Applied Artificial Intelligence (COMP 534)
THIRD ASSIGNMENT - Semantic Segmentation

Sr. No.	Name	Student ID
1	Rahul Nawale	201669264
2	Sanya Upadhyay	201680389
3	Saibharghav Pokala	201675984

In this assignment we have used deep learning models to solve a semantic segmentation task.

Dataset Introduction:

Dataset: Cambridge Labeled Objects in Video:

1. It consists of 101 images (960x720 pixel) in which each pixel was manually assigned to one of the following 32 object classes that are relevant in a driving environment: shown in Fig 1.
2. The "void" label indicates an area which ambiguous or irrelevant in this context.
3. The colour/class association is given in the file label_colors.txt. Each line has the R G B values (between 0 and 255) and then the class name.

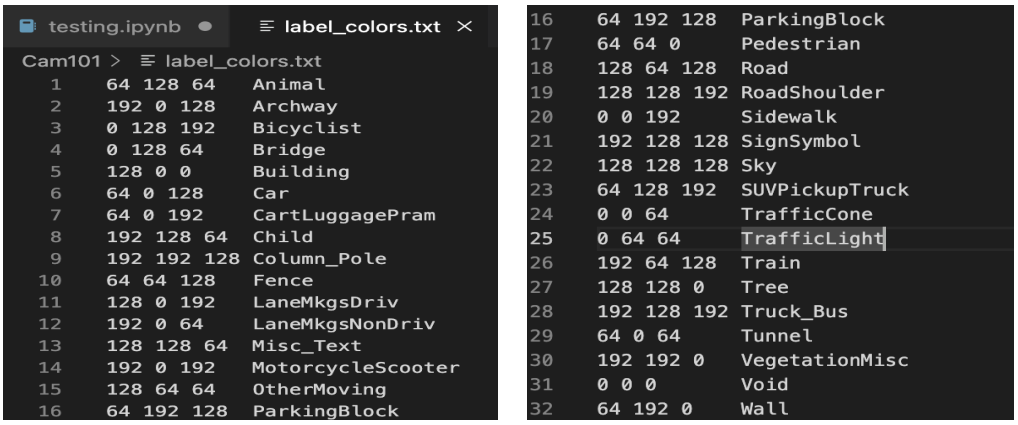


Fig 1 : 32 classes mentioned in labels_colors.txt file provided with dataset

The pixel labelling in the provided dataset enables training computer vision models with high precision in identifying and categorising objects present in a driving setting.

Output from the code in the file of the submission:

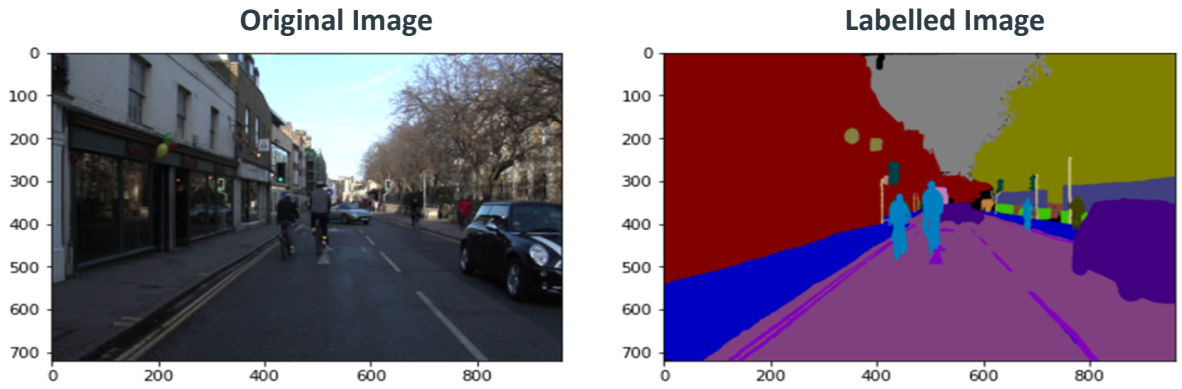


Fig 2 : CODE OUTPUT – Visualisation of the Dataset

Data Augmentation:

1. Data augmentation is a technique used to increase the diversity of the training data by applying transformations to the images and their corresponding labels.
2. The code uses the **Albumentations** library to perform data augmentation on a training dataset of images and their corresponding segmentation masks.
3. An augmentation pipeline is defined using the Compose function from the Albumentations library. In this case, the pipeline consists of a single HorizontalFlip transformation with a probability of 1.
4. The transformation is applied to each training image and its corresponding mask using the transform function. The transformed images and masks are appended to the lists.
5. This increases the size of the training dataset and can help prevent overfitting and improve the model's ability to generalise to new data.

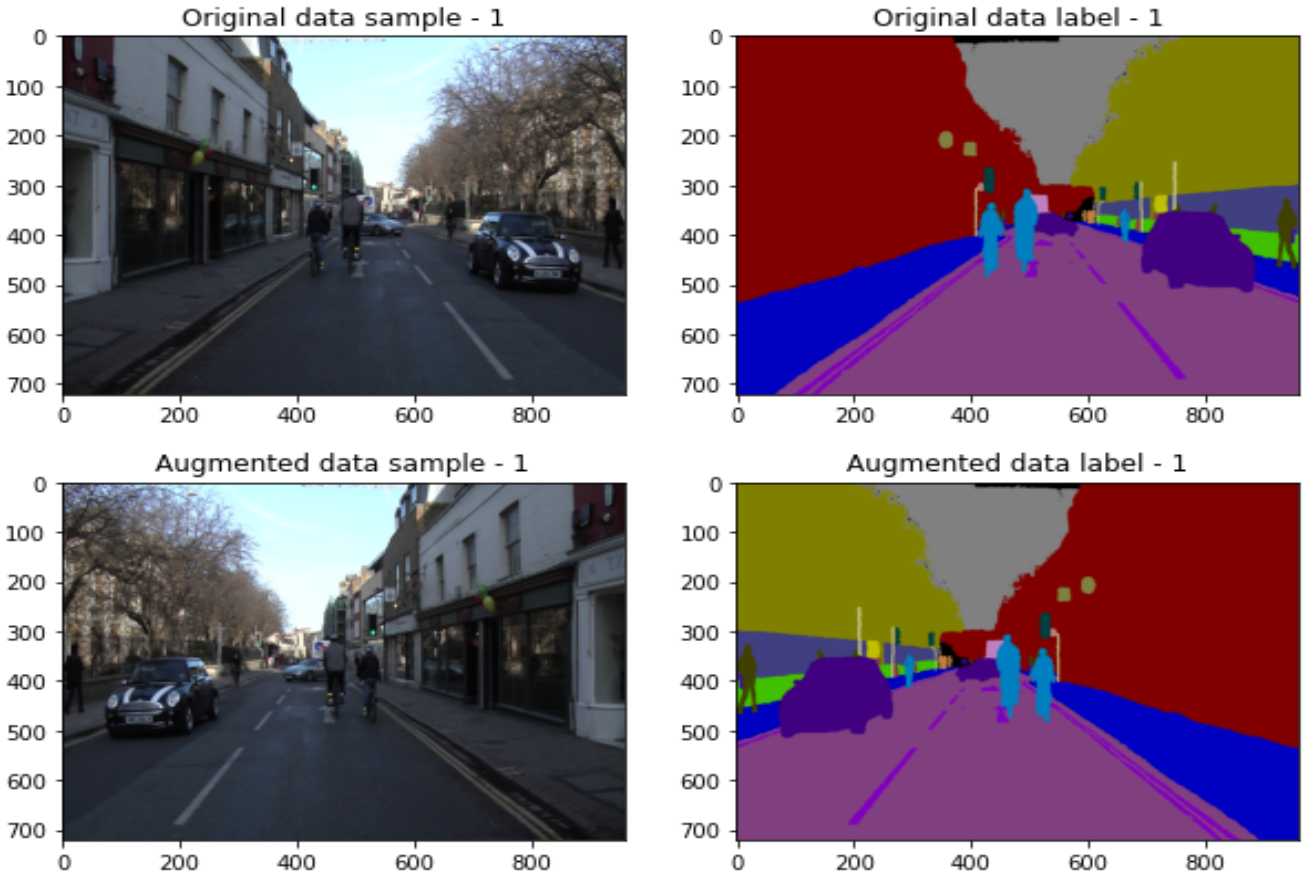


Fig 3: Code Output - Data Augmentation

Model Introduction:

We have used the following model to perform semantic segmentation:

1. FCN
2. UNET
3. DeeplabV3+

We have described hyperparameters used, structure of models and results of the training and testing in the next section.

Hyperparameters:

Various hyperparameters tested for the models are described in the following table.

Hyperparameters Used	FCN	UNET	DeepLab
Activation Function	Softmax, ReLU, ELU	ReLU, ELU	ReLU, Sigmoid
L1 Regularization	0.0 to 0.1	0.0 to 0.01	0.0 to 0.1 (step size 0.01)
L2 Regularization	0.0 to 0.1	0.0 to 0.01	0.0 to 0.1 (step size 0.01)
Drop Out	True(0.5) or False	True(0.5) or False	-
Optimizers	SGD, ADAM	SGD, ADAM	SGD, ADAM
Learning Rate	1e-4, 1e-2, sampling='log'	1e-4, 1e-3, 1e-2, 1e-1	1e-4, 1e-3, 1e-2
Epochs	5, 10	5, 10	5,10
Loss Function	'categorical_crossentropy', 'weighted_pixel_cross_entropy', 'dice_loss', 'pixel_wise_cross_entropy'	'categorical_crossentropy', 'weighted_pixel_cross_entropy', 'dice_loss', 'pixel_wise_cross_entropy'	'categorical_crossentropy', 'weighted_pixel_cross_entropy', 'pixel_wise_cross_entropy'
Metrics	Mean IoU, Pixel Accuracy	Mean IoU, Pixel Accuracy	Pixel Accuracy, Accuracy

FCN Structure and hyperparameters description:

1. The FCN model is created with a ResNet50 backbone. The ResNet50 model is used as a pre-trained feature extractor and its weights are frozen.
2. The output layer is created for the FCN task with a 1x1 convolutional layer followed by upsampling layers.
3. L1 and L2 regularization coefficients for the output layer are tunable hyperparameters.
4. The optimizer (either Adam or SGD) and its parameters (learning rate, beta_1, beta_2, epsilon for Adam; learning rate, momentum, nesterov for SGD) are tunable hyperparameters.
5. The loss function (categorical cross-entropy, weighted pixel cross-entropy, dice loss, or pixel-wise cross-entropy) and the metric (accuracy, mean IoU, dice coefficient, or pixel accuracy) are tunable hyperparameters.
6. A random search tuner is created to search the best hyperparameters.
7. The model is built with the optimal hyperparameters and trained.

FCN Design: Hyperparameters and Techniques: Reasoning

1. L1 and L2 regularisation coefficients for the output layer are tunable because they control the amount of

regularization applied to the model's weights. Regularization can help prevent overfitting by encouraging the model to learn sparse or small weights.

2. The optimizer and its parameters are tunable because different optimization algorithms work better for different problems.
3. The loss function is tunable because different loss functions can be more suitable for different problems.
4. The metric is tunable because different metrics can be more suitable for evaluating the performance of the model on different problems.
5. Transfer learning is used by using a pre-trained ResNet50 model as a feature extractor. This can improve the performance of the model by leveraging prior knowledge learned from a large dataset.
6. Upsampling layers are used to upsample the output of the ResNet50 model to produce an output with the same spatial dimensions as the input image. This is necessary for semantic segmentation tasks where a per-pixel prediction is required.

UNet Model: U-Net with ResNet50 backbone

The Model involves the following steps:

1. The U-Net model is created with a ResNet50 backbone. The ResNet50 model is used as a pre-trained feature extractor and its weights are frozen.
2. The output of the ResNet50 model is upsampled using a series of Conv2DTranspose layers with batch normalization, activation, and dropout.
3. The upsampled output is concatenated with the output of intermediate layers from the ResNet50 model using skip connections.
4. The final output is produced by applying another Conv2DTranspose layer with a sigmoid activation function.

U-Net with ResNet50 backbone: Hyperparameters and Techniques: Reasoning

1. Transfer learning is used by using a pre-trained ResNet50 model as a feature extractor. This can improve the performance of the model by leveraging prior knowledge learned from a large dataset.
2. Skip connections are used to concatenate the upsampled output with the output of intermediate layers from the ResNet50 model. This can help improve the performance of the model by allowing it to use both low-level and high-level features for making predictions.
3. Batch normalization is used to help the model converge faster by normalizing the activations of the previous layer at each batch.
4. Activation functions are used to introduce non-linearity to the model, which is essential for learning complex functions.
5. Dropout is used to prevent overfitting and improve the generalization of the model by randomly dropping out a fraction of the neurons during training.
6. A sigmoid activation function is used in the final output layer because it produces outputs between 0 and 1, which can be interpreted as probabilities for binary classification tasks.

DeepLabV3+

DeepLabV3+ is a popular convolutional neural network (CNN) model for semantic segmentation tasks.

1. The convolution_block function is defined, which performs a sequence of operations including a convolutional layer, batch normalization, and ReLU activation. This function is used to build convolutional blocks within the model.
2. PyramidPooling function implements the Dilated Spatial Pyramid Pooling module, which helps capture multi-scale contextual information. It takes an input tensor and performs average pooling followed by convolutional blocks with different dilation rates. The output of each block is concatenated with an upsampled version of the average-pooled tensor, resulting in a fused representation of multiple scales.
3. The deeplabV3Plus function defines the overall DeepLabV3+ model. It takes input shape and the number of target classes as parameters.
4. The model begins with an input layer, followed by a pre-trained **ResNet50 backbone**, initialized with weights from the ImageNet dataset. The ResNet50 model is truncated after the "conv4_block6_2_relu" layer, which provides a feature map with rich spatial information.
5. The feature map from the ResNet50 backbone is then passed through the PyramidPooling function, which extracts multi-scale features using dilated convolutions.
6. Two branches are created to capture different levels of detail: input_a and input_b. input_a is an upsampled version of the output from PyramidPooling, while input_b is obtained from an intermediate layer of the ResNet50 backbone ("conv2_block3_2_relu"). input_b is also processed with a convolutional block to reduce the number of filters.
7. The two branches are concatenated along the channel axis, merging the multi-scale features with lower-level details.
8. Additional convolutional blocks are applied to refine the combined features.
9. The output feature map is upsampled to match the original input size using bilinear interpolation.
10. Finally, a 1x1 convolutional layer is applied to produce the final output logits, representing the predicted class probabilities for each pixel.

Techniques that can affect the performance of a Model (FCN, UNET, DeepLabV3+) for semantic segmentation include:

- **Data augmentation**: Applying random transformations to the training data can increase the diversity of the training set and help prevent overfitting. This can improve the model's ability to generalize to new data.
- **Regularization**: Adding L1 or L2 regularization to the model's weights can help prevent overfitting by encouraging the model to learn sparse or small weights. This can improve the model's ability to generalize to new data.
- **Dropout**: Adding dropout layers to the model can help prevent overfitting by randomly dropping out units during training. This can improve the model's ability to generalize to new data.
- **Early stopping**: Stopping training early when the validation loss stops improving can help prevent overfitting by preventing the model from overfitting to the training data. This can improve the model's ability to generalize to new data.
- **Transfer learning**: Using a pre-trained model as a feature extractor can improve the performance of the

model by leveraging prior knowledge learned from a large dataset. This can improve the model's ability to learn from a smaller dataset.

Overfitting

- Overfitting occurs when the model performs well on the training data but poorly on new data. This can happen when the model is too complex relative to the amount of training data available. Underfitting occurs when the model performs poorly on both the training data and new data. This can happen when the model is not complex enough to capture the underlying patterns in the data.
- To deal with overfitting, techniques such as data augmentation, regularization, dropout, and early stopping can be used. These techniques help prevent overfitting by increasing the diversity of the training set, encouraging sparse or small weights, randomly dropping out units during training, and stopping training early when validation loss stops improving.
- To deal with underfitting, techniques such as increasing the complexity of the model or using transfer learning can be used. These techniques help improve performance by allowing the model to capture more complex patterns in the data or leveraging prior knowledge learned from a large dataset.

RESULTS and CODE OUTPUT:

Best Parameters found after hyperparameter tuning :

Hyperparameters	FCN	UNet	DeepLab
Activation Function	relu	relu	relu
L1 Regularization	0.0	0.0	0.08
L2 Regularization	0.0	0.0	0.01
Drop out	0.5	0.0	0.1
Optimizers	adam	adam	sgd
Learning Rate	0.0032336	le-4	0.01
Epochs	10(for 5 to 10)	10(for 5 to 10)	5
Loss Function	categorical crossentropy	categorical crossentropy	categorical crossentropy

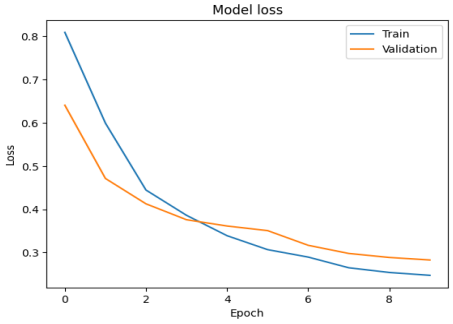
Table: Best Hyperparameter

FCN Model Results:(After 10 Epochs)

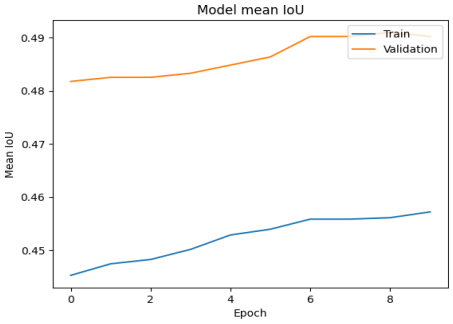
- **loss**: 0.3169
- **Pixel_accuracy**: 0.9042
- **Mean_iou** : 0.4877

Test pixel accuracy: 0.9041608572006226

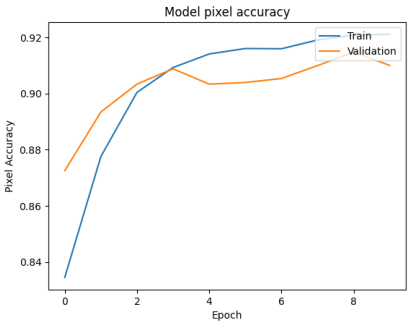
Test mean IoU: 0.48771



Loss vs Epoch

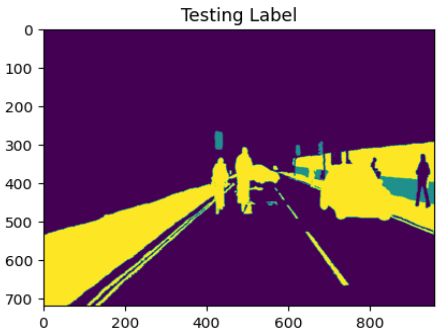


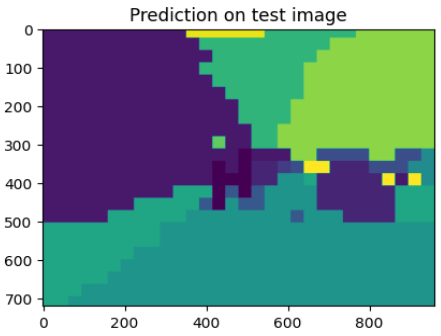
Mean IoU vs Epoch



Pi

FCN Prediction



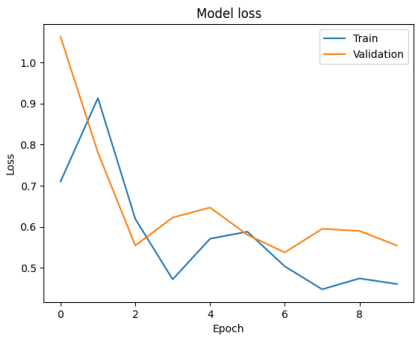


Comments on Result:

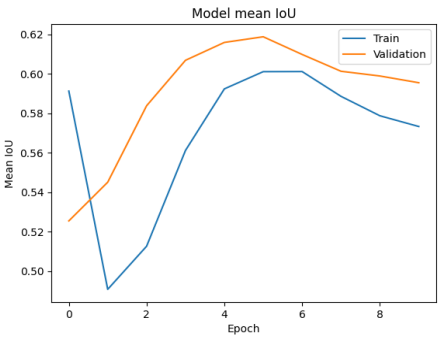
1. The FCN with ResNet50 backbone model achieved a training loss of 0.3169, a mean IoU of 0.4877, and a pixel accuracy of 0.9042. On the test set, the model achieved a mean IoU of 0.4877125918 and a pixel accuracy of 0.9041608572006226.
2. FCN architectures are simpler and more straightforward to implement. They can still achieve good performance on semantic segmentation tasks by using a pre-trained feature extractor and upsampling layers to produce per-pixel predictions.

UNet Results:

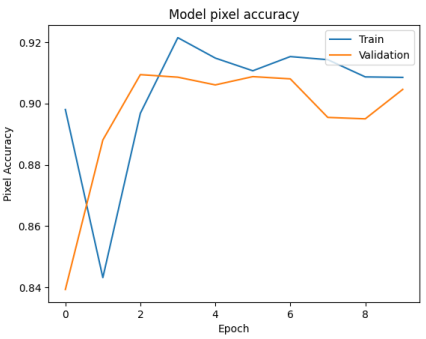
1. **loss**: 0.5775
2. **Mean_iou** : 0.5851
3. **Pixel_accuracy**: 0.9018
4. **Test mean IoU**: 0.5850850343704224
5. **Test pixel accuracy**: 0.9017722606658936



Loss vs Epoch

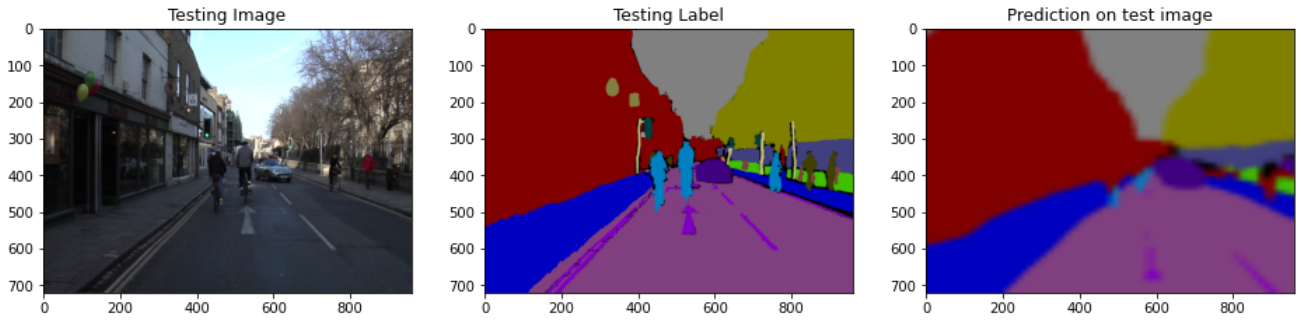


Mean IoU vs Epoch



Pixel Accuracy vs Epoch

Prediction:



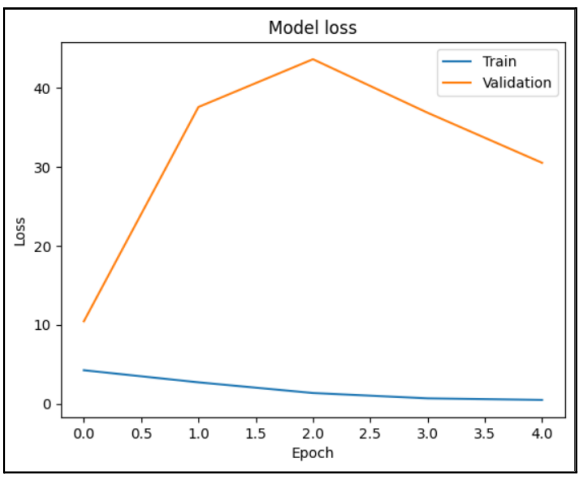
Comments on Result:

1. The U-Net with ResNet50 backbone model achieved a training loss of 0.5775, a mean IoU of 0.5851, and a pixel accuracy of 0.9018. On the test set, the model achieved a mean IoU of 0.5850850343704224 and a pixel accuracy of 0.9017722606658936.
2. U-Net with ResNet50 backbone model achieved slightly better performance in terms of mean IoU on both the training and test sets. However, the FCN with ResNet50 backbone model achieved slightly better performance in terms of pixel accuracy on both the training and test sets.
3. U-Net architectures are known for their ability to effectively combine low-level and high-level features using skip connections. This can help improve the performance of the model on semantic segmentation tasks by allowing it to use both fine-grained details and high-level contextual information when making predictions.

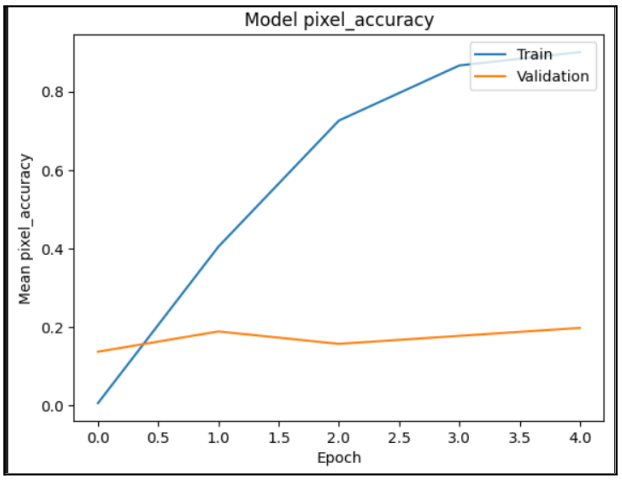
DeepLabV3+ Results:

After 5 Epochs
Categorical_crossentropy loss: 0.4686
Pixel accuracy : 0.9008

Result

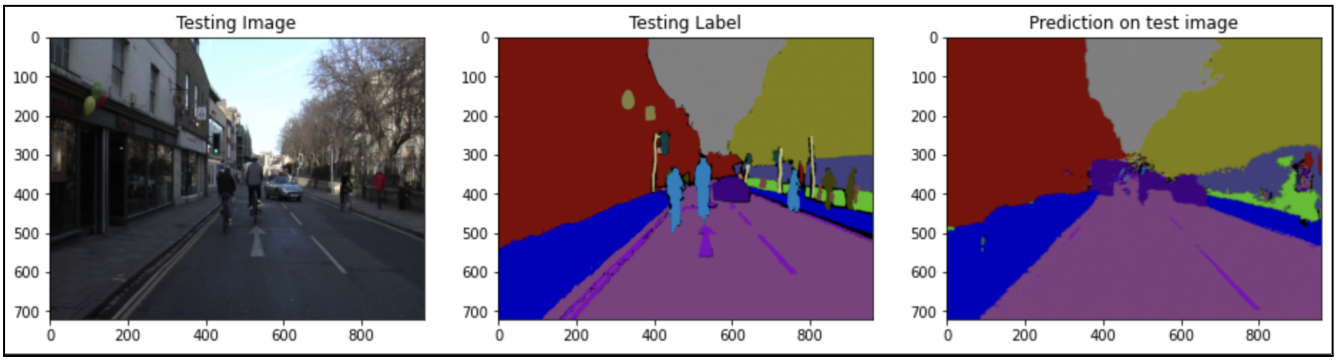


Loss vs Epoch



Pixel Accuracy vs Epoch

Prediction:



Comments on Results:

The value of 0.4686 represents the average loss computed using the categorical cross-entropy loss function. This loss is commonly used for multi-class classification problems. Lower values indicate better performance, as it means the model's predicted class probabilities are closer to the true class labels.

The value of 0.9008 represents the pixel accuracy metric, which measures the percentage of correctly classified pixels in the segmentation task. A higher pixel accuracy indicates better performance, as it means a larger portion of the pixels in the predicted segmentation map match the ground truth labels.

Conclusion:

FCN, U-Net, and DeepLabv3+ each have their own strengths and trade-offs. FCN is faster and simpler, U-Net performs well in limited datasets with preserved spatial information, and DeepLabv3+ excels in capturing multi-scale contextual information. The choice of model depends on the specific requirements of the segmentation task and the available resources.

A table comparing UNet, FCN, and DeepLabv3+ along with their pros and cons:

Model	UNet	FCN (Fully Convolutional Network)	DeepLabv3+
Architecture	U-shaped network with skip connections	Encoder-decoder architecture with skip connections	Encoder-decoder architecture with atrous convolutions
Pros	- Excellent performance for biomedical image segmentation tasks	- Captures both local and global context information	- Effective at handling both small and large objects
	- Strong performance for small dataset sizes	- Suitable for real-time applications	- Achieves state-of-the-art results on various datasets
	- Handles class imbalance well	- Can process images of any size	- Resistant to object scale variations

Cons	**- Limited receptive field**	**- May struggle with fine-grained details**	**- More complex architecture to train**
	- May overfit with larger datasets	**- Can be computationally expensive for large images**	**- Longer training time compared to other models**
	- May not perform well on large objects	**- Requires careful tuning of hyperparameters**	

- These pros and cons are not exhaustive and may vary depending on the specific implementation, dataset, and task requirements.
- It's important to consider these factors when selecting a model for your semantic segmentation task.

Your own reflection on this project, for example, what you have learned via this project, which part you could do better next time, etc.

- Through this project, We have learned that semantic segmentation is a challenging task that requires careful consideration of various factors such as model architecture, dataset size, and computational resources. While UNet, FCN, and DeepLabv3 are popular models for semantic segmentation, each has its strengths and weaknesses, as highlighted in the comparison table.

- Moving forward, to improve the accuracy of the model, we could explore various techniques such as data augmentation, transfer learning, and fine-tuning hyperparameters. Additionally, we could also consider using more advanced models such as Mask R-CNN or PANet, which have shown promising results in semantic segmentation tasks.

- Furthermore, we could also consider optimizing the code for faster computation time and better memory management. This could involve using parallel computing techniques, optimizing the use of GPU, and minimizing unnecessary computations.

- Overall, this project has been a valuable learning experience, and we look forward to applying the knowledge gained to future projects.