Engineering and Technology Journal e-ISSN: 2456-3358
Volume 08 Issue 07 July-2023, Page No.- 2397-2406
DOI: 10.47191/etj/v8i7.03, I.F. – 7.136

Machine Learning Unet-Based Segmentation of Sentinel-1A Satellite Images

U. I. Nduanya[1], K. A. Akpado[2], S. N. Ogili[3]
[1,2]Nnamdi Azikiwe University, Awka, Nigeria
[3]Enugu University of Science and Technology, Enugu, Nigeria

ABSTRACT: Image segmentation is an area of study which has generated a lot of interest in engineering in the last few years. Segmentation of Sentinel 1A satellite images is an area that is not as common as that of its optical counterparts as a result of the appearance of its images. The focus of this paper is the segmentation of Sentinel-1A Synthetic Aperture Radar (SAR) satellite images using UNet, a Deep learning technique designed specifically for semantic segmentation of images. The model was developed with Python programming language in keras, which is a deep learning Python API running on the Tensorflow framework. The satellite images were acquired from the European Space Agency's (ESA) Copernicus satellite acquisition hub. The performance evaluation was carried out with Jaccard Index or Intesection over Union (IoU) and Accuracy metrics. The IoU metric measures the number of pixels common between the target and prediction masks divided by the total number of pixels present across both masks while the Accuracy metric reports the percent of pixels in the image which were correctly classified. The results recorded in the evaluation had IoU as 0.42 and an Accuracy of 95%. This show a high performance of the model in segmenting images.

KEYWORDS: Synthetic Aperture Radar, UNet, Semantic Segmentation, Sentinel-1A, Copernicus Satellite.

1.0 INTRODUCTION

Image segmentation is the detection and classification of individual objects within an image. It determines the outlines of objects inside an image at the pixel level. This segmentation is often achieved by the application of a machine learning technique, Deep Learning, on images. Deep learning has become a very well used machine learning tool in every field in the past three decades. As computational power advanced, there was progress in the area of Artificial Neural Networks (ANN) and the attempt to increase the number of hidden neurons was successful. This successful development of multiple hidden layers improved prediction and thus, brought about Deep Learning. Deep Learning methods are a set of methods that allows machines to automatically figure out the representations required for detection or classification by using its multiple levels of layers to extract higher level features from the raw input (LeCun et al., 2015). DL refers to the development and application of Deep Neural Networks. As a result of the great success in the increase of hidden layers in ANN and the birth of Deep Neural Networks, there has been an explosion of different variants of Deep Neural Network and UNet is one of them.

The UNet architecture is a revolution in the area of computer vision. It is a Convolutional Neural Network that was developed for biomedical image segmentation but which performed so well, that it has been applied in every field imaginable, in Engineering: self-driving cars, agriculture, to hydrology, etc. The U-Net architecture employs an encoder-decoder cascade structure. The encoder is the first half of the architecture while the decoder is the second half. These two parts are also referred to as the contracting path and the expansive path. The UNet model derives its name from the shape of its architecture. It appears similar to the letter U. In CNN, the image is transformed into a vector which is frequently utilized in classification problems but in UNet an image is first transformed into a vector before being transformed back into an image using the same mapping. This reduces the distortion by preserving the original structure of the image. The network was built on a Fully Convolutional Network (FCN) architecture (Ronneberger et al., 2015). This FCN architecture was modified and extended to work with fewer training images to yield more precise segmentations. A 512 x 512 image segmentation takes less than a second on a modern GPU.

The application of UNet in image segmentation of SAR satellite images is fairly recent. Synthetic Aperture Radar is a radar technology for high resolution ground imaging and moving target identification for use on both manned and unmanned aerial platforms. The Sentinel-1A SAR satellite's major objective is to monitor the land and the sea. Using radar imaging, the SAR satellite can acquire images in all weather situations and can operate day and night in a 2-pole orbit (Copernicus, 2022).

SAR satellite image segmentation is being applied more in the field of engineering as a result of its numerous advantages in technology especially in remote sensing applications. Remote sensing technologies use various recording tools to collect data on objects and infrastructure on the Earth's surface without making direct contact (Shah, Seker, Hameed,

and Draheim, 2019). It has been applied in simple visual interpretation (Sanyal & Lu, 2004), improvements in forecasting (Hostache et al., 2018), image change detection technology (Clement et al., 2018), road detection (Yang et al., 2019), image processing (Hordiiuk et al., 2019), land cover mapping (Benbahria et al., 2019) and image despeckling (Lattari et al., 2019). It has also been applied to flood detection, Kang et al. (2018) developed a Flood Detection model for SAR Images via Fully Convolutional Network. Nemni et al. (2020) designed a Fully Convolutional Neural Network for Rapid Flood Segmentation in Synthetic Aperture Radar Imagery; flood mapping: Li et al. (2018) developed a two-step automatic change detection chain for quick flood mapping based on Sentinel-1 SAR images and flood prediction: Garcia-Pintado et al. (2015) presented a Satellite-supported flood forecasting model in river networks. The model assimilated satellite-based water level observations. SAR's application in remote sensing is as a result of its cloud-penetrating and day-and-night operational capabilities.

The segmentation of Sentinel-1A SAR satellite images with UNet involves successive compression of data into a lower-dimensional representation by the contracting path and the conversion of this data back to the original image dimension by the expansive path. In recent research works such as (Jaisakthi et al., 2021; Zhang & Xia, 2021; Hernandez et al., 2021; Katiyar et al., 2020) the detection of flooded areas in Sentinel-1A SAR satellite images was achieved with UNet but it was trained with few images (small dataset) thereby reducing the efficiency of the model as it performed poorly when tested on different satellite images.

2.0 METHODOLOGY

2.1. Image Acquisition

The images were downloaded from the Copernicus open access hub. The images acquired and processed in this study are S1A data corresponding to Lokoja close to the lower basin of the Niger River; data from the upper part of the Niger up to Tungan papa, Kebbi and data from Benue river up to Jalingo, Taraba, Nigeria.

Timelapses of these parts of the two rivers were created in the hub and broken up into frames to form the dataset.

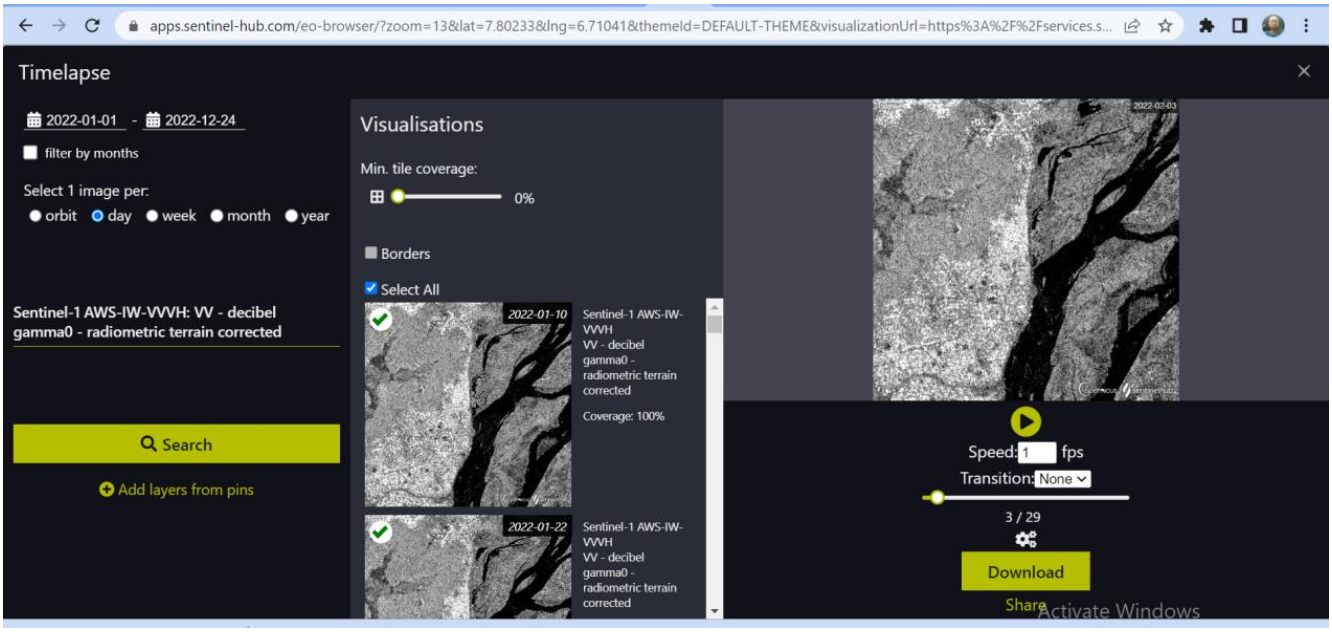
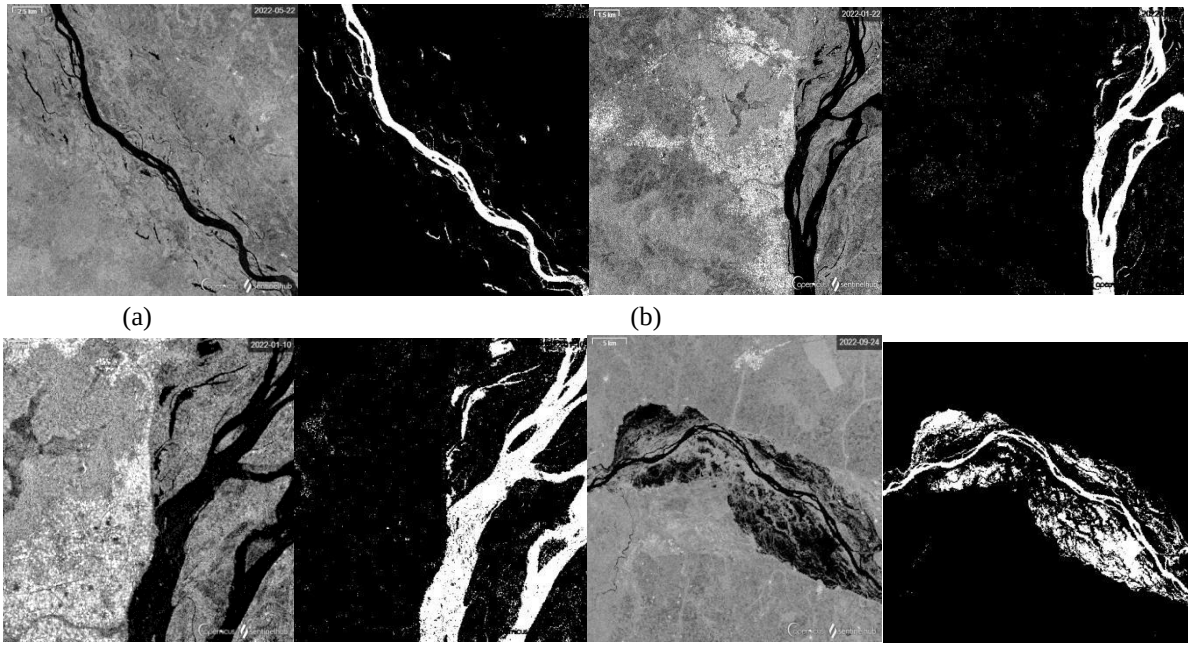


Fig. 1: Timelapse creation and download from Copernicus data hub

The chosen period is between January to December, 2022. The dates of the frames within the chosen period was selected by the system. These are the dates on which the satellite captured the selected areas.

The images were pre-processed in SNAP software following the recommendation of Copernicus. They were also further pre-processed with Python in Keras before feeding them into the model.

Some of the images and their masks used in the dataset are shown below:



(a) (b)

(c) (d)

Fig. 2: (a) to (d) Images and their masks from the dataset

2.2 The UNet Model

UNet is a U-shaped encoder-decoder network architecture, which consists of four encoder blocks and four decoder blocks that are connected via a bridge. U-Net consists of Convolution Operation, Max Pooling, ReLU Activation, Concatenation and Up Sampling Layers with three sections: contraction, bridge/bottleneck, and expansion section. The UNET architecture has an encoder and a decoder (a contracting path and an expansive path). The contracting path is made up of a normal convolutional network, which consists of multiple convolutions. Each of the convolution is followed by a Rectified Linear Unit (ReLU), a Batch Normalization to prevent overfitting and a max pooling operation for downsampling and spatial dimension reduction. Skip connections are used at every step by concatenating the output of the transposed convolution layers with the feature maps from the Encoder, which is at the same level. This is done to recover the loss of the border pixels in every convolution. Concatenation of feature maps helps to give localized information. The expansive part is made up of convolutions, ReLU, batch normalization and transpose convolutions or upsampling for spatial dimension increment. The final layer uses a Softmax activation function for multi-class output.

The size of the input data is 128 x 128 x 3. Each purple bar corresponds to a multi-channel feature map. The number of channels is denoted at the top of the bar. The size of feature maps for each layer is denoted on the left of the bar. The gray bars concatenated with purple bars are the copies of the feature maps from the encoder.

The developed model is a forty-two layer network with 23 Convolutional layers (nineteen convolutions and four transpose convolutions), ReLU, Batch normalization, Max pool and Up-sampling layers. This architecture enables very precise semantic segmentation results.

This modified UNet model utilized:

- Three input channels and three output channels.
- Zero padding.
- Adam optimizer (with a default learning rate of 0.001).
- Batch normalization is applied after ReLU instead of the dropout layers used in the model described by Ronnerberg (2015).
- Categorical Cross-Entropy loss.

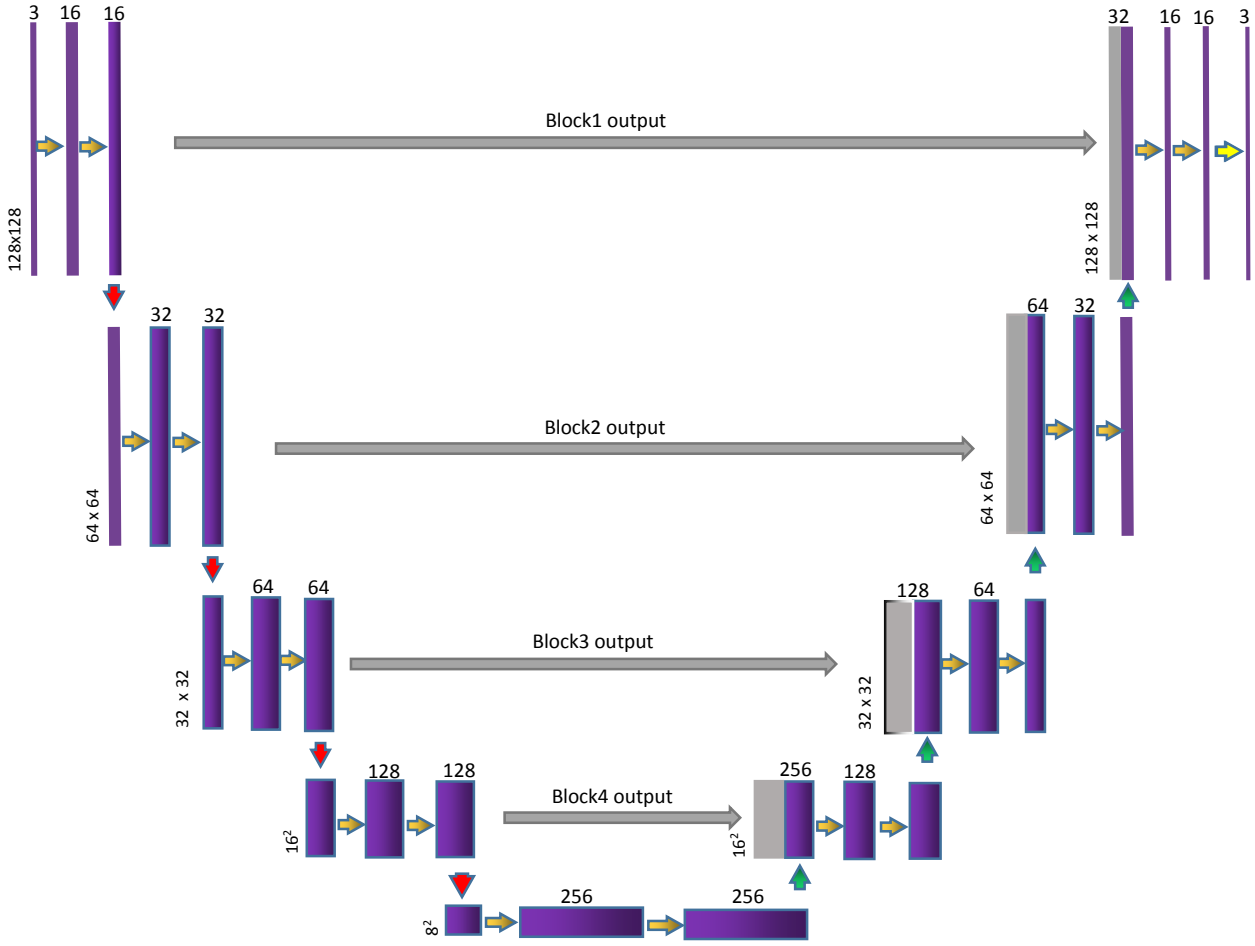


Fig. 3: The UNet model architecture

The summary of the layers is shown in Table 1.

Table 1: Summary of the layers of the UNet model

Model: "FPWS_model"			
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 256, 256, 3)]	0	[]
lambda (Lambda)	(None, 256, 256, 3)	0	['input_1[0][0]']
conv2d (Conv2D)	(None, 256, 256, 16)	448	['lambda[0][0]']
batch_normalization (BatchNormalization)	(None, 256, 256, 16)	64	['conv2d[0][0]']
conv2d_1 (Conv2D)	(None, 256, 256, 16)	2320	['batch_normalization[0][0]']
max_pooling2d (MaxPooling2D)	(None, 128, 128, 16)	0	['conv2d_1[0][0]']
conv2d_2 (Conv2D)	(None, 128, 128, 32)	4640	['max_pooling2d[0][0]']
batch_normalization_1 (BatchNormalization)	(None, 128, 128, 32)	128	['conv2d_2[0][0]']
conv2d_3 (Conv2D)	(None, 128, 128, 32)	9248	['batch_normalization_1[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 32)	0	['conv2d_3[0][0]']
conv2d_4 (Conv2D)	(None, 64, 64, 64)	18496	['max_pooling2d_1[0][0]']
batch_normalization_2 (BatchNormalization)	(None, 64, 64, 64)	256	['conv2d_4[0][0]']
conv2d_5 (Conv2D)	(None, 64, 64, 64)	36928	['batch_normalization_2[0][0]']
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 64)	0	['conv2d_5[0][0]']
conv2d_6 (Conv2D)	(None, 32, 32, 128)	73856	['max_pooling2d_2[0][0]']

batch_normalization_3 (BatchNormalization)	(None, 32, 32, 128)	512	['conv2d_6[0][0]']
conv2d_7 (Conv2D)	(None, 32, 32, 128)	147584	['batch_normalization_3[0][0]']
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 128)	0	['conv2d_7[0][0]']
conv2d_8 (Conv2D)	(None, 16, 16, 256)	295168	['max_pooling2d_3[0][0]']
batch_normalization_4 (BatchNormalization)	(None, 16, 16, 256)	1024	['conv2d_8[0][0]']
conv2d_9 (Conv2D)	(None, 16, 16, 256)	590080	['batch_normalization_4[0][0]']
conv2d_transpose (Conv2DTranspose)	(None, 32, 32, 128)	131200	['conv2d_9[0][0]']
concatenate (Concatenate)	(None, 32, 32, 256)	0	['conv2d_transpose[0][0]',
			'conv2d_7[0][0]']
conv2d_10 (Conv2D)	(None, 32, 32, 128)	295040	['concatenate[0][0]']
batch_normalization_5 (BatchNormalization)	(None, 32, 32, 128)	512	['conv2d_10[0][0]']
conv2d_11 (Conv2D)	(None, 32, 32, 128)	147584	['batch_normalization_5[0][0]']
conv2d_transpose_1 (Conv2DTranspose)	(None, 64, 64, 64)	32832	['conv2d_11[0][0]']
concatenate_1 (Concatenate)	(None, 64, 64, 128)	0	['conv2d_transpose_1[0][0]',
			'conv2d_5[0][0]']
conv2d_12 (Conv2D)	(None, 64, 64, 64)	73792	['concatenate_1[0][0]']
batch_normalization_6 (BatchNormalization)	(None, 64, 64, 64)	256	['conv2d_12[0][0]']
conv2d_13 (Conv2D)	(None, 64, 64, 64)	36928	['batch_normalization_6[0][0]']
conv2d_transpose_2 (Conv2DTranspose)	(None, 128, 128, 32)	8224	['conv2d_13[0][0]']
concatenate_2 (Concatenate)	(None, 128, 128, 64)	0	['conv2d_transpose_2[0][0]','con v2d_3[0][0]']
conv2d_14 (Conv2D)	(None, 128, 128, 32)	18464	['concatenate_2[0][0]']
batch_normalization_7 (BatchNormalization)	(None, 128, 128, 32)	128	['conv2d_14[0][0]']
conv2d_15 (Conv2D)	(None, 128, 128, 32)	9248	['batch_normalization_7[0][0]']
conv2d_transpose_3 (Conv2DTranspose)	(None, 256, 256, 16	2064	['conv2d_15[0][0]']
concatenate_3 (Concatenate)	(None, 256, 256, 32)	0	['conv2d_transpose_3[0][0]', 'conv2d_1[0][0]']
conv2d_16 (Conv2D)	(None, 256, 256, 16)	4624	['concatenate_3[0][0]']
batch_normalization_8 (BatchNormalization)	(None, 256, 256, 16	64	['conv2d_16[0][0]']
conv2d_17 (Conv2D)	(None, 256, 256, 16)	2320	['batch_normalization_8[0][0]']
conv2d_18 (Conv2D)	(None, 256, 256, 3)	51	['conv2d_17[0][0]']
==========	==========	=====	==========
Total params: 1,944,083			
Trainable params: 1,942,611			
Non-trainable params: 1,472			
_____ _	_____ _	_____ __	_____ _____

2.3 Model Training

The model was trained on a satellite image dataset of 1,412 images of before, during and after flood. The development and training were done on Intel Core i5 CPU @ 2.40GHz - 2.50 GHz with 8.00 GB RAM. The model was trained on the dataset for 100 training epochs with Binary Cross entropy as the loss function and Adam Optimizer as the optimization algorithm with default parameters α = 0.001, β_1 = 0.9, β_2 = 0.999 and $\epsilon = 10^{-8}$.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)[\frac{\delta L}{\delta w_t}] v_t = \beta_2 v_{t-1} + (1-\beta_2)[\frac{\delta L}{\delta w_t}] \quad (1)$$

where m_t = aggregate of gradients at time t [current] (initially, m_t = 0); m_{t-1} = aggregate of gradients at time t-1, w_t = weights at time t; δL = derivative of Loss Function; δw_t = dervative of weights at time t; β_1 and β_2 = decay rates of average of gradients.

2.4 Model Training Flowchart

The flowchart in figure 4 depicts the development and training of the model. The dataset is loaded and pre-processed. The pre-processing involves the cropping and re-sizing of the images to fit the design of the model. The images are then augmented. The cropping, re-sizing and augmentation of the images increases the size of the dataset. The hyper-parameters are set and the training of the model is done. Afterwards, the model is validated and evaluated. A prediction test is also done and the improved model is saved.

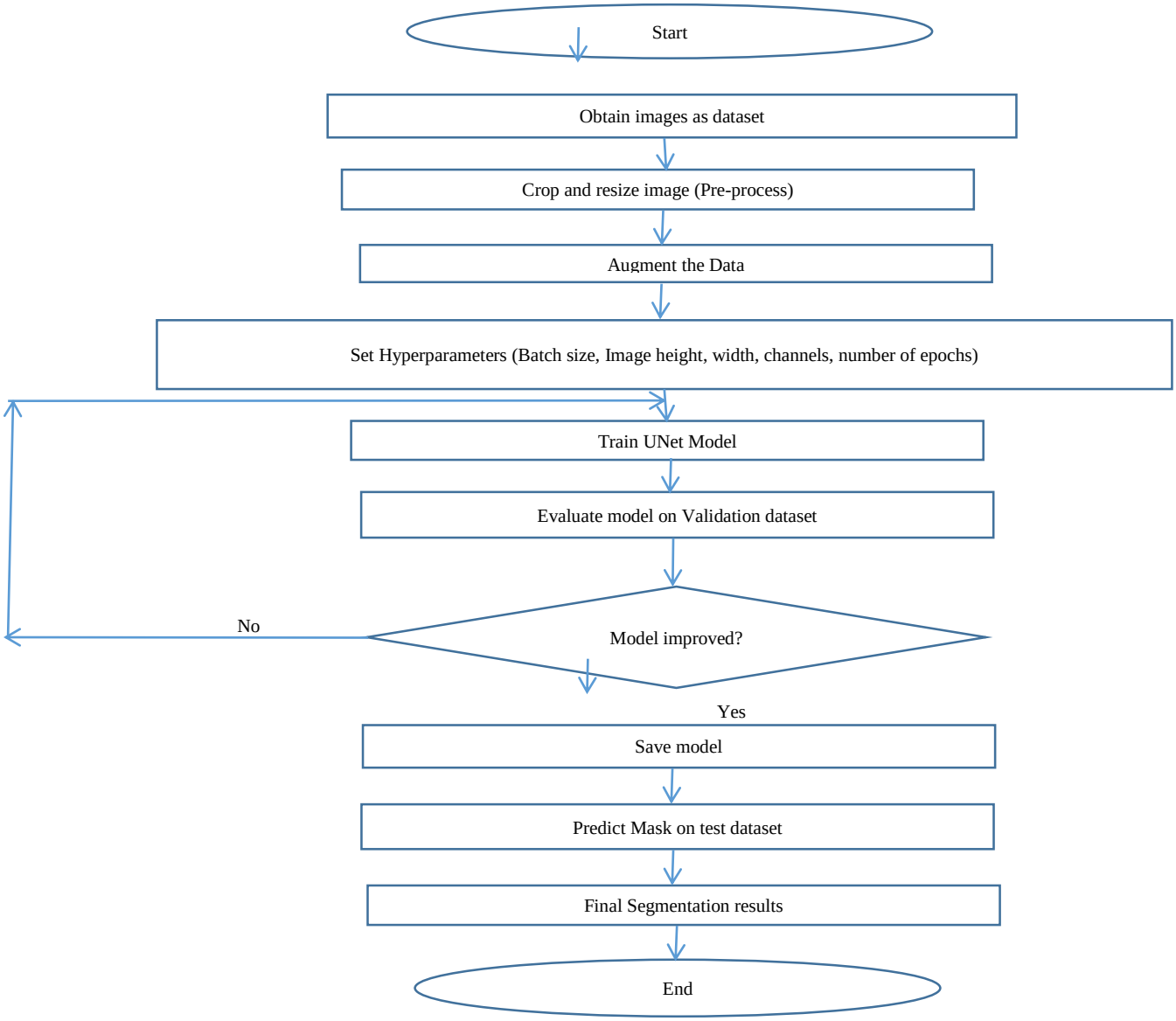


Fig. 4: Flowchart of the Model training process

To adjust the input size of the designed UNet, the training and validation images were re-sized to 128 x 128 pixels resolutions. The validation was implemented and the training loss and validation accuracy are recorded accordingly.

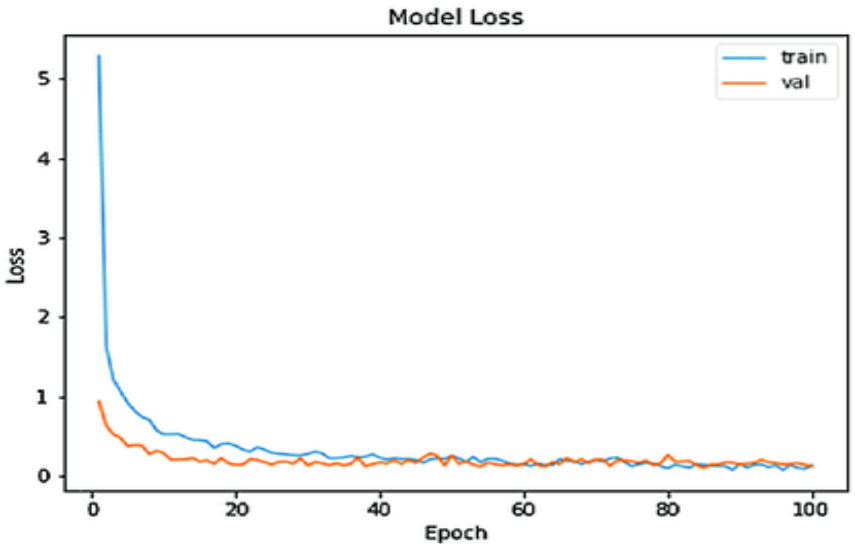


Fig. 5: Training and Validation Loss for the epochs

The training and validation loss is shown in figure 5. The training loss maintains at first and then descends rapidly after the 20 epochs. The validation loss shows a gradual descent and the maintenance of a particular margin, together with the training loss, with slight variations. Then the convergence of the model at the last epoch.

3.0 RESULTS AND DISCUSSION

The evaluation metrics used were Accuracy and Jaccard Index or Intersection over Union (IoU). These are the evaluation indicators commonly used in image segmentation. Figure 6 presents the training and validation accuracy.

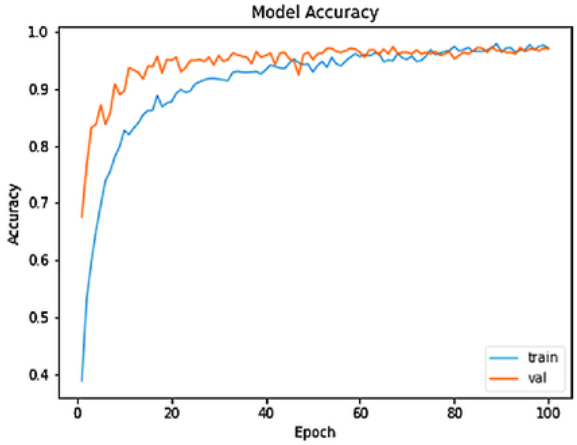


Fig. 6: UNet model accuracy

When the class representation is small within the image, this metric can often produce deceptive results because the measure is skewed in reporting how effectively you identify negative cases (that is, where the class is not present).

The jaccard index or intersection over union is a method to quantify the percent overlap between the target mask and the prediction output. This method calculates the performance of the model by calculating the intersection and union between the Ground Truth and the Prediction. It measures the number of pixels common between the target and prediction masks divided by the total number of pixels present across both masks.

$$IoU = \frac{target \cap prediction}{target \cup prediction} \quad (2)$$

The intersection (A∩B) comprises of pixels from both the prediction mask and the ground truth mask, while the union (A∪B) is made up of all pixels from either the prediction or target mask.

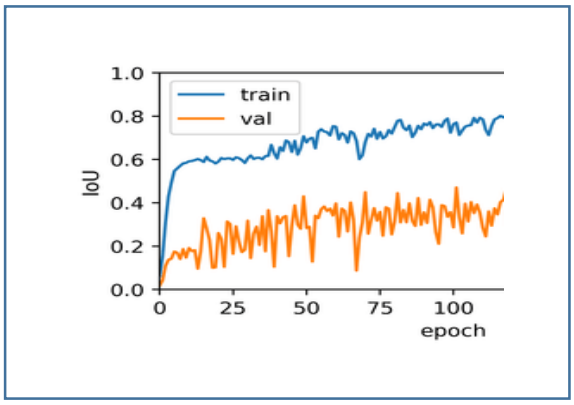


Fig. 7: Training and Validation IoU

The model's IoU value was not a steady rise for the validation curve as is depicted in figure 7.

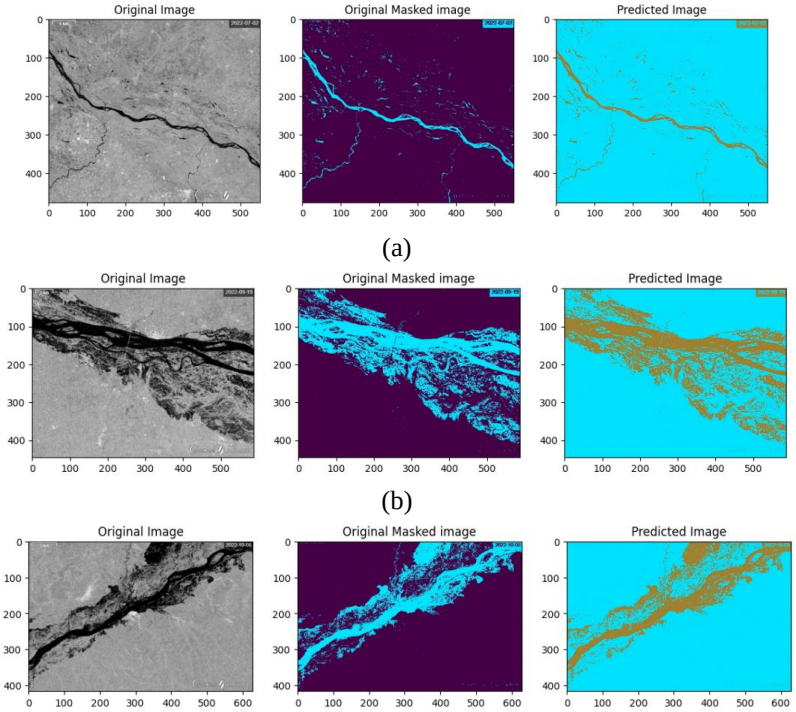
```
model.evaluate(X_test, y_test, verbose=1)

Out[112]: 3/3 [==============================] - 1s 310ms/step - loss: 0.1213 -
```

Fig. 8: Model evaluation showing results of metrics

During training, the learning rate was adjusted from 0.001 to 0.0001. This reduced the loss and increased the accuracy and IoU.

3.1 Predictions from the trained Model

The flood detection using SAR satellite data performed quite well according to the evaluation factors. The detection of both non-flood and flood areas was done very well. UNet does not require an extremely large dataset (unlike CNN) therefore, the number of images in the training dataset was considered adequate. On all plots, the borders of the flood areas are included to facilitate the visualization of the flood area.



(c)**Fig. 9: (a) - (c) Flood detection results**

4.0 CONCLUSION

In this study, a modified UNet model was proposed and developed. The model strongly learned the features of the SAR satellite data. It was trained on 1,412 SAR satellite images for 5 hours on Intel Core i5 CPU @ 2.40GHz - 2.50 GHz with 8.00 GB RAM. The training was done for 100 Epochs of 16 per batch with Binary Cross entropy as the loss function and Adam Optimizer as the optimization algorithm. The result of the extensive experiments on the SAR dataset yields 95% accuracy and 42% IoU score. The experiments and comparisons with the standard U-Net model show that the proposed model achieves good segmentation performance, with a 5% improvement in Accuracy but a significant reduction in performance in the IoU score.

The model achieved satisfactory segmentation of the images but the IoU score is still significantly low, which shows that there might be misclassification of images when tested on other datasets. Improvement is possible with respect to the IoU of the training and validation dataset. The training and validation curves started out steady but increased in fluctuations as training progressed. There is also a wide gap between the two curves.

5.0 RECOMMENDATION

There is room for improvement in the performance of the developed UNet model. The hyperparameters can be further adjusted to achieve better performance.

REFERENCES

1. Benbahria, Z., Smiej, M. F., Sebari, I. and Hajji, H. (2019). Land cover intelligent mapping using transfer learning and semantic segmentation. *Proceedings 7th Mediterranean Congress Telecommunication*, pp. 1–5.
2. Clement, M. A., Kilsby, C. G. and Moore, P. (2018). Multi-temporal synthetic aperture radar flood mapping using change detection. *Journal of Flood Risk Management*, vol. 11, no. 2, pp. 152–168.
3. Copernicus (August 3, 2022). *Mission ends for Copernicus Sentinel-1B satellite*. https://www.esa.int/Applications/Observing_the_Earth/Copernicus/Sentinel-1/Mission_ends_for_Copernicus_Sentinel-1B_satellite
4. Garcia-Pintado, J., Mason, D., Dance, S. L., Cloke, H., Neal, J. C., Freer, J., & Bates, P. D. (2015). Satellite-supported flood forecasting in rivernetworks: A real case study. *Journal of Hydrology*, 523, 706–724. https://doi.org/10.1016/j.jhydrol.2015.01.084
5. Hernández, D., Cecilia, J. M., Cano, J.-C., & Calafate, C. T. (2021). Flood Detection Using Real-Time Image Segmentation from Unmanned Aerial Vehicles on Edge-Computing Platform. *Remote Sensing*, *14*(1).
6. Hordiiuk, D., Oliinyk, I., Hnatushenko, V. and Maksymov, K. (2019). "Semantic Segmentation for Ships Detection from Satellite Imagery," IEEE 39th International Conference on Electronics and Nanotechnology (ELNANO), Kyiv, Ukraine, pp. 454-457, DOI: 10.1109/ELNANO.2019.8783822.
7. Hostache, R., Chini, M., Giustarini, L.,Neal, J., Kavetski, D., Wood, M., Co-rato, G., Pelich, R. & Matgen, P. (2018). Near- Real- Time Assimilation of SAR- Derived Flood Maps for ImprovingFlood Forecasts. Water Resources Re -search, 54(8), 5516–5535. https://doi.org/10.1029/2017WR022205
8. Jaisakthi, S. M., Dhanya, P. R., Jitesh Kumar, S. (2021). Detection of Flooded Regions from Satellite Images Using Modified UNET. *4th International Conference on Computational Intelligence in Data Science (ICCIDS)*, Chennai, India. pp.167-174, 10.1007/978-3-030-92600-7_16. hal-03772928
9. Kang, W., Xiang, Y., Wang, F., Wan, L., & You, H. (2018). Flood Detection in Gaofen-3 SAR Images via Fully Convolutional Networks. *Sensors*, *18*(9), 2915. MDPI AG. Retrieved from http://dx.doi.org/10.3390/s18092915
10. Katiyar, V.; Tamkuan, N. and Nagai, M. (2020) Flood area detection using SAR images with deep neural. In Proceedings of the 41st Asian Conference of Remote Sensing—Asian Association of Remote Sensing, Deqing China, 9–11 November 2020.
11. Lattari, F. et al. (2019). "Deep learning for SAR image despeckling," *Remote Sensing*, vol. 11, no. 13, Art. no. 1532.
12. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* 521, 436–444 (2015). https://doi.org/10.1038/nature14539
13. Li, Y.; Martinis, S.; Plank, S. and Ludwig, R. (2018). An automatic change detection approach for rapid flood mappingin Sentinel-1 SAR data. *International Journal of Applied Earth Observation and Geoinformatics*, 73, 123–135
14. Nemni, E., Bullock, J., Belabbes, S., & Bromley, L. (2020). Fully Convolutional Neural Network for Rapid Flood Segmentation in Synthetic Aperture Radar Imagery. *Remote Sensing*, *12*(16), 2532. MDPI AG. Retrieved from http://dx.doi.org/10.3390/rs12162532
15. 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. https://doi.org/10.1007/978-3-319-24574-4_28

16. Sanyal, J. and Lu, X. (2004). "Application of remote sensing in flood management with special reference to monsoon asia: A review,"*Natural Hazards: J. Int. Soc. Prevention Mitigation Natural Hazards*, vol. 33, no. 2, pp. 283–301.
17. Shah S.A., Seker D.Z., Hameed S., and Draheim D. (2019). The Rising Role of Big Data Analytics and IoT in Disaster Management: Recent Advances, Taxonomy and Prospects. *IEEE Access.* 2019;**7**:54595–54614. doi: 10.1109/ACCESS.2019.2913340.
18. Yang, X. et al. (2019). "Road detection and centerline extraction via deep recurrent convolutional neural network U-Net," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 9, pp. 7209–7220.
19. Zhang, L., & Xia, J. (2021). Flood Detection Using Multiple Chinese Satellite Datasets during 2020 China Summer Floods. *Remote Sensing*, *14*(1), 51. MDPI AG. Retrieved from http://dx.doi.org/10.3390/rs14010051