CHAPTER 5

SEMANTIC SEGMENTATION OF CRACKS ON MASONRY SURFACES USING DEEP LEARNING (DL) TECHNIQUES

5.1 General

The term "masonry surfaces" describes faces of building blocks such as- bricks, stones or concrete that are mortared together. Numerous infrastructure components, such as structures, bridges, walls, and historical monuments, utilise these materials. Masonry varieties include brick, stone, and concrete block masonry, each with a distinctive texture, arrangement, and structural characteristics. When creating crack detection models, it's crucial to consider this diversity because different types of masonry may have varied crack patterns and appearances. Infrastructure, such as highways, railways, and bridges, play a major role in the development of any country. Surface cracks can be caused by various factors such as weathering, structural loads, and seismic activity etc. These cracks can lead to significant damage if not detected and repaired on time. For uninterrupted movement of traffic, structural health monitoring of structure and removal of the same is mandatory in case of any defect. Cracks are common defects on concrete or masonry surfaces. As the concrete surface has a uniform texture, any crack available on the concrete surface can easily be detected. Nevertheless, in the case of masonry surface, the image surface has uneven texture, so crack detection over masonry surface is a complex task compared to a concrete surface or asphalt surface.

Most Indian railway bridges older than 50 years are having masonry substructures. Traffic density, axle load, and speed have increased significantly since the construction date of

the bridge. Due of the increase in traffic density, axle loading, and speed of trains, there is a major probability of old masonry substructure getting cracked, so frequent inspection of all bridges is mandatory for detecting cracks at the initial stage and maintenance of them within the time limit. Previously, visual inspection was the main technique for crack detection. Since the physical inspection of all the bridges is a time-consuming, laborious, imprecise, uncertain, and costly affair. In many bridge locations, it is almost impossible to reach in person due to the unavailability of space and support system. The semantic segmentation of cracks can overcome all the above shortcomings without much human interventions. Health monitoring of bridge substructure can be done using the approach mentioned in this study.

Detecting cracks can be challenging, especially on rough surfaces such as masonry. This research study focuses on the detection of surface cracks on masonry surfaces using deep learning techniques. This study compares the performance of various networks trained using deep learning techniques for semantic segmentation of cracks on masonry surfaces.

5.2 Methods of crack detection

Traditional methods (edge detection, filtering, thresholding, morphological operations), Classical ML-based methods (SVM, random forest, Neural Network (NN)) and DL-based methods (Convolutional neural network (CNN)) are the three categories of methods available for image crack detection. Deep learning-based crack detection techniques can be categorized into image classification, object detection, and Semantic Segmentation (SS).

5.2.1 Traditional Methods

Traditional methods (Woods et al. 2004) are defined with their advantages and limitation below:

1. Edge Detection: Edge detection techniques, like Canny edge detection, Sobel, or Prewitt filters, highlight edges in the image. Cracks often have distinct edge characteristics, making edge detection useful for detecting image cracks.

   These methods can quickly identify boundaries between objects, making them suitable for detecting straight and well-defined crack edges. They are computationally efficient and can be applied to real-time applications. However, these are noise-sensitive and unable to handle complex crack shapes or faint cracks well.

2. Filtering: Filtering techniques, such as Gaussian or median filtering, are used to enhance crack features or remove noise from the images, which can improve crack detection results.

   These techniques can enhance crack features and reduce noise, improving the quality of crack images for subsequent analysis. They are simple and easy to implement. However, Selecting the appropriate filter size and parameters can be challenging. Filtering may inadvertently smooth cracked edges, making them less distinguishable from the background.

3. Thresholding: Thresholding techniques binarize the image based on a predefined threshold value. Simple thresholding can be useful for segmenting cracks from the background.

   Thresholding can efficiently convert grayscale images to binary representations. Nevertheless, choosing an appropriate threshold is critical. Threshold-based methods may not handle varying lighting conditions or crack intensities well.

4. Morphological Operations: Morphological operations, such as dilation and erosion, can refine the detected crack regions, fill gaps, or remove small noise regions.

   These operations can refine crack regions and connect disjointed segments. They are effective in removing small noise regions. However, these operations may introduce unwanted artifacts, and their performance highly depends on the choice of structuring elements. They may not adequately handle complex crack shapes.

5.2.2 Machine learning (ML) based Methods

ML-based methods (Bishop and Nasrabadi 2006) are defined with their advantages and limitation below:

1. SVM (Support Vector Machines): SVMs are supervised learning algorithms that provide a way to categorize data using an optimal hyperplane. In the context of crack detection, SVM can be trained to distinguish between crack and non-crack regions based on handcrafted features.

   In high-dimensional feature spaces, SVM performs well, making it suitable for large-scale tasks. SVM can handle overfitting and non-linear decision boundaries. However, it is costly, require domain expertise, and sensitive to noisy data.

2. Random Forest: In Random Forest, multiple decision trees are combined to make predictions. It is robust and effective for classification tasks, including crack detection, where multiple trees vote on the class labels.

   This method uses an ensemble of decision trees, which helps to reduce overfitting and improve generalization. It can handle large datasets, and measure feature importance. However, the ensemble nature of this method can make it less interpretable than individual decision trees. Moreover, it may not perform well on imbalanced datasets.

3. Neural Networks (NNs): Artificial Neural Network (ANN) is the popular network in the ML-based techniques. It is a type of machine learning algorithm that aims to recognize complex patterns and relationships in data through a series of interconnected artificial neurons. These neurons, also known as nodes or units, are organized into layers, consisting of input, hidden, and output layers.

   The advantages of using ANN for image-based crack detection include their proficiency in pattern recognition, automatic feature extraction, adaptability to various

crack types, and improved performance with deep learning techniques like CNNs. However, ANN's limitations are more time consumption, costly data collection, and overfitting.

5.2.3. Deep learning (DL) based Methods

Convolutional Neural Networks (CNNs) have gained tremendous popularity for crack detection due to their ability to automatically learn features directly from raw images (Garcia-Garcia et al. 2017).

CNNs consist of multiple convolutional layers that learn various spatial features from the input image. The layers are typically followed by pooling and fully connected layers for classification. CNNs can learn hierarchical representations, making them well-suited for complex tasks like crack detection.

CNNs have demonstrated exceptional performance in various image-related tasks. They can automatically learn hierarchical features from data, making them well-suited for crack detection. Transfer learning with pre-trained models can boost performance on limited datasets. However, CNNs require a large amount of labeled data for training. Fine-tuning deep architectures on crack datasets demands significant computational resources. CNNs may not generalize well to unseen crack patterns or different environmental conditions.

5.3 Semantic Segmentation (SS) of cracked images

Semantic segmentation technique is used for pixel level crack detection, here each pixel is classified as a cracked or a non-cracked pixel. Currently, it is most preferred technique due to its precision capability. The detection of cracks in surfaces with smooth textures, such as RCC structures and pavement, has been presented in previous studies. However, very few studies

offered masonry surface crack detection. The most popular technique for crack segmentation in recent years is semantic segmentation of images. In this study, semantic segmentation of masonry surfaces has been done using various DL techniques. This study utilizes an image-based crack detection method using FCN to overcome the challenges of conventional image-based methods that require complex image pre-processing techniques for crack feature extraction.

5.3.1 Segmentation Models

For the semantic segmentation of cracks, the segmentation models U-Net (Ronneberger et al. 2015), FPN (Lin et al. 2017), DeepLabV3+ (Chen et al. 2017) (Chen et al. 2018), and PSPNet (Zhao et al. 2017) are integrated with several CNNs acting as the network's backbone. Backbone network used in the study are inceptionv3 (Szegedy et al. 2016), mobilenet (Howard et al. 2017), resnet34 (He et al. 2016), resnet50 (Simonyan et al. 2014), and vgg19 (Simonyan et al. 2014). Two types of loss functions, BCE and binary focal loss (Lin et al. 2020), are used herein study. Segmentation models U-Net, FPN, and PSPNet used in the present study are Fully Convolutional Networks (FCN). DeepLabV3+ is a DCNN type network. Different models with different pre-trained models as the backbone are used to improve model performances in the segmentation processes. The methodology is common for every base model and backbone used in the present study. FCN model (Yang et al. 2018) is useful in the semantic segmentation of cracks. The feature extraction backbone present in FCN is identical to CNN; however, a fully convolutional layer rather than a fully connected layer is employed to get spatial map output for each class. A schematic diagram of FCN is shown in Figure 5.1.

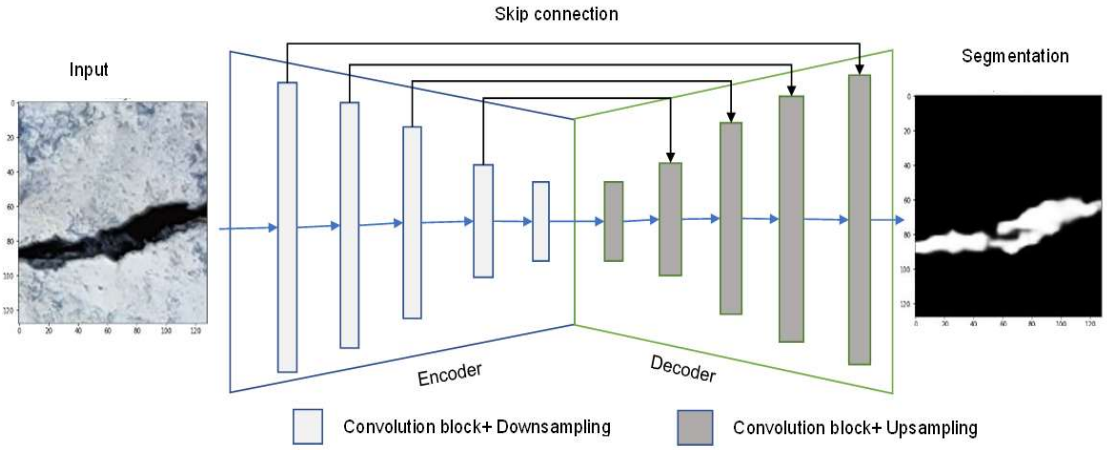


Figure 5.1: Schematic diagram of Fully Convolutional Networks (FCN).

Size of the input image used in the present study for U-Net, FPN, DeepLabV3+, and PSPNet segmentation model is (128,128,3), (128,128,3), (128,128,3), and (384,384,3). According to Yakubovskiy's work (2019), the U-Net, FPN, and PSPNet models are being implemented with various CNNs as the backbone. Segmentation models implemented in the study are briefly described as follows:

5.3.1.1 U-Net

U-Net (Ronneberger et al. 2015) is an encoder-decoder architecture designed for biomedical image segmentation, but its effectiveness has been demonstrated in various segmentation tasks, including crack detection on masonry surfaces. The architecture of segmentation model U-Net can be seen in Figure 5.2.

The U-Net architecture consists of two main parts: the contracting path and the expanding path.

Contracting Path (Encoder): This part of the network is responsible for capturing context and reducing the spatial resolution of the input image. It is composed of several convolutional layers

followed by max-pooling operations. These operations progressively downsample the feature maps, allowing the network to learn high-level features and semantic information.

Expanding Path (Decoder): The expanding path is used for precise localization of objects. It involves upsampling the feature maps using transposed convolutions (deconvolutions). A concatenation is then performed between the upsampled feature maps and the contracting paths corresponding to these upsampled feature maps. The concatenation enables the network to recover spatial information lost during downsampling and perform detailed segmentation.

U-Net architecture is simple and efficient, making it computationally inexpensive. The skip connections between the contracting and expanding paths facilitate precise localization of objects. It can work well with limited labeled data due to its effective skip connections.

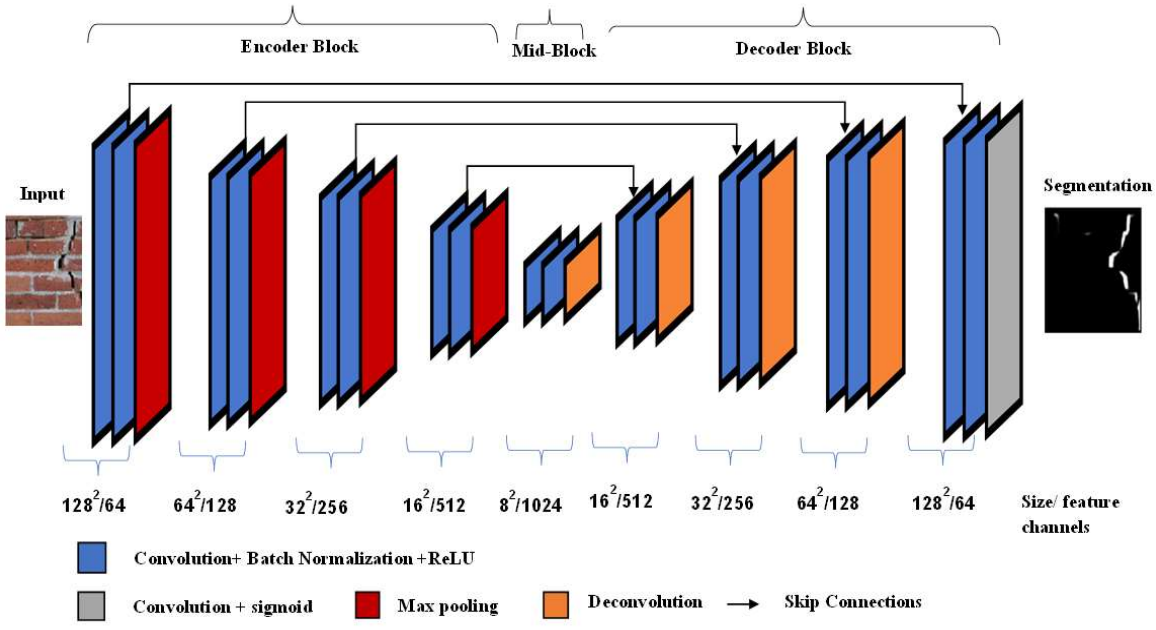


Figure 5.2: Illustration of the architecture of segmentation model U-net used in this study.

5.3.1.2 FPN (Feature Pyramid Network)

FPN (Lin et al. 2017) is designed to address the challenge of capturing multi-scale features for object detection and segmentation tasks. FPN introduces a top-down architecture with lateral connections to combine multi-scale features from different layers of a backbone network (usually a pre-trained classification network like ResNet). The architecture of the segmentation model FPN can be observed in Figure 5.3(a).

Bottom-up Path (Backbone Network): The bottom-up path consists of several convolutional layers from a pre-trained classification network. It extracts hierarchical features with increasing receptive fields.

Top-down Path (FPN): The top-down path starts from the highest-resolution feature map and uses upsampling (e.g., through interpolation) to generate higher-resolution feature maps. These upsampled feature maps are then combined with lateral connections from the corresponding layers in the bottom-up path.

Skip Connections (Lateral Connections): The lateral connections help to connect low-level and high-level features, facilitating the flow of information from different scales. This integration of multi-scale features allows the network to handle objects of various sizes more effectively.

FPN effectively captures features at multiple scales, making it well-suited for handling objects of different sizes. It enhances the performance of object detection and segmentation models by incorporating context from different feature levels.

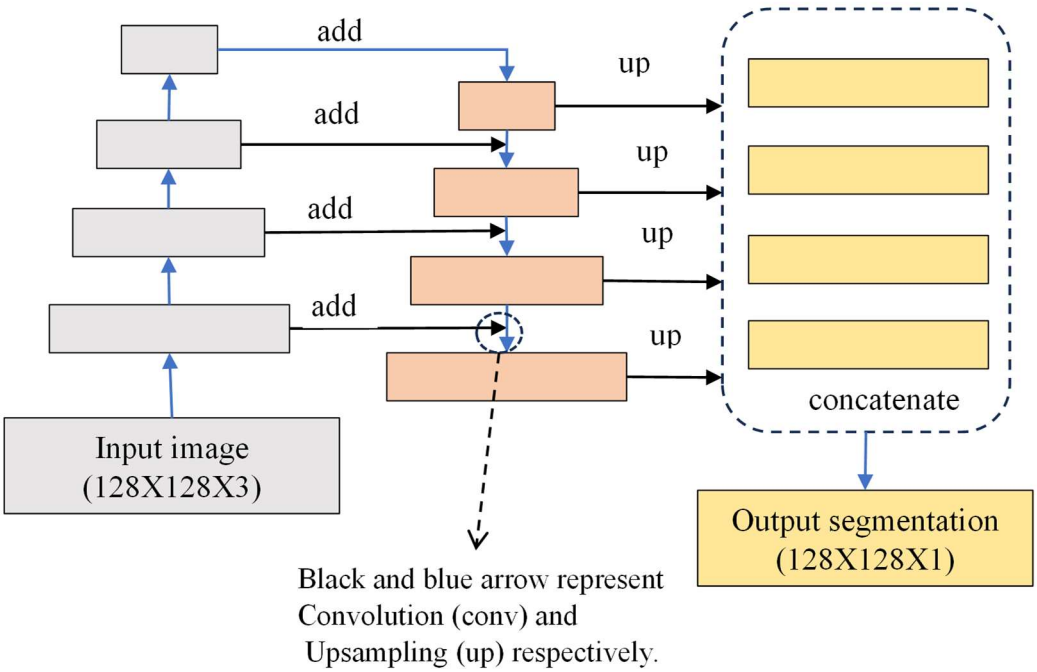


Figure 5.3(a): Illustration of the architecture of segmentation model FPN.

5.3.1.3 DeepLabV3+

DeepLabV3+ (Chen et al. 2017, 2018) is an extension of the DeepLab series, focusing on capturing long-range context information while maintaining fine-grained details. DeepLabv3+ builds upon the DeepLabv3 model and introduces an additional decoder module to refine the segmentation results further. The architecture of segmentation model DeepLabV3+ is provided in Figure 5.3(b).

Atrous (Dilated) Convolution: DeepLabv3+ uses atrous convolutions with various dilation rates to capture multi-scale context. Using atrous convolutions, the network can increase the receptive field without adding more parameters.

Encoder-Decoder with ASPP: The encoder part of the network uses atrous spatial pyramid pooling (ASPP) to capture contextual information at multiple scales. ASPP applies multiple

parallel atrous convolutions with different dilation rates to the same input feature map. The outputs of these convolutions are concatenated, providing rich context information.

Decoder: The decoder module in DeepLabv3+ involves upsampling the feature maps using bilinear interpolation. It also incorporates skip connections from the encoder to refine the segmentation output.

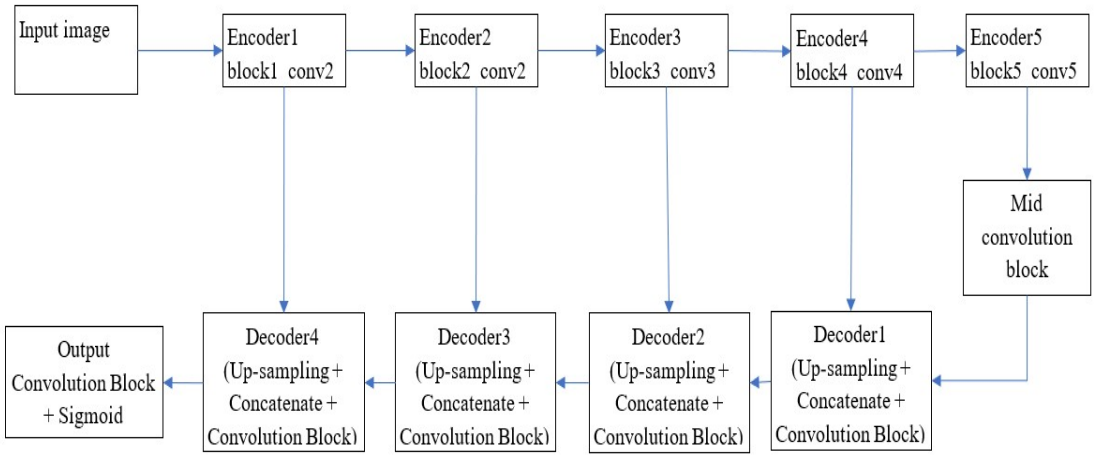


Figure 5.3(b): Illustration of the architecture of segmentation model DeepLabv3+.

DeepLabv3+ effectively captures global and local contextual information, making it suitable for tasks requiring accurate localization and context understanding. The ASPP module enables multi-scale analysis without introducing a large number of parameters. Skip connections aid in preserving fine details during the upsampling process.

5.3.1.4 PSPNet (Pyramid Scene Parsing Network)

PSPNet (Zhao et al. 2017) is designed to capture context at different scales using pyramid pooling modules. PSPNet utilizes a pyramid pooling module to aggregate multi-scale contextual information. The architecture of segmentation model PSPNet can be observed in Figure 5.3(c).

Pyramid Pooling Module: The core of PSPNet is the pyramid pooling module, which divides the input feature map into a fixed number of regions. It then performs global pooling (usually average pooling) in each region. These pooled features are then concatenated and passed through convolutional layers to generate a fixed-size representation, which encodes multi-scale context.

Encoder-Decoder Architecture: PSPNet adopts an encoder-decoder architecture. The encoder is typically a pre-trained classification network like ResNet or VGG, which captures hierarchical features. The pyramid pooling module is added to the encoder to capture global context. The decoder then upsamples the feature maps and refines the segmentation results.

PSPNet effectively captures multi-scale context through the pyramid pooling module. It achieves competitive performance with reduced computational complexity compared to other methods that rely heavily on dilated convolutions. It is flexible and can be combined with various backbone architectures.

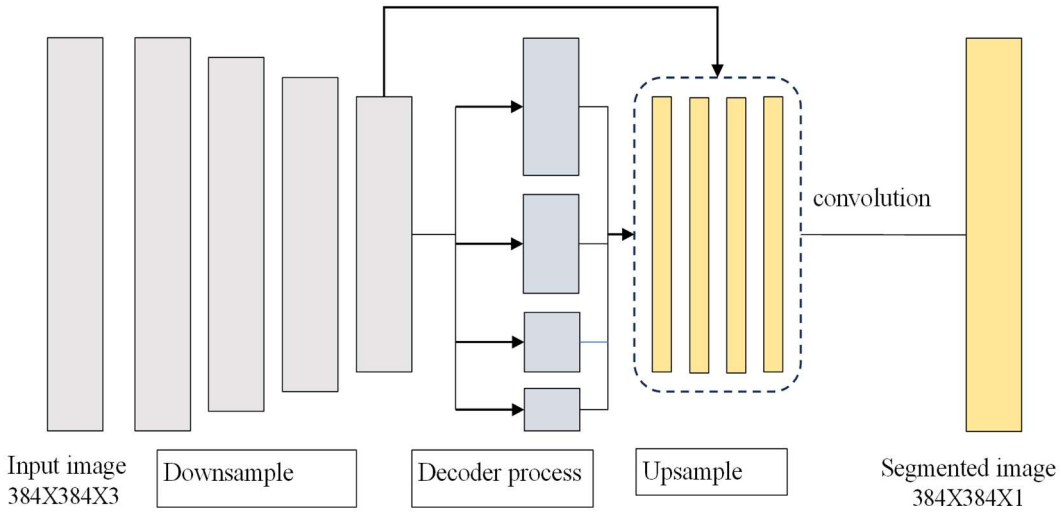


Figure 5.3(c): Illustration of the architecture of segmentation model PSPNet.

5.3.2 Training Configuration

Photo have been collected from various sources. Dataset for the images is taken from Dais et al. (2021). There is total 240 images of size 224X224X3 in the masonry dataset. Additionally, 60 images of existing masonry box culvert bridges are added to the masonry dataset. Pictures are taken with an oneplus8 smartphone. The images utilized in the study are of old existing masonry culvert bridges at Banaras Hindu University in India. 70%, 15%, and 15% images of the masonry dataset are used for training, validation and testing, respectively. Number of epochs, batch size and initial learning rate used in the training process are 40, 16, and 0.0001 respectively. The networks are implemented using TensorFlow as the back-end and Keras (Chollet 2015), a Python-based high-level neural network API. The networks are operated on a laptop with an AMD Ryzen7 4800H Octa-Core processor running at 2.90 GHz, 16 GB of RAM, and a 4 GB GEForce GTX 1650Ti GPU.

Adam (Kingma et al. 2015) has been used as an optimizer for the training of networks. Constant learning rate has been maintained during training. Binary Cross Entropy (BCE) and binary focal loss (Lin et al. 2020) are used as loss functions to reduce the false negatives due to the imbalance between cracked pixels and background pixels. The model has been trained with 40 epochs and batch size set to sixteen.

5.3.3. Performance metrics

If predicted and ground-truth semantically segmented cracked images are given. Four components of the confusion matrix (Figure 5.3(d)) can be obtained corresponding to each pixel.

If a cracked pixel is predicted correctly, it is True Positive (TP).

If a non-cracked pixel is predicted correctly, it is True Negative (TN).

If a non-cracked pixel is predicted incorrectly, it is False Positive (FP).

If a cracked pixel is predicted incorrectly, it is False Negative (FN).

Based on the TP, TN, FP, and FN following performance metrics can be evaluated.

For Binary cross entropy (BCE) as loss function

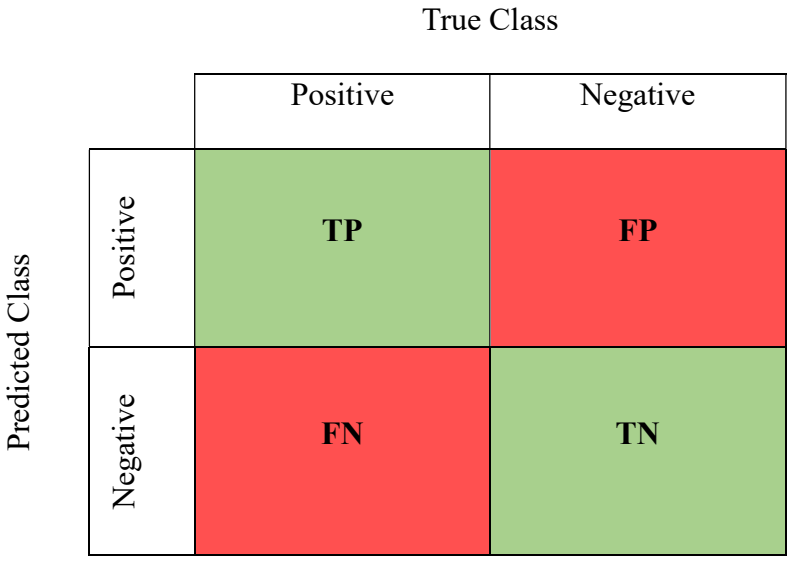


Figure 5.3(d): Confusion matrix for multi-class DL model

$$\text{Loss} = -\text{Y} \times \log\bar{Y} + (1 - \text{Y}) \times \log(1 - \bar{Y}) \quad (5.1)$$

For Binary focal loss as loss function γ

$$\text{Loss} = -\alpha \times (1 - \text{p})^{\gamma} \times \log(\text{p}); \text{ if } \text{Y} = 1 \quad (5.2a)$$

$$\text{Loss} = -(1 - \alpha) \times \text{p}^{\gamma} \times \log(1 - \text{p}); \quad \text{if } \text{Y} = 0 \quad (5.2b)$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \quad (5.3)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.4}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.5}$$

$$\text{Dice coefficient (DC)} = \frac{2 \times \text{TP}}{\text{FP} + \text{FN} + 2 \times \text{TP}} \tag{5.6}$$

$$\text{Intersection over Union (IoU)} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{5.7}$$

$$\text{F1 score} = 2 \times \left(\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}\right) \tag{5.8}$$

where Y is the ground truth which takes values 0 and 1 for non-cracked and cracked pixels, respectively. $\overline{\text{Y}}$ is the prediction value in the range of 0 to 1. The harmonic mean of precision and recall is the F1 score. p is the predicted probability of the positive class, α is a weighting factor balances the contribution of positive and negative examples, and γ is a focusing parameter that down-weights the gift of uncomplicated measures and emphasizes the complex models.

5.4 Discussion of results

The segmentation model DeepLabV3+ has three networks, the segmentation model FPN has eight networks, the segmentation model PSPNet has three networks, and the segmentation model Unet has nine networks. Twenty-three networks have been trained over the masonry dataset; the network nomenclature is presented in Table 5.1.

Table 5.1: Nomenclature of networks

S.No.	Network index	Segmentation Model	Backbone	Loss function	Network name
1	#1a	DeepLabV3+	resnet50	BCE	DeepLabV3+ (#1a)
2	#1b	DeepLabV3+	resnet50	focal loss	DeepLabV3+ (#1b)
3	#1c	DeepLabV3+	vgg19	BCE	DeepLabV3+ (#1c)
4	#2a	FPN	inceptionv3	BCE	FPN (#2a)
5	#2b	FPN	inceptionv3	focal loss	FPN (#2b)
6	#2c	FPN	mobilenet	BCE	FPN (#2c)
7	#2d	FPN	mobilenet	focal loss	FPN (#2d)
8	#2e	FPN	resnet34	BCE	FPN (#2e)
9	#2f	FPN	resnet34	focal loss	FPN (#2f)
10	#2g	FPN	vgg19	BCE	FPN (#2g)
11	#2h	FPN	vgg19	focal loss	FPN (#2h)
12	#3a	PSPNet	mobilenet	BCE	PSPNet (#3a)
13	#3b	PSPNet	resnet34	BCE	PSPNet (#3b)
14	#3c	PSPNet	vgg19	BCE	PSPNet (#3c)
15	#4	Unet	absent	BCE	Unet (#4)
16	#4a	Unet	inceptionv3	BCE	Unet (#4a)
17	#4b	Unet	inceptionv3	focal loss	Unet (#4b)
18	#4c	Unet	mobilenet	BCE	Unet (#4c)
19	#4d	Unet	mobilenet	focal loss	Unet (#4d)
20	#4e	Unet	resnet34	BCE	Unet (#4e)
21	#4f	Unet	resnet34	focal loss	Unet (#4f)
22	#4g	Unet	vgg19	BCE	Unet (#4g)
23	#4h	Unet	vgg19	focal loss	Unet (#4h)

5.4.1 Performance metrics of all networks on training dataset

This section presents the results obtained by analysis of the training dataset of cracked masonry data (Dais et al. 2021) using different pre-trained networks. Training has been done for 40 epochs. Various parameters like the number of parameters, the weight of network, average analysis time per epoch, and performance metrics like loss, accuracy, precision, recall, dice coefficient, Intersection over Union, and F1 score are presented in Table 5. 2. Formulas for metrics presented in Table 5. 2 can be seen from Eqn. 5.1 to Eqn. 5.8. In each case, the results on training and validation datasets for a specific segmentation model with a backbone are poor in the case of a binary focal loss when compared to the same produced using binary cross entropy.

Table 5.2: Various parameters and performance metrics of pre-trained networks

S.No.	Network	Parameters (millions)	Weights (Mb)	Training time(s) per epoch	Loss score	Accuracy	Precision	Recall	Dice coefficient	IoU	F_1 score
1	DeepLabV3+ (#1a)	11.8	139.6	46.0	1.9	98.7	88.2	88.1	**84.7**	**73.5**	**56.2**
2	DeepLabV3+ (#1b)	11.8	139.6	46.0	0.4	98.6	80.8	90.1	58.7	41.7	53.7
3	DeepLabV3+ (#1c)	13.2	299.3	48.0	4.7	97.9	84.5	73.1	**66.2**	**49.7**	**47.8**
4	FPN (#2a)	25.0	294.6	56.0	1.3	98.8	91.6	91.6	**90.0**	**81.9**	**59.3**
5	FPN (#2b)	25.0	294.6	56.0	0.3	98.6	81.9	92.7	71.7	56.1	55.3
6	FPN (#2c)	6.1	71.9	44.0	1.3	98.8	91.9	92.0	**90.4**	**82.6**	**59.6**
7	FPN (#2d)	6.1	71.9	44.0	0.3	98.7	83.0	92.7	75.6	60.9	55.8
8	FPN (#2e)	23.9	281.1	54.0	1.5	98.7	91.3	91.1	89.3	80.8	58.9
9	FPN (#2f)	23.9	281.1	54.0	0.4	98.5	81.8	90.7	63.5	46.5	54.4
10	FPN (#2g)	22.9	268.5	75.0	4.7	97.5	78.9	71.0	66.6	50.0	44.7
11	FPN (#2h)	22.9	268.5	75.0	1.2	97.6	73.1	77.6	40.3	25.4	45.1
12	PSPNet (#3a)	12.9	152.1	240.0	4.8	97.5	80.1	71.9	**68.5**	**52.4**	**45.3**

13	PSPNet (#3b)	2.7	32.5	73.0	1.7	94.6	79.3	69.3	67.0	47.1	43.2
14	PSPNet (#3c)	1.85	21.4	90	2	98.6	89.5	87.3	**85.5**	**74.7**	**56.4**
15	Unet (#4)	8.6	67.7	30.0	12.8	71.4	44.8	8.6	18.1	8.2	8.8
16	Unet (#4a)	29.9	352.0	33.0	1.7	98.7	91.1	90.5	**86.6**	**76.4**	**58.5**
17	Unet (#4b)	29.9	352.0	33.0	0.5	98.5	80.9	89.5	47.4	31.2	53.5
18	Unet (#4c)	8.3	98.1	19.0	1.5	98.8	91.7	91.4	**88.0**	**78.7**	**59.2**
19	Unet (#4d)	8.3	98.1	19.0	0.3	98.7	82.7	92.6	52.7	35.9	55.6
20	Unet (#4e)	24.4	287.3	28.0	2.1	98.6	90.2	87.4	84.2	72.8	56.7
21	Unet (#4f)	24.4	287.3	28.0	0.6	98.3	78.4	88.9	47.8	31.5	52.0
22	Unet (#4g)	29.1	340.9	52.0	2.9	98.3	87.8	83.2	79.3	65.8	53.9
23	Unet (#4h)	29.1	340.9	52.0	1.0	97.9	77.1	80.3	37.9	23.6	48.0

Each model analysed over total 40 epochs. All metrics are in percentage. Metrics given above is obtained on training data at last epoch (40^{th} epoch). Two best-performing networks corresponding to each segmentation model are indicated by bold numbers.

5.4.2 Epoch wise dice coefficient, IoU, and F1 score

The metrics dice coefficient, IoU, and F1 score for all networks are shown in Figures 5.4, 5.5, and 5.6 respectively.

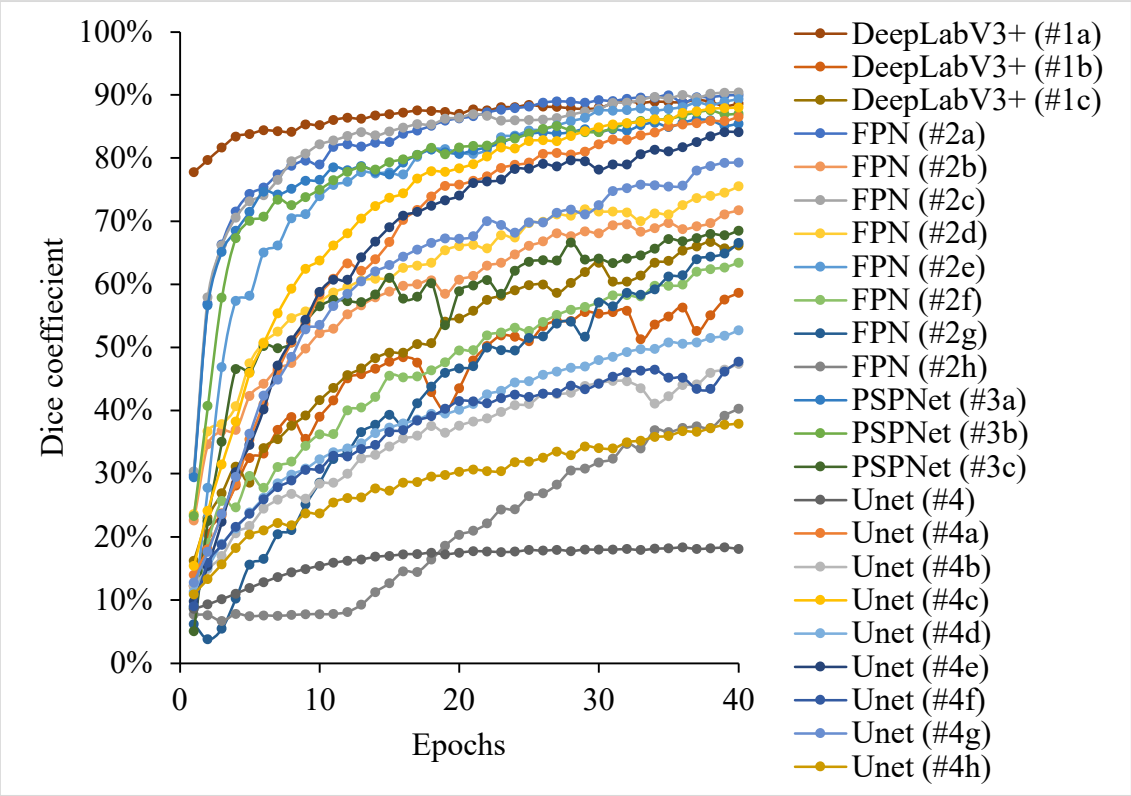


Figure 5.4: Dice coefficient of all networks over training dataset

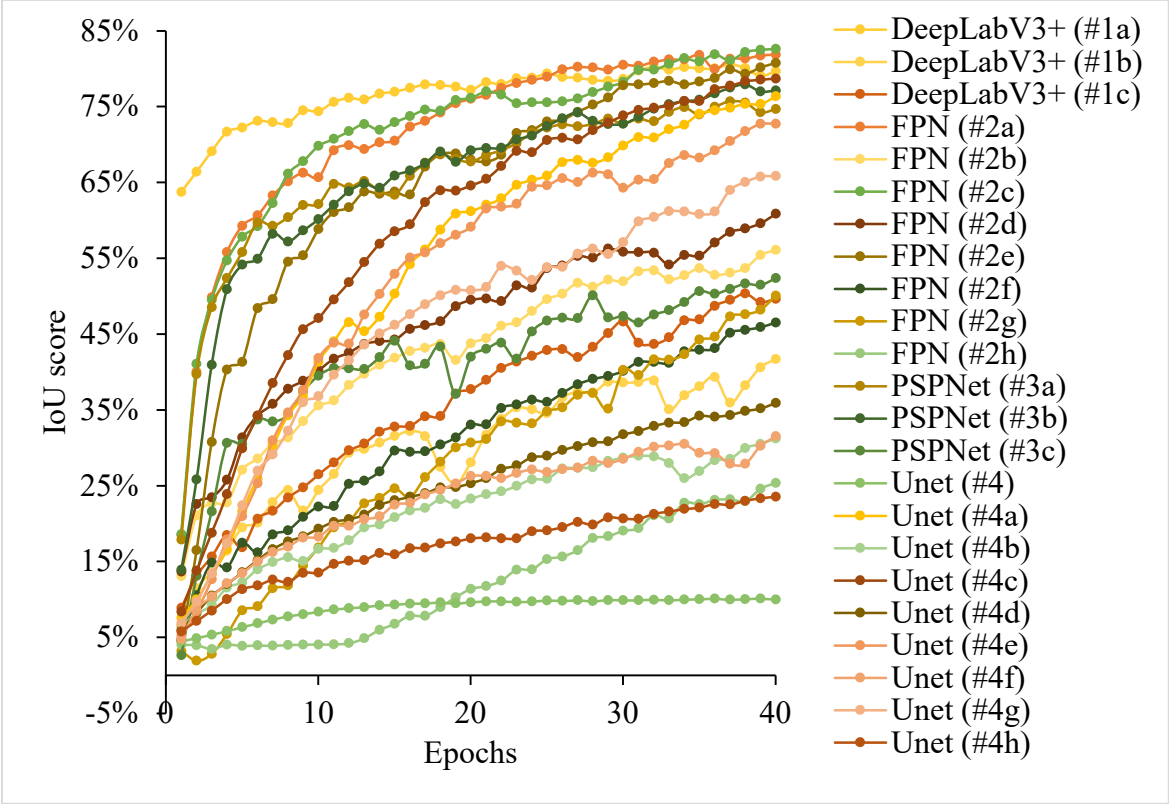


Figure 5.5: IoU score of all networks over training dataset

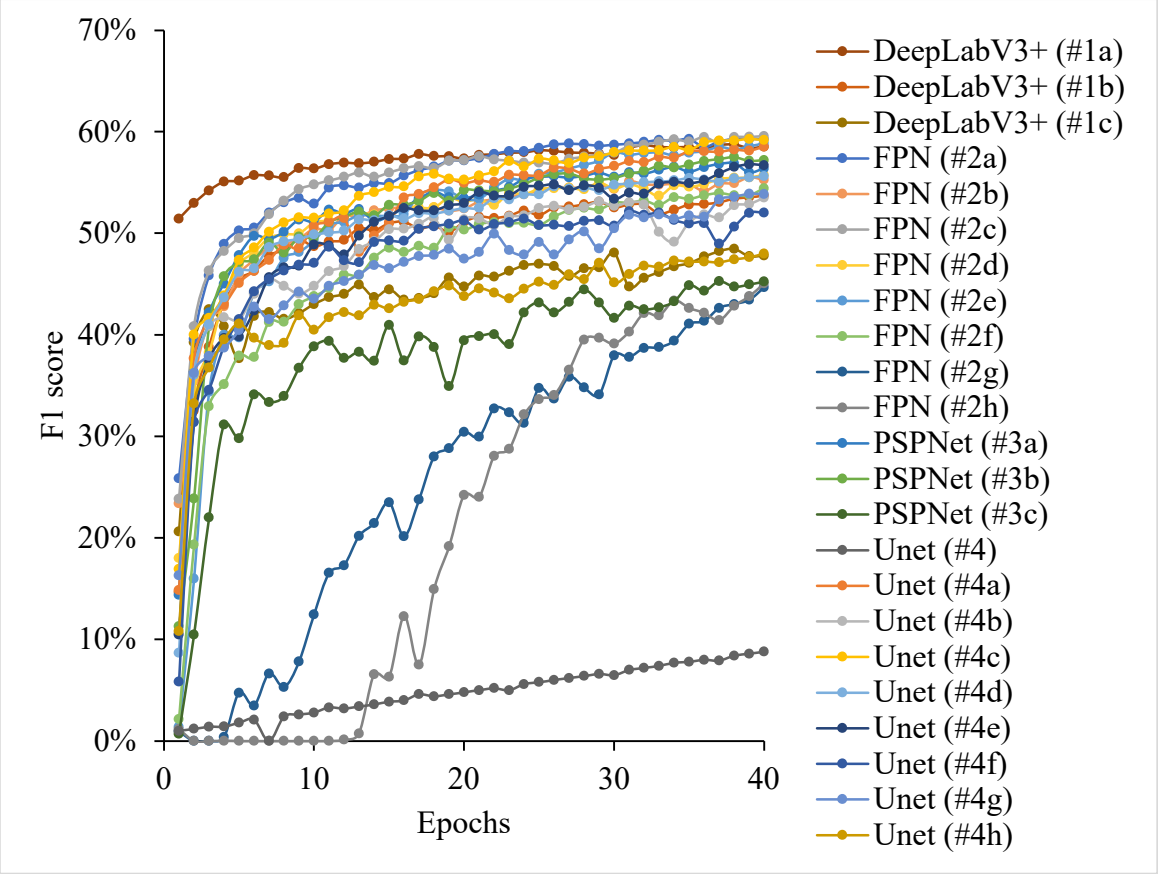


Figure 5.6: F1 score of all networks over training dataset

5.4.3 Segmentation model wise performance metrics

Figure 5.7 and Figure 5.8 illustrate various performance metrics for all networks in the segmentation models FPN and Unet. Figure 5.9 depicts various performance metrics for all networks of both segmentation model DeepLabV3+ as well as PSPNet. Based on performance, each segmentation model's top two networks are selected for further analysis. A total of eight of the twenty-three networks have been further analysed over validation and testing dataset.

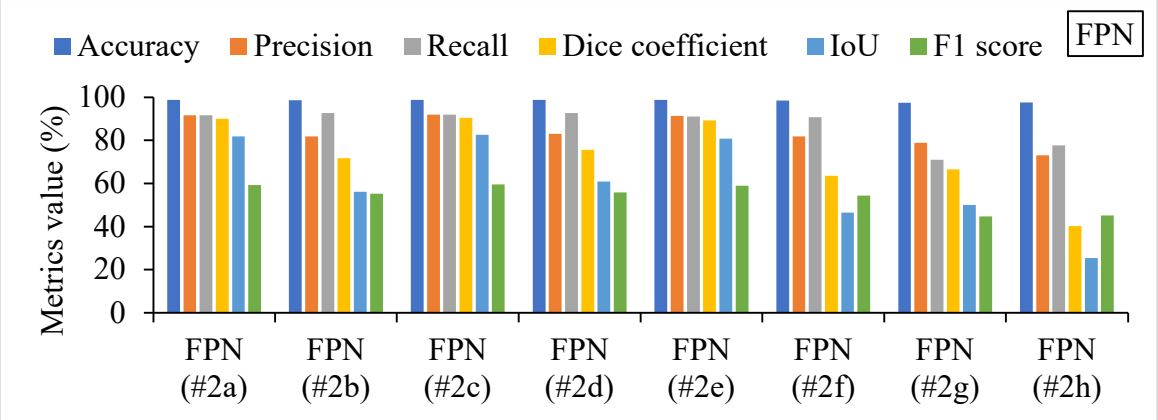


Figure 5.7: Performance metrics of segmentation model FPN for the 40th epoch

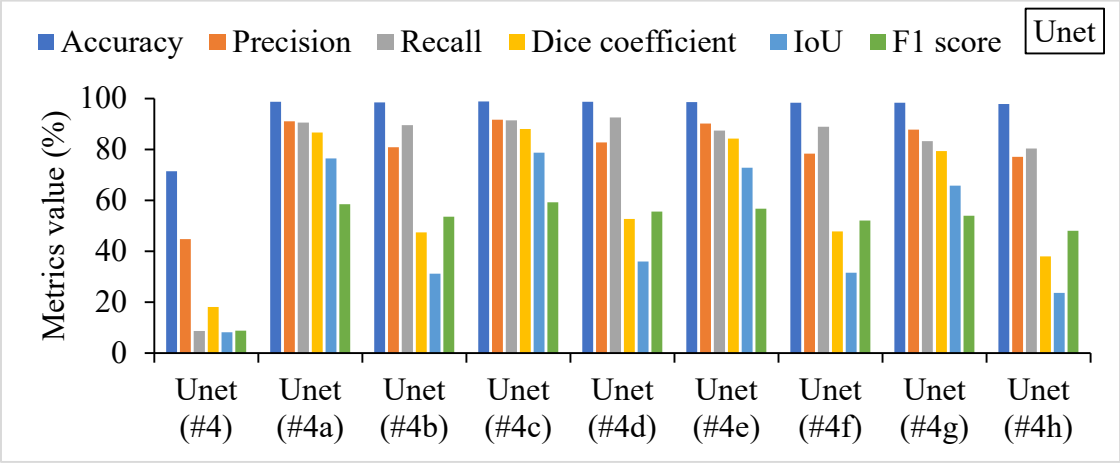


Figure 5.8: Performance metrics of segmentation model Unet for the 40th epoch

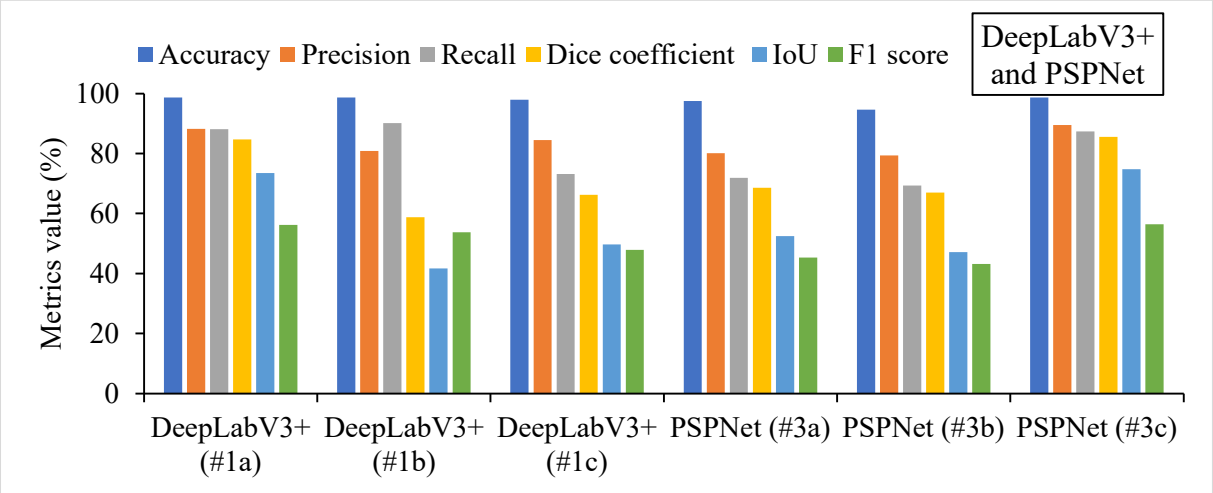


Figure 5.9: Performance metrics of segmentation models DeelabV3+ and PSPNet for the 40th epoch

5.4.4 Training time and validation loss

As can be observed from Table 5.2, the segmentation model without a backbone network, Unet (#4) has the lowest performance of any segmentation model that includes a backbone network. Training time per epoch for all network presented in Table 5.2 is shown in Figure 5.10. Networks are arranged in ascending order of per epoch training time for better clarity.

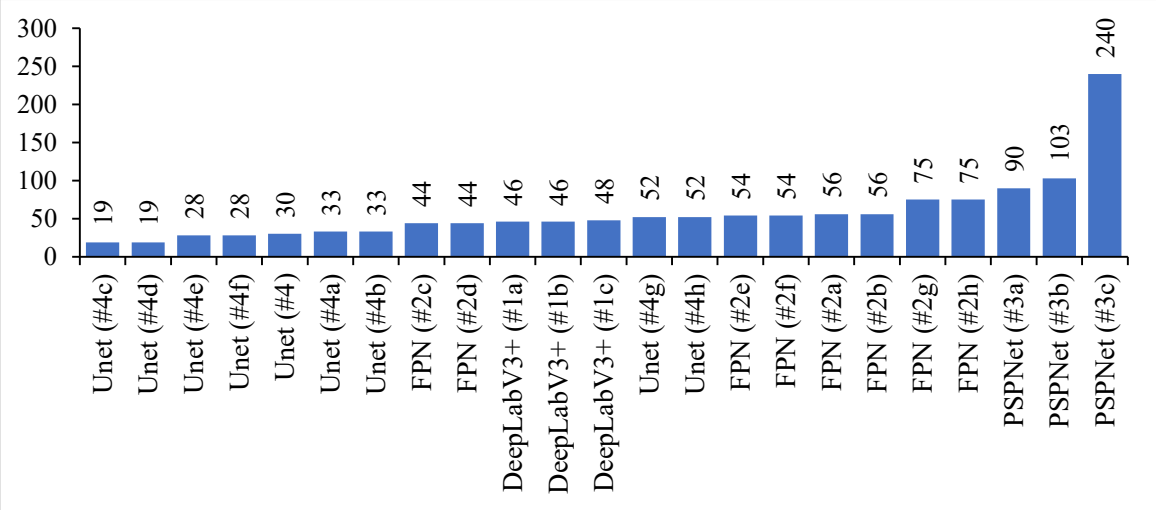


Figure 5.10: Training time (in seconds) per epoch for all networks.

As we can see from Figure 5.10, there is no change in training time of network on changing loss function. Training time is used as a measure of network efficiency. On the basis of average training time per epoch, the most and least efficient networks are Unet (#4c) and PSPNet (#3c), respectively. The loss score is calculated on the validation dataset at every epoch. The loss score on eight selected networks is shown in Figure 5.11.

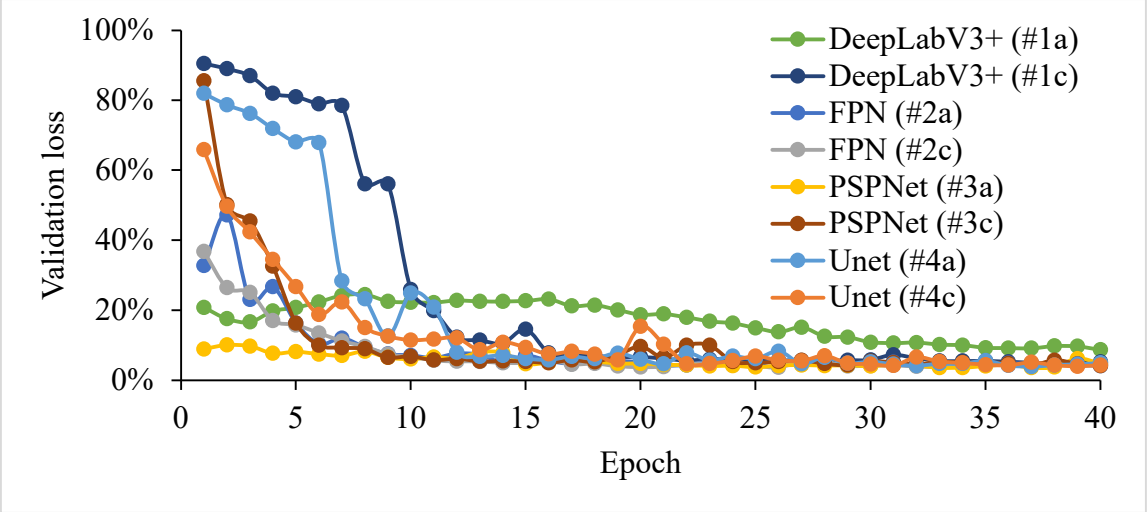


Figure 5.11: Loss score by selected networks on validation dataset.

5.4.5 Performance metrics of selected networks on training, validation, and testing dataset

Each segmentation model's top two best-performing networks and their performance metrics are presented separately in Table 5.3 for the training dataset. Bar chart of performance metrics on training dataset for eight selected networks is shown in Figure 5.12.

Table 5.3: Performance metrics evaluated on training dataset

S.No.	Network	Loss score	Accuracy	Precision	Recall	Dice coefficient	IoU	F_1 score
1	DeepLabV3+ (#1a)	1.9	98.7	88.2	88.1	84.7	73.5	56.2
2	DeepLabV3+ (#1c)	4.7	97.9	84.5	73.1	66.2	49.7	47.8
3	FPN (#2a)	1.3	98.8	91.6	91.6	90.0	81.9	59.3
4	FPN (#2c)	1.3	98.8	91.9	92.0	90.4	82.6	59.6
5	PSPNet (#3a)	2.0	98.6	89.5	87.3	85.5	74.7	56.4
6	PSPNet (#3c)	4.8	97.5	80.1	71.9	68.5	52.4	45.3
7	Unet (#4a)	1.7	98.7	91.1	90.5	86.6	76.4	58.5
8	Unet (#4c)	1.5	98.8	91.7	91.4	88.0	78.7	59.2

Each network analysed over total 40 epochs. All metrics are in percentage. Metrics given above are belong to last epoch (40^{th} epoch).

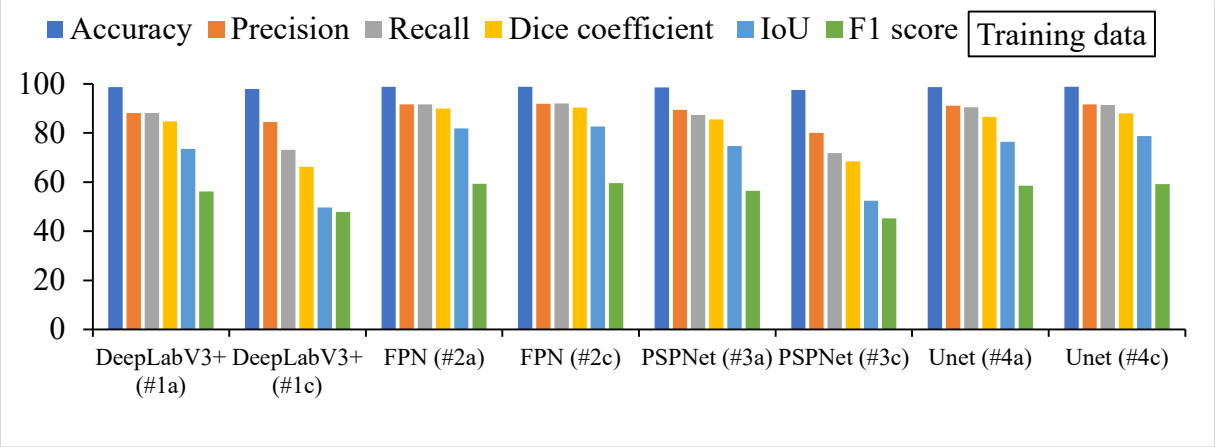


Figure 5.12: Bar chart of performance metrics on training dataset for 40th epoch.

Table 5.4 represents performance metrics evaluated on validation dataset for eight selected networks. Bar chart of performance metrics on validation dataset for eight selected networks is shown in Figure 5.13.

Table 5.4: Performance metrics evaluated on validation dataset

S.No.	Network	Loss score	Accuracy	Precision	Recall	Dice coefficient	IoU	F_1 score
1	DeepLabV3+ (#1a)	4.8	98.2	92.0	68.3	78.1	62.4	51.6
2	DeepLabV3+ (#1c)	5.2	97.7	90.0	62.4	68.2	52.3	44.3
3	FPN (#2a)	4.1	98.2	88.1	73.4	78.5	64.7	49.5
4	FPN (#2c)	4.6	98.2	90.3	71.3	78.4	64.7	49.2
5	PSPNet (#3a)	4.9	98	91.5	62.1	72.3	56.9	44.8
6	PSPNet (#3c)	4.1	98.0	85.9	61.1	66.7	50.3	43.1
7	Unet (#4a)	4.3	98.1	84.9	72.6	74.7	59.9	47.9
8	Unet (#4c)	5.0	98.0	90.8	64.3	71.9	56.3	45.8

Each network analysed over total 40 epochs. All metrics are in percentage. Metrics given above are belong to last epoch (40th epoch).

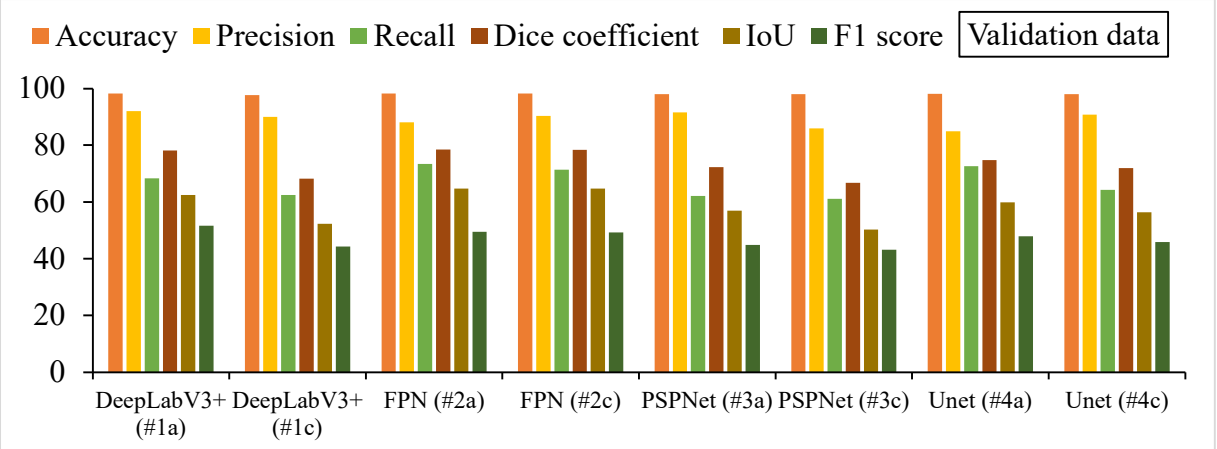


Figure 5.13: Bar chart of performance metrics on validation dataset for 40th epoch.

Table 5.5 shows performance metrics evaluated on testing dataset on eight selected networks on the testing dataset. Bar chart of performance metrics on testing dataset for eight selected networks is shown in Figure 5.14. Since the BCE loss function gives better results than the focal loss function, only BCE function-based model is used in testing, and their capability is checked.

Table 5.5: Performance metrics evaluated on testing dataset

S.No.	Network	Accuracy	Precision	Recall	Dice coefficient	IoU	F_1 score
1	DeepLabV3+ (#1a)	98.5	92.2	73.1	83.1	72	54.4
2	DeepLabV3+ (#1c)	97.2	96.8	48.0	59.3	42.9	40.1
3	FPN (#2a)	98.5	90.6	87.1	86.9	74.9	59.3
4	FPN (#2c)	98.2	92.4	83.9	85.6	75.4	56.3
5	PSPNet (#3a)	98.2	93.6	44.8	65.7	47.6	38.7
6	PSPNet (#3c)	97.6	81.8	53.8	58.2	41.6	38.6
7	Unet (#4a)	98.4	89.4	81.9	80.1	67.4	47.9
8	Unet (#4c)	98.4	94.5	72.8	75.6	61.4	50.1

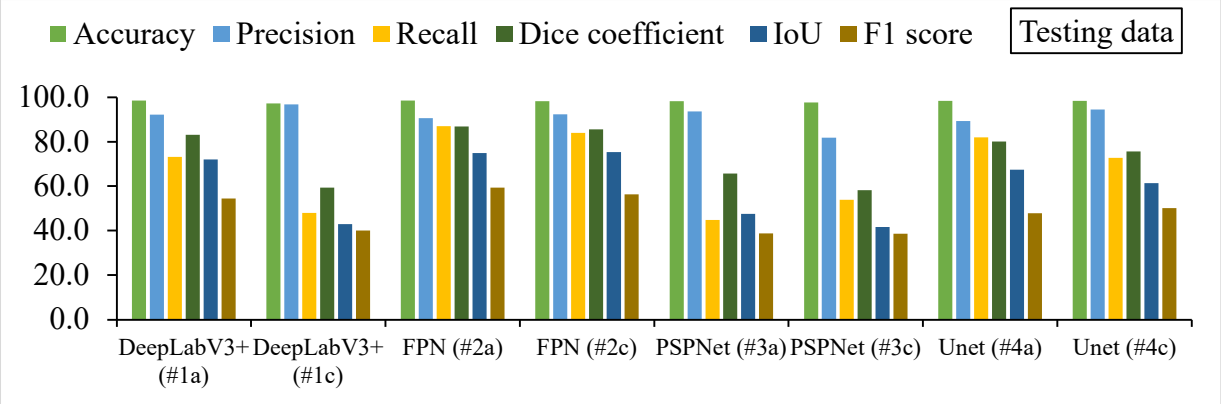
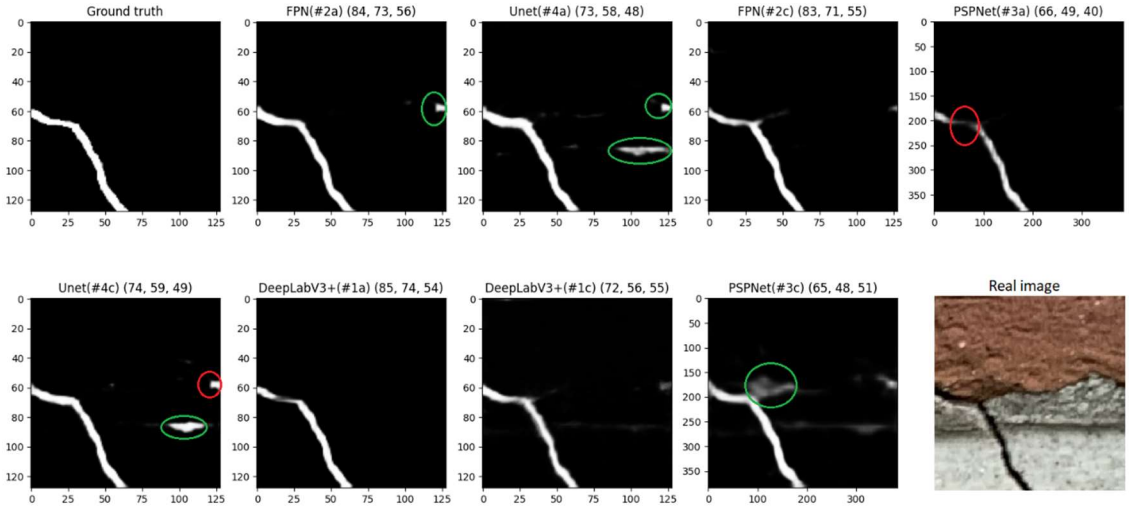


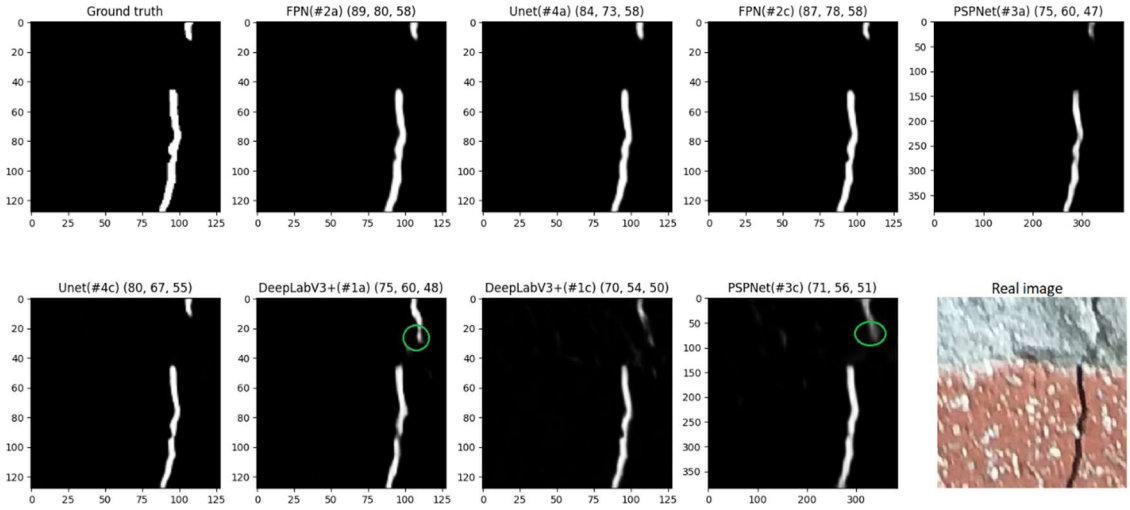
Figure 5.14: Bar chart of performance metrics on testing dataset.

5.4.6 Semantically segmented images

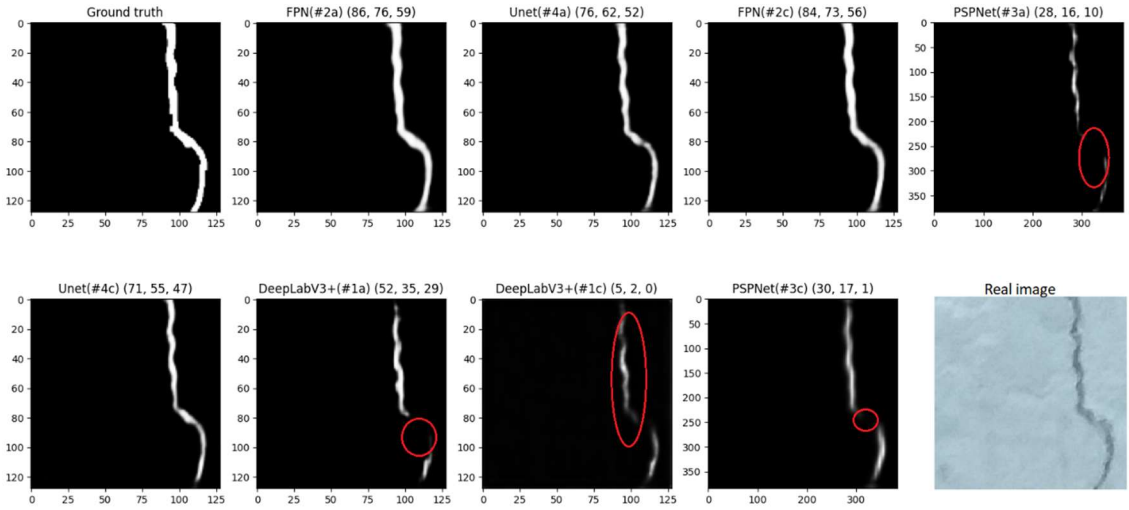
Semantic segmentation of cracked masonry surfaces is shown in Figure 5.15. At the top, each segmented image is labelled with network name followed by three metrics: dice coefficient, IoU, and F1 score. All performance metrics are in percentage. BCE loss function has been applied in each network shown in Fig 15. A total of eight images are shown in Figure 5.15 and each image is predicted by eight networks, namely FPN(#2a), Unet(#4a), FPN(#2c), PSPNet(#3a), Unet(#4c), DeepLabV3+(#1a), DeepLabV3+(#1c), PSPNet(#3c) respectively to demonstrate the network performance. Green and red circles indicate false positive (FP) and false negative (FN) detections, respectively, in the predicted images shown in Figure 5.15(a) to 15(h).


Ground truth
FPN(#2a) (84, 73, 56)
Unet(#4a) (73, 58, 48)
FPN(#2c) (83, 71, 55)
PSPNet(#3a) (66, 49, 40)
Unet(#4c) (74, 59, 49)
DeepLabV3+(#1a) (85, 74, 54)
DeepLabV3+(#1c) (72, 56, 55)
PSPNet(#3c) (65, 48, 51)
Real image

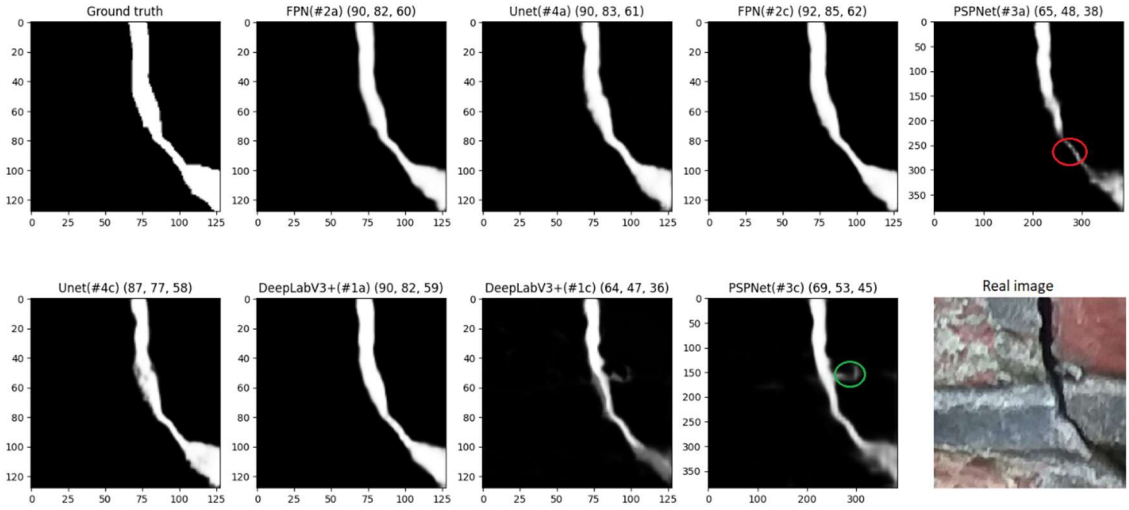
(a)


Ground truth
FPN(#2a) (89, 80, 58)
Unet(#4a) (84, 73, 58)
FPN(#2c) (87, 78, 58)
PSPNet(#3a) (75, 60, 47)
Unet(#4c) (80, 67, 55)
DeepLabV3+(#1a) (75, 60, 48)
DeepLabV3+(#1c) (70, 54, 50)
PSPNet(#3c) (71, 56, 51)
Real image

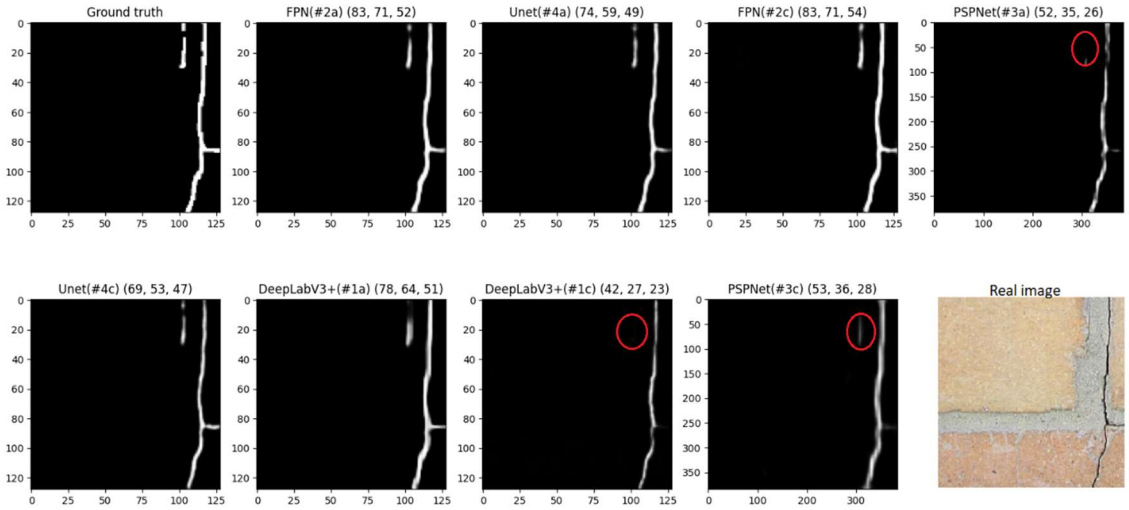
(b)


Ground truth
FPN(#2a) (86, 76, 59)
Unet(#4a) (76, 62, 52)
FPN(#2c) (84, 73, 56)
PSPNet(#3a) (28, 16, 10)
Unet(#4c) (71, 55, 47)
DeepLabV3+(#1a) (52, 35, 29)
DeepLabV3+(#1c) (5, 2, 0)
PSPNet(#3c) (30, 17, 1)
Real image

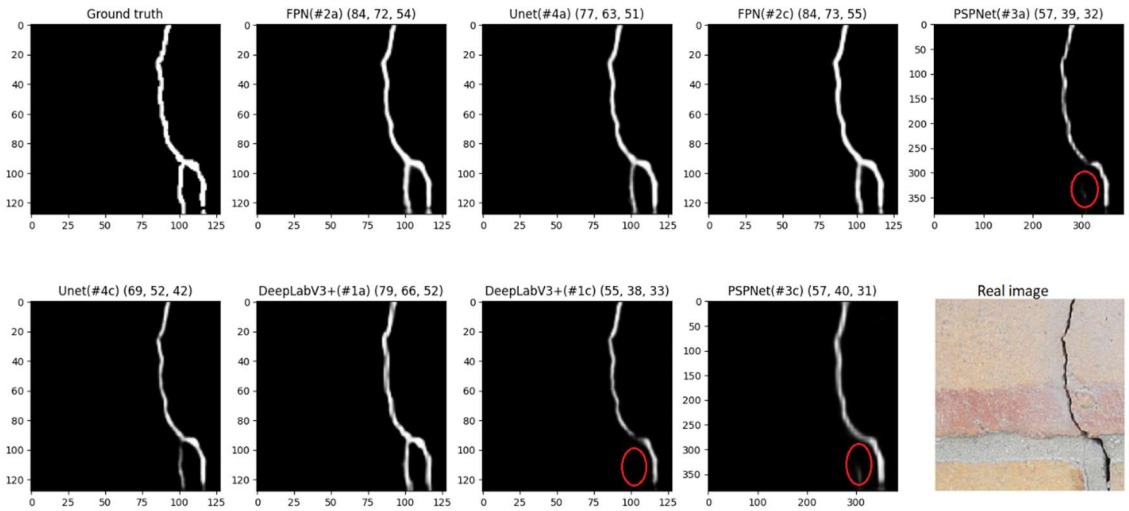
(c)


Ground truth
FPN(#2a) (90, 82, 60)
Unet(#4a) (90, 83, 61)
FPN(#2c) (92, 85, 62)
PSPNet(#3a) (65, 48, 38)
Unet(#4c) (87, 77, 58)
DeepLabV3+(#1a) (90, 82, 59)
DeepLabV3+(#1c) (64, 47, 36)
PSPNet(#3c) (69, 53, 45)
Real image

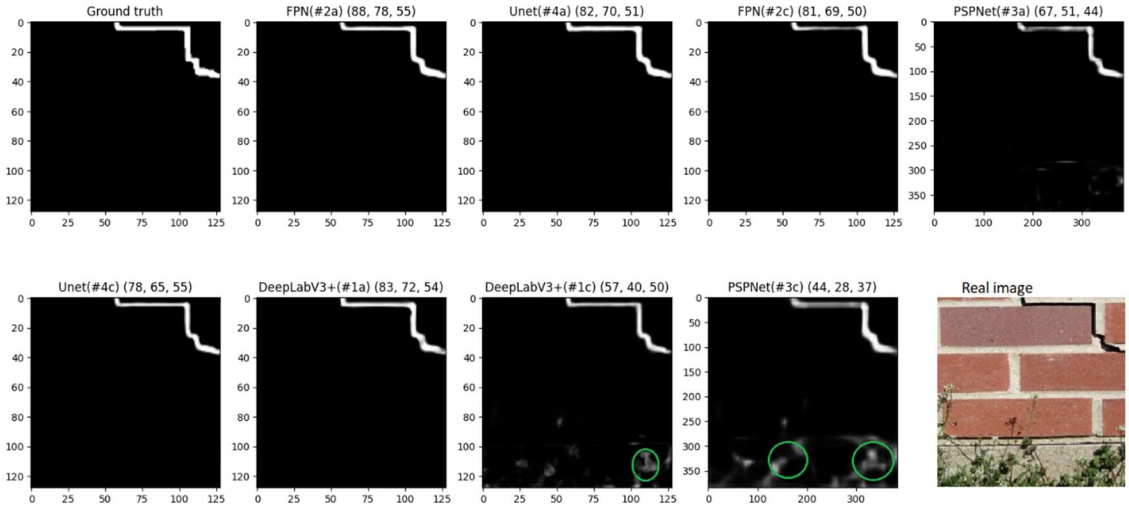
(d)


Ground truth
FPN(#2a) (83, 71, 52)
Unet(#4a) (74, 59, 49)
FPN(#2c) (83, 71, 54)
PSPNet(#3a) (52, 35, 26)
Unet(#4c) (69, 53, 47)
DeepLabV3+(#1a) (78, 64, 51)
DeepLabV3+(#1c) (42, 27, 23)
PSPNet(#3c) (53, 36, 28)
Real image

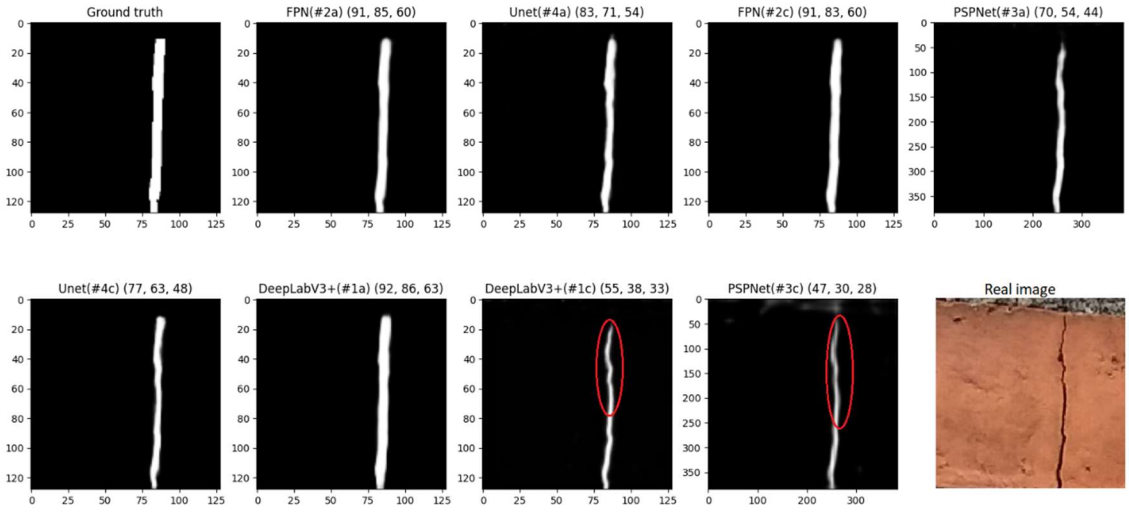
(e)


Ground truth
FPN(#2a) (84, 72, 54)
Unet(#4a) (77, 63, 51)
FPN(#2c) (84, 73, 55)
PSPNet(#3a) (57, 39, 32)
Unet(#4c) (69, 52, 42)
DeepLabV3+(#1a) (79, 66, 52)
DeepLabV3+(#1c) (55, 38, 33)
PSPNet(#3c) (57, 40, 31)
Real image

(f)



(g)



(h)

Figure 5.15: Semantic segmentation of cracked masonry surfaces labelled with network name followed by three metrics(dice coefficient, IoU,F1 score).

(a) image with shadow (b) uneven surface with small crack (c) low contrast image (d) uneven surface with wider crack (e) surface with fine crack-1 (f) surface with fine crack-2 (g) surface with vegetation (h) cracked brick surface

Challenging examples are taken to demonstrate the performance of networks. Image with shadow Figure 5.15(a), uneven surface Figure 5.15(b) and 15(d), low intensity contrast Figure 5.15(c), very fine cracks Figure 5.15(e) and Figure 5.15(f) and natural turbulence-like vegetation Figure 5.15(g) are taken.

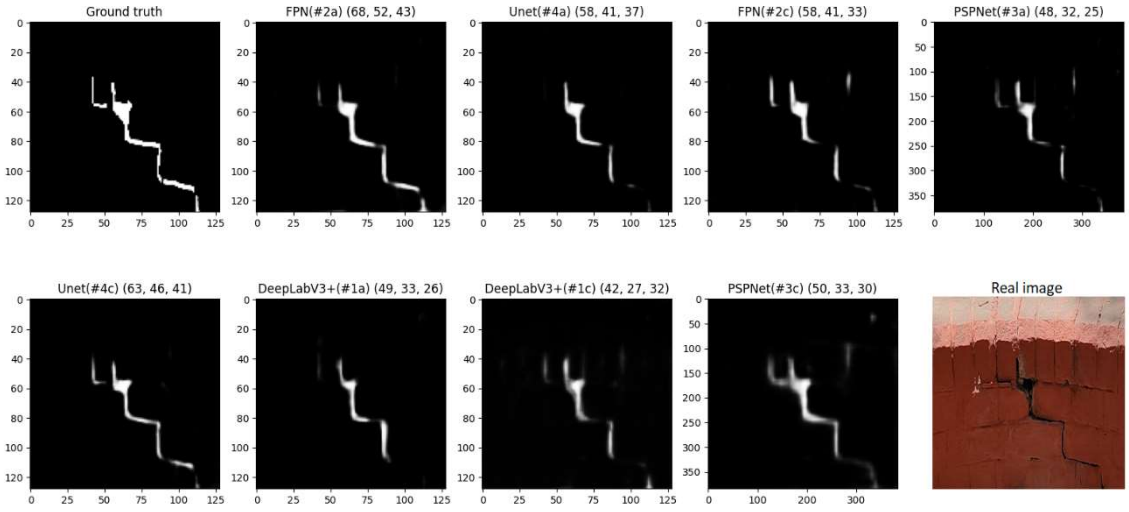
In Fig 15(a), some networks show false positive at the shadow part of the image, and only PSPNet(#3a) shows false negative detection. In Fig 15(b), false positives are detected by DeepLabV3+(#1a) and PSPNet(#3c) at damaged mortar part due to uneven surface. False negative detections are shown at many pixels by DeepLabV3+(#1c), PSPNet(#3a), DeepLabV3+(#1a) and PSPNet(#3c) in Figure 5.15(c) due to low-intensity contrast adjacent to the cracked portion. In Fig 15(d), false negative and false positive detections are shown by PSPNet(#3a) and PSPNet(#3c), respectively. In both Figure 5.15(e) and Figure 5.15(f), false negative pixels are shown at finer crack locations by PSPNet(#3a), DeepLabV3+(#1a), and PSPNet(#3c). False positive pixels are detected at the vegetation part of the image Figure 5.15(g) by DeepLabV3+(#1c) and PSPNet(#3c). DeepLabV3+(#1c) and PSPNet(#3c) detect false negative pixels in Fig 15(h). The top three best-performing models on the images with real-world complexities are FPN(#2a), FPN(#2c), and DeepLabV3+(#1a), respectively.

Depending on the backbone employed, the performance of the models varies greatly, with some backbones yielding superior outcomes to others. For instance, the mobilenet backbone outperformed the resnet34 backbone in the FPN architecture, and the same is true for the U-Net and PSPNet architectures. Similar results are seen in the Deeplabv3+ architecture, where the resnet50 backbone outperformed the vgg19 backbone. The top three networks based on three performance metrics (dice coefficient, IoU, F1 score) out of six metrics given in Table 5. 5 are FPN(#2a) (86.9%, 74.9%, 59.3%), FPN(#2c) (85.6%, 75.4%, 56.3%), DeepLabV3+(#1a) (83.1%, 72%, 54.4%) respectively. The results of this study show that three networks FPN(#2a), FPN(#2c), and DeepLabV3+(#1a) can also accurately detect finer surface cracks on masonry surfaces.

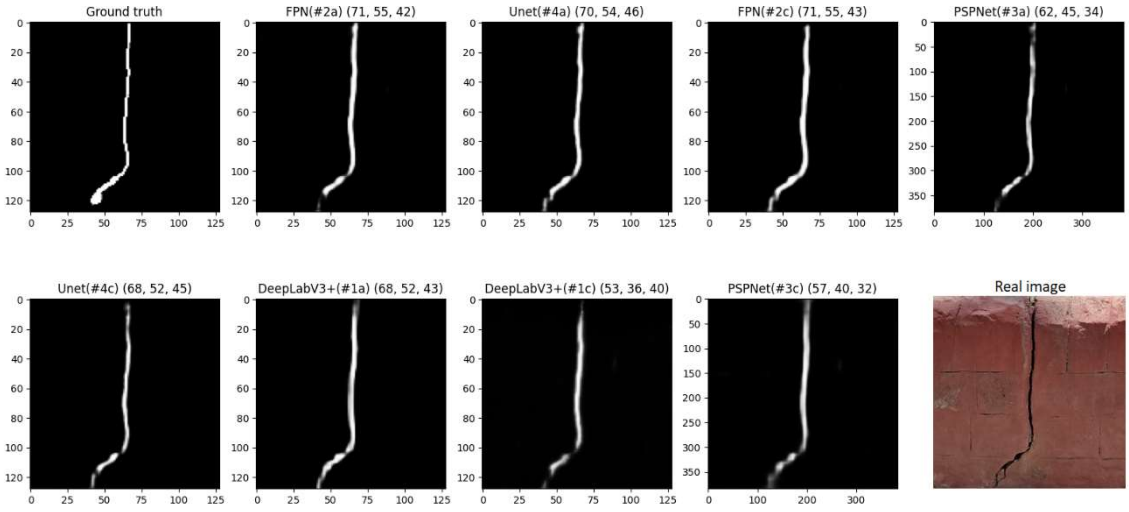
5.5 Case study

A case study on existing small masonry culvert bridges is also conducted to evaluate the trained network's performance in reality. Networks are tested using pictures of existing small masonry culvert bridges at Banaras Hindu University, India. A full-size image of bridge culvert is shown in Figure 5.16, cracked part of culvert is highlighted by rectangular box and segmentation of cracked portion is shown in Figure 5.17(a). Figures 17(a) through 17(e) exhibit segmented images with performance metrics (dice coefficient, IoU, and F1 score). However, some of the pictures had shadows due to the acute angle with the masonry surface when the image was taken with an oneplus8 smartphone. The three networks namely FPN(#2a), FPN(#2c), and DeepLabV3+(#1a) outperform among eight selected networks on photos with existing complications like shadow, vegetation, etc. As trained networks are also tested on existing masonry culverts and their performance are examined. This study can significantly aid the detection of cracks in the masonry substructure of old railway bridges.

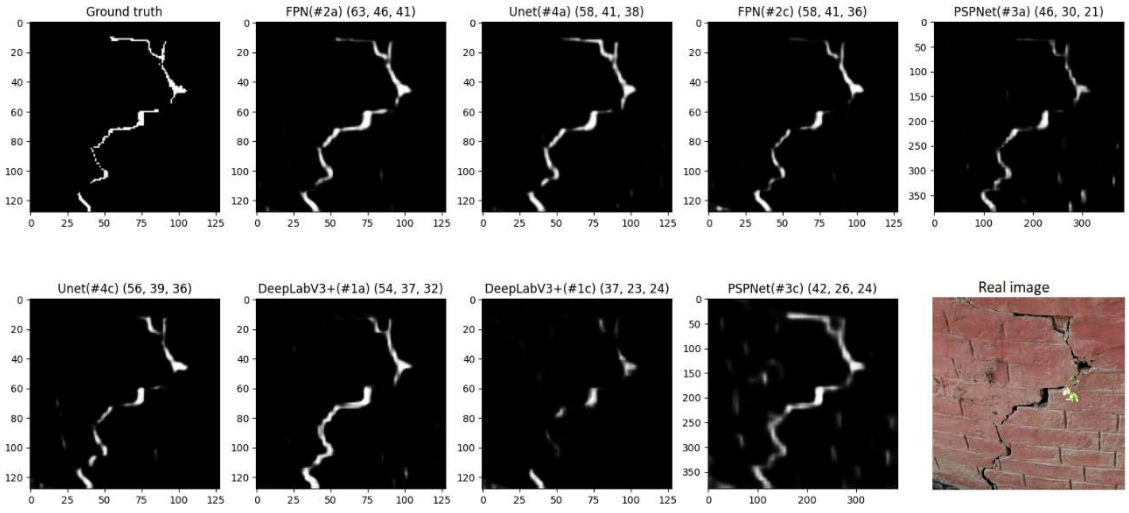
Figure 5.16: Existing old masonry culvert bridge at Banaras Hindu University, India


Ground truth
FPN(#2a) (68, 52, 43)
Unet(#4a) (58, 41, 37)
FPN(#2c) (58, 41, 33)
PSPNet(#3a) (48, 32, 25)
Unet(#4c) (63, 46, 41)
DeepLabV3+(#1a) (49, 33, 26)
DeepLabV3+(#1c) (42, 27, 32)
PSPNet(#3c) (50, 33, 30)
Real image

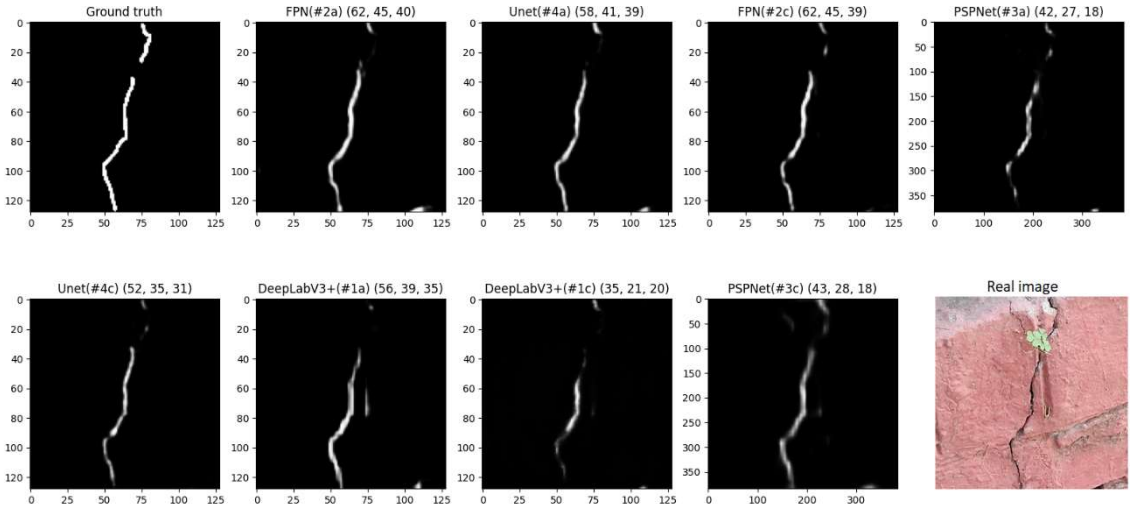
(a)


Ground truth
FPN(#2a) (71, 55, 42)
Unet(#4a) (70, 54, 46)
FPN(#2c) (71, 55, 43)
PSPNet(#3a) (62, 45, 34)
Unet(#4c) (68, 52, 45)
DeepLabV3+(#1a) (68, 52, 43)
DeepLabV3+(#1c) (53, 36, 40)
PSPNet(#3c) (57, 40, 32)
Real image

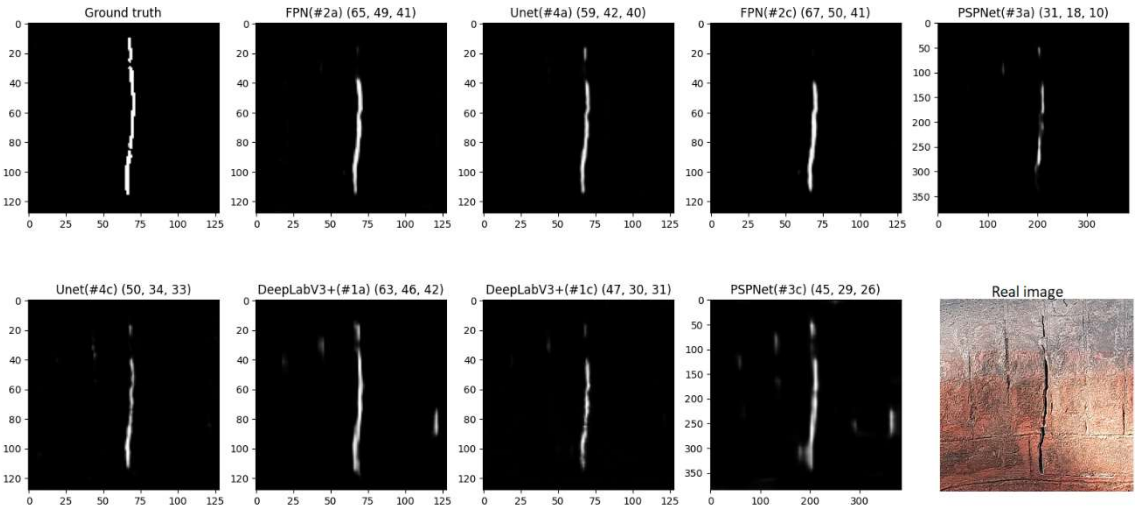
(b)


Ground truth
FPN(#2a) (63, 46, 41)
Unet(#4a) (58, 41, 38)
FPN(#2c) (58, 41, 36)
PSPNet(#3a) (46, 30, 21)
Unet(#4c) (56, 39, 36)
DeepLabV3+(#1a) (54, 37, 32)
DeepLabV3+(#1c) (37, 23, 24)
PSPNet(#3c) (42, 26, 24)
Real image

(c)



(d)



(e)

Figure 5.17: Semantic segmentation of existing cracked masonry culvert bridges labelled with network name followed by three metrics(dice coefficient, IoU,F1 score).

(a) image with shadow (b) surface with uniform crack (c) map pattern cracked surface (d) surface with vegetation (e) surface with an unclear crack pattern

5.6 Concluding remarks

5.6.1 Study summary

Based on the considered dataset and performance metrics used in the present study, findings from the study are summarized as follows:

- Two best performing networks for segmentation model Deeplabv3+, FPN, PSPNet, and Unet are Deeplabv3+(#1a) & Deeplabv3+(#1c), FPN(#2a) & FPN(#2c), PSPNet(#3a) & PSPNet(#3c), and Unet(#4a) & Unet(#4c) respectively.
- The accuracy of all networks except Unet(#4) used in the study is more than 95 percent in the training, validation and testing, which indicates that networks precisely segment cracks from masonry data.
- Based on average training time per epoch, the most and least efficient networks are Unet(#4c) and PSPNet(#3c).
- Backbone resnet34 performs poorly with each segmentation network compared to other backbone networks for binary focal loss and binary cross-entropy loss function.
- On the basis of performance metrics (dice coefficient, IoU, and F1 score), the best three networks are FPN(#2a) (86.9%, 74.9%, 59.3%), FPN(#2c) (85.6%, 75.4%, 56.3%), DeepLabV3+(#1a) (83.1%, 72%, 54.4%) respectively.
- In each case, the results on training and validation datasets for a specific segmentation model with a backbone are poor in the case of a binary focal loss when compared to the same produced using binary cross entropy.
- The segmentation model without a backbone network has the lowest performance of any segmentation model with a backbone network.

5.6.2 Conclusion

Deep learning (DL)-based semantic segmentation models has successfully been applied in semantic segmentation of cracks over masonry surfaces. The performance of various deep learning architectures with different backbone networks has been investigated to segment cracks. The performance of trained networks has also been checked over existing small masonry culvert bridges. Result shows that trained networks can detect finer cracks in different environmental conditions. This study presents deep learning-based effective techniques for semantic segmentation of cracks over masonry surfaces. It will be valuable in crack detection over masonry surfaces like masonry buildings, masonry boundary walls, old masonry railway bridges, etc.

Since trained networks semantically segmented even finer cracks in different environmental conditions over existing small masonry culvert bridges, presented techniques can also be used in crack detection over masonry surfaces in old railway bridges. In addition to reducing the time and effort previously required for the inspectors to inspect and document masonry structures, the proposed method also minimizes human error.