# Semantic Segmentation using Convolutional Neural Networks

**Prabhav Gaur**
pgaur@eng.ucsd.edu

**Hsuan-lin Her**
hsher@eng.ucsd.edu

**Ashish Arun Farande**
afarande@ucsd.edu

## Abstract

In this project, we perform the task of semantic segmentation on TAS500 dataset. We develop Convolutional Neural Network (CNN) so as to train and predict the segmentation masks of the images. We experiment with various CNN architectures and also several data augmentation methods that are state of the art. We also implement or own architecture named HAPNet. We then evaluate the performance of these structures using parameters such as loss, accuracy and Jaccard index (IoU). We also visualize the segmented output for the first image in the test set. An IoU accuracy of 51% was achieved with the Baseline Model. This was increased to 56% using the data augmentation and weighted loss, which address the issue of real world data. Further, a well known model for image segmentation was implemented and compared with the baseline model, which yielded approximately 56% IoU accuracy. In addition, we make use of ResNet in order to transfer the learning and train a model for our dataset, which yields a better accuracy (62% IoU). At the end, we design and present a new model which is inspired from the models which performed better at ImageNet and our understanding of different Models. This model was able to yield 63% validation IoU accuracy.

## 1 Introduction

Semantic Segmentation is a process of classifying each pixel in the image under a certain category. Which makes it crucial in various applications, ranging from scene understanding to inferring support relationship among objects in autonomous vehicles, robotics and healthcare. The early methods relied on the low level cues from the image, which were analysed using the image texture, but in the recent days these have been superseded by the advancements in the machine learning algorithms. The current deep learning model are able to achieve tasks like object detection, classification and recognition in images. Now, there has been a lots of interest in scene understanding in most of the applications like autonomous driving and Health care. As understanding the scene helps the system take more precise and accurate decision. Hence, keeping scene understanding for autonomous vehicles as our motivation we do feasibility study on Fully Convolutional Networks (FCNs) [2], U-Net [3], ResNet18 [4] and discuss on new architecture which are inspired from the well known models.

To study semantic segmentation better, we use TAS500 dataset, which is obtained from autonomous driving in unstructured environments. The dataset offers fine-grained vegetation and terrain classes that can be used to learn various natural and artificial objects and obstacles that is often encountered while autonomous driving, Each pixel in the training and validation images is labelled with the corresponding class of the object displayed at that location of the image. These labels include classes such as buildings, bushes, sky, etc. Each label has a corresponding color which encodes the object associated with that label. This dataset has a total 9 coarse and 23 fine-grained object categories. For this work, we will be only using the 9 categories for supervised learning.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

## 2 Related Work

Convolutional Neural Networks [1] is the state of the art for semantic segmentation. In this report, we have used Fully Convolutional Networks (FCNs) [2], U-Net [3] and ResNet18 [4]. We also used Inception Network [5] in our proposed architecture and combine it with prevalent models to improve the performance. FCNs have upsampling layer to balance the contraction of convolutional layers. Hence, the output of the network can be restored to the size of input and pixel-wise class can be predicted. U-Net, which is upgrade on FCNs in terms of performance, consists of two parts: encoder and decoder. Encoder is usually responsible for contraction of network and by combining all the pixels of image influence the answer to the question 'what?'. Decoder is implemented after the encoder and directly takes information from previous encoder layers, where the dimension of image is still large. This helps to determine the location of the class and thus answers the question 'where?'. The name U-Net comes from U-shaped architecture where the first half of U curve represents downsampling while the second half represents upsampling. With skipped connections added, it overcomes the spatial information loss during downsample process, increase the resolution of prediction. ResNet is a deep convolutional network network, that also has skipped connections to allow the propagation of gradients. The propagation tackles the vanishing gradient problem. ResNet18 is a special case of ResNet, is 18 layers deep and is pre-trained on 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. Similarly ResNet34 and ResNet152, with larger number of layers, are pre-trained. Inception network consists of many layers of inception module. Each inception module has filter of various sizes in parallel and the output of these parallel channels are concatenated at the output which becomes input to the next layer. The idea behind inception module is to allow the network to choose appropriate filter to detect different objects.

## 3 Method

### 3.1 Data Descriptions

A total of 443, 103, 103 images are provided for train, test, validation set with labels at each pixel. A total of 10 classes exist in the labels, with unequal distribution 1.

Table 1: Dataset label description

| Class | Name | Number of Samples |
|---|---|---|
| 0 | Terrain | 22829796 |
| 1 | Vegatation | 78633099 |
| 2 | Construction | 5283067 |
| 3 | Vehicle | 1152484 |
| 4 | Sky | 13038040 |
| 5 | Object | 881615 |
| 6 | Human | 185824 |
| 7 | Animal | 20445 |
| 8 | Void | 6125114 |
| 9 | Undefined | 137236 |

#### 3.1.1 Preprocessing

We calculate the mean and standard deviation of each channel from the training set. Then we use the pytorch transform Normalize function to normalize each image in train/test/validation. This function z-score the image according to the mean and standard deviation for all three channels.

#### 3.1.2 Data augumentation

We tried using 3 pytorch data augmentation transformations with probability $p = 0.5$. This means that for each iteration, each image(and its label) has a probability $p = 0.5$ to be transformed. The transformations include: horizontal flipping, vertical flipping and color jitter. Both flipping operations

are appled to image and labels. "Color jitter" randomly tunes the brightness and saturation of the image, and is only applied to the image.

### 3.2 Evaluation metrics

In order to evaluate the efficiency of the model, we need to have a use a method which can give a accuracy which makes sense semantically. Over here we make use of two metrics: Pixel accuracy and Intersection-over-union. We will observe that the later one would be the better way to find the accuracy of the semantic segmentation problem.

1. **Pixel Accuracy:** In this method we find the percent of the pixels classified correctly. But as we can see from the distribution above that the background objects dominate the data-set. In this scenario, the model can just fit classify the pixels as vegetation/terrain/sky and achieve very high accuracy, as these class make up most of the portion of the dataset. Unfortunately, the imbalanced classes can not be eradicated from the real world data. Hence, the metric of pixel accuracy wont be able to provide a good information of the model.

$$\text{Pixel Accuracy} = \frac{\text{Total Correct Predictions}}{\text{Total Number of Samples}}$$

   Also, the Void class was being ignored while calculating the pixel accuracy.

2. **Intersection-Over-Union:** The IoU is the area of the overlap between the predicted segmentation and the ground truth divided by the total area covered by both for a class. Later we take a mean of IoUs of all the classes. This is commonly used metric for the segmentation problems.

$$\text{IoU} = \frac{\text{Intersection}}{\text{Union}}$$

   Also, the Void class was being ignored while calculating the IoU.

### 3.3 Baseline Model

Our Baseline Architecture is a standard Encoder Decoder network in which the size of the image is reduced and the number of channels are increased using convolution before a non-linear layer. This convolution layer is fed to a ReLU non-linear activation. Every non-linear ReLU activation is followed by a batch normalization layer. And at the end of the network, we have Convolution layer such that it takes an input with 32 channels and produced output of with 10 channels, which is the total number of classes in our case.

We are using Cross Entropy Loss criterion with Adam Optimizer, as it is currently the best optimizer.

In summary the Baseline Network is as follows:

1. **Architecture:** Encoder-Decoder Network
2. **Activation Function:** ReLU
3. **Loss Criterion:** Cross Entropy
4. **Optimizer** Adam
5. **Batch Normalization:** Yes
6. **Final Layer:** 2D Convolution layer with 10 output channels

Table 2: Baseline Architecture

| Sr No | Layer | In-Channels | Out-Channels | Kernel-Size | Padding | Stride |
|---|---|---|---|---|---|---|
| 1 | conv1 | 3 | 32 | 3 | 1 | 2 |
| 2 | conv2 | 32 | 64 | 3 | 1 | 2 |
| 3 | conv3 | 64 | 238 | 3 | 1 | 2 |
| 4 | conv4 | 128 | 256 | 3 | 1 | 2 |
| 5 | conv5 | 256 | 512 | 3 | 1 | 2 |
| 6 | deconv1 | 512 | 512 | 3 | 1 | 2 |
| 7 | deconv2 | 512 | 256 | 3 | 1 | 2 |
| 8 | deconv3 | 256 | 128 | 3 | 1 | 2 |
| 9 | deconv4 | 128 | 64 | 3 | 1 | 2 |
| 10 | deconv5 | 64 | 32 | 3 | 1 | 2 |
| 11 | Classifier | 32 | 10 | 3 | 1 | 2 |

### 3.4 Imbalanced class problem

Imbalanced Class problem is a problem of biased distribution of classes is biased or skewed. The disproportion of the imbalanced classes, can occur easily in the semantic segmentation where the pixels are the data. In the distribution 1, we can observe that the 'Vegetation' samples are far more compared to the 'Animal' sample. In the real world, when the images are collected at different points, it intuitively makes sense that the terrain/vegetation/sky would cover large area of the picture compared to any other category, as these would act as the background of the image. Moreover, the class imbalance can be observed in the many applications from health care to fraud detection. Hence, we need a more generic way to resolve this issue.

#### 3.4.1 Weighted Loss

There are many solutions proposed to the issue of the Imbalanced class. One of them is Weighted loss, in which we penalise more to the class with lesser samples, where as penalise less to the class with more samples. In this method, we calculate the weights for each class such that they are inversely proportional to the number of samples. This way we will have higher weight for the class with less number of samples. And these weights we multiplied to the respective class while calculating the loss.

$$W_k = \frac{1}{N_k}$$

Where, $N_k$ is the number of samples in class k.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{C} W_k t_k \log y_k$$

During the testing phase, multiple relationships between the weights and the number of samples were tried out. As the direct relationship of inversely proportional was tried out and it seemed to overfit the data. So, new relation like inversely proportional to the log of number of samples was tried out and it seemed to work well, as it was less aggressive.

$$W_k = \frac{1}{\log N_k}$$

### 3.5 Models

#### 3.5.1 Transfer learning

We applied ResNet for transfer learning. Using pretrained ResNet as the encoder, The last two layers - average pooling and fully connected layers are removed from ResNet. I tried 3 ResNets with different depth: ResNet18, ResNet34 and ResNet152. For the decoder, we initially use naive decoder from the baseline model. This model has no residuel or skipped connections, and therefore was termed "Awful ResNet". We further implemented a decoder with skipped connections, dubbed as "Descent ResNet".During training, the encoder does not update weights. We only train the decoder.All networks are train with $lr = 0.05$.

#### 3.5.2 Unet

Out implementation of UNet is shown in fig. 2. We made slight modification to the original architecture of UNet [3] to suit our needs. The major change is instead of using cropping to match dimension size between encoder and decoder layers, we use zero padding as shown by dashed boxes in fig. 2. This is done so that the size lost during 3x3 convolutions can be restored and dimension of output image matches the original input dimension. We also use batchnorm after every convolution and transpose convolution layer. The activation function in each layer is ReLU. On the encoder side, we have 5 encoder modules that give 64, 128, 256, 512 and 1024 output channels respectively. Each encoder module consists of two stacked 3x3 convolutional layers followed by a maxpool layer of stride 2. In the last encoder module, maxpool is removed and we start climbing back into the decoder layers. In the decoder part, we have 4 decoder modules. Each decoder modules start with a 3x3 transpose convolution and output is zero padded to allow the encoder output from same level to be concatenated. This is then followed by two stacked 3x3 convolutional layers. This process is repeated in all modules till the output of model is available back on the first level, which is obtained by 1x1 convolutional layer to give output channels equal to number of classes.

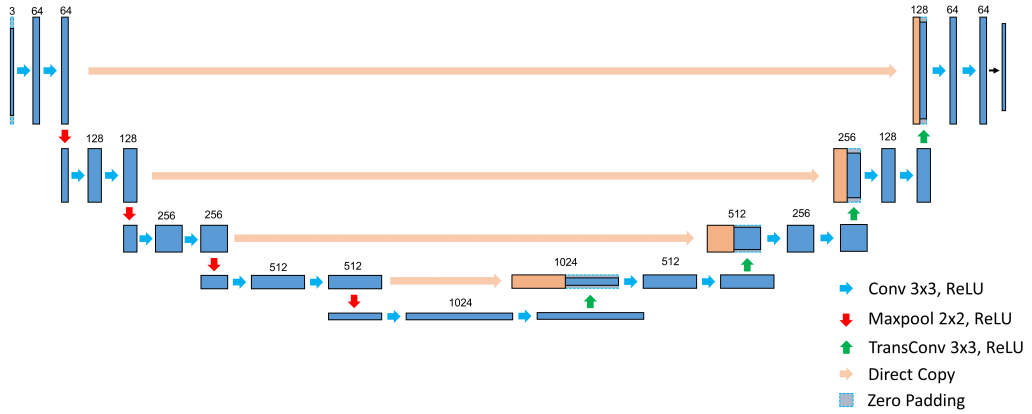


Figure 1: UNet Architecture

### 3.6 HAPNet: Our Own Architecture

From reading the literature on different models used in the Convolution Neural Network, for different application we were able to build a consensus that the Neural Network is able to learn better when the Neural Network is given a choice to choose the receptive field it wants to use. This is why the GoogLeNet was able to do better after Alexnet. The inception module in the GoogLeNet gave freedom of choosing a receptive field between 1x1, 3x3, 5x5, before a linearity was applied. The same trend is being followed up in the ResNet and Dense Network, which tend to work surprisingly good. In these networks, the skip connections play a role of creating the receptive fields of different sizes. A convolution of 3x3 in 2 different layers provided an overall 5x5 receptive field. In addition, these networks have the skip which the network would learn to skip if that specific layer is increasing the loss.

Also, from the models like UNet and SegNet which are prominent in the semantic segmentation, we were able to learn the importance of the contraction of the images as we proceed forward in the layers, which helps in the model to learn scale in-variance and also reduce the number of computations. Moreover, these contractions should be expanded as the prediction should be the classification of the pixels.

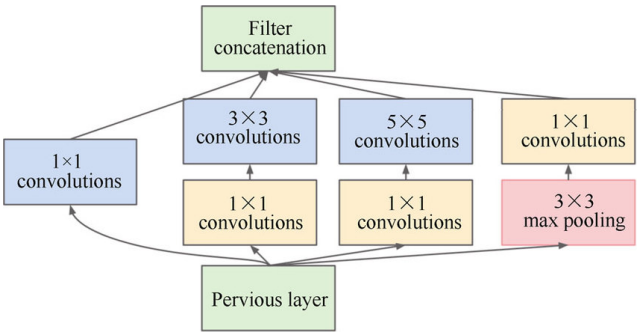


Figure 2: Orignal Inception Module

Due to the lack of computational power with the resources available, we tried to optimize the inception module in the computational direction, such that it affects the performance minimally. The following decisions were made while coming up with **HAP_Inception**

1. **Minimal number of Channels per layer:** Due to the space limitation in the GPU, we couldn't store huge number of parameters in the GPU. We decided not use more number of channels per layer. So, In our case we boosted the number of channel to 32 in stem and gradually increased the number of channels to 128 and kept it stagnant. Moreover, more number of channels would learn more internal representations. We were left with no other choice.
2. **Remove Bottle Neck:** The bottleneck receptive field was one of the innovative ideas to reduce the number of channels before we moved to the 3x3 or 5x5 convolution. Unfortunately this didnt make sense in our case, where we did have many channels coming in each layer.
3. **Remove Max Pooling:** Remove the max pooling layer present inside the inception and put it outside the inception.
4. **Replace the 5x5 receptive field with two 3x3 receptive field:** This is a crazy hack, as the overall output remains the same but with lesser operation.

Hence, below is the customized inception

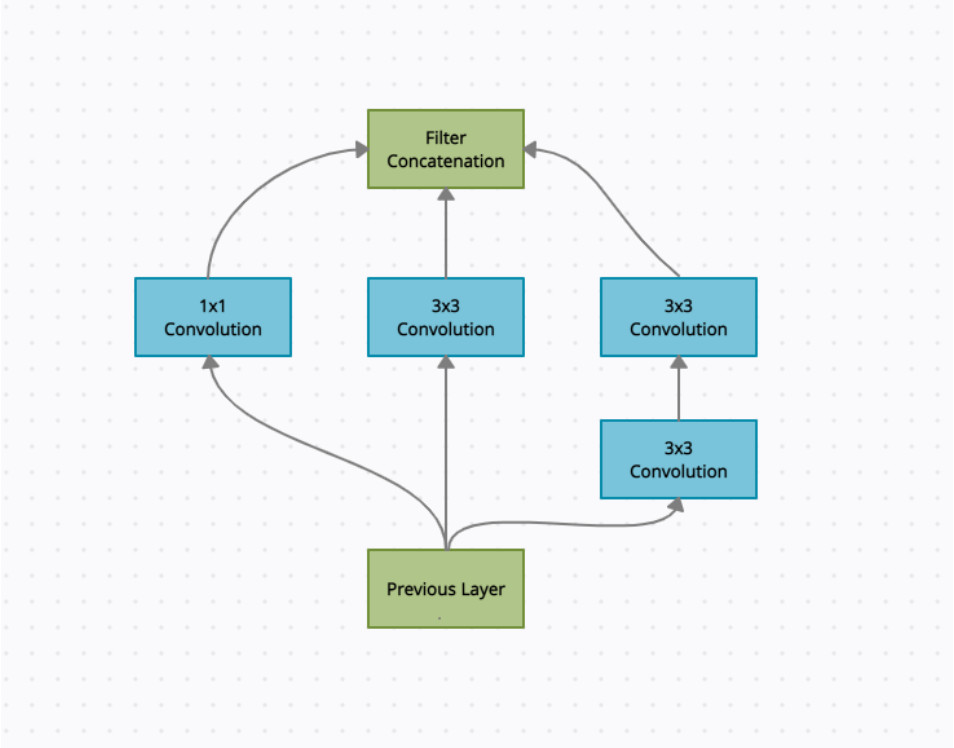


Figure 3: Customized Inception Module

Table 3: Baseline Architecture

| Sr No | Layer | In-Channels | Out-Channels | Kernel-Size | Padding | Stride |
|---|---|---|---|---|---|---|
| 1 | conv1 | 3 | 16 | 3 | 1 | 1 |
| 2 | conv2 | 16 | 32 | 3 | 1 | 1 |
| 3 | Inception1 | 32 | (16+32+16) | 3 | 1 | 1 |
| 4 | Max-Pooling | 64 | 64 | 2 | 0 | 2 |
| 5 | Inception2 | 64 | (32+64+32) | 3 | 1 | 1 |
| 6 | Max-Pooling | 128 | 128 | 2 | 0 | 2 |
| 7 | Inception3 | 128 | (32+64+32) | 3 | 1 | 1 |
| 8 | Max-Pooling | 128 | 128 | 2 | 0 | 2 |
| 9 | Inception4 | 128 | (32+64+32) | 3 | 1 | 1 |
| 10 | Max-Pooling | 128 | 128 | 2 | 0 | 2 |
| 11 | Deconv1 | 128 | 128 | 3 | 0 | 2 |
| 12 | Inception5 | 128 | (32+64+32) | 3 | 1 | 1 |
| 13 | Deconv2 | 128 | 128 | 3 | 0 | 2 |
| 14 | Inception6 | 128 | (32+64+32) | 3 | 1 | 1 |
| 15 | Deconv3 | 128 | 128 | 3 | 0 | 2 |
| 16 | Inception7 | 128 | (32+64+32) | 3 | 1 | 1 |
| 17 | Deconv4 | 128 | 64 | 3 | 0 | 2 |
| 18 | Inception8 | 64 | (16+32+16) | 3 | 1 | 1 |
| 19 | Classifier | 32 | 10 | 3 | 1 | 2 |

Following is the architecture:

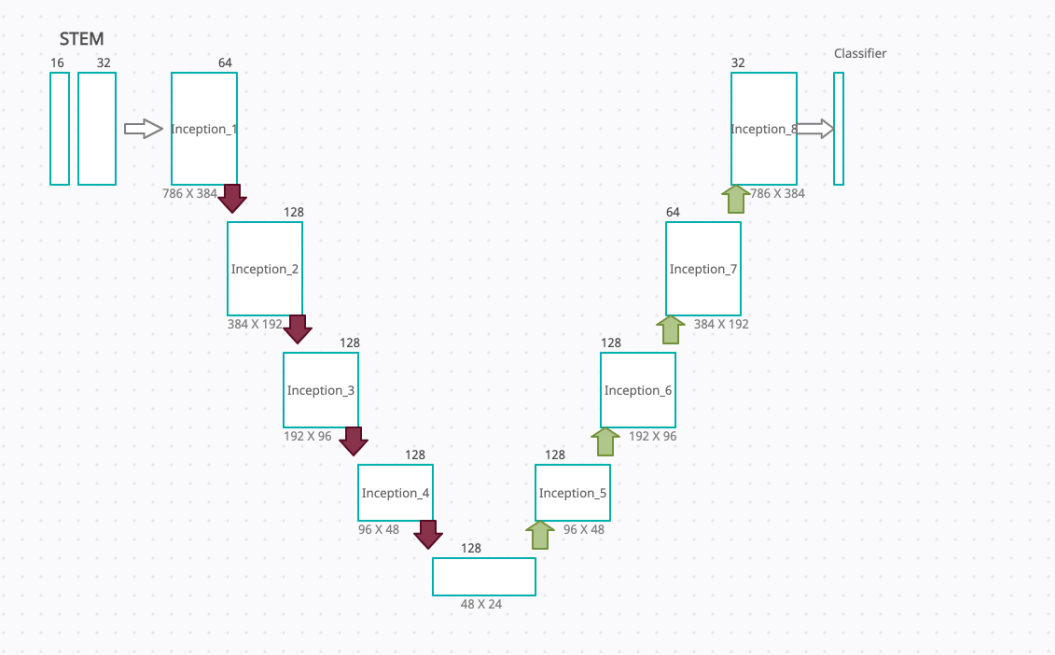


Figure 4: HAPNet Architecture

# 4 Results

## 4.1 Baseline Model

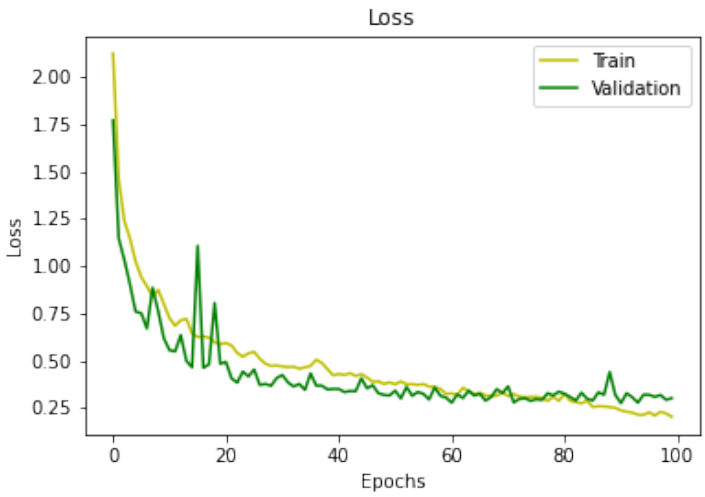


Figure 5: Loss Curve for the Baseline Model

The base line model was able to achieve the following accuracy over the test set:

1. **IoU Accuracy:** 51.6
2. **Pixel Accuracy:** 89.66

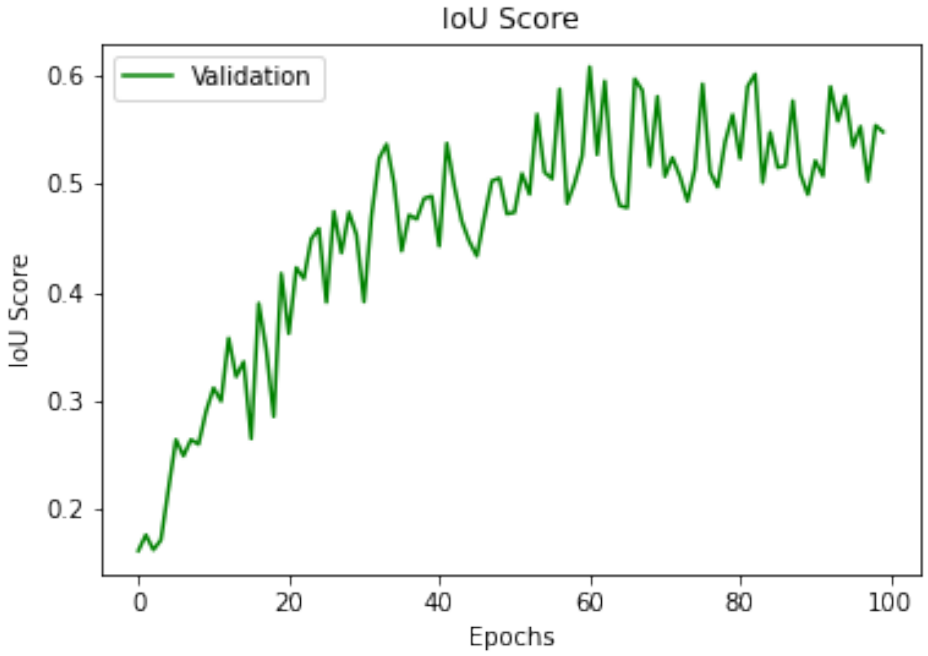


Figure 6: IoU Accuracy on the Validation set for the Baseline Model

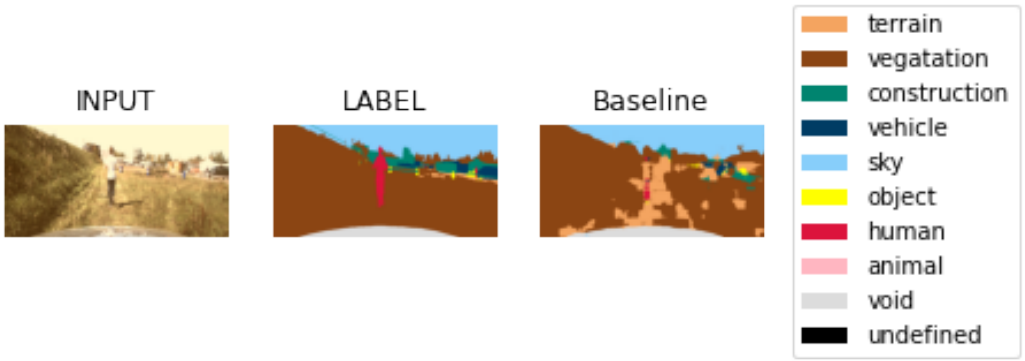


Figure 7: Visualization of the Pixel Classification and the ground truth (LABEL)

### 4.2 Data Augumentation

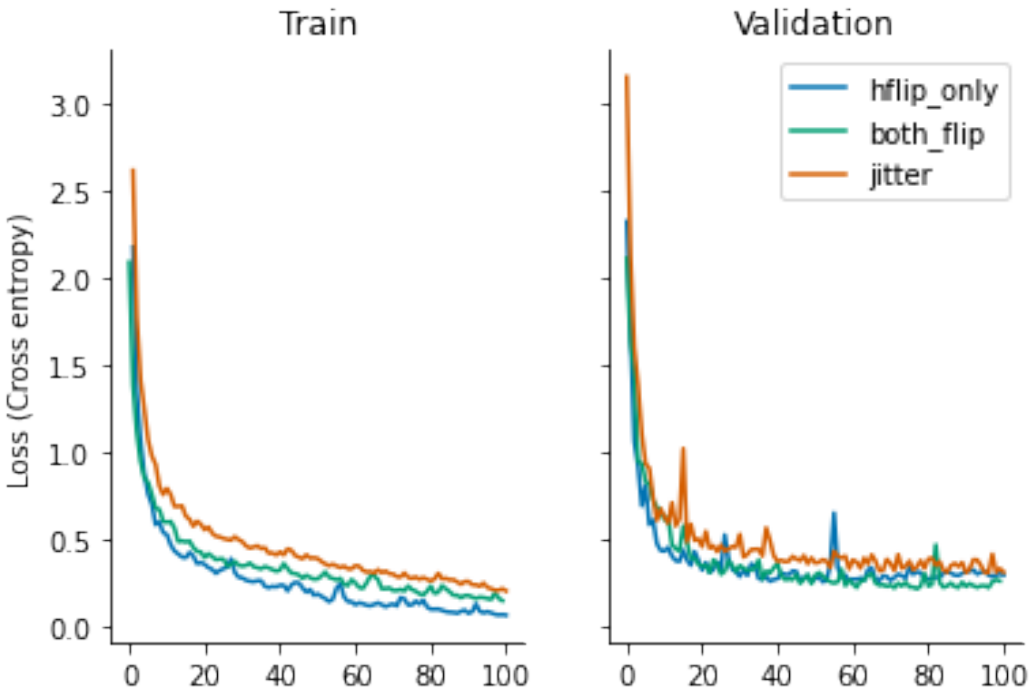


Figure 8: Loss Curve for Various Data Augmentation. **abbreviations** Horizontal flip(hflip), Flipping on both sides (both_flip and color jitter (jitter)

Table 4: Test Accuracy and Iou (Data Augmentation) **abbreviations:** Horizontal flip(hflip), Flipping on both sides (both_flip and color jitter (jitter)

| cond | hflip_only | both_flip | jitter |
|---|---|---|---|
| Test IoU | 0.554630 | 0.572559 | 0.498569 |
| Test Pixel Acc | 0.914117 | 0.918269 | 0.911771 |
| Iou(class 0) | 0.850377 | 0.718516 | 0.737696 |
| Iou(class 1) | 0.872570 | 0.888671 | 0.892172 |
| Iou(class 2) | 0.276401 | 0.146128 | 0.300396 |
| Iou(class 3) | 0.441739 | 0.241526 | 0.365730 |
| Iou(class 4) | 0.913750 | 0.727224 | 0.867019 |
| Iou(class 5) | 0.120796 | 0.041273 | 0.065651 |
| Iou(class 6) | 0.319719 | 0.064296 | 0.161299 |
| Iou(class 7) | 0.000000 | 0.000000 | 0.000000 |
| Iou(class 8) | 0.898148 | 0.868225 | 0.852894 |

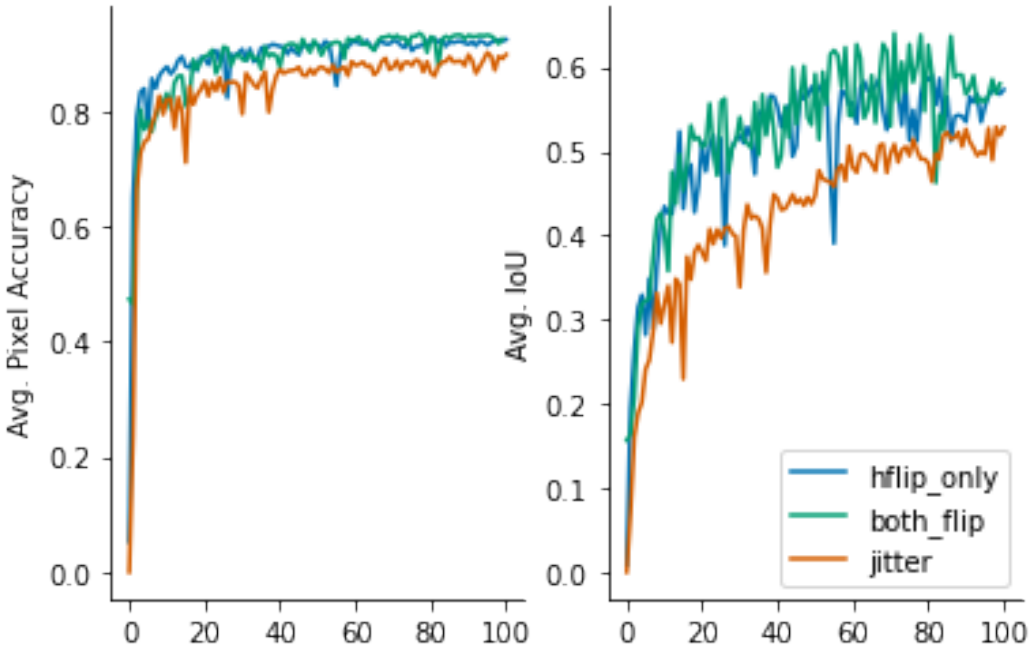


Figure 9: Accuracy and IoU per training epoch (Data Augmentation)**abbreviations:** Horizontal flip(hflip), Flipping on both sides (both_flip and color jitter (jitter)

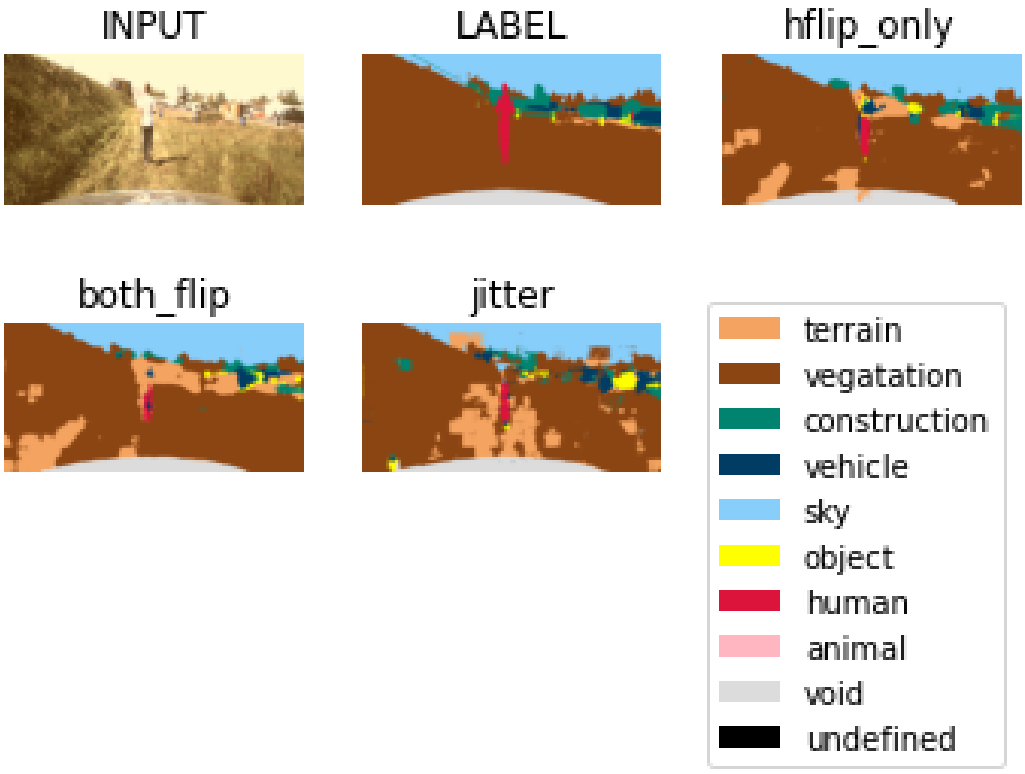


Figure 10: Showcase the classification result (Data Augmentation) **abbreviations:** Horizontal flip(hflip), Flipping on both sides (both_flip and color jitter (jitter)

### 4.3 Weighted Loss

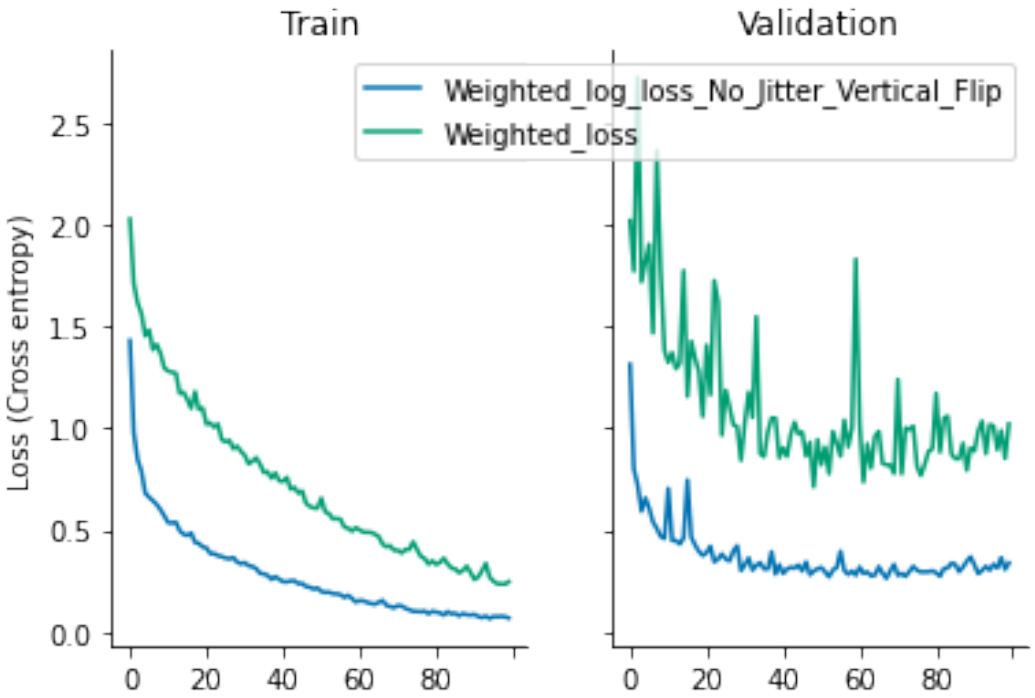


Figure 11: Loss Curve for the Baseline Model with Weighted Loss

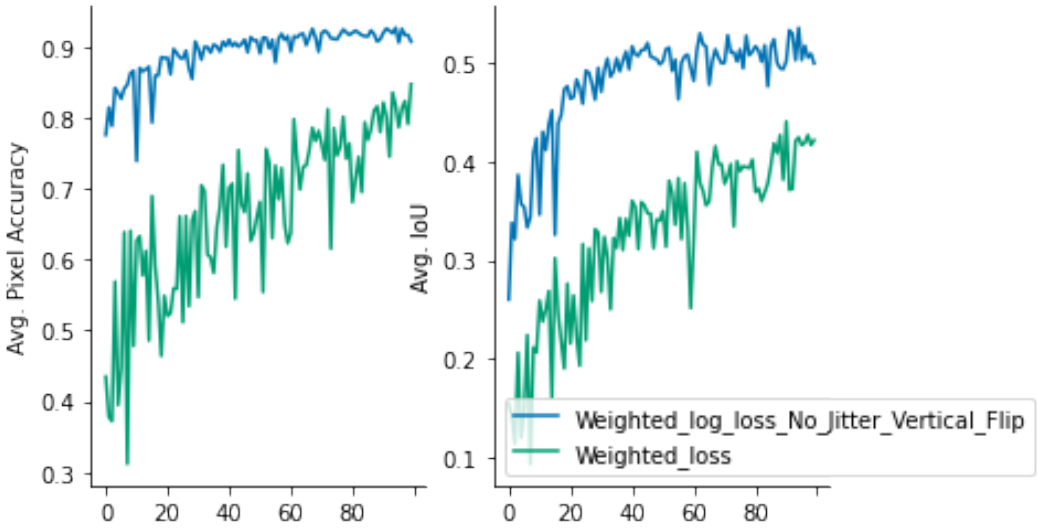


Figure 12: IoU Accuracy on the Validation set for the Baseline Model with Weigted Loss

The base line model was able to achieve the following accuracy over the test set:

1. **IoU Accuracy:** 54.26
2. **Pixel Accuracy:** 93.27

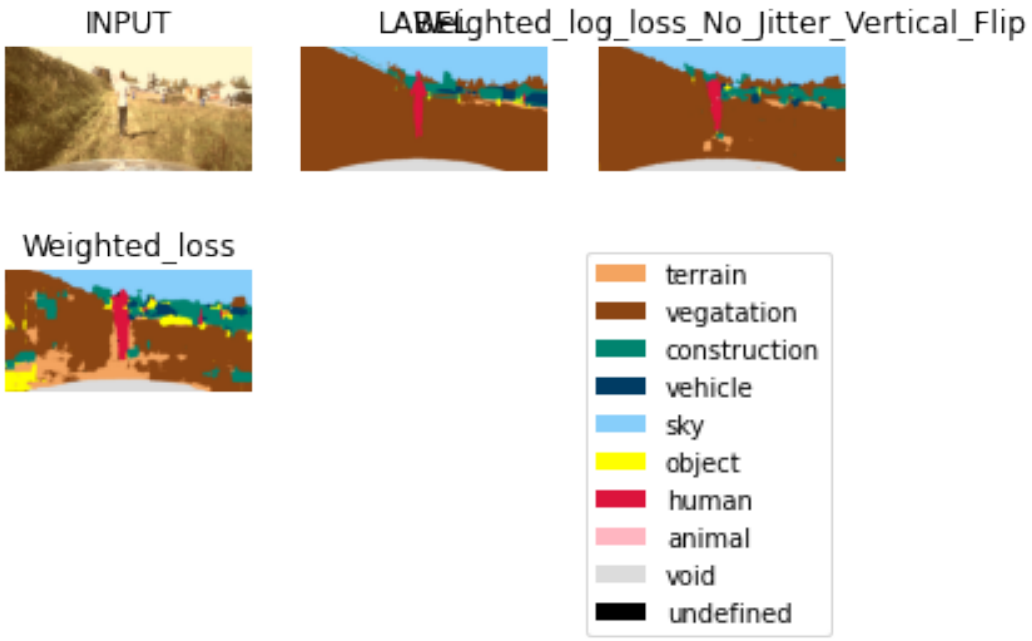


Figure 13: Visualization of the Pixel Classification and the ground truth (LABEL)

**Note:** In the above figure there is a overlap between the title of two images i.e. "LABEL" and "Weighted Log loss with no jitter and vertical Flip"

### 4.4 Transfer learning

Table 5: Test Accuracy and Iou for transfer learning models. For row "decoder", "FCN" stands for the decoder from basic_fcn.py. "SkipCon" stands for "skipped connections"

| cond | Awful_Res18 | Awful_Res34 | Awful_Res152 | Awful_Res34_w | Des_Res34_we | Des_Res34 |
|---|---|---|---|---|---|---|
| Test IoU | 0.535189 | 0.589994 | 0.586883 | 0.459606 | 0.491012 | 0.620499 |
| Test Pixel Acc | 0.881406 | 0.911894 | 0.92577 | 0.836795 | 0.873265 | 0.933215 |
| Iou(class 0) | 0.706959 | 0.650176 | 0.755762 | 0.603787 | 0.687631 | 0.748441 |
| Iou(class 1) | 0.828423 | 0.890818 | 0.90352 | 0.794452 | 0.832409 | 0.912359 |
| Iou(class 2) | 0.260771 | 0.516926 | 0.345337 | 0.259008 | 0.206052 | 0.335245 |
| Iou(class 3) | 0.788595 | 0.79199 | 0.759851 | 0.574618 | 0.517109 | 0.635939 |
| Iou(class 4) | 0.827058 | 0.822991 | 0.873603 | 0.77351 | 0.875549 | 0.93458 |
| Iou(class 5) | 0.139454 | 0.05744 | 0.151299 | 0.144381 | 0.048673 | 0.101068 |
| Iou(class 6) | 0.167901 | 0.340127 | 0.226461 | 0.149314 | 0.343507 | 0.440056 |
| Iou(class 7) | 0.0 | 0.0 | 0.0 | 0.052434 | 0.032986 | 0.0 |
| Iou(class 8) | 0.857447 | 0.857725 | 0.852843 | 0.784956 | 0.875191 | 0.887293 |
| ResNet size | ResNet18 | ResNet34 | ResNet152 | ResNet34 | ResNet34 | ResNet34 |
| weighted | False | False | False | True | True | False |
| decoder | FCN | FCN | FCN | FCN | skipCon | skipCon |

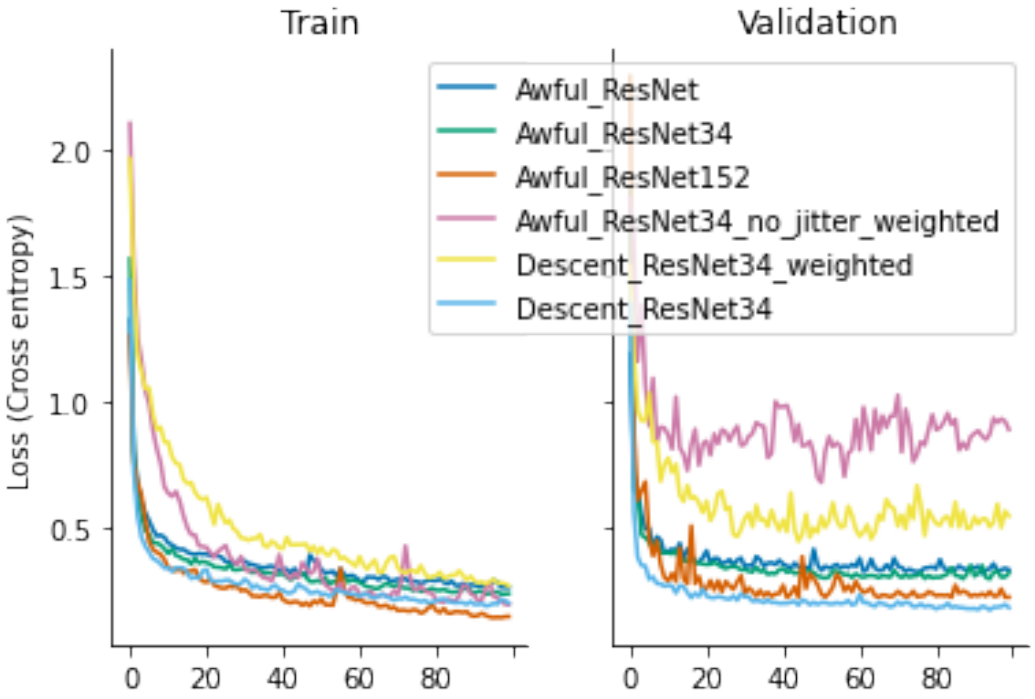


Figure 14: Loss Curve for Various Transfer Learning model **abbreviations:** Awful_ResNet uses the basic decoder; Descent_ResNet added skipped connections.

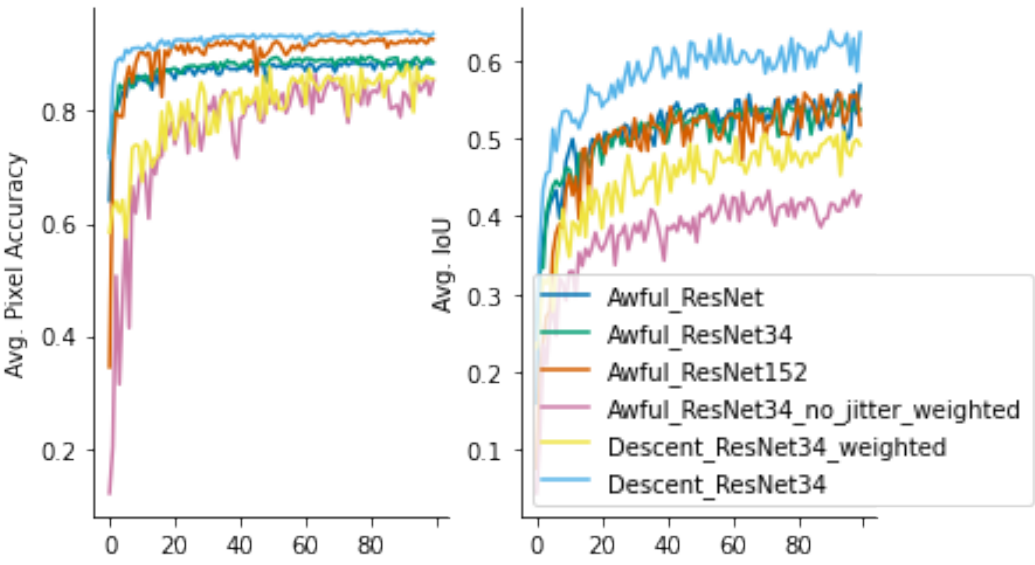


Figure 15: Accuracy and IoU per training epoch, Transfer Learning model **abbreviations:** Awful_ResNet uses the basic decoder; Descent_ResNet added skipped connections.

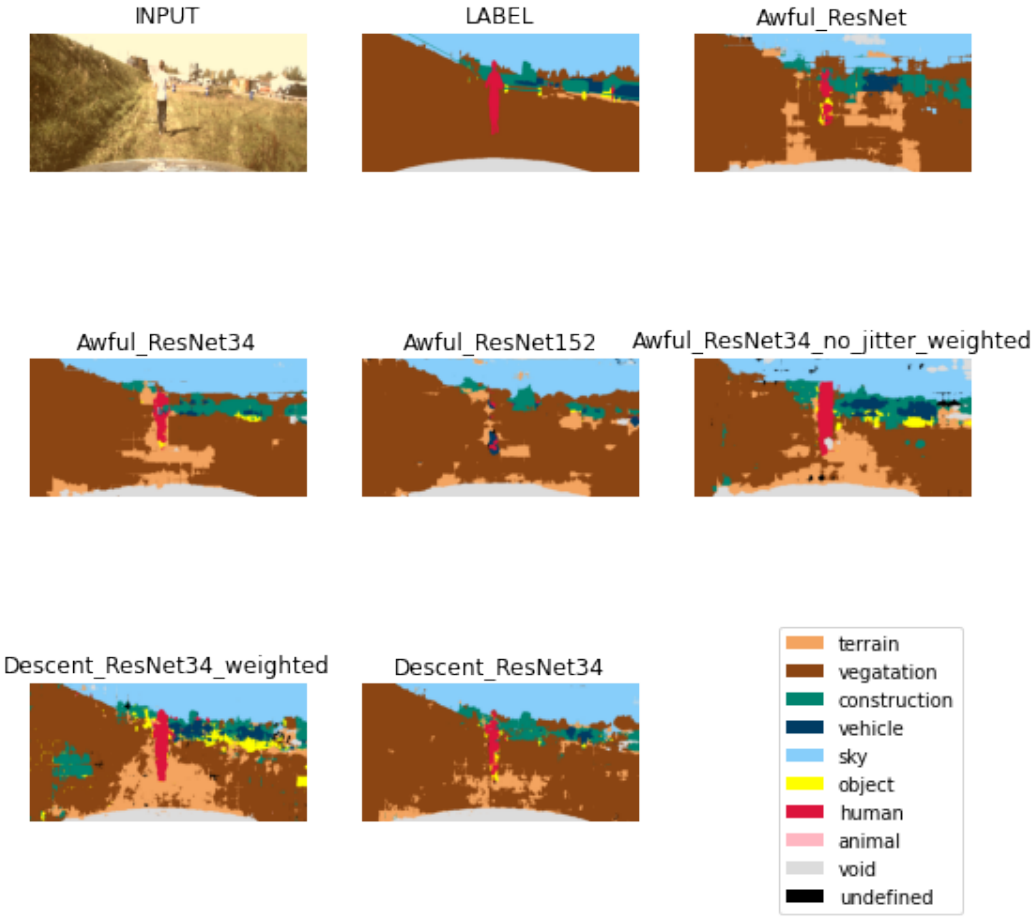


Figure 16: Transfer learning visualization **abbreviations:** Awful_ResNet uses the basic decoder; Descent_ResNet added skipped connections.

### 4.5 Unet

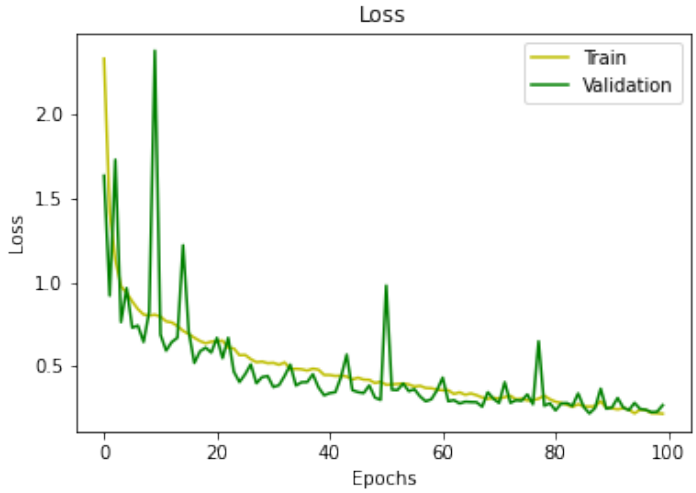


Figure 17: Loss Curve for Unet

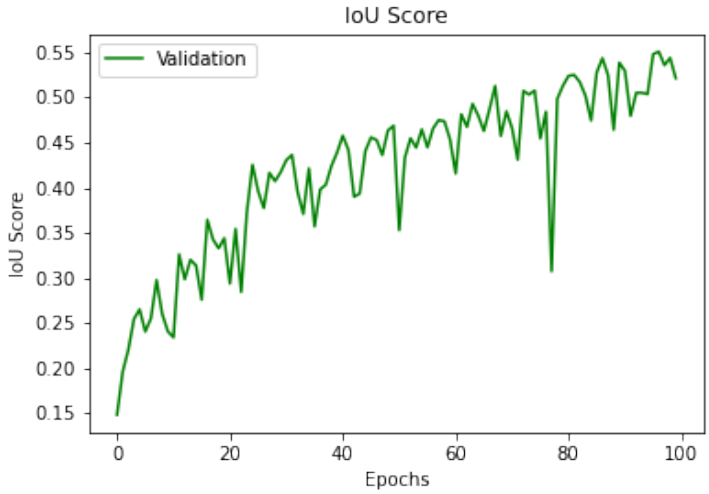


Figure 18: Accuracy Curve for Unet


Figure 19: Showcase for UNet architecture

### 4.6 HAPNet: Our Own Architecture

Test performance:

1. **IoU Accuracy:** 55.16
2. **Pixel Accuracy:** 93.48

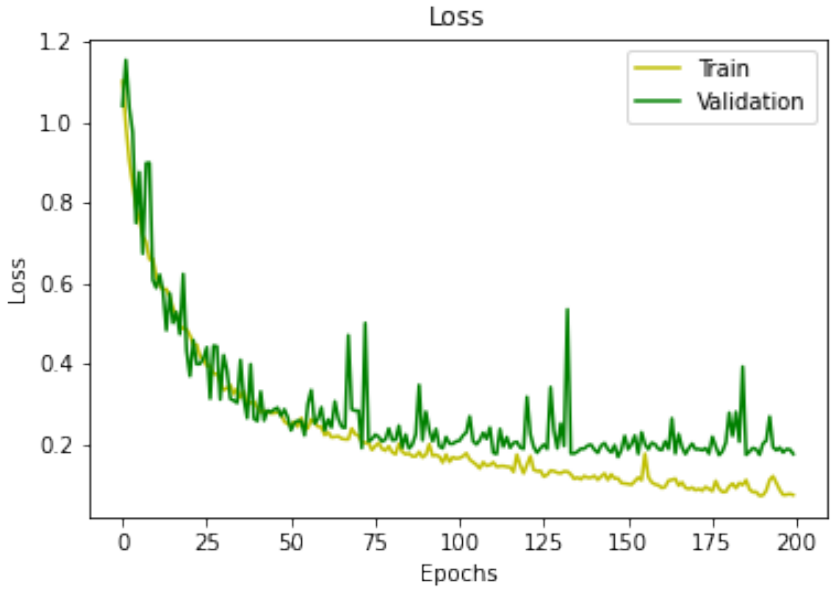


Figure 20: Loss Curve for the HAPNet Model

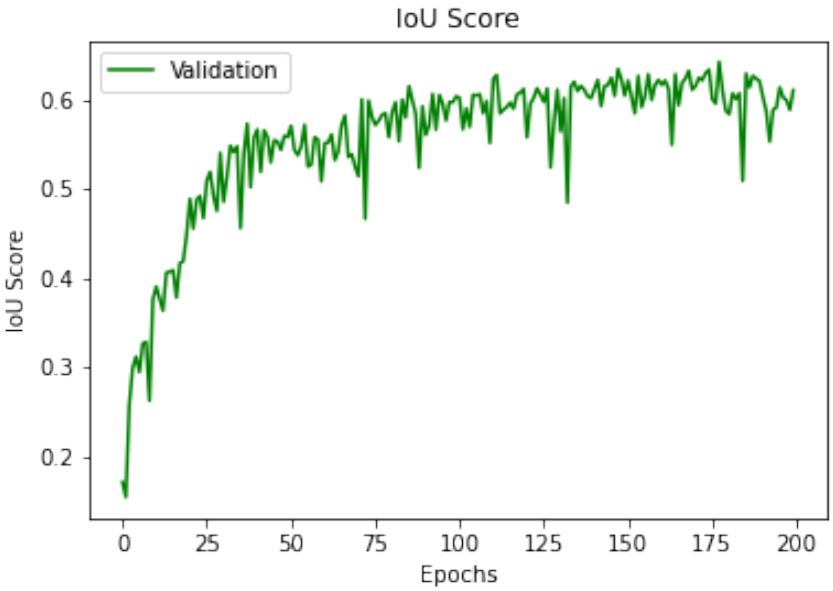


Figure 21: IoU Accuracy on the Validation set for the HAPNet Model

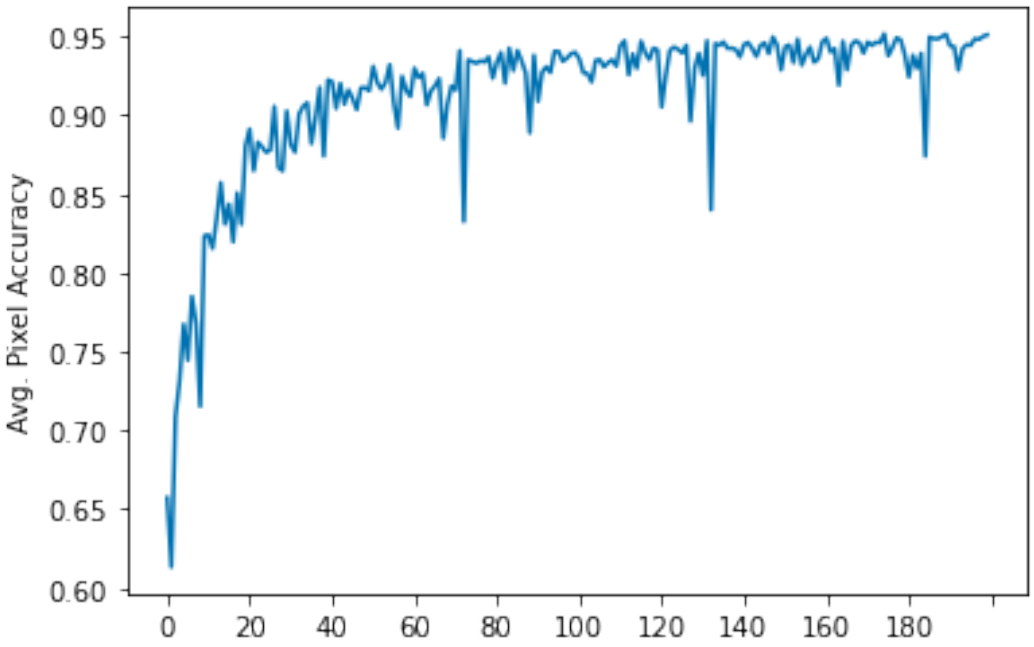


Figure 22: Pixel Accuracy on the validaiton set for HAPNet Model

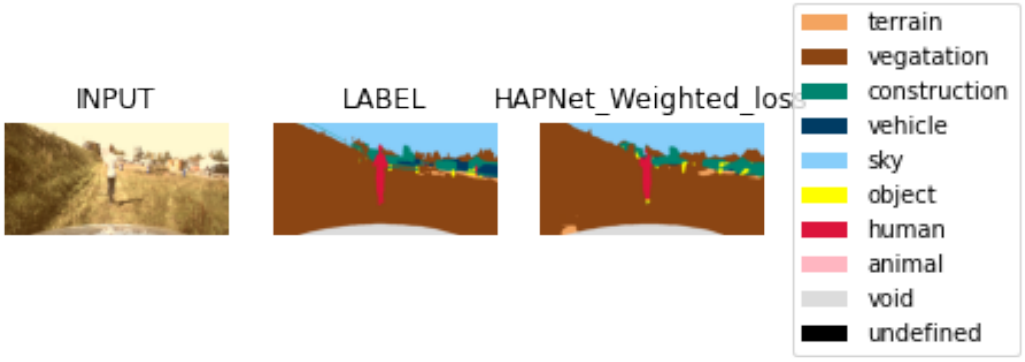


Figure 23: Visualization of Pixel Classification

# 5 Discussion

## 5.1 Evaluation over Data Augmentation

Our study started with implementing data augmentation on baseline model. Comparing data augmentation result 4 with baseline 3, data augmentation certainly performs better in some aspects. For example, the hflip is able to predict much better than baseline. This can be attributed to the property of lateral symmetry of human body. In comparison flipping both dimension does not work as it has harder time recognising human body upside down. Other than few object-specific improvements, data augmentation doesn't have a clear cut improvement on baseline as it is still unable to recognise difference between terrain and vegetation.

When looking at the training curve (Fig 8 and 9), we observe that the training is slower, probably because in the presence of data augmentation, the task is more complex - the network has to recognize human of jittered colors, it has to learn about the shape of the human. Also, we are suprised to find that color jitter does not do that much good to our performance. Compared to the baseline model (IoU 51%), it has an even lower IoU (49%). Although it is best to compare through multiple iteration of training, with 1 trial it is certain that jittering is not very helpful. It might be because due to the small amount of data, color remains a very important information. Alternatively, judging by the training curve, it seems that the training has not fully converged yet. So maybe we need to train it for a few more epochs.

## 5.2 Evaluation over Weighted Loss

In order to address the imbalance of the classes in the dataset, we used the weighted loss. Where we penalised the class with less samples more compared to the one with the higher samples. Unfortunately if the data is repeated, which is quite the case in the pixel classification, where the pixel values will keep on repeating, the model starts to overfit. And this is exactly what we observed as shown in the figure below:

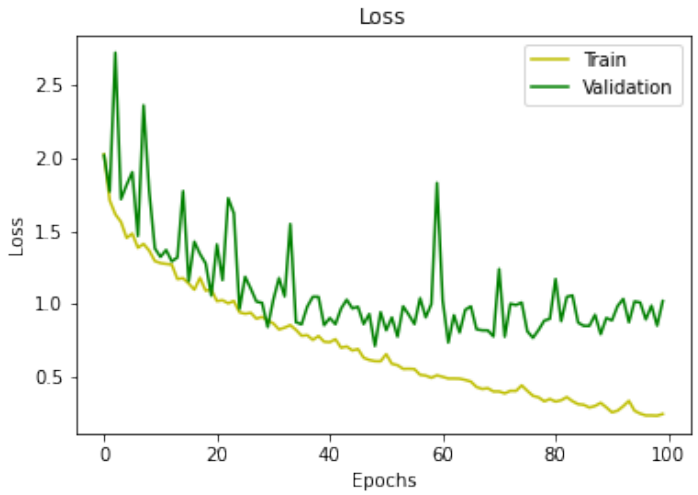


Figure 24: Loss over the base model with weighted loss

The probable reason why this behaviour is observed is because of the aggressive penalise. We are trying to divide by the number samples, which is very high in our problem, where the examples are the pixels. We tried to combat this issue by reduce the effect of penalty by using the log of the number of samples. And this tends to reduces the over-fitting.This also helped the loss to be reduced to 0.1 from 0.5. And the test accuracy increased from 48% to 54%.

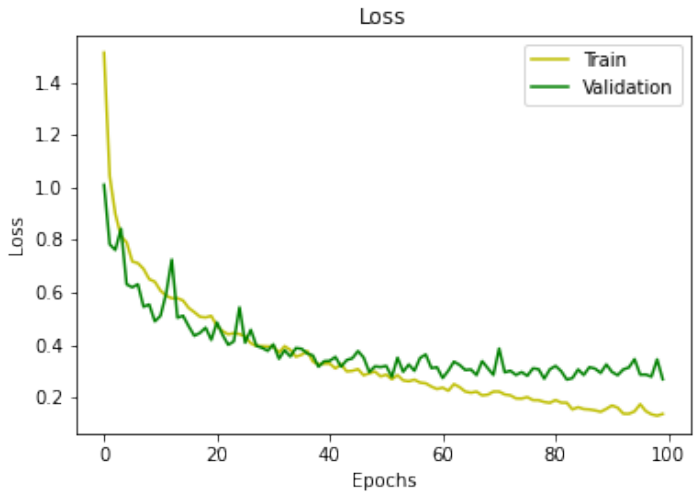


Figure 25: Loss over the base model with weighted log loss

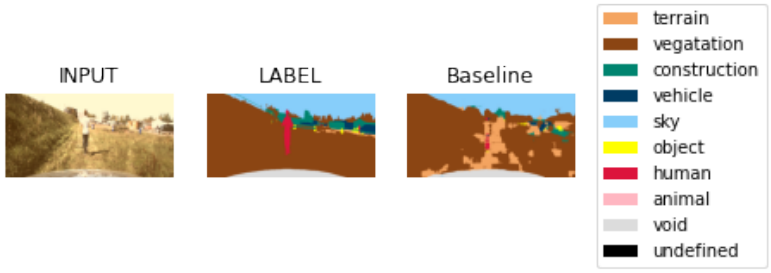


Figure 26: Visualization of the Pixel Classification without weighted loss

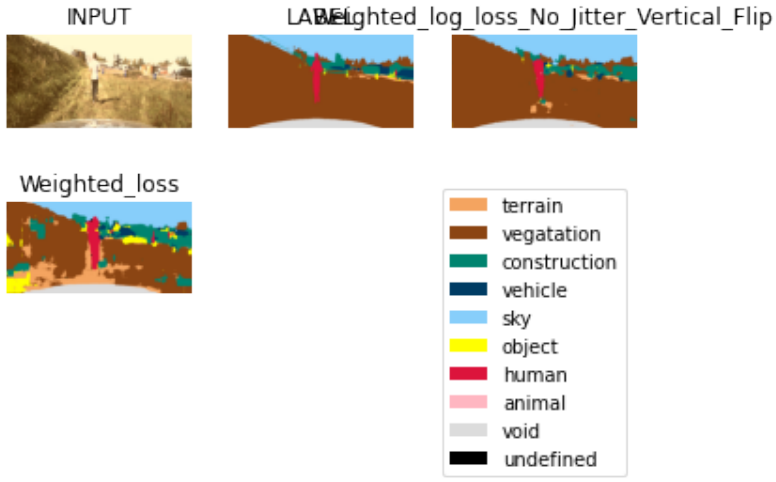


Figure 27: Visualization of the Pixel Classification with weighted loss

We can also observe from the distribution of classes that the human class is very low compared to the terrain/vegetation/sky. Due, to this the human(red) colour in the visualisation part of the results does not get classified accurately 26). But, when we use the weighted log loss the human label gets classified accurately 27).

From this we can conclude that weighted log loss would be a preferable way to address imabalanced class in the case of semantic segmentation.

### 5.3 Evaluation over Transfer Learning

Next we compare transfer learning and use pre-trained Residual Network to this end. ResNet is able to identify the person much better that baseline and only data augmentation. This could be attributed to larger number of images that ResNet learn and thus help it identify a person, that can have various shape and color due to clothing, much better. But identifying the vegetation still remains an issue. The ResNet recognises that there is vegetation present and various architectures get it correct at different places but there are patches that vegetation gets confused with terrain and construction. To improve the predicted area for vegetatation, we use UNet which has the useful property of identifying global location and the context at the same time . This can be seen fig. 14, where UNet is able to identify vegetation much more uniformly, both on the left and near the skyline where vegetation is predominant. It is understandable that it still has trouble recognising vegetation near the person as it is minimal and is identical to a road.

We also observe that when using deeper ResNet, we were able to improve the results, as we see using ResNet18 we get mean IoU of 0.53, and 0.58 with ResNet34 and ResNet152. It is because deeper ResNet can learn more intricate features that is helpful to recognize objects 5. Training with Resnet is also faster. It can reach an mean IoU of 40% at very early epoch, because the encode does not learn it from scratch. However, even with ResNet, our result was still not significantly better than baseline (IoU 58%), until skipped connections are added. Looking at the predictions 16 the predictions from vanilla decoders (the "Awful ResNets") are very patchy, signaling a loss of resolution. Adding skipped connections (the "Descent ResNets"), allow us to preserve spatial information from the encoder, therefore increase resolution. This idea is actually the same as that of Unet. There are some advantages with using ResNet or transfer learning: As the encoder is pretrained, we don't need to retrain things and therefore can reduce the number of parameters, and improve running time. The downside is if the dataset is very different from the dataset that is used to pretrain, the features learned cannot be sucessfully applied. During training, we observed that if we input unnormalized image into ResNet, the performance will be worse than baseline. It might be because that without proper preprocessing, the raw image display different features ResNet don't recognize.

**improvements that can be done to transfer learning**

1. try adding dense connections to the decoder as well
2. try out different, newer transfer learning model
3. train for more epochs
4. at later iterations, fine tune the encoder by allow gradient to modify the weights (adjust to the dataset)

#### 5.3.1 Unets

Lastly, we tried to train a Unet from scratch. From the previous section, "Descent ResNet" is fairly similar to Unet. Looking at the performance, the mean IoU is around 55%, similar to augumented baseline models. This might be due to a multitude of reasons. First, Unet has a lot of parameters to train. Given our limited GPU memory, we can only go at a batch size of 1-4. This makes the training unstable and slow. Second, our dataset is relatively small compare the the number of parameters, therefore it cannot fully learn the features. Thirdly, judging from the training curve 17, we might need to tune the learning rate and the number of epoch quite a bit. However, given the resource and time permitted, that was not possible. In the context of a small dataset, a similar architecture, the "Descent Net" utilize transfer learning to solve this problem.

**improvements**

1. reducing the input dataset so that we can increase batch size

2. reduce hidden layer channels
3. add dense connections to the decoder and encoders
4. train for more epochs, fine tune learning rate
5. get a bigger GPU
6. use pre-trained U-net

### 5.4 Evaluation over HAPNet (Own Architecture)

The model designed by us tends to saturate at approximately 62% accuracy for the validation set. In addition, the visualisation of the best pixel classification looks very accurate.

The benefits of the HAPNet model are

1. It takes lesser time for computation. Approximately 10% better than the baseline model
2. It has better accuracy compared to the baseline model. The HAPNet is able to reach 65% IoU accuracy over the validation set
3. It is modular, hence addition of new inceptions is easily possible
4. It learns faster compared to the baseline model

The HAPNet was computationally inexpensive because it was using lesser channel outputs, implying lesser parameter. We had to use lesser parameters because of limitation of the resources. We were not able to store the parameters in the GPU for processing for large number of parameters.

The accuracy of the HAPnet is high because the HAPNet is the inspired from the idea that we will let the Model learn the size of the receptive field. This gives the Model freedom to optimize the loss in additional directions,

As, the model is modular we can edit the inception in future like addition of another branch with 3 3x3 convolution receptive fields.

We have discussed about the approach followed in the development of this Model.

**Improvements**

1. Addition of more number of channels to the layers.
2. Densely connecting the network using skip connections.
3. Addition of another branch to the inception using three 3x3 convolution receptive fields
4. Different regularization methods can be tried in order to avoid the over-fitting as the epochs increase.

## 6 Individual Contributions to the Project:

1. **Ashish:** IoU, Pixel accuracy, Weighted loss, invented "Our own architecture", the HAPNet!!. Also made a good code base to work with. Write many of the discussion in HapNet.
2. **Prabhav:** Worked on the implementation of the Unet and development of HAPNet. In addition worked on the Abstarct, Introduction, Method, Results and Discussion and references.
3. **Hsuan-lin:** Normalization, Data Augmentation, Transfer learning, Visualizing labels. Write about those sections mainly.

## 7 References

## References

[1] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. **86**, no. 11, pp. 2278-2324, Nov. 1998.

[2] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[3] Ronneberger O., Fischer P., Brox T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab N., Hornegger J., Wells W., Frangi A. (eds) *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. MICCAI 2015. Lecture Notes in Computer Science, vol 9351. Springer, Cham.

[4] J. Zhang, Z. Wang, S. Zhang, G. Wei, Z. Xiong and M. Yang, "Image Semantic Segmentation Based on the GAN Auxiliary Network," *2021 IEEE International Conference on Progress in Informatics and Computing (PIC)*, 2021, pp. 118-124.

[5] C. Szegedy et al., "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1-9.

[6] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, Serge Belongie, "Class-Balanced Loss Based on Effective Number of Samples," *2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1-9.