Foundation of Deep Learning
Kaggle Competition

Team: Check Mic
Members: Zhengxiao YING, Yue CHEN, Jiaqian MA

Overview

We have trained four models (two for Unet, and two for Deeplabv3) to achieve image auto-segmentation, and we compared their performance.

1.Data Description

This dataset contains uni-sized 261 train images containing pixel-wise annotations and 112 test images which were acquired by a small UAV (sUAV) at the area on Houston, Texas. These images were obtained in order to assess the damages on residential and public properties after Hurricane Harvey. In total there are 25 categories of segments (e.g. roof, trees, pools, background, etc.).

2.Data Pre-processing

Our data pre-processing consists in data preparation and data augmentation.

2.1 Data Preparation

After loading all the training and testing images, then we transformed images into array form for further processing.

2.2 Data Augmentation

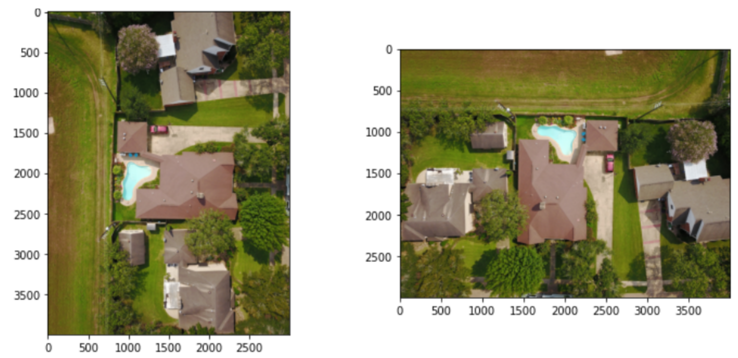

Fig 1. Original image(left) and image after data augmentation(right)

This dataset only provides 261 images for training which might not be enough for training a good deep neural networks model. Data augmentation can help to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a neural network model.[1]

[1] Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0

Data augmentation is not applied on test and validation data. Since test and validation data should be representative of the original data and is left unmodified for unbiased evaluation. Normally, geometric transformations, flipping, color modification, cropping, rotation, noise injection and random erasing are used to augment image in deep learning. In this case, we rotate the images and their corresponding masks by 90 degrees, and then flip them horizontally and vertically, both with a 100% probability.

3.Model Building

3.1 Optimize Strategy and Loss Function

3.1.1 Optimizer

We applied two optimizers to minimize the loss functions.

1) **SGD**[2]
   Stochastic gradient descent (SGD) is an iterative method for optimizing an objective function. It can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient by an estimate thereof (calculated from a randomly selected subset of the data). Especially in high-dimensional optimization problems this reduces the computational burden, achieving faster iterations in trade for a lower convergence rate.

2) **Adam**[3]
   Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iteratively based on training data. Adam combines the advantages of two other extensions of stochastic gradient descent. Adam's advantages: The method is too fast and converges rapidly. Rectifies vanishing learning rate, high variance. Adam's disadvantages: Computationally costly.

In this task, since the dataset is not too big, we assume that Adam will have a better performance and meanwhile the cost computation can be accepted here.

3.1.2 Loss Function

For multi-classification problems, there are a number of loss functions to choose from, we used cross-entropy loss and dice loss for model evaluation .

1) **Cross-Entropy Loss**[4]
   Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. In this multiclass classification problem, we calculate a separate loss for each class label per observation and sum the result. The method to calculate Cross Entropy Loss is as follows:

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

   Where M is number of classes (25 in our case), y is binary indicator (0 or 1) if class label c is the correct classification for observation o, p is predicted probability observation o is of class c.

[2] Sra, S., Nowozin, S., & Wright, S. J. (Eds.). (2012). Optimization for machine learning. Mit Press.
[3] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. ICLR 2015. arXiv preprint arXiv:1412.6980, 9.
[4] Zhang, Z., & Sabuncu, M. R. (2018, January). Generalized cross entropy loss for training deep neural networks with noisy labels. In the 32nd Conference on Neural Information Processing Systems (NeurIPS).

2) **Dice Loss**
   Dice Loss is widely used in medical image segmentation tasks to address the data imbalance problem. The Dice coefficient can be used to compare the pixel-wise agreement between a predicted segmentation and its corresponding ground truth. The formula is given by:

$$\frac{2 * |X \cap Y|}{|X| + |Y|}$$

   where X is the predicted set of pixels and Y is the ground truth. The Dice coefficient is defined to be 1 when both X and Y are empty. Dice score is the mean of the Dice coefficients for each (Image, Label) pair in the test set.

3.2 U-Net[5]

3.2.1 Model description

Based on a fully convolutional network, U-Net works with very few training images and yields more precise segmentations. The main idea is to supplement a usual contracting network by successive layers, where pooling operators are replaced by upsampling operators. Hence, these layers increase the resolution of the output. In order to localize, high resolution features from the contracting path are combined with the upsampled output. A successive convolution layer can then learn to assemble a more precise output based on this information.
In the upsampling part U-Net has a large number of feature channels, which allow the network
to propagate context information to higher resolution layers. As a consequence, the expansive path is more or less symmetric to the contracting path, and yields a u-shaped architecture.

3.2.2 Model Architecture

The network architecture consists of a contracting path (Encoder) and an expansive path(Decoder). The contracting path follows the typical architecture of a convolutional network.It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a ReLU and a 2x2 max pooling operation with stride 2 for downsampling.
At each downsampling step we double the number of feature channels. Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution (“up-convolution”) that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU.

3.3 DeepLabv3+[6]

3.3.1 Model description

Fully Convolutional Neural Networks (FCNs) are often used for semantic segmentation. One challenge with using FCNs on images for segmentation tasks is that input feature maps become smaller while traversing through the convolutional & pooling layers of the network. This causes loss of information about the images and results in output where predictions are of low resolution and object boundaries are fuzzy. The DeepLab model addresses this challenge by using Atrous convolutions and Atrous Spatial Pyramid Pooling (ASPP) modules. This architecture has evolved over several generations:

[5] Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham.
[6] https://developers.arcgis.com/python/guide/how-deeplabv3-works/

DeepLabV1 uses Atrous Convolution and Fully Connected Conditional Random Field (CRF) to control the resolution at which image features are computed. DeepLabV2 uses ASPP to consider objects at different scales and segments with much improved accuracy. Apart from using Atrous Convolution, DeepLabV3 uses an improved ASPP module by including batch normalization and image-level features. It gets rid of CRF (Conditional Random Field) as used in V1 and V2.

3.3.2 Model Architecture

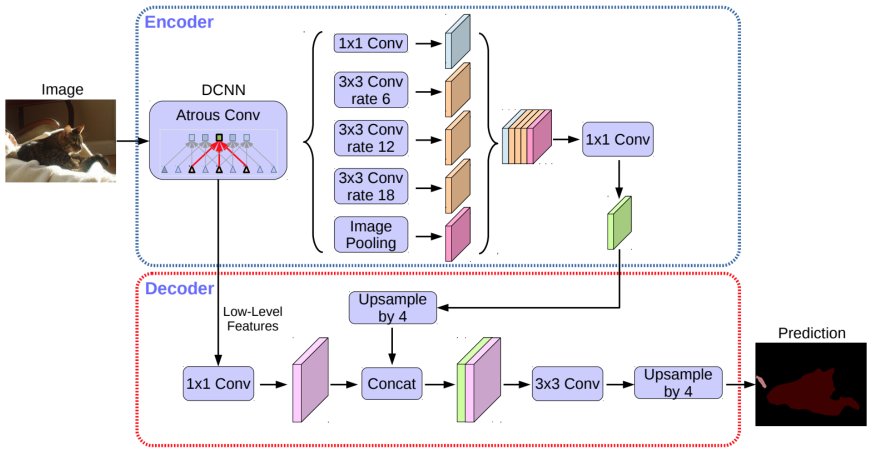

Fig 2. DeepLabV3+ Model Architecture

DeepLabV3+ is an extension of the DeepLabV3 model, which has the following architecture:

- Features are extracted from the backbone network (VGG, DenseNet, ResNet).
- To control the size of the feature map, atrous convolution is used in the last few blocks of the backbone.
- On top of extracted features from the backbone, an ASPP network is added to classify each pixel corresponding to their classes.
- The output from the ASPP network is passed through a 1 x 1 convolution to get the actual size of the image which will be the final segmented mask for the image.

DeepLabV3+ extends DeepLabV3 by adding an encoder-decoder structure. The encoder module processes multiscale contextual information by applying dilated convolution at multiple scales, while the decoder module refines the segmentation results along object boundaries.

4.Model Training and Evaluation

4.1 Metrics and Dataset Split

To compare different models and pick the best performer, we trained several models and looked at their performance on several metrics. Metrics include Training Loss, Validation Loss, Training Loss in the entire dataset, Test Dice Coefficient (on Kaggle). To calculate the Training Loss and Validation Loss, we split the entire training set into training and validation sets with a validation set ratio of 0.15.

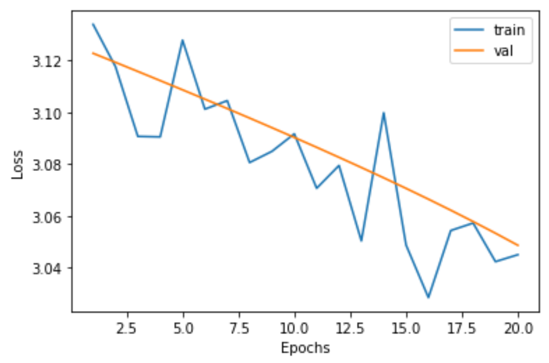

Fig 3. Train loss and validation loss of UNet_1 model (optimizer = SGD)

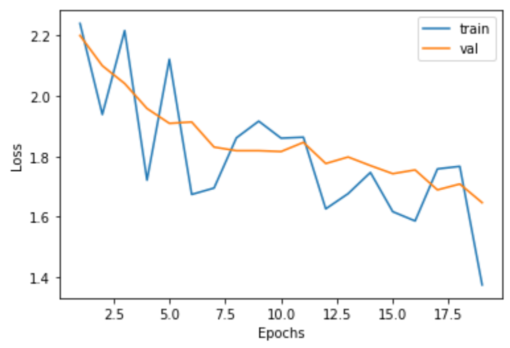

Fig 4. Train loss and validation loss of UNet_2 model (optimizer = Adam)

4.2 Model Training

For U-Net models, we trained basic models with Cross-Entropy Loss, 0.0005 Learning Rate for 20 epochs. The slow convergence speed of SGD (Fig 3,Fig 4) is in line with what we said 3.1, Adam has a better performance and meanwhile the cost computation can be accepted here. So we continue to use Adam in later models.

For DeepLabV3+models, we used a Learning Rate of 0.001 and the loss function *SparseCategoricalCrossentropy*, which is available in Keras and does not require one-hot representation. For DeepLabV3+1 models, we use a ResNet50 pretrained on ImageNet as the backbone model, and we use the low-level features from the *conv4_block6_2_relu* block of the backbone. In DeepLabV3+_1, we did not apply data augmentation and have run 30 epochs. As for DeepLabV3+_2, we applied data augmentation including vertical and horizontal flips as well as 90-degree rotations. We ran 60 epochs for DeepLabV3+_2 since the enlarged dataset requires much more epochs for model metrics to converge.

4.3 Evaluation

Compared with two state-of-the-art semantic segmentation approaches, DeepLabv3+ performs better than U-Net. For U-Net models, as we can see, the model optimized by Adam performs better than by SGD. Both U-Net models have not converged and the one used SGD is far from converging.

For DeepLabV3+ models, metrics converged after certain epochs and the one without data augmentation outperformed the one with data augmentation. It might be due to overfitting since the model with data augmentation performs better in Training Loss while worse in validation Loss and in Test Dice Coefficient.

Table 1. Model performance comparison

Model	**Data Augmentation**	**Optimizer**	**Training Loss**	**Validation Loss**	**Training Loss in the entire dataset**	**Test Dice Coefficient (on Kaggle)**
UNet_1	/	SGD	3.0450	3.0486	2.9149	0.5838
UNet_2	/	Adam	1.3748	1.6470	1.3636	0.6075
DeepLabV3+_1	/	Adam	0.2802	0.9862	/	0.7627
DeepLabV3+_2	√	Adam	0.2660	1.0108	/	0.7213

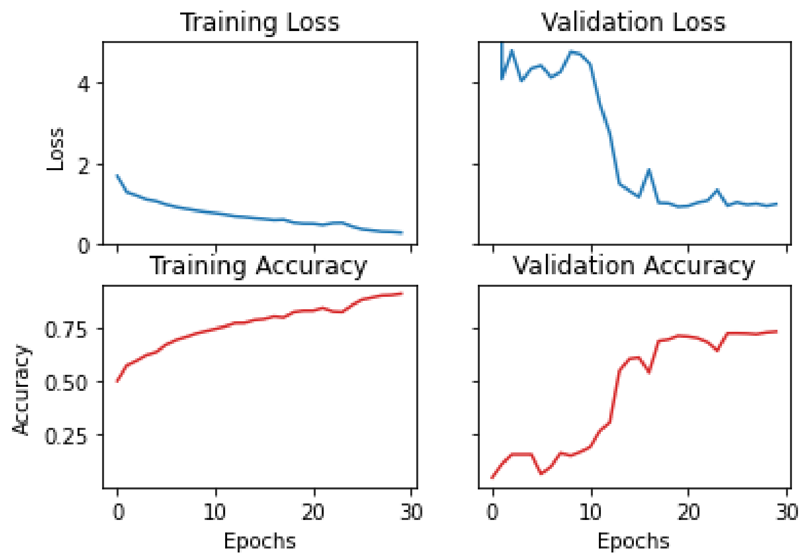

Fig 5. Train Loss and Accuracy v.s. Validation Loss and Accuracy of DeepLabV3+_1 model (optimizer = Adam)

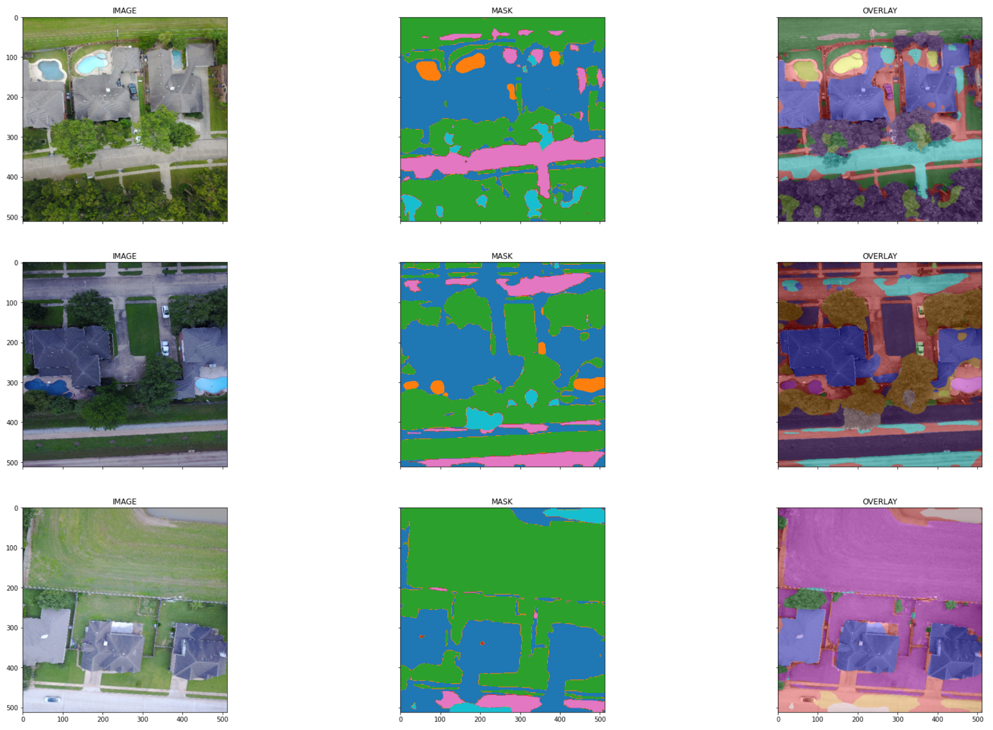

Fig 6. Examples of Segmented Test Images inferred by DeepLabV3+_1 model

5. Discussion

5.1 Limitation

We used cross-entropy loss on training and validation, while testing scores were based on dice loss. The inconsistency of the loss functions might make the test results not as good as expected.

Besides, for multi-class semantic segmentation problems with imbalanced data, choosing a suitable loss function is important. Cross-entropy loss evaluates the class predictions for each pixel vector individually and then averages over all pixels, we're essentially asserting equal learning to each pixel in the image. This can be a problem if various classes have unbalanced representation in the image, as training can be dominated by the most prevalent class.

Also, during the training of DeepLabV3+ models on Google Colaboratory, we encountered the ResourceExhaustedError when we tried to increase batch size or image size of our training data. Therefore, the capacity of the GPU could be limiting the performance of the model as it’s difficult to choose the optimal parameters.

5.2 Further improvement

As we can see in Fig 6, our model has generated relatively accurate segmentation for test images, but some segmentation isn’t still flawed. In some test images, trees cannot be very accurately segmented and the boundary of roads is usually mistakenly segmented, which may be because the shadows on the road have darker pixels and were thus not identified as a part of the road. By looking into the segmented test images and analyzing the shortcomings of our segmentation, we can use different methods to improve the segmentation accuracy of certain specific classes.

Moreover, we only tried ResNet50 and ResNet152 as the backbone model. A recent paper pointed out that employing the Xception model as network backbone on top of DeepLabV3+ can further improve the performance of the model.[7]

If cross entropy loss is used, we can weight this loss for each output channel in order to counteract a class imbalance present in the dataset and improve the model performance.[8]

Image semantic segmentation is a challenge recently tackled by end-to-end deep neural networks. One of the main issues between all the architectures is to take into account the global visual context of the input to improve the prediction of the segmentation. The state-of-the-art models use architectures trying to link different parts of the image in order to understand the relations between the objects.The pixel-wise prediction over an entire image allows a better comprehension of the environment with a high precision. [9]Scene understanding is also approached with keypoint detection, action recognition, video captioning or visual question answering. Thus, the segmentation task combined with these other issues using multi-task loss should help to outperform the global context understanding of a scene.

[7] Chen L C, Zhu Y, Papandreou G, et al. Encoder-decoder with atrous separable convolution for semantic image segmentation[C]//Proceedings of the European conference on computer vision (ECCV). 2018: 801-818.
[8] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3431-3440.
[9] A. Arnab et al., "Conditional Random Fields Meet Deep Neural Networks for Semantic Segmentation: Combining Probabilistic Graphical Models with Deep Learning for Structured Prediction," in IEEE Signal Processing Magazine, vol. 35, no. 1, pp. 37-52, Jan. 2018, doi: 10.1109/MSP.2017.2762355.