformers. *CoRR*, abs/2105.15203.