# Semantic Segmentation of a multi-modal dataset for Off-Road Robotics (RELLIS-3D)

**ANKUR GURIA, 17039**
Project guide: **Dr. Sujit P. B.**
Dept. of Electrical Engineering and Computer Science, IISER Bhopal

## Introduction & Objective:

In this project, we have tried to perform **semantic segmentation** i.e. pixel-wise classification of a digital image to partition it into multiple segments, of a **multimodal dataset collected in an off-road environment**, **RELLIS-3D**, which contains annotations for 13,556 LiDAR scans and **6,235 images**. We have used the images only. The data was collected on the Rellis Campus of Texas A&M University. It comprises **20 classes** viz. sky, grass, tree, bush, concrete, mud, person, puddle, rubble, barrier, log, fence, vehicle, object, pole, water, asphalt, and building.

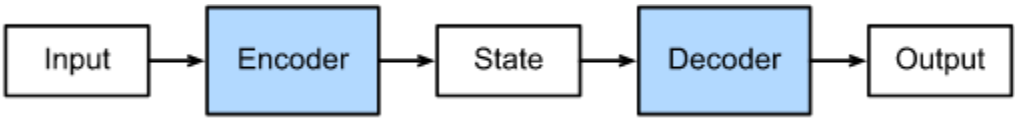
Our objective is to design a **lightweight (computationally less expensive)** algorithm to solve the above problem to be able to get **higher frames per second** while running the algorithm in real-time.

## First experiment:

We tried using the less complex start-of-the-art neural networks with encoder-decoder architectures to directly segment 20 classes.

## Theory:

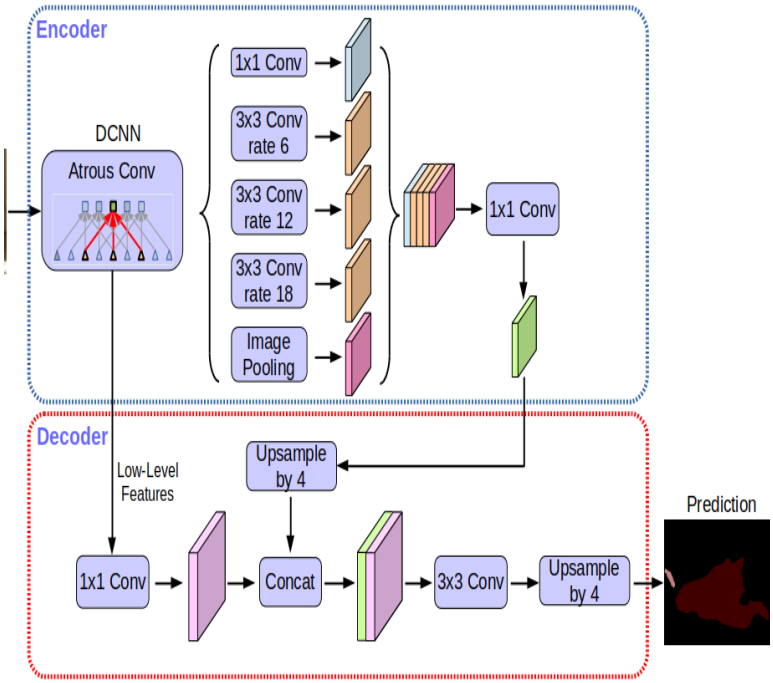
We have used an **encoder-decoder model** where an encoder is a network that takes the input and output of a feature map/vector/tensor. These feature vector hold the information, the features, that represents the input. The decoder is again a network (usually the same network structure as the encoder but in opposite orientation) that takes the feature vector from the encoder and gives the best closest match to the actual input or intended output. We have used this architecture as this is **computationally less expensive** and has a plethora of literature using the same.



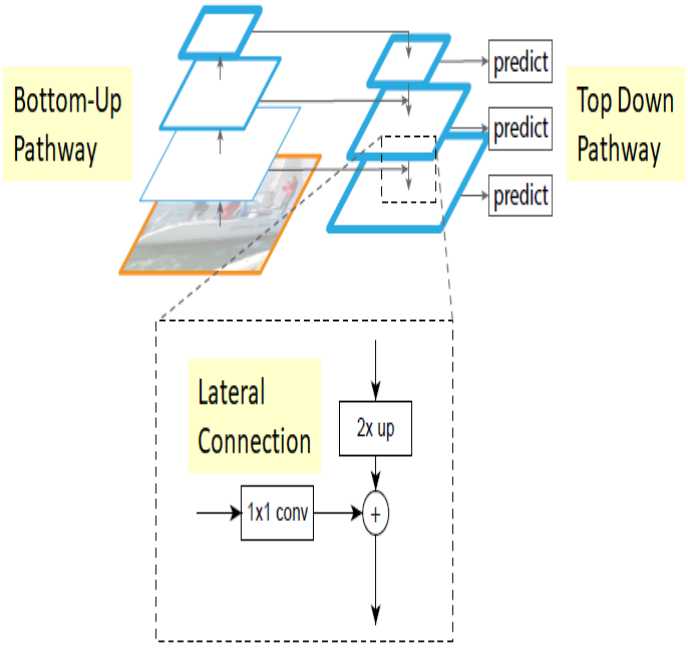

- The encoder is also referred to as the **backbone** of the model
- Few standard backbones **EfficientNet B0**, **ResNet 32**, **ResNet 34** etc.

**Decoders** that we experimented with are:

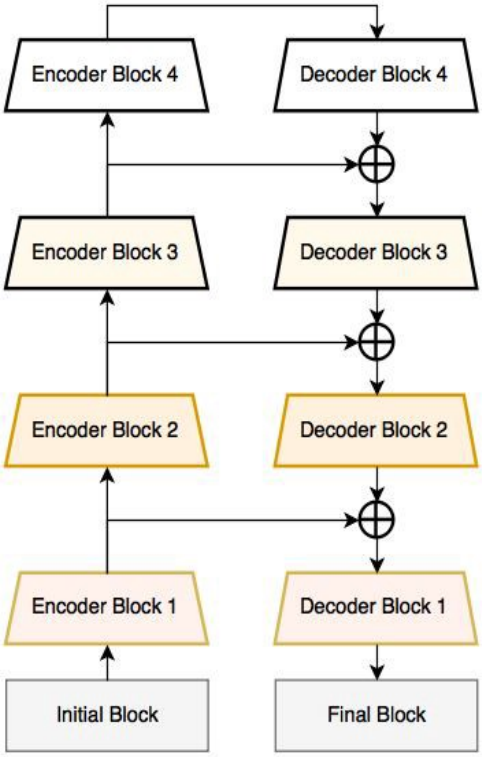
### DeepLab V3Plus



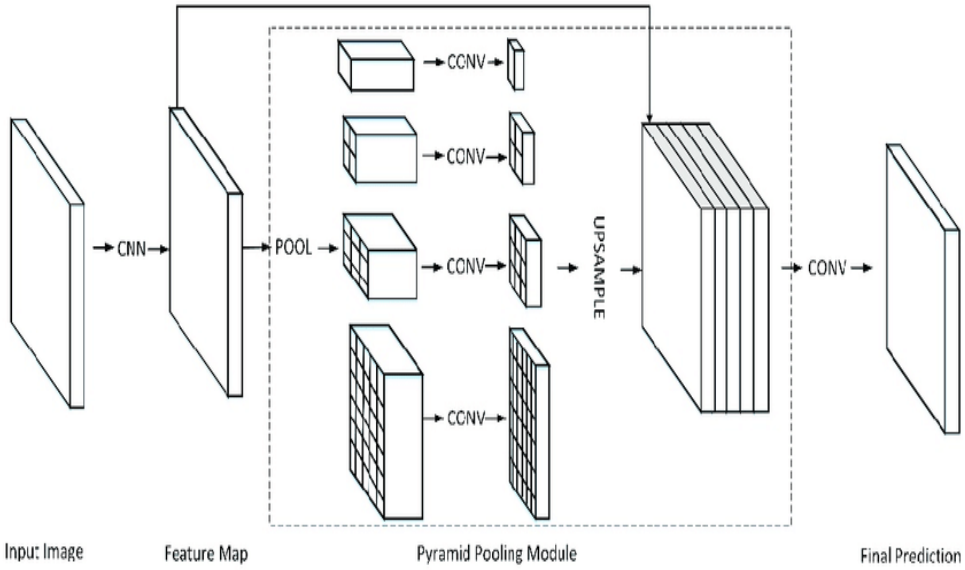

### Feature Pyramid Network



**LinkNet**




**Pyramid Scene Parsing Network**




Few salient features of the above mentioned neural networks are:

a) **DeepLabV3Plus**: It has **atrous (or dilated) convolutions** which are regular convolutions with a factor that allows us to expand the filter's field of view. The idea is to provide the model with multi-scale information. To do that, it adds a series of atrous convolutions with different dilution rates. These rates are designed to capture long-range context.
b) **Feature Pyramid Network:** It uses a **pyramid** of the same image at a **different scale** and combines low-resolution, semantically strong features with high-resolution, semantically weak features via a top-down pathway and lateral connections.
c) **LinkNet: Skip connections** residual blocks tries to efficiently share the information learned by the encoder with the decoder after each downsampling block. This proves to be better than using pooling indices in the decoder or just using fully convolutional networks in the

decoder. Not only this feature forwarding technique gives us good accuracy values but also enables us to have **few parameters** in our decoder.

d) **Pyramid Scene Parsing Network**: The **pyramid pooling module** is the main part of this model. The feature map from the backbone is pooled at different sizes after which upsampling takes place on the pooled features to make them the same size as the original feature map. This technique fuses the features at **different scales** hence aggregating the overall context of the image.

## Experiments & Observations:

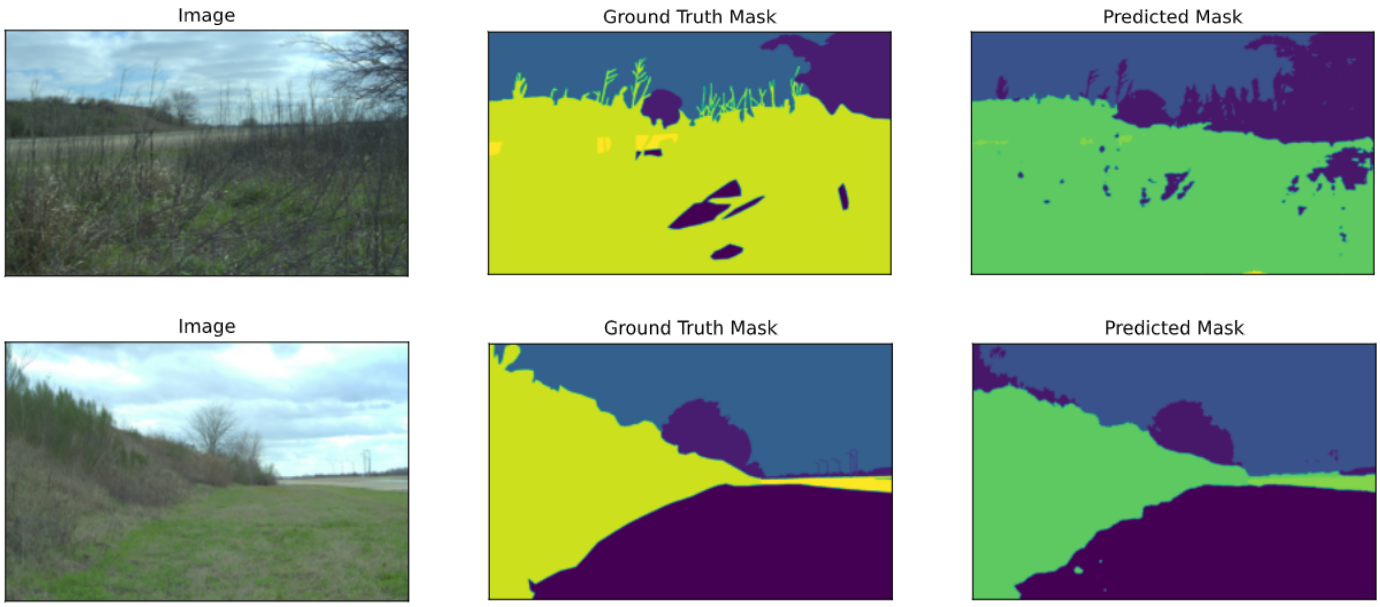
Grond-truth masks: Here we have created the ground-truth masks with unique values in each channel e.g. (0, 1) values in channel 1, (0, 2) values in channel 2, and so on unlike having (0, 1) i.e. one-hot encoding in every channel.

Loss Function: **Dice Loss function**

| **Model** | **Mean IOU** | **F-Score** | **Dice Loss** |
|---|---|---|---|
| **FPN + EfficientNet-B0 (Concat)** | **0.319** | 0.4826 | 0.5177 |
| **FPN + EfficientNet-B0** | **0.3174** | 0.4809 | 0.5194 |
| LinkNet + ResNet-32 | 0.3058 | 0.4675 | 0.5325 |
| PSPNet + EfficientNet-B0 | 0.2577 | 0.4082 | 0.592 |
| **DeeplabV3Plus + ResNet-34** | **0.3317** | 0.4423 | 0.5079 |
| LinkNet + EfficientNet-B0 | 0.2892 | 0.4414 | 0.578 |

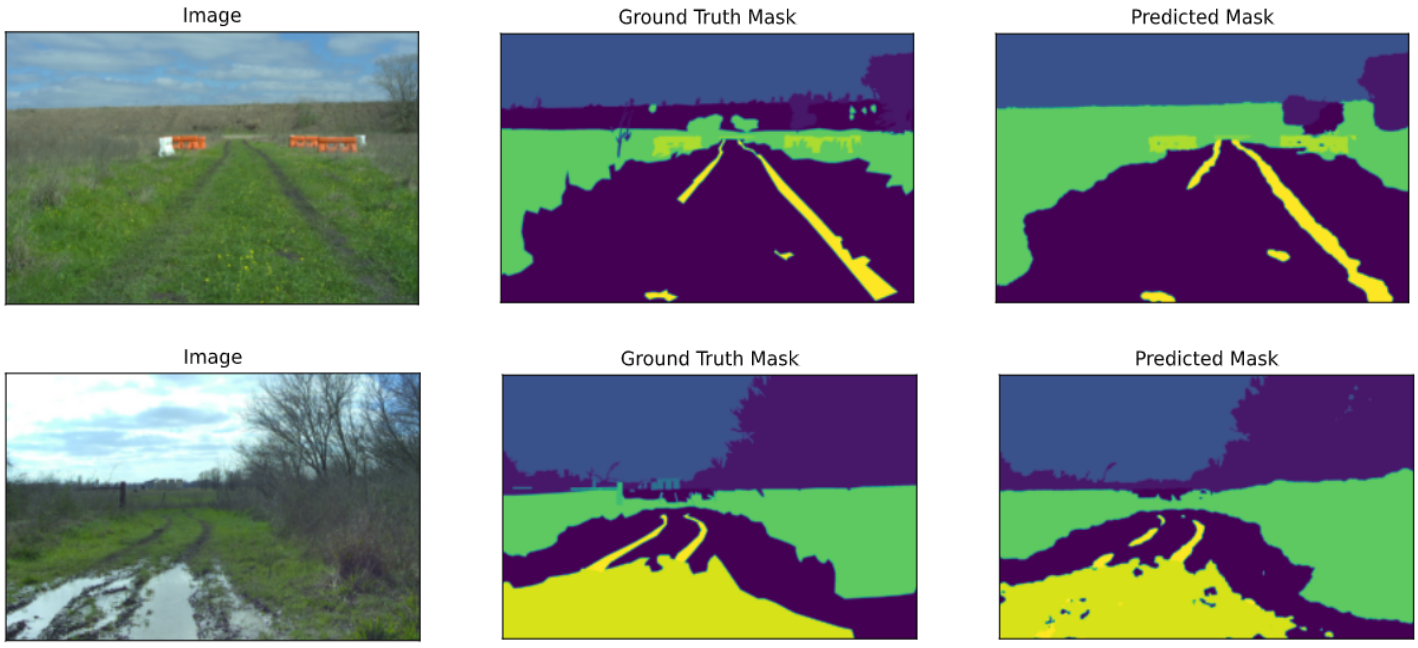
**LinkNet predictions:**

- Captured the **boundaries** of the object classes



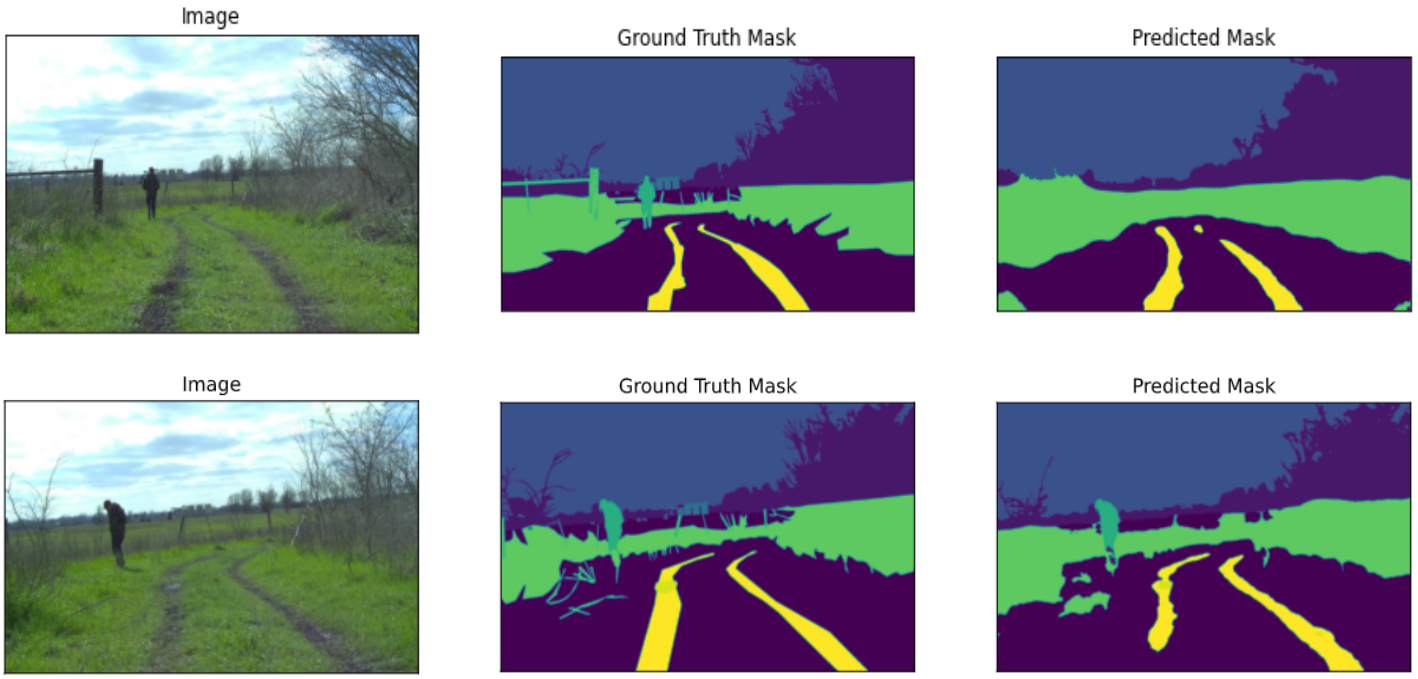

**FPN predictions:**

- Detected **small objects**, puddles, mud, rubble, etc., classes



**DeepLabV3Plus predictions:**

- Predicted the **person class** and **body of the object classes** really well




**Reasons for low scores:**

1) **Huge class Imbalance:** 7/20 classes comprise more than 80% of the total number of pixels in the dataset used.

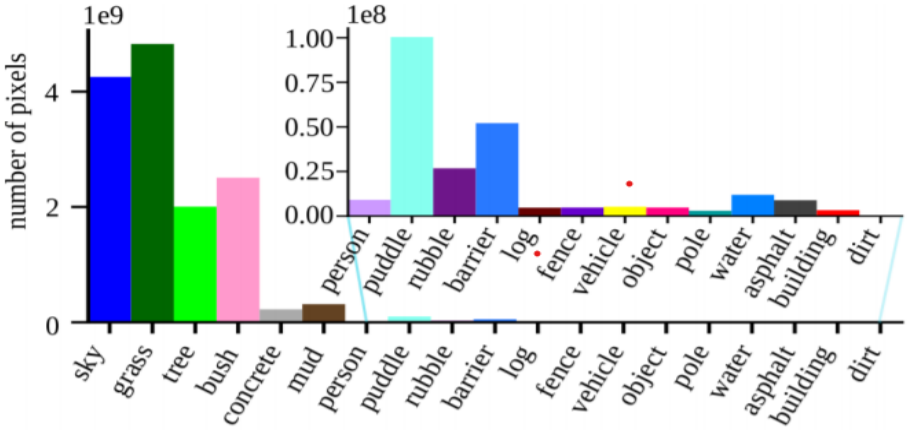
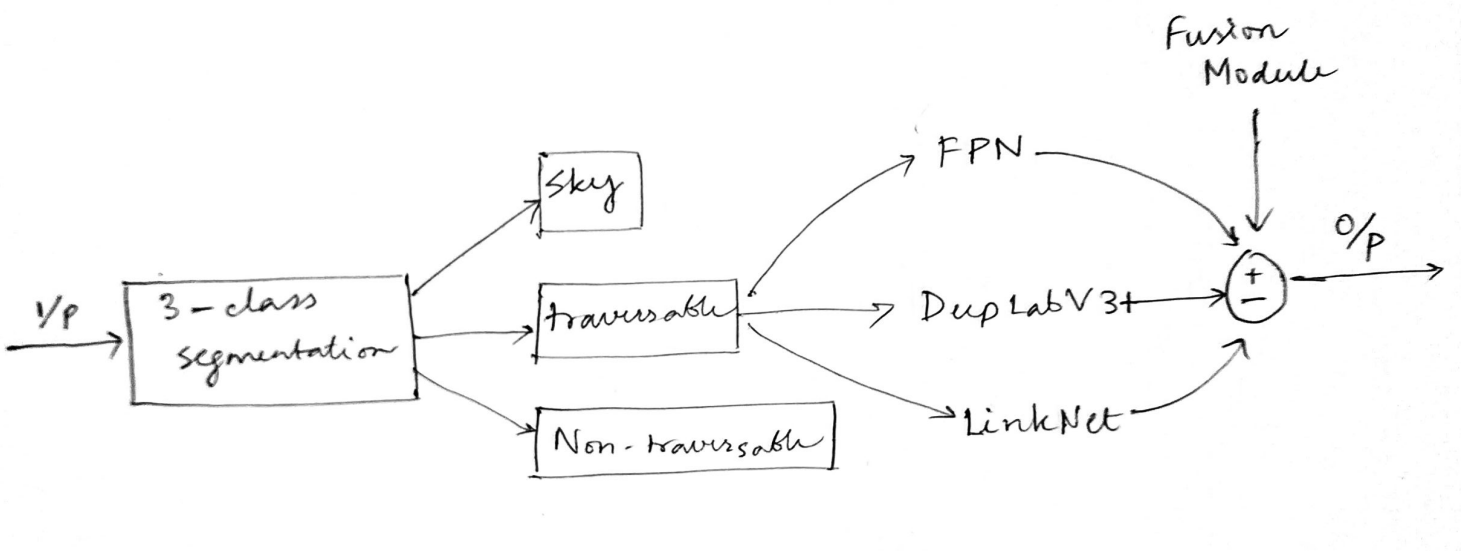



Fig. 3: Image Label distribution. The number of labeled points per class are shown.

2) The off-road environment's **unclear boundaries** also cause problems for both algorithms and human labelers.

## New hypothesis:

- We reduced to 20 classes to **3-classes** viz. **Taversable**, **Non-traversable,** and **Sky** classes.
- Then segment the **Traversable** class semantically by combining the predictions of **FPN**, **PSPNet,** and **Deeplab v3plus.**




- Try better loss functions like the **Tversky**, **Focal Tversky**, a **combination of dice, and binary cross-entropy,** etc.

## Theory:

After the architecture design, the major element of a neural network is its loss function which actually drives the learning in the network. The loss functions we chose for our problem are**:**

1) **Dice Loss:**

The dice coefficient is a **measure of overlap** of the predicted mask and the ground truth. Since it does not account for the background class, it cannot dominate over the smaller distribution classes. It works wonders for class imbalances.

$$DL(y, \hat{p}) = 1 - \frac{2y\hat{p} + 1}{y + \hat{p} + 1}$$

2) **Tversky Index:**

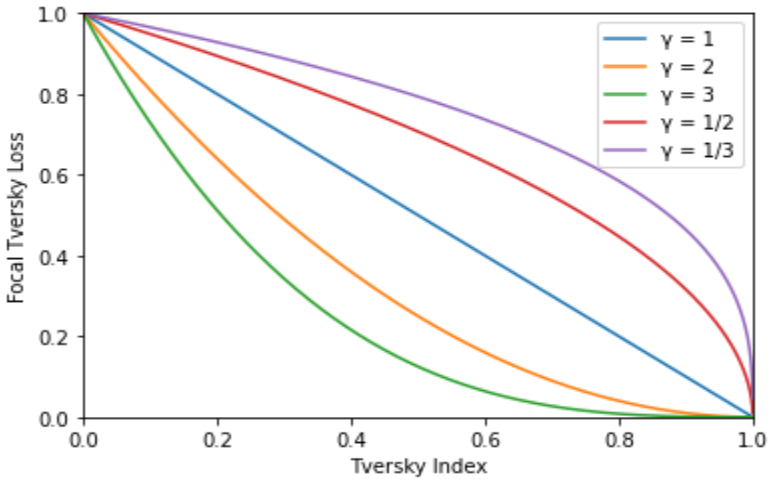
The Tversky Index adds two parameters, **α** and $\beta$, where **α + $\beta$ = 1**. By setting the value of **α** > $\beta$, you can penalize false negatives more. This becomes useful in highly imbalanced datasets where the additional level of control yields better small-scale segmentations.

$$\mathrm{TI} = \frac{\mathrm{TP}}{\mathrm{TP} + \alpha\mathrm{FN} + \beta\mathrm{FP}}$$

3) **Focal Tversky Loss:**

- γ is a parameter that controls the non-linearity of the loss
- For γ < 1, the gradient of the loss is higher for examples where TI > 0.5 which forces the model to focus more on such examples
- Towards the end of the training, the model still tries to learn even though Tversky Index is nearing convergence
- For class imbalance, γ is kept greater than 1. This helps in a higher loss gradient for instances where TI < 0.5. This forces the model to focus on harder examples, especially small-scale segmentations which usually receive low TI scores.
-

$$\mathrm{FTL} = (1 - \mathrm{TI})^{\gamma}$$




## Combining predictions:

Largely there can be three ways to combine multiple neural networks into one:

1. Have the two neural networks independent and train them separately, but combine the predictions just like **ensemble learning**
2. Have the two networks separate until some point on the networks and make a combination layer somewhere before the output layer
3. Build a neural network from scratch using logics and algorithms of all the neural networks

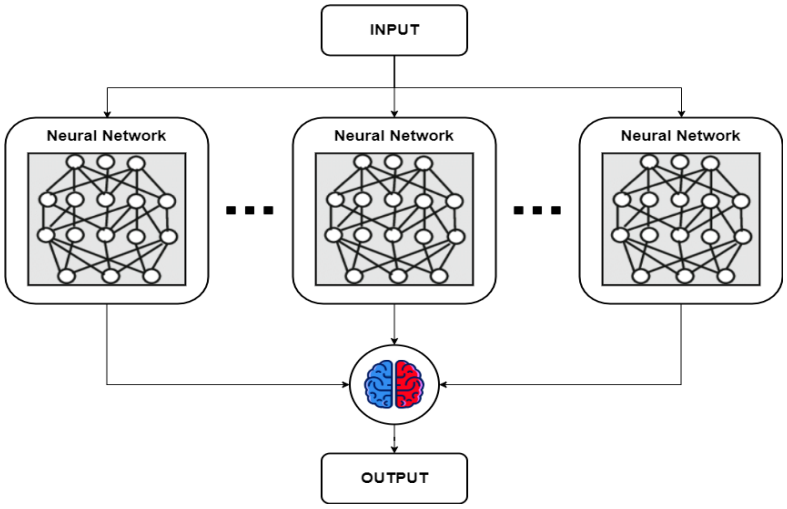
Drawbacks of method 2 & 3:

- Connecting all neural networks at some point results in a single large network with high computational complexity
- Combining the logic of neural networks is a very complicated task

Thus, we went ahead with method 1 of ensemble learning.

**Ensemble Learning**

- We train multiple models “good at different aspects” instead of a single model and **combine the predictions** from these models to get a single output mask.



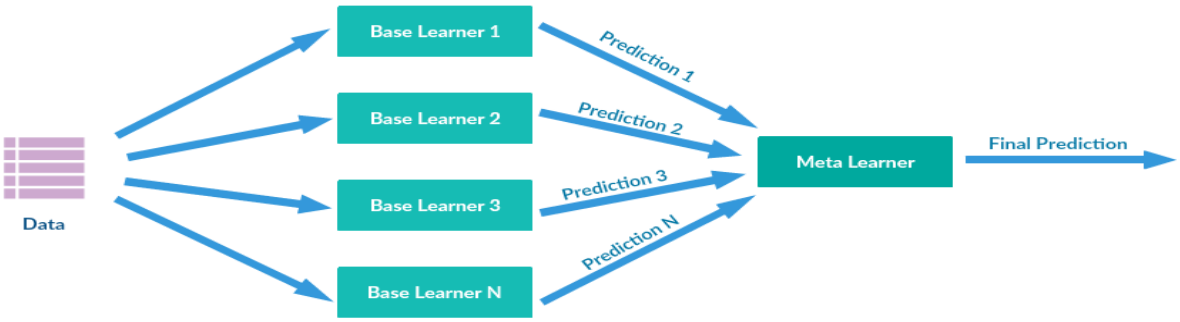

Neural network models can learn complex nonlinear relationships in the data. A downside of this flexibility is its stochastic nature, which leads to high variance.

- **Benefits of ensembling:**
  - Ensembling **adds a bias** that in turn counters the variance of a single neural network model. The resultant **predictions are less sensitive** to the specifics of the training data, choice of the training scheme, and the serendipity of a single training run
  - From the law of large numbers, we have an idea that averaging multiple models trained on the same dataset give us a spread for improved reliability, as it will further **decrease the standard deviation** of the accuracy of an ensemble on the test dataset. The ensemble can thus result in **better predictions** than any single best model.

**Ensembling Methods:**

- **Averaging:** The simplest way to combine is to calculate the average of the predictions from each of the sub-models.
- 
- **Weighted Averaging:** This can be improved slightly by weighting the predictions from each model, where the weights are small positive values and the sum of all weights equals one, allowing the weights to indicate the percentage of trust or expected performance from each model.

- **Stacked generalization (Stacking):** One of the advanced methods of ensemble learning is to train an entirely new model to learn how to best combine the contributions from each submodel. This method is a two-step process:
    - **Level 0**: The level 0 data is the training dataset inputs and level 0 models learn to make predictions from this data.
    - **Level 1**: The level 1 data takes the output of the level 0 models as input and the single level 1 model, or meta-learner, learns to make predictions from this data.
    - The meta-learner has to be trained on a separate dataset to the examples used to train the level 0 models to avoid overfitting. For that, we split the training dataset into a train and validation set. The level 0 models are then trained on the train set. The level 1 model is then trained using the validation set.




- **Integrated stacking:** The sub-networks can be embedded in a **larger multi-headed neural network** that then learns how to best combine the predictions from each input sub-model.

## Experiments:

➢ We used **DeepLabV3Plus** and **FPN** for the 3-class segmentation**.**

➢

➢ Trained **DeepLabV3Plus, FPN,** and **LinkNet** separately for the Traversable class segmentation.

➢

- ➢ We experimented with **weighted averaging** and **stacking ensembling** techniques to combine the predictions of the sub-models.
- ➢ **DeepLabV3Plus** and **FPN** were implemented as meta-learners in stacking.
- ➢
- ➢ **Tversky loss**, **Focal Tversky loss,** and **Dice loss functions** were tried in both the 3-class segmentation, base learner, and meta-learner network of stacking ensembles.

## Observations:

3-class segmentation: (Sky, Traversable, Non-traversable) (**Focal Tversky Loss**)

| **Decoder + Backbone** | **Mean IOU** |
|---|---|
| FPN + EfficientNet-B0 | **0.929** |
| DeepLab V3Plus + ResNet-34 | **0.9617** |

Overall segmentation (after combining predictions) (**Focal Tversky Loss**)

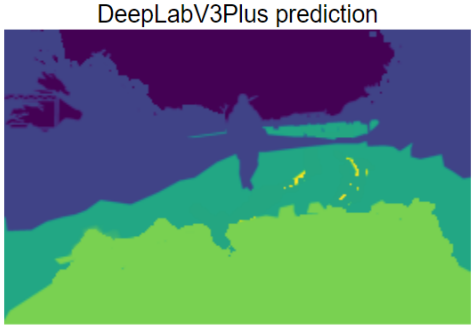
| **Ensemble Technique** | **Mean IOU** |
|---|---|
| Weighted Average | 0.3914 |
| FPN + EfficientNet-B0 | **0.5828** |
| DeepLab V3Plus + ResNet-34 | **0.6347** |

**Overall mask prediction example 1:**

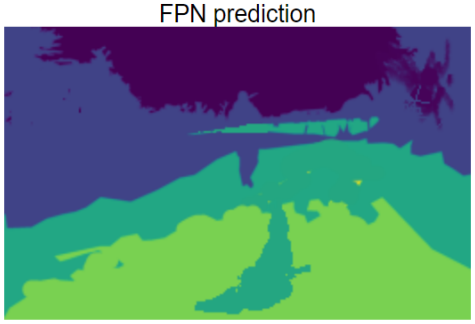
Original Image

3-class segmentation prediction

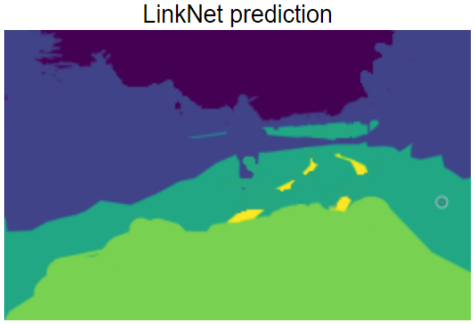
DeepLabV3Plus prediction



FPN prediction



LinkNet prediction



Ensemble prediction (DeeplabV3Plus)

- 3-class segmentation prediction came out really well as evident from the mean IOU score.
- LinkNet was able to capture parts of the mud strides on the grass.
- FPN got the boundaries much better than the other sub-model predictions.
- DeepLabV3Plus detected the person class and had maximum pixels correctly classified.
- The DeepLabV3Plus meta-learner rightly got the hints of the various objects in the image from the sub-models and did a great job at combining the sub-model predictions.

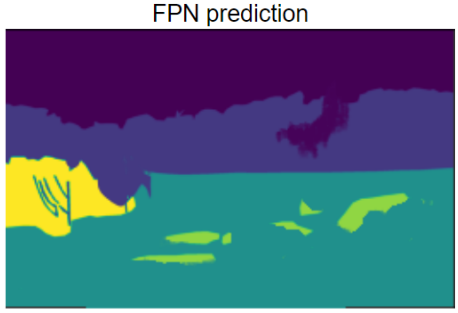
**Overall mask prediction example 2:**

Original Image

3-class segmentation prediction

DeepLabV3Plus prediction

FPN prediction



LinkNet prediction

Ensemble prediction (DeepLabV3Plus)

- We can see from this example too that the algorithm did a great job at predicting the final mask, comparing with the original image; as defining a clear boundary between the mud and the grass can be a difficult task even for a human; but, the model was still able to segment the mud in the grass.
- Even though the predictions are not this well for all examples of the test dataset but, these examples give us validation of our hypothesis.

## Limiting factors & Scope of improvement:

- **More Training** - Generally state of the art models are trained for 400-500 epochs to get the best results. Whereas, our algorithm neural networks had been trained for only 7-10 epochs due to scarcity of computational resources (each sub-model requires 16GB of RAM to train)
- **Rescaling base learner predictions** - A major limiting factor is that because all the sub-models have different architectures their prediction has different dimensions which require re-scaling before they are fed into the meta-learner network for combining. This leads to a loss in spatial information of the sub-model predictions.
- **Quality of data** - The data was collected in the rainy season and thus many boundaries (e.g. ones between grass and bush are not clear) which makes the traversable class harder to segment than the other classes.
- **Class imbalance** - Still exists e.g. most of the space in the traversable part is filled with grass whereas, the mud, puddles, etc. are quite less in density as compared to grass.
- **More data** - Most of the literature on computer vision states that feeding more data into the network can lead to better efficiency, Though there is a chance of overfitting after a certain limit, we have used much lesser data in comparison to that limit.

## Conclusion:

Even though the mean IOU scores aren't too high enough to put this algorithm to test in the real world but we need to keep in mind that this solution is much more computationally efficient than most of the current state-of-the-art solutions. Thus we can declare the experiments to be a success and we have been able to come up with a lightweight algorithm that can be used by an off-road vehicle to perform semantic segmentation in real-time at high frames per second, to be able to decide which direction to move in, with confidence.

Nowadays, to solve a computer vision problem, there is always a

hunch to create a new neural network architecture or use the current state-of-the-art algorithm for that particular task which is generally a very complex model without understanding how it works and why it works the way it works. I believe that to start solving a problem like this we should try to first break the problem down into smaller parts and try to find a simple solution to each of those parts. Then try to combine all those smaller task solutions into one to solve the entire problem. The outcome of this project stands as proof of this thought.

## References:


- Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam: Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation: arXiv: 1802.02611
- Abhishek Chaurasia, Eugenio Culurciello: LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation: arXiv: 1707.03718
- Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie: Feature Pyramid Networks for Object Detection: arXiv: 1612.03144
- Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, Jiaya Jia: Pyramid Scene Parsing Network: arXiv: 1612.01105
- Riyaz Sikora, O'la Hmoud Al-laymoun: A Modified Stacking Ensemble Machine Learning Algorithm Using Genetic Algorithms