Accredited SINTA 2 Ranking
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2024
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online at: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 8 No. 6 (2024) 779 - 787 e-ISSN: 2580-0760

Performance and Efficiency Comparison of U-Net and Ghost U-Net in Road Crack Segmentation with Floating Point and Quantization Optimization

Haidhi Angkawijana Tedja[1*], Onno W. Purbo[2]
[1,2]Department of Computer Science, Informatics, Institut Teknologi Tangerang Selatan, Tangerang Selatan, Indonesia
[1]haidhiangkawijana@gmail.com, [2]onno@indo.net.id

Abstract

This study presents a comprehensive comparison of U-Net and Ghost U-Net for road crack segmentation, emphasizing their performance and memory efficiency across various data representation formats, including FP32, FP16, and INT8 quantization. A dataset of 12,480 images was used, with preprocessing steps such as binarization and normalization to improve segmentation accuracy. Results show that Ghost U-Net achieved a marginally higher performance, with an IoU of 0.5041 and a Dice coefficient of 0.6664, compared to U-Net's IoU of 0.5034 and Dice coefficient of 0.6662. Ghost U-Net also demonstrated significant memory efficiency, reducing GPU usage by up to 60% in FP16 and INT8 formats. However, a sharp decline in performance was observed for Ghost U-Net in the INT8 format, where the IoU dropped to 0.2038 and the Dice coefficient to 0.3227, whereas U-Net maintained stable performance across all formats. These findings suggest that Ghost U-Net is preferable for applications prioritizing memory efficiency and inference speed, while U-Net may be better suited for tasks requiring consistent accuracy across different quantization levels. This study underscores the importance of considering both performance stability and memory efficiency when selecting models for deployment in real-world applications.

Keywords: U-Net; Ghost U-Net; image segmentation; memory efficiency; quantization

How to Cite: H. A. Tedja and Onno W. Purbo, "Performance and Efficiency Comparison of U-Net and Ghost U-Net in Road Crack Segmentation with Floating Point and Quantization Optimization", *J. RESTI (Rekayasa Sist. Teknol. Inf.)* , vol. 8, no. 6, pp. 779 - 787, Dec. 2024.
DOI: https://doi.org/10.29207/resti.v8i6.6089

1. Introduction

Cracks on concrete surfaces are a key indicator that reveals the safety and degradation level of a structure. To maintain the health and reliability of buildings, regular inspection and monitoring of surface cracks are crucial [1]. In the context of infrastructure maintenance, the process of crack detection and road segmentation is one of the critical aspects that requires high accuracy, considering that small and complex cracks on road surfaces are difficult to detect manually [2].

The application of deep learning-based computer vision methods has proven to improve the accuracy and efficiency of crack detection, with U-Net being one of the widely used models for high-quality image segmentation, including in the medical and infrastructure fields. However, as the need for computational efficiency increases, especially on devices with limited resources such as edge computing and mobile devices, a lighter and faster model is needed [3]. Ghost U-Net; a variant of U-Net, was developed to address this issue by using the Ghost module, which reduces the number of parameters and memory usage while maintaining competitive accuracy and improving inference efficiency, especially on low-precision data formats such as FP16 and INT8.

Previous research conducted by Lingyu Sun et al. has shown that Ghost U-Net performs well on underwater imagery [4]. Research by Yinghan Xu et al. also demonstrated good results on medical images [5]. In addition to image segmentation, Ghost U-Net has also shown satisfactory results in the image deblurring case studied by Feng Ziliang [6]. Our Ghost U-Net model also shows improvements in GPU resource utilization and inference time compared to the research conducted by Alessandro Di Benedetto et al., although in terms of metrics, it is still lower [7]. That study achieved 6.7 frames per second (FPS), while our Ghost U-Net

Received: 29-09-2024 | Accepted: 13-12-2024 | Published Online: 28-12-2024

achieved 99.04 FPS for FP32 and 156.53 FPS for FP16. This makes our model suitable for running on embedded devices.

Many studies have examined the application of various image segmentation methods to detect road damage, such as the use of the SDDNet model [8] and DeepCrack [9]. Research related to floating point customization and quantization has been conducted by Mohammad Hossein et al. [10] using the U-Net model for medical images. To the best of our knowledge, no one has yet applied Ghost U-Net nor studied the effects of floating point customization and quantization of U-Net and Ghost U-Net models for road crack segmentation. This study focuses on comparing the performance of U-Net and Ghost U-Net on road crack data. Additionally, this research evaluates the stability of the model on different floating point representations and model quantization.

2. Research Methods

This research aims to compare two models for image segmentation, namely U-Net and Ghost U-Net. The first stage of the research involves data collection, and preprocessing, followed by training both models with varying learning rates. After training, the best model from both U-Net and Ghost U-Net will be selected based on testing performance. The best model will then be converted into different floating-point representations for further evaluation, using metrics such as GPU memory usage and prediction speed.

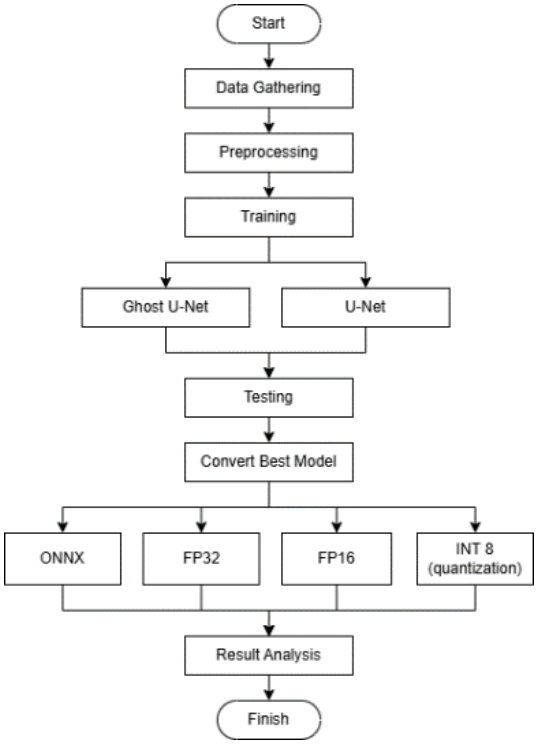

Figure 1. Research flow

The diagram in Figure 1 illustrates the workflow for comparing the performance of the Ghost U-Net and U-Net models in a specific task, such as image segmentation. Figure 1 is an explanation of the steps in the diagram: The first stage is gathering the data that will be used to train the models. The collected data undergoes preprocessing to ensure consistency before being used in the models. Following preprocessing, both the Ghost U-Net and U-Net models are trained using the data with five different learning rates for each model. Once the training phase is complete, the models are tested to evaluate their performance. The best-performing model from these tests is then selected and converted into various formats for performance optimization, including ONNX and different floating-point formats like FP32 and FP16, as well as quantization to INT8. Finally, the performance results across these formats are analyzed to determine the best model and configuration.

Semantic segmentation is the process of pixel-level classification, where each pixel in an image is assigned a specific category. Meanwhile, instance segmentation not only classifies pixels as in semantic segmentation but also distinguishes different objects within the same category [11]. In this research, we will work on semantic segmentation.

U-Net was initially developed as a fully convolutional network model for medical image segmentation [12]. U-Net is an autoencoder model where the encoder part extracts features and information, while the decoder part learns and reconstructs the information from the encoder [13].

Table 1. U-Net Architecture

Block	Output shape	Params
DownConv 1	[64,128,128]	38.720
DownConv 2	[128,64,64]	221.550
DownConv 3	[256,64,64]	885.248
DownConv 4	[512,32,32]	3.539.968
Bottleneck	[1024,16,16]	14.157.824
UpConv 1	[512,32,32]	9.176.576
UpConv 2	[256,64,64]	2.294.528
UpConv 3	[128,128,128]	573.824
UpConv 4	[64, 256, 256]	143.552
Ouput Layer	[1, 256, 256]	65
Total Params		31.031.855

Table 1 presents the architecture of a U-Net model, detailing the layers in the encoder (DownConv), bottleneck, and decoder (UpConv) phases. It includes the output shape and the number of parameters for each layer. The model begins with four downsampling convolutional layers (DownConv) that reduce spatial dimensions while increasing feature maps, followed by a bottleneck layer with the highest number of parameters. The decoder consists of four upsampling convolutional layers (UpConv) that restore the spatial dimensions while reducing the number of feature maps. Finally, the model ends with an output layer that produces a [1,256,256] output. The total number of parameters in the model is approximately 31 million, making this U-Net architecture quite large. To address this, we developed Ghost U-Net which uses ghost modules as the backbone to significantly reduce the number of parameters while maintaining performance.

Ghost U-Net is a U-Net model that uses the Ghost module as its backbone. Due to the large complexity of U-Net's backbone, the model requires significant memory, making it unsuitable for devices with limited memory, such as embedded devices [5]. The model also demonstrates strong performance, as shown by high pixel accuracy and IoU, despite having fewer parameters [14]. By utilizing the Ghost module as the backbone, the model is expected to provide informative features with fewer parameters [15].

Table 2. Ghost U-Net Architecture

Block	Output shape	Params
Ghost DownConv 1	[64,128,128]	2.976
Ghost DownConv 2	[128,64,64]	13.952
Ghost DownConv 3	[256,64,64]	52.480
Ghost DownConv 4	[512,32,32]	203.264
Bottleneck	[1024,16,16]	14.157.824
Ghost UpConv 1	[512,32,32]	2.497.536
Ghost UpConv 2	[256,64,64]	626.176
Ghost UpConv 3	[128,128,128]	157.440
Ghost UpConv 4	[64, 256, 256]	39.808
Ouput Layer	[1, 256, 256]	65
Total Params		17.751.521

Table 2 presents the architecture of a Ghost U-Net model, which utilizes ghost modules to reduce the number of parameters compared to the standard U-Net. Each block of the model, from the Ghost DownConv layers in the encoder to the Ghost UpConv layers in the decoder, is detailed with its output shape and parameter count. The model follows a similar structure to U-Net, with downsampling in the encoder, a bottleneck layer, and upsampling in the decoder. However, by using ghost modules, the total parameters of Ghost U-Net are significantly reduced to approximately 17.75 million, compared to the 31 million parameters in the standard U-Net. This reduction makes Ghost U-Net a more efficient alternative while retaining the original functionality of U-Net.

3. Results and Discussions

The dataset used in this research is a road crack dataset obtained from Kaggle, designed for image segmentation tasks. This dataset consists of 12,480 pairs of images, with a size of 448 x 448 x 3 for the input (x) and 448 x 448 for the labels (y). The input images (x) are RGB images with three channels, while the label images (y) are grayscale images with only one channel. These images will be resized to 256 x 256 to reduce computational load and improve efficiency [16].

Image segmentation generally requires a binary mask to define the area to be predicted by the model. The binarization process converts pixel values into two categories: the object to be predicted by the model, which is marked by a pixel value of 255 or 1, and the background, which is marked by a value of 0. This process is crucial in identifying the area targeted for detection by the model [17]. The collected dataset still contains pixel values ranging from 0 to 255. Figure 2 is a sample of the mask and the proportion of unique pixel values.

In Figure 3, pixels with a value of 0 (black) account for 86.8%, while those with a value of 255 (white) makeup 9.3%. The remaining 3.9% are pixels with values other than 0 and 255. Due to the inconsistency in pixel values, a binarization method will be applied to convert all masks into a binary format. In this study, a threshold of 100 will be used, meaning that all pixels with an intensity value above 100 will be considered as the foreground (typically converted to 255), and pixels with an intensity below that will be considered as the background (typically converted to 0). After going through the binary thresholding process, we will get an image like Figure 4.

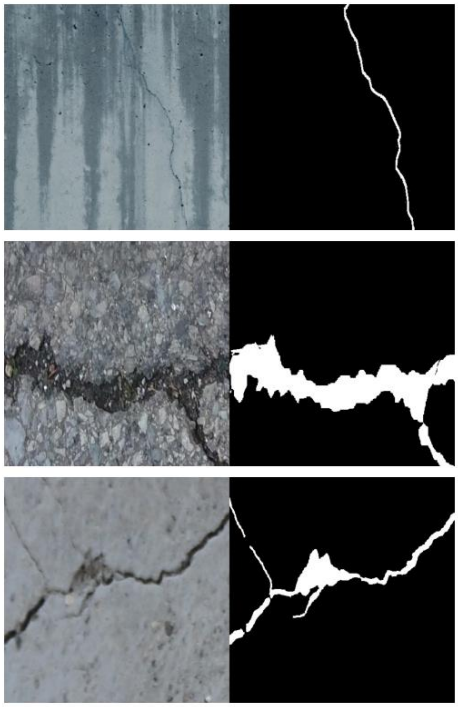

Figure 2. Input image (left) and label mask (right)

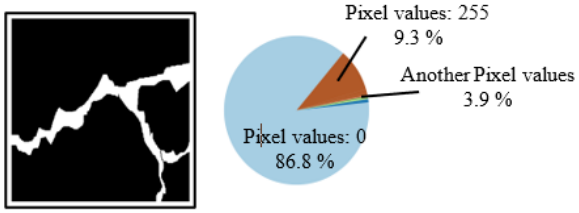

Figure 3. Proportion of Pixels Values

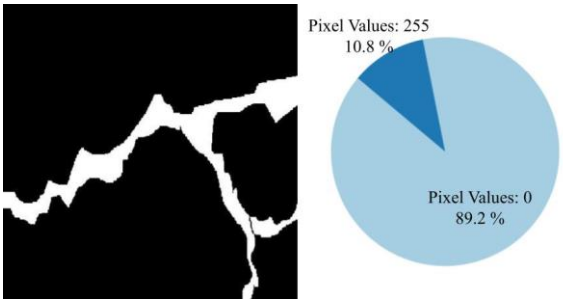

Figure 4. Binarized Image

After harmonizing all labels, the next step in preprocessing is normalization. Normalization is a preprocessing stage where data is scaled so that each feature contributes equally and maintains a consistent value range [18]. Additionally, normalization plays a crucial role in improving the model's generalization ability, ensuring that the model can consistently recognize patterns in varied data, leading to optimal prediction performance on both training data and new, unseen data [19]. In this study, normalization will be performed by dividing each pixel value by 255 [20].

Both models will be trained with a data split of 80:10:10 for training, validation, and testing. The training and validation data will be used during the training process, while the testing data will be employed to evaluate the final results of the models and determine which model performs best [21]. The metrics used for evaluation will include IoU [22], Dice Coefficient [23], and a combination of Dice Loss and Binary Cross Entropy (BCE) as the loss function [24].

Intersection over Union (IoU) is widely used in various tasks, particularly in segmentation. In segmentation tasks, IoU evaluates the accuracy of pixel-wise predictions, which reflect the quality of the learned features. Specifically, IoU is calculated by taking the area of overlap between the predicted segmentation and the ground truth, divided by the area of their union using Equation 1.

$$IoU = \frac{TP}{TP+FN+FP} \quad (1)$$

This metric provides a comprehensive measure of how well the model distinguishes between different classes at the pixel level, making it essential for assessing segmentation performance [25]. The third measure used for further evaluation of network models is the Dice coefficient (F1 score), which is defined as twice the number of common pixels from a given class in the reference image and predicted image, divided by the sum of the pixels belonging to that class in both the reference and predicted images.

$$Dice\ Coefficient = \frac{2TP}{2TP+FN+FP} \quad (2)$$

The Dice coefficient in Equation 2 is positively correlated with the IoU measure; however, in the case of IoU, individual poor segmentation results have a more severe impact on the overall evaluation of the model [26].

For the loss function, we adopt the combination of Binary Cross Entropy (BCE) and Dice Loss, inspired by studies such as those conducted by Vishal Rajput [27] ass shown in Equations 3 and 4.

$$Dice\ Loss = 1 - Dice\ Coefficient \quad (3)$$

$$BCE = -\frac{1}{n}\sum_{i-1}^{n}(Y_i \cdot \log \hat{Y}_i\,(1 - Y_i) \cdot \log(1 - \hat{Y}_i) \quad (4)$$

These works demonstrated that Dice Loss provides superior pixel-wise accuracy, especially in segmentation tasks, while BCE offers better generalization and resistance to noise and adversarial attacks. By merging both loss functions, we aim to improve the model's precision, robustness, and overall performance, achieving a balance between accurate segmentation and resilience against perturbations.

It should be noted that the use of custom representation e.g. ONNX, FP32, FP16, and INT8 is done after training is completed (post-training quantization), with the aim of improving inference efficiency [28]. We used the PyTorch framework for modeling and training, utilizing the default floating point format, which is FP32, due to its balance between precision and computational efficiency.

The training process uses 2x T4 GPU from Kaggle kernel. We will use 50 epochs and a batch size of 16 due to limited computing power and time. We also use checkpoint callbacks to save the model with the lowest loss during the 50 epochs of training. The training results are shown in Tables 3 and 4.

Table 3. U-net training result

Learning Rate	Loss	Iou	Dice
0,1	2,1966	0,029	0,0563
0,01	0,5455	4×10^{-8}	0
0,001	0,2398	0,4471	0,615
0,0001	0,1965	0,5195	0,6812
0,00001	0,2207	0,4837	0,6477

Table 4. Ghost U-net training result

Learning Rate	Loss	Iou	Dice
0,1	0	0,5047	0,6679
0,01	0,198	0,5146	0,677
0,001	0,195	0,5195	0,6809
0,0001	0,1962	0,5186	0,68
0,00001	0,2424	0,4888	0,6529

Tables 3 and 4 show the best results from the validation data during training. The U-Net model performed best when trained with a learning rate of 0.0001, while Ghost U-Net achieved its best results at a learning rate of 0.001. In numerical terms, the performance of U-Net is superior to that of Ghost U-Net, although the difference is not significant, as the values are very close to each other. The final results of the models will be tested on the testing data to measure performance on unseen data during the training process. The model with the best performance on the testing data will be selected for further evaluation using ONNX and TensorRT as shown in Tables 5 and 6.

The best model is the one trained with a learning rate of 0.0001, whether it be U-Net or Ghost U-Net. The U-Net model achieved a loss of 0.2031, an IoU of 0.5034, and a Dice coefficient of 0.6662. In contrast, the Ghost U-Net model recorded a loss of 0.2012, an IoU of 0.5041, and a Dice coefficient of 0.6664. Overall, Ghost U-Net performs slightly better than U-Net.

After obtaining the best model, we will conduct further testing to compare the performance and resources used by each model. The testing will be performed using the

Open Neural Network Exchange (ONNX) framework, which is user-friendly and suitable for use in embedded devices [29]. In addition to ONNX, we will also test using TensorRT, which enables fast predictions [30] and supports floating-point customization and quantization [31]. In this testing, we will use both 32-bit and 16-bit floating-point representations. The higher the bit, the wider the range of values that can be calculated, resulting in more precise predictions but requiring more resources [32]. This is because calculations with 32-bit floating-point representation can cover a broader value range, meaning they can represent very small to very large numbers with higher accuracy and precision [33].

Table 5. U-net on testing data

Learning Rate	Loss	Iou	Dice
0,1	$1{,}98\times10^{40}$	0,00055	0,00111
0,01	0,544	4×10^{-8}	0
0,001	0,2449	0,4322	0,5996
0,0001	0,2031	0,5034	0,6662
0,00001	0,2334	0,455	0,6198

Table 6. Ghost U-net on testing data

Learning Rate	Loss	Iou	Dice
0,1	0,213	0,488	0,6524
0,01	0,203	0,4998	0,663
0,001	0,2029	0,5017	0,6646
0,0001	0,2012	0,5041	0,6664
0,00001	0,2522	0,4658	0,631

Previous research has demonstrated that ONNX provides significant improvements in inference speed compared to other frameworks. In a study involving 25,600 OCT B-scans, the binary prediction results (0-normal, 1-B-scan of interest) were identical between TensorFlow and ONNX inferences in both CPU and GPU modes. However, the average inference execution times showed notable differences. In CPU mode, the average execution time for one macular cube was 8.99±0.09 seconds for TensorFlow and 3.57±0.15 seconds for ONNX, resulting in a speedup of 60.29%. In GPU mode, the average execution times were 0.55±0.03 seconds for TensorFlow and 0.35±0.03 seconds for ONNX, leading to a speedup of 36.36% [34]

Additionally, in a study conducted by Zhang et al., experiments running YOLOv3 on the NVIDIA Jetson Nano revealed that using TensorRT with fp16 quantization resulted in the lowest power consumption, fastest running speed, and least memory usage. The fastest inference time achieved was 13.1 FPS with a 320×192 resolution. This demonstrates that the fp16 model is an effective technique to reduce computational resources while maintaining fast inference speeds, particularly in edge AI applications where power and memory efficiency are critical factors [35].

We will also implement quantization, where the model is converted to an 8-bit integer representation, allowing for faster inference and reduced memory usage, albeit at the cost of accuracy [36]. This occurs because floating-point data types are transformed into integer numbers, allowing calculations to be completed more quickly with less memory, thereby reducing the overall latency of the model and increasing inference speed at the expense of accuracy [37]. The testing will utilize an RTX 3050 mobile GPU, with the parameters to be evaluated being inference time and GPU resource usage. A total of 12,475 images will be used for this evaluation. The results of the tests will be illustrated in in Figure 5.

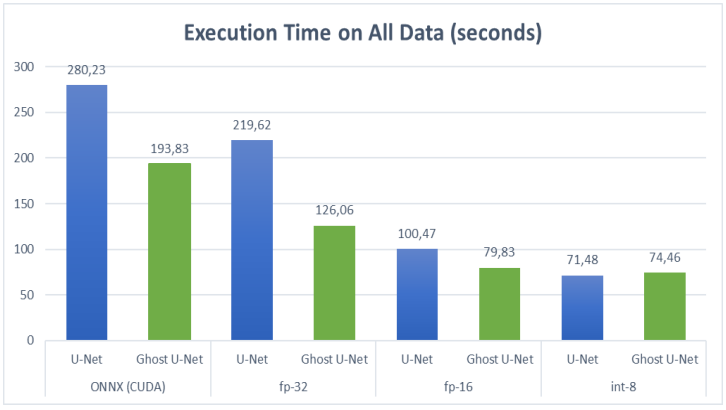

Figure 5. Inference time

The diagram in Figure 5 illustrates a comparison of execution times between U-Net and Ghost U-Net across several data formats, including ONNX (CUDA), FP32, FP16, and INT8. In general, Ghost U-Net demonstrates faster execution times than U-Net, with the only exception being the INT8 format. For the ONNX (CUDA) format, U-Net takes 280.23 seconds to complete the task, while Ghost U-Net finishes in 193.83 seconds. This results in a time difference of 86.4 seconds, indicating that Ghost U-Net is 30.8% faster than U-Net in this format. Moving to the FP32 format, U-Net records an execution time of 219.62 seconds, whereas Ghost U-Net reduces this time to 126.06 seconds. The difference of 93.56 seconds represents the largest time savings, making Ghost U-Net 42.6% more efficient in FP32. In the FP16 format, U-Net's execution time is 100.47 seconds, while Ghost U-Net completes the process in 79.83 seconds, saving 20.64 seconds and showing a 20.5% improvement in speed. However, in the INT8 format, the trend reverses slightly, with U-Net

being marginally faster than Ghost U-Net. U-Net completes the task in 71.48 seconds, compared to Ghost U-Net's 74.46 seconds, making Ghost U-Net 4.2% slower in this format.

In summary, Ghost U-Net consistently outperforms U-Net in terms of execution time across all formats, except for the quantized INT8 format, where U-Net holds a slight advantage. This suggests that Ghost U-Net excels particularly in higher-precision formats such as ONNX (CUDA), FP32, and FP16, offering significant improvements in efficiency.

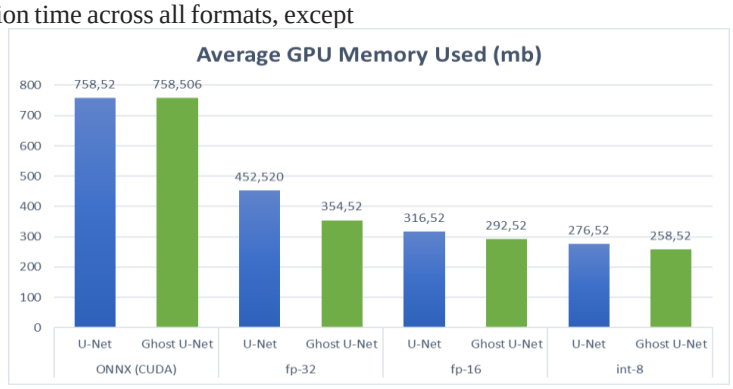

Figure 6. Inference resource

Figure 6 illustrates the average GPU memory usage (in megabytes) for both U-Net and Ghost U-Net models across different data representations: ONNX (CUDA), FP32, FP16, and INT8. The first comparison shows that U-Net and Ghost U-Net consume nearly the same amount of memory, approximately 758.52 MB, when using the ONNX (CUDA) format, with a negligible difference. However, as smaller data representations are used, significant memory savings become evident. When using FP32, U-Net consumes 452.52 MB, while Ghost U-Net reduces memory usage to 354.52 MB, showcasing Ghost U-Net's efficiency. The trend continues with FP16, where U-Net uses 316.52 MB and Ghost U-Net further reduces this to 292.52 MB. The most optimized memory usage is observed with the INT8 representation, where U-Net consumes 276.52 MB, and Ghost U-Net uses the least, at 258.52 MB. Overall, Ghost U-Net consistently uses less GPU memory than U-Net across all representations, especially when employing smaller data types like FP16 and INT8. These results suggest that using reduced data representations, such as FP16 and INT8, can lower GPU memory usage by nearly 60% compared to the FP32 and ONNX formats, offering a clear advantage in memory efficiency without compromising model performance. Further testing will involve evaluating performance metrics with the same dataset.

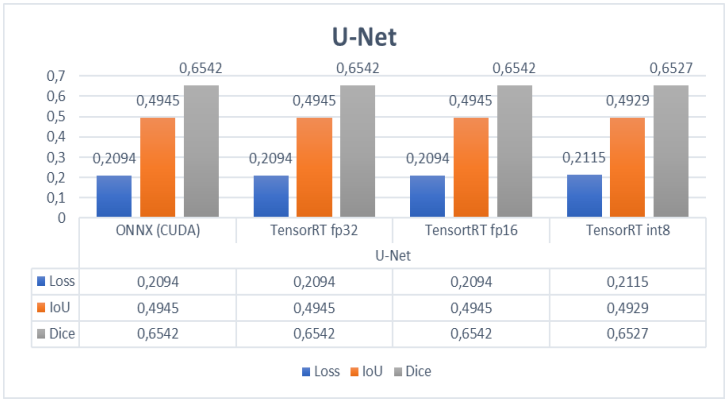

Figure 7. U-Net performance on all representation

The graph in Figure 7 presents a performance comparison of the U-Net model using several data representation formats, including ONNX (CUDA), TensorRT FP32, TensorRT FP16, and TensorRT INT8. The results indicate a high level of consistency across most configurations in terms of loss, Intersection over Union (IoU), and Dice coefficient. For the loss metric, U-Net maintained a stable value of 0.2094 across ONNX, TensorRT FP32, and TensorRT FP16 representations. However, a slight increase to 0.2115 is observed in the TensorRT INT8 representation.

In terms of the IoU (Intersection over Union), the model achieved a score of 0.4945 for ONNX, FP32, and FP16 representations, showing a steady performance across these configurations. A minor reduction to 0.4929 occurred when the model was run using the INT8 representation, indicating a slight decrease in segmentation accuracy at this reduced precision level.

For the Dice coefficient, which measures the overlap between the predicted segmentation and the ground truth, the U-Net model consistently scored 0.6542 across ONNX, FP32, and FP16. Like the IoU, the Dice

coefficient showed a minor drop to 0.6527 when using the INT8 format.

In summary, while U-Net's performance remains relatively consistent across ONNX, FP32, and FP16 data representations, a slight degradation in performance is observed when using the INT8 format. This is evident from the marginal increases in loss and small decreases in both IoU and Dice coefficient. Despite these reductions, the differences are minimal, making INT8 a viable option when prioritizing memory efficiency over the finest performance metrics.

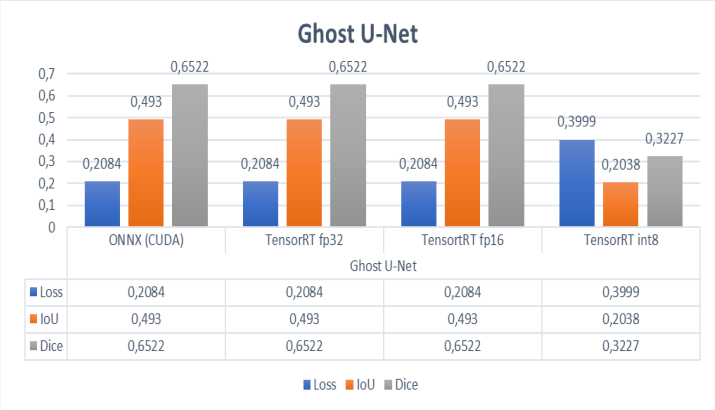

Figure 8. Ghost U-Net performance on all representation

The graph in Figure 8 depicts the performance of the Ghost U-Net model across four different data representation formats: ONNX (CUDA), TensorRT FP32, TensorRT FP16, and TensorRT INT8. For the first three formats (ONNX, FP32, and FP16), the performance remains relatively consistent, with a loss value of 0.2084, IoU (Intersection over Union) of 0.493, and a Dice coefficient of 0.6522. This stability indicates that Ghost U-Net performs reliably across these representations, maintaining accuracy in segmentation tasks.

However, when transitioning to the TensorRT INT8 format, there is a noticeable degradation in performance. The loss value increases significantly to 0.3999, indicating that the model's predictions deviate more from the ground truth compared to the other formats. Similarly, the IoU drops sharply to 0.2038, suggesting a considerable reduction in segmentation accuracy. The Dice coefficient also decreases substantially to 0.3227, reflecting a significant loss in overlap between predicted and actual segmentations.
In comparison to the U-Net model, Ghost U-Net demonstrates less stability, particularly in the INT8 representation. While both models experience some performance decline with INT8, Ghost U-Net's metrics show a much larger drop, particularly in terms of IoU and Dice coefficient. This suggests that Ghost U-Net may be less suited to lower-precision formats like INT8, where memory savings come at the expense of substantial performance degradation. Further analysis will include visualizing the segmentation results for each model and representation to better understand these performance differences.

The results of the tests from Figures 9, 10 and 11 indicate that both the U-Net and Ghost U-Net models perform well, except for the Ghost U-Net that was quantized to 8-bit integer. This model fails to segment images effectively, even producing no output at all. This issue is likely due to the model's complexity and the tasks it is performing, as U-Net utilizes the Ghost module as its backbone to reduce the number of parameters. The use of the Ghost module can decrease the model's robustness [38]. When robustness decreases, the model becomes more sensitive to the noise generated by the quantization process [39].

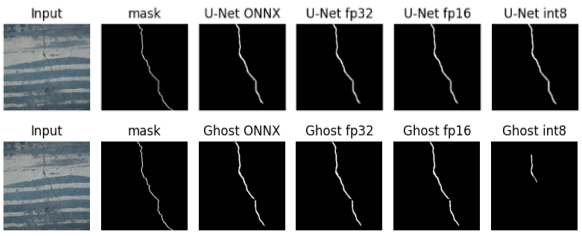

Figure 9. Inference result 1

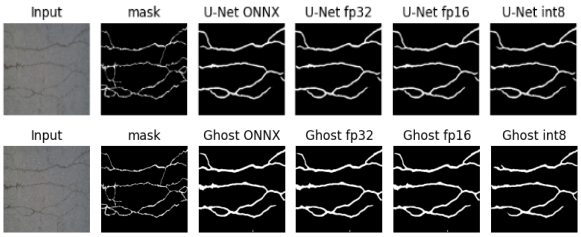

Figure 10. Inference result 2

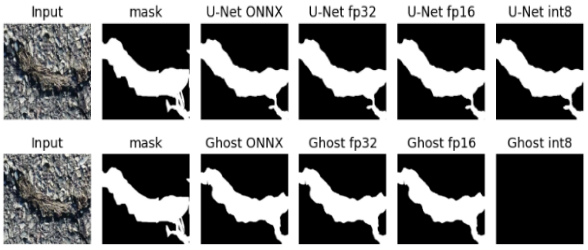

Figure 11. Inference result 3

4. Conclusions

This research focuses on road crack image segmentation using two deep learning models, U-Net and Ghost U-Net, with a dataset from Kaggle consisting of 12,480 pairs of images. The images were preprocessed using binarization and normalization techniques to facilitate segmentation, with binary masks defining objects and backgrounds. The training data was divided into training, validation, and testing sets in an 80:10:10 ratio. The metrics used were IoU, Dice Coefficient, and a combined loss function of binary cross-entropy and Dice loss. The models were trained for a total of 50 epochs. Ghost U-Net demonstrated advantages in model efficiency by reducing the number of parameters from 31 million (U-Net) to approximately 17 million without significant performance loss. U-Net achieved the best results with a learning rate of 1e-4, resulting in a loss of 0.1965, IoU of 0.5195, and a Dice coefficient of 0.6812, while Ghost U-Net showed very similar performance at a learning rate of 1e-3, with a loss of 0.195, IoU of 0.5195, and a Dice coefficient of 0.6809. During testing with the test data, both models achieved their best results at a learning rate of 1e-4, with U-Net achieving a loss of 0.2031, IoU of 0.5034, and Dice of 0.6662, while Ghost U-Net obtained a loss of 0.2012, IoU of 0.5041, and Dice of 0.6664. This indicates that, although Ghost U-Net has fewer parameters, it can rival U-Net's performance. After training and identifying the best model, we evaluated the performance of both models on various data representations (FP32, FP16, and INT8) using the ONNX and TensorRT frameworks. Ghost U-Net excelled in inference speed, being 20%-40% faster and using approximately 35MB less memory across all data representations. In terms of metrics—IoU, Dice, and loss—the U-Net model achieved consistent results with slight degradation in the INT8 format. Conversely, Ghost U-Net maintained stability in ONNX, FP32, and FP16 formats but experienced a significant performance drop in INT8, with a notable increase in loss and sharp decreases in IoU and Dice coefficient. This demonstrates that models with fewer parameters can be sensitive to quantization. These findings highlight the importance of selecting an appropriate model based on application priorities, whether efficiency or stability is preferred. To our knowledge, there has been no prior research specifically analyzing the impact of floating-point customization and quantization (FP16, FP32, INT8) on the performance of deep learning models, particularly in the context of road crack segmentation. Therefore, this research makes a significant contribution to bridging the literature gap related to memory efficiency and model stability based on data representation formats. The results provide essential insights for researchers and practitioners in selecting and adjusting deep learning models according to computational needs and application accuracy. Further research will focus on exploring more advanced quantization techniques to improve the accuracy of quantized models. Additionally, lighter and more efficient model architectures will be investigated. Finally, larger and more diverse datasets will be used to enhance the model's generalization capabilities.

Acknowledgements

Dataset from Kaggle and contributions from ONNX and TensorRT developers are greatly appreciated for for supporting this research.

References

[1] B. Kim, N. Yuvaraj, K. R. Sri Preethaa, and R. Arun Pandian, "Surface crack detection using deep learning with shallow CNN architecture for enhanced computation," *Neural Comput. Appl.*, vol. 33, no. 15, pp. 9289–9305, 2021 https://doi.org/10.1007/s00521-021-05690-8.

[2] M. Zhang and J. Xu, "A semantic segmentation model for road cracks combining channel-space convolution and frequency feature aggregation," *Sci. Rep.*, vol. 14, no. 1, p. 16038, 2024 https://doi.org/10.1038/s41598-024-66182-y

[3] L. Xiaolin, R. C. Panicker, B. Cardiff, and D. John, "Multistage Pruning of CNN Based ECG Classifiers for Edge Devices," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, 2021, pp. 1965–1968. https://doi.org/10.1109/EMBC46164.2021.9630588.

[4] L. Sun, W. Li, and Y. Xu, "Ghost-UNet: Lightweight model for underwater image enhancement," *Eng. Appl. Artif. Intell.*, vol. 133, p. 108585, 2024, https://doi.org/10.1016/j.engappai.2024.108585.

[5] Y. Xu, Q. Li, S. He, and B. Awudong, "Ghost-Unet: An efficient convolutional neural network for spine MR image segmentation: Lightweight Segmentation Method for Spine MRI," in *Proceedings of the 2022 4th International Conference on Robotics, Intelligent Control and Artificial Intelligence*, in RICAI '22. New York, NY, USA: Association for Computing Machinery, 2023, pp. 1159–1163. https://doi.org/10.1145/3584376.3584581.

[6] Z. Feng, J. Zhang, X. Ran, D. Li, and C. Zhang, "Ghost-Unet: multi-stage network for image deblurring via lightweight subnet learning," *Vis. Comput.*, 2024 https://doi.org/10.1007/s00371-024-03315-4.

[7] A. Di Benedetto, M. Fiani, and L. M. Gujski, "U-Net-Based CNN Architecture for Road Crack Segmentation," *Infrastructures*, vol. 8, no. 5, 2023 https://doi.org/10.3390/infrastructures8050090.

[8] W. Choi and Y.-J. Cha, "SDDNet: Real-Time Crack Segmentation," *IEEE Trans. Ind. Electron.*, vol. 67, no. 9, pp. 8016–8025, 2020 https://doi.org/10.1109/TIE.2019.2945265.

[9] Y. Liu, J. Yao, X. Lu, R. Xie, and L. Li, "DeepCrack: A deep hierarchical feature learning architecture for crack segmentation," *Neurocomputing*, vol. 338, pp. 139–153, 2019 https://doi.org/10.1016/j.neucom.2019.01.036.

[10] M. AskariHemmat *et al.*, "U-Net Fixed-Point Quantization for Medical Image Segmentation," in *Large-Scale Annotation of Biomedical Data and Expert Label Synthesis and Hardware Aware Learning for Medical Imaging and Computer Assisted Intervention*, L. Zhou, N. Heller, Y. Shi, Y. Xiao, R. Sznitman, V. Cheplygina, D. Mateus, E. Trucco, X. S. Hu, D. Chen, M. Chabanas, H. Rivaz, and I. Reinertsen, Eds., Cham: Springer International Publishing, 2019, pp. 115–124. https://doi.org/10.1007/978-3-030-33642-4_13

[11] R. Azad *et al.*, "Medical Image Segmentation Review: The Success of U-Net," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–20, 2024 https://doi.org/10.1109/TPAMI.2024.3435571.

[12] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *CoRR*, vol. abs/1505.04597, 2015, [Online]. Available: http://arxiv.org/abs/1505.04597

[13] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni,

"U-Net and Its Variants for Medical Image Segmentation: A Review of Theory and Applications," *IEEE Access*, vol. 9, pp. 82031–82057, 2021 https://doi.org/ 10.1109/ACCESS.2021.3086020.

[14] I. A. Kazerouni, G. Dooly, and D. Toal, "Ghost-UNet: An Asymmetric Encoder-Decoder Architecture for Semantic Segmentation From Scratch," *IEEE Access*, vol. 9, pp. 97457–97465, 2021 https://doi.org/ 10.1109/ACCESS.2021.3094925

[15] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "GhostNet: More Features From Cheap Operations," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1577–1586. https://doi.org/10.1109/CVPR42600.2020.00165.

[16] M. Kumaravelan Umashankarand Nivedita, "Localized Super Resolution for Foreground Images Using U-Net and MR-CNN," in *Computer Vision and Machine Intelligence Paradigms for SDGs*, S. M. and W. S.-H. Kannan R. Jagadeeshand Thampi, Ed., Singapore: Springer Nature Singapore, 2023, pp. 25–39. https://doi.org/10.1007/978-981-19-7169-3_3

[17] J. Pandey Shreyaand Bharti, "Review of Different Binarization Techniques Used in Different Areas of Image Analysis," in *Evolution in Signal Processing and Telecommunication Networks*, J. and S. S. C. and B. V. Chowdary P. Satish Ramaand Anguera, Ed., Singapore: Springer Singapore, 2022, pp. 249–268. https://doi.org/10.1007/978-981-16-8554-5_25

[18] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Appl. Soft Comput.*, vol. 97, p. 105524, 2020, doi: https://doi.org/10.1016/j.asoc.2019.105524.

[19] P. L. Delisle, B. Anctil-Robitaille, C. Desrosiers, and H. Lombaert, "Realistic image normalization for multi-Domain segmentation," *Med. Image Anal.*, vol. 74, p. 102191, Dec. 2021, doi: 10.1016/J.MEDIA.2021.102191.

[20] X. Pei *et al.*, "Robustness of machine learning to color, size change, normalization, and image enhancement on micrograph datasets with large sample differences," *Mater. Des.*, vol. 232, p. 112086, 2023, doi: https://doi.org/10.1016/j.matdes.2023.112086.

[21] F. Maleki, N. Muthukrishnan, K. Ovens, C. Reinhold, and R. Forghani, "Machine Learning Algorithm Validation: From Essentials to Advanced Applications and Implications for Regulatory Certification and Deployment," *Neuroimaging Clin.*, vol. 30, no. 4, pp. 433–445, Nov. 2020 https://doi.org/10.1016/j.nic.2020.08.004.

[22] F. van Beers, A. Lindström, E. Okafor, and M. Wiering, "Deep Neural Networks with Intersection over Union Loss for Binary Image Segmentation," in *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*, SciTePress, Mar. 2019, pp. 438–445. https://doi.org/10.5220/0007347504380445.

[23] A. W. Setiawan, "Image Segmentation Metrics in Skin Lesion: Accuracy, Sensitivity, Specificity, Dice Coefficient, Jaccard Index, and Matthews Correlation Coefficient," in *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, 2020, pp. 97–102. https://doi.org/10.1109/CENIM51130.2020.9297970

[24] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, 2020, pp. 1–7. https://doi.org/10.1109/CIBCB48159.2020.9277638

[25] Z. Liang, Z. Zhang, M. Zhang, X. Zhao, and S. Pu, "RangeIoUDet: Range Image based Real-Time 3D Object Detector Optimized by Intersection over Union," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 7136–7145. https://doi.org/10.1109/CVPR46437.2021.00706

[26] Z. Krawczyk, J. S. ´NSKI, J. Starzyński, B. Pol, and A. Tech, "Segmentation of bone structures with the use of deep learning techniques," *Bull. Polish Acad. Sci. Tech. Sci.*, 2023 https://doi.org/10.3390/app14177557

[27] V. Rajput, "Robustness of different loss functions and their impact on networks learning capability," *ArXiv*, vol. abs/2110.0, 2021 https://doi.org/10.2139/ssrn.4065778

[28] Z. Liu, Y. Wang, K. Han, W. Zhang, S. Ma, and W. Gao, "Post-Training Quantization for Vision Transformer," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. W. Vaughan, Eds., Curran Associates, Inc., 2021, pp. 28092–28103. https://doi.org/10.48550/arXiv.2106.14156

[29] A. Shridhar, P. Tomson, and M. Innes, "Interoperating Deep Learning models with ONNX.jl," *Proc. JuliaCon Conf.*, vol. 1, no. 1, p. 59, 2020 https://doi.org/10.21105/jcon.00059

[30] E. Jeong, J. Kim, and S. Ha, "TensorRT-Based Framework and Optimization Methodology for Deep Learning Inference on Jetson Boards," *ACM Trans. Embed. Comput. Syst.*, vol. 21, no. 5, Oct. 2022 https://doi.org/10.1145/3508391

[31] L. Stäcker *et al.*, "Deployment of Deep Neural Networks for Object Detection on Edge AI Devices with Runtime Optimization," in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021, pp. 1015–1022. https://doi.org/10.1109/ICCVW54120.2021.00118

[32] C. Gernigon, S.-I. Filip, O. Sentieys, C. Coggiola, and M. Bruno, "Low-Precision Floating-Point for Efficient On-Board Deep Neural Network Processing," in *2023 European Data Handling & Data Processing Conference (EDHPC)*, 2023, pp. 1–8. https://doi.org/10.23919/EDHPC59100.2023.10396014

[33] T. Yu *et al.*, "Collage: Light-Weight Low-Precision Strategy for LLM Training," in *Proceedings of the 41st International Conference on Machine Learning*, R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, Eds., in Proceedings of Machine Learning Research, vol. 235. PMLR, Sep. 2024, pp. 57459–57479. [Online]. Available: https://proceedings.mlr.press/v235/yu24d.html

[34] H. Ren, S. Duca, and N. D'Souza, "Comparison of Open Neural Network Exchange (ONNX) and TensorFlow based inferences for the B-scan of interest algorithm," *Invest. Ophthalmol. Vis. Sci.*, vol. 64, no. 8, p. 213, Jun. 2023.

[35] J. Suo, X. Zhang, S. Zhang, W. Zhou, and W. Shi, "Feasibility Analysis of Machine Learning Optimization on GPU-based Low-cost Edges," in *2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*, 2021, pp. 89–96. https://doi.org/10.1109/SWC50871.2021.00022

[36] M. Dörrich, M. Fan, and A. M. Kist, "Impact of Mixed Precision Techniques on Training and Inference Efficiency of Deep Neural Networks," *IEEE Access*, vol. 11, pp. 57627–57634, 2023 https://doi.org/10.1109/ACCESS.2023.3284388

[37] D. Liu, M. Jiang, and K. Pister, "LLMEasyQuant – An Easy to Use Toolkit for LLM Quantization," 2024. [Online]. Available: https://arxiv.org/abs/2406.19657

[38] C. Devaguptapu, D. Agarwal, G. Mittal, and V. N. Balasubramanian, "An Empirical Study on the Robustness of NAS based Architectures," *CoRR*, vol. abs/2007.08428, 2020 https://doi.org/10.48550/arXiv.2406.19657

[39] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A White Paper on Neural Network Quantization," *CoRR*, vol. abs/2106.08295, 2021 https://doi.org/10.48550/arXiv.2406.19657

