Optimizing U-Net Performance: Evaluation of State-of-the-Art Encoders and Optimizers on Human Segmentation Dataset.

Mohammad Omar Faruk[1,*], Mohammad Anwar Hosen[1], Michael Johnstone[1], and Aziz Ashfak[2]

***Abstract*— This study investigates the impact of various encoders and optimizers on the performance of U-Net-based models for human segmentation tasks. A comprehensive evaluation was conducted using multiple EfficientNet, ResNet, and MobileNet encoders in combination with ten different optimization algorithms. The performance was assessed using standard metrics such as validation/train loss and Intersection over Union (IoU). Among the tested configurations, lightweight encoders such as EfficientNet-B0, EfficientNet-B1, and EfficientNet-Lite0 demonstrated superior segmentation capabilities, with EfficientNet-B0 paired with the Adamax optimizer achieving the best performance, recording a validation IoU of 0.8507 and a training IoU of 0.8914. Statistical analyses, including ANOVA, confirmed significant differences in segmentation performance across encoders but not across optimizers, suggesting that encoder architecture plays a more pivotal role in accuracy. These results underscore the importance of selecting efficient encoder backbones over traditional deep CNNs for optimal performance in human segmentation tasks.**

I. Introduction

Image segmentation is a fundamental task in computer vision that involves partitioning an image into meaningful regions, often at the pixel level. It enables precise localization and classification of objects within a scene, making it essential for applications such as medical imaging, autonomous driving, and surveillance. Among its applications, human segmentation is particularly important for understanding and interpretation of human figures, behaviors, and body parts, which has important applications in fields such as surveillance, human-computer interaction, healthcare, and robotics [1]. Semantic operates at the pixel level, classifying each individual. By pixel-level labels to different objects of humans, semantic segmentation authorizes decision-making systems to ensure dynamic scenarios [2]. Among the numerous architectures proposed, U-Net has gained prominence for its symmetric encoder-decoder structure, which effectively captures contextual information through a contracting path and recovers spatial precision using a corresponding expansive path with skip connections [3].

Human semantic segmentation requires pixel-level labeled datasets and a robust encoder-decoder architecture, such as U-Net [4]. The integration of advanced encoder backbones—including EfficientNet-B0, EfficientNet-B1, EfficientNet-Lite0, EfficientNet-B6, MobileNetV2, ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152 enhances both the efficiency and performance of the model. This encoder-decoder framework enables end-to-end learning by combining high-level abstract feature extraction with precise spatial reconstruction, making it particularly effective in handling the challenges posed by diverse human poses, occlusions, and complex backgrounds. The human segmentation dataset used in this study offers pixel-wise ground truth mask labels, providing a rich and detailed representation of various human segmentation scenarios [5].

Recent research emphasizes the need for efficient encoder architectures to enable real-time human segmentation on resource-constrained devices [6]. For example, Zhang et al. employed a U-Net variant integrated with MobileNetV2, demonstrating competitive performance on human parsing datasets while significantly reducing the parameter count and inference time [7]. In another study, You, D, et al., [8] proposed an EfficientNet-B0-based segmentation network that maintained high segmentation accuracy across multiple human-centric datasets, despite a drastically reduced computational footprint.

Despite these advancements, achieving an optimal trade-off between segmentation accuracy and computational efficiency remains a significant challenge, especially in unconstrained human-centric environments [9]. These studies highlighted the feasibility of using compact encoders without severely compromising performance. However, the adaptability of such models across diverse human segmentation datasets, particularly those with varying image resolutions and pose diversity, still warrants further investigation.

The human segmentation dataset captures a variety of human activity scenarios, offering a valuable benchmark for evaluating model performance in real-world conditions. The dataset, curated primarily consists of images sourced from Google Image Search. Although relatively small in size, the dataset is well-suited for encoder-decoder architectures such as U-Net due to its high-resolution images and detailed pixel-wise annotations. These characteristics make it particularly valuable for addressing the challenges associated with human image segmentation tasks.

This paper investigates the performance of ten encoder backbones—EfficientNet-B0, EfficientNet-B1, EfficientNet-Lite0, EfficientNet-B6, MobileNetV2, ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152—in combination with ten optimizers: SGD, AdaGrad, RMSprop, AdaDelta, AdamW, NAdam, RAdam, Adamax, and ASGD. Each encoder-optimizer pair is evaluated over 100 training iter-

[1] Institute for Intelligent Systems Research and Innovation, Deakin University, Waurn Ponds, Australia. Emails: {o.faruk, anwar.hosen, michael.johnstone}@deakin.edu.au

[2] Noakhali Science and Technology University, Noakhali, Bangladesh. Email: aziz1516@student.nstu.edu.bd

*Corresponding author: Mohammad Omar Faruk (o.faruk@deakin.edu.au)

ations to analyze the model's behavior in terms of loss and Intersection over Union (IoU) metrics. The goal of this study is to identify the most effective combinations of encoders and optimizers for achieving optimal performance on the human segmentation dataset.

II. Literature Review

A. Image Segmentation Models

Image segmentation is a critical process in computer vision, aimed at assigning semantic labels to individual pixels in an image, thereby enabling the extraction of precise object boundaries. Traditional approaches such as thresholding, clustering, and region-growing techniques have been progressively replaced by deep learning-based methods due to their superior performance and generalization capabilities. Fully Convolutional Networks (FCNs) pioneered end-to-end learning for pixel-wise classification and laid the foundation for later advancements in semantic segmentation models [10].

More recently, architectures such as DeepLab [11], PSPNet [12], and SegNet [13] have contributed to improved spatial accuracy and multi-scale context awareness. These models incorporate innovations like atrous convolutions, pyramid pooling, and encoder-decoder structures. Moreover, transfer learning has emerged as a valuable approach by utilizing pretrained backbones (e.g., ResNet, EfficientNet) to extract robust feature representations for segmentation tasks, especially in limited data scenarios. Studies have also shown that the choice of encoder significantly influences model performance, particularly in dense prediction tasks such as human segmentation [14].

B. U-Net Model

The U-Net architecture, initially proposed for biomedical image segmentation [15], has gained popularity across diverse domains due to its symmetric encoder-decoder structure and efficient use of skip connections. These skip connections help preserve spatial information lost during downsampling and allow for accurate reconstruction of high-resolution segmentation masks. The original U-Net model employs a relatively shallow encoder, but subsequent adaptations have integrated more powerful backbones like EfficientNet, MobileNet, and ResNet to enhance feature extraction and semantic richness [10] [16].

Recent implementations, such as those based on the segmentation-models.pytorch (SMP) library, offer modular flexibility in selecting encoders and customizing architectures. This has facilitated comparative research evaluating the trade-offs between model complexity, accuracy, and training efficiency across different U-Net variants. The architecture's compatibility with transfer learning and various loss functions (e.g., Dice Loss, BCEWithLogits) makes it especially suitable for binary segmentation problems like human figure detection in complex backgrounds.

C. Statistical Analysis in Image Segmentation

While most segmentation studies focus on model architecture and performance metrics, statistical evaluation methods such as Analysis of Variance (ANOVA) are increasingly used to assess the significance of performance differences across experimental configurations [17] [18]. These techniques help determine whether improvements in segmentation accuracy are due to genuine architectural superiority or random variations.

In segmentation studies involving multiple model configurations (e.g., different encoders and optimizers), ANOVA provides a formal way to compare groups and validate conclusions beyond simple metric comparison. For instance, significant F-values and low p-values (e.g., $p < 0.001$) indicate that the encoder or training configuration meaningfully impacts the model's effectiveness. Such statistical insights not only strengthen the experimental claims but also guide future research in selecting optimal architectures for specific segmentation tasks [19].

III. Methodology

A. Study Design

This study presents a systematic evaluation of performance optimization strategies for the U-Net architecture in human segmentation tasks. Specifically, it investigates the impact of ten state-of-the-art encoder backbones and ten distinct optimization algorithms on segmentation accuracy and training efficiency. A benchmark dataset was first selected to facilitate a thorough exploration of encoder and optimizer configurations within the U-Net framework. The dataset underwent extensive preprocessing to ensure compatibility with the model inputs.

For model construction, ten advanced encoder architectures were integrated into the U-Net design. To further enhance training performance, ten widely used and effective optimizers were employed. During model compilation, appropriate learning rates and a consistent loss function were applied to maintain experimental consistency and fairness across all configurations. Each model variant was trained for 100 epochs to ensure reliable performance evaluation.

The overall methodology is organized into the following key components: (i) Dataset Selection, (ii) Data Preprocessing, (iii) Encoder Integration, (iv) Optimizer Configuration, (v) Model Architecture, (vi) Experimental Setup, and (vii) Statistical Analysis. This comprehensive framework enables a robust assessment of how encoder and optimizer choices affect U-Net performance in human segmentation applications.

B. Dataset

The human segmentation dataset captures a range of human activity scenarios and provides high-resolution ground truth images along with comprehensive segmentation masks. It contains a total of 290 ground truth images and corresponding mask images. Additionally, a CSV file is included to store the file paths of each image-mask pair, facilitating efficient data loading and preprocessing. Due to its image

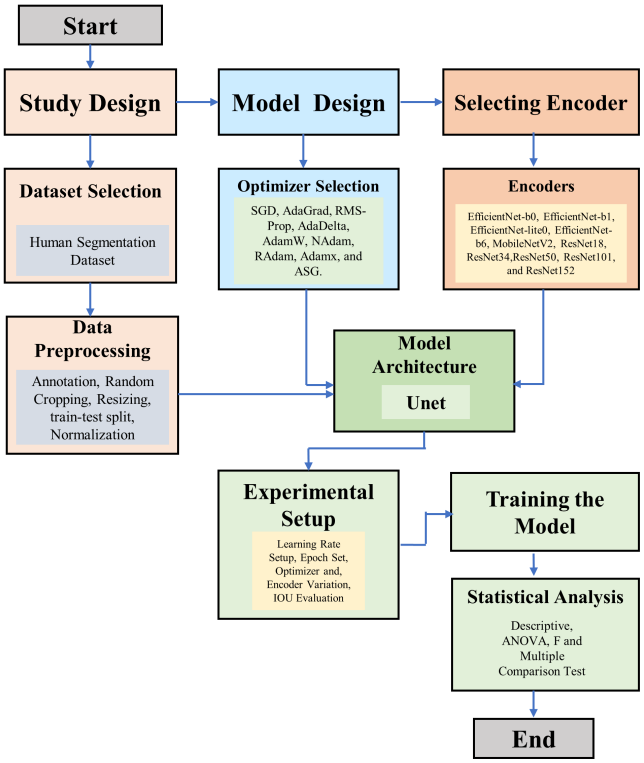

Fig. 1. The flow diagram of the study design.

quality and annotation precision, the dataset is well-suited for encoder-decoder architectures, particularly U-Net. The dataset is publicly available on GitHub and serves as a useful resource for benchmarking human segmentation models.

C. Data Preprocessing

The dataset is structured to support semantic segmentation tasks, comprising original RGB images, corresponding ground mask images, and a CSV file that specifies the file paths for all annotated data. Upon reading the CSV file, both the ground truth and mask images are loaded into memory. The ground truth images are subsequently converted into a format compatible with the model's training requirements. During the preprocessing phase, all images were resized to 320×320 pixels, with a batch size of 16. Out of a total of 290 images, 232 were allocated for training and 58 for testing. Data augmentation was applied exclusively to the training set, where 50% of the images underwent horizontal and vertical flipping. Additionally, RandomResizedCrop was employed with a target resolution of 256×256 pixels. No augmentation techniques were applied to the validation set. Finally, normalization was performed on all images in both the training and validation datasets to ensure consistency in the input distribution.

D. Encoders

Encoders serve a critical function in semantic segmentation by transforming raw input data into abstract, high-dimensional feature representations [20]. Their architectural robustness directly affects the quality of feature extraction, enabling the encoding of both spatial structures and semantic key components for precise pixel-wise classification in downstream segmentation tasks [21].

This study investigates a diverse set of encoder backbones, including EfficientNet-b0, EfficientNet-b1, EfficientNet-lite0, EfficientNet-b6, MobileNetV2, ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152.

Within the EfficientNet series, B0 and B1 are optimized for lightweight segmentation, offering an effective balance between accuracy and efficiency. EfficientNet-B6 provides deeper hierarchical features suited for more complex segmentation, while EfficientNet-lite0 is tailored for deployment in resource-constrained environments such as mobile devices. MobileNetV2 introduces architectural innovations such as inverted residuals and linear bottlenecks, which significantly reduce computational overhead—making it suitable for real-time and edge-based applications [22]. The ResNet family offers a scalable range of models: ResNet-18 and ResNet-34 support faster training and capture low- to mid-level features, whereas deeper variants like ResNet-50, ResNet-101, and ResNet-152 integrate bottleneck layers to extract rich, multi-scale features, enhancing performance in cluttered or high-complexity scenes [23]. The details of each of the encoders is discussed below:

EfficientNet is a family of convolutional neural networks (CNNs) designed for image classification and other visual recognition tasks [24]. The key motivation behind EfficientNet is to achieve a better trade-off between accuracy and computational efficiency, making it suitable for a wide range of devices and applications, from mobile devices to large-scale cloud computing systems. There are three types of layers such as: Stem, MBConv Blocks, and Head [25].

- **Stem:** Conv 3×3, 32 filters, stride 2 + BN (Batch Normalization) + ReLU6
- **MBConv Blocks:** 7 stages of MBConv (depthwise separable conv + SE + expansion) with varying kernel sizes, strides, and expansion ratios [26]
- **Head:** Conv 1×1 (1280 channels) + Global Avg Pool + Dense layer with SoftMax

EfficientNet models are popular for their accuracy-to-efficiency ratio, especially suited for both cloud-based and edge deployments.

EfficientNet-B0: EfficientNet-B0 serves as the baseline in the EfficientNet family. The full architecture has about 82 layers in total (counting convolution, batch norm, activations, etc.) and has 5.3M parameters [27]. The input size of the image is 224×224. Metrics values of this encoder: Top-1 Accuracy (ImageNet): ∼77.1% and Top-5 Accuracy (ImageNet): ∼93.3%. EfficientNet-B0 is slightly slower but more accurate [28].

EfficientNet-B1: EfficientNet-B1 is a scaled-up version of EfficientNet-B0, introduced by Tan and Le in *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (ICML 2019)* [29]. The full architecture has about 88 layers in total (counting convolution, batch norm, activations, etc.) and has approximately 7.8M parameters and the input size of the image is 240×240 [30]. Metrics values of this

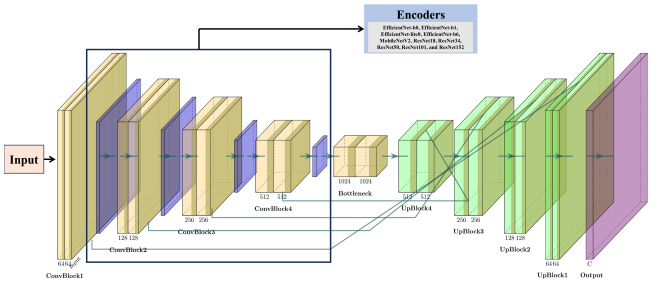

Fig. 2. Architecture of the U-Net model incorporating a range of encoder blocks.

encoder: Top-1 Accuracy (ImageNet): ∼79.1% and Top-5 Accuracy (ImageNet): ∼94.4% [29]. EfficientNet-B1 offers better accuracy than B0 with only a slight increase in compute, making it a strong candidate for balanced efficiency and performance.

EfficientNet-Lite0: The full architecture has about 78 layers in total (counting convolution, batch norm, activations, etc.) and has approximately 4.7M parameters and input size of the image is 224×224 with metrics values of this encoder: Top-1 Accuracy (ImageNet): ∼75.3%. Top-5 Accuracy (ImageNet): ∼92.3%. EfficientNet-Lite0 is designed for fast and efficient inference on edge devices.

EfficientNet-B6: EfficientNet-B6 is a much deeper and wider network. The full architecture has about 98 layers in total (counting convolution, batch norm, activations, etc.) and has approximately 43M parameters and the input size of the image is 528×528 with metrics values of this encoder: Top-1 Accuracy (ImageNet): ∼84.0% and Top-5 Accuracy (ImageNet): ∼96.7% [31]. EfficientNet-B6 offers significantly higher accuracy compared to smaller variants like B0 and B1, but at the cost of increased computation and memory usage [32]. It is best suited for high-performance vision tasks on powerful GPUs or server environments.

MobileNetV2: MobileNetV2 is a lightweight convolutional neural network (CNN) architecture developed by Google, designed specifically for mobile and embedded vision applications [33]. It builds on the original MobileNet by introducing inverted residuals and linear bottlenecks, which significantly improve efficiency without sacrificing accuracy and the full architecture has about 88 layers in total (counting convolution, batch norm, activations, etc.) and has approximately 3.4M parameters [34]. The input size of the image is 224×224. Metrics values of this encoder: Top-1 Accuracy (ImageNet): ∼71.8% and Top-5 Accuracy (ImageNet): ∼91.0% [34].

ResNet is a deep convolutional neural network architecture introduced by Kaiming He et al. in 2015, designed to enable the training of extremely deep networks by addressing the vanishing gradient problem through residual learning and skip connections [35].

ResNet18: ResNet18 is the smallest model in the ResNet family, using basic residual blocks [36]. It offers a good balance between model simplicity and performance, making it suitable for baseline experiments, fast training, and deployment in low-resource environments. The full architecture has about 71 layers in total (counting convolution, batch norm, activations, etc.) and has approximately 11.7M parameters with input size of the image is 224×224 and the metrics values of this encoder: Top-1 Accuracy (ImageNet): ∼69.8% and Top-5 Accuracy (ImageNet): ∼89.1% [37].

ResNet34: ResNet34 deepens the architecture compared to ResNet18, enabling improved feature representation while maintaining the simplicity of basic residual blocks. It strikes a strong balance between computational efficiency and performance, often used for segmentation and classification in mid-scale applications. The full architecture has about 95 layers in total (counting convolution, batch norm, activations, etc.) and has approximately 21.8M parameters with the input size of the image is 224×224 and the metrics values of this encoder: Top-1 Accuracy (ImageNet): ∼73.3% and Top-5 Accuracy (ImageNet): ∼91.4% [38].

ResNet50: ResNet50 introduces bottleneck residual blocks, allowing the network to go deeper without significantly increasing computational cost [39]. The full architecture has about 177 layers in total (counting convolution, batch norm, activations, etc.) and has approximately 25.6M parameters with the input size of the image is 224×224 and the metrics values of this encoder: Top-1 Accuracy (ImageNet): ∼76.2% and Top-5 Accuracy (ImageNet): ∼92.8% [40].

ResNet101: ResNet101 deepens the network further using bottleneck residual blocks, providing enhanced representational power and better feature hierarchies [41].The full architecture has about 344 layers in total (counting convolution, batch norm, activations, etc.) and has approximately 44.5M parameters with the input size of the image is 224×224 and metrics values of this encoder: Top-1 Accuracy (ImageNet): ∼77.4% and Top-5 Accuracy (ImageNet): ∼93.5% [42].

ResNet152: ResNet152 is the deepest model in the original ResNet family, offering very strong feature extraction capabilities due to its depth and bottleneck architecture [43]. The full architecture has about 475 layers in total (counting convolution, batch norm, activations, etc.) and has approximately 60.2M parameters and the input size of the image is 224×224. with the metrics values of this encoder: Top-1 Accuracy (ImageNet): ∼78.3% and Top-5 Accuracy (ImageNet): ∼94.1% [44].

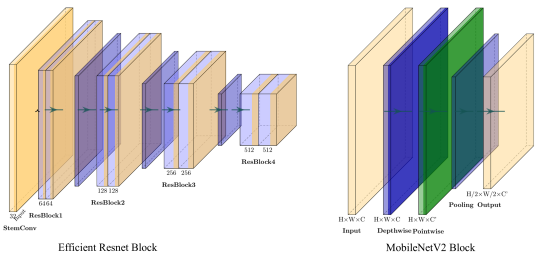

Fig. 3. Presentation of the Efficient-ResNet and MobileNet encoder blocks.

E. Optimizers

Optimizers play a vital role in neural network training by minimizing the loss function and enhancing model accuracy [45]. They are among the most influential elements in fine-tuning performance, directly impacting convergence speed and generalisation. In this study, we experimented with ten widely used optimisation algorithms: SGD, AdaGrad, RMS-Prop, AdaDelta, AdamW, NAdam, RAdam, Adamx, and ASG. Since the effectiveness of each optimiser is closely tied to the choice of learning rate, a constant learning rate of 0.03 was maintained across all experiments to ensure consistency in evaluation.

SGD is a frequent update of model parameters and requires less memory, but has high variance [46]. In AdaGrad, the learning rate changes adaptively with iterations, but when the learning rate becomes a very small number, it will cause the dead neuron problem [47]. RMS-Prop is a special version of Adagrad where the learning rate gets adjusted automatically, but it is slow in learning. AdaDelta is an extension of Adagrad, where it does not need to set a default learning rate. Adam Optimizer is the most popular and famous gradient descent optimisation algorithm. It is easy to implement, computationally efficient, and has few memory requirements. AdamW is a variant of Adam that decouples weight decay and offers better generalization than Adam. NAdam incorporates Nesterov momentum into Adam and achieves faster convergence than Adam in some cases. RAdam rectifies the variance of adaptive learning rates in the early training stages, but it has a higher computational cost. Adamx is a variant of Adam based on the infinity norm And it is less commonly used. ASG performs stochastic updates, but averages the parameters over time.

F. Model Architecture

In this study, we adopt a deep learning architecture based on the U-Net framework, implemented using Pytorch for the task of semantic segmentation. The model is designed to predict binary masks with high spatial fidelity from RGB input images [**?**]. The model leverages modern convolutional neural network techniques to achieve precise pixel-wise segmentation. We integrate ten robust encoder backbones in our model architectures, such as: EfficientNet-b0, EfficientNet-b1, EfficientNet-lite0, EfficientNet-b6, MobileNetv2, ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152. The encoder is responsible for generating deep feature representations of the input images, while the decoder reconstructs precise pixel-wise segmentation masks from these features.

G. Experimental Setup

The human segmentation experiments were conducted using a custom-built deep learning model implemented in PyTorch. The architecture was based on the U-Net framework, widely recognized for its effectiveness in biomedical and semantic segmentation tasks. The U-Net backbone was dynamically configured using the segmentation_models.pytorch (smp) library, allowing for modular experimentation with different encoder architectures [48].

The model, defined in the Segmentation model class, instantiated a U-Net architecture with a specified encoder, passed via the global ENCODER variable. Each encoder was initialized with pretrained weights from ImageNet, controlled by the WEIGHTS parameter. The model accepted RGB input images (in_channels=3) and produced a single-channel output (classes=1) corresponding to binary segmentation masks.

During the forward pass, the model computed output logits using the selected U-Net configuration. When ground truth masks were available during training, the output was passed through two complementary loss functions: the Dice Loss (for overlap-based segmentation accuracy) and Binary Cross-Entropy with Logits Loss (BCEWithLogitsLoss), which captures pixel-wise classification performance. The total training loss was computed as the sum of these two losses, enabling the model to optimize both regional accuracy and pixel-level confidence. During inference (i.e., when no masks were provided), the model simply returned the raw segmentation logits for further post-processing or evaluation.

All experiments were conducted on a CUDA-enabled GPU, ensuring efficient training across the dataset. The Human Segmentation dataset was used, with training metadata specified in the CSV file. The training configuration included the following hyperparameters: image size of 320×320 pixels, batch size of 16, learning rate (LR) of 0.003, and training over 25 epochs. The model was trained and evaluated using this fixed configuration while systematically varying the encoder and optimizer combinations for comparative analysis.

H. Statistical Analysis

To evaluate the influence of different encoder and optimizer combinations on human segmentation performance, a total of 100 experiments were conducted. These experiments tested 10 encoder architectures—including EfficientNet-B0, B1, B6, Lite0, MobileNetV2, and ResNet-18, 34, 50, 101, and 152—each combined with 10 different optimizers (Adam, RMSprop, Adadelta, Adagrad, SGD, Adamax, ASGD, AdamW, NAdam, and RAdam). All encoder variants were initialized with ImageNet-pretrained weights. Training was performed on a consistent human segmentation dataset, with performance measured using validation loss, validation Intersection over Union (IoU), training loss, and training IoU.

In the first phase of analysis, descriptive statistics were computed to provide an overall view of model performance. Mean values of each metric were calculated separately for every optimizer and encoder. This summary helped identify general trends, such as which encoders consistently yielded lower validation loss or higher IoU across various optimizers. The computed means were compiled into comparative tables, allowing clear benchmarking between methods.

To further visualize these trends, a series of graphical representations were created. These included bar plots comparing the average validation loss, training loss, validation IoU, and training IoU for all encoders and optimizers.

Figures 4 through 8 illustrate these comparisons across models. These visualizations provided intuitive insights into model behavior—highlighting, for instance, the consistent superiority of EfficientNet-B0, Lite0, and B1 variants in both validation and training phases, and the relatively weaker performance of deeper ResNet models like ResNet-101 and ResNet-152.

In the second phase, inferential statistical analysis was performed using one-way ANOVA. This test assessed whether the differences in performance metrics across groups were statistically significant. When comparing encoders, the ANOVA revealed significant effects on all four metrics ($p < 0.001$), indicating that the encoder architecture has a substantial impact on segmentation outcomes. In contrast, the optimizer comparison yielded non-significant p-values (p ¿ 0.05), suggesting that the choice of optimizer had minimal influence when the encoder was held constant.

IV. Result and Discussion

The results section presents the findings from extensive experiments conducted to evaluate the performance of various encoder and optimizer combinations for human segmentation. Performance was assessed using key metrics such as validation loss, training loss, validation IoU, and training IoU. Both descriptive statistics and inferential analyses were applied to interpret the outcomes. Graphs and tables summarize the results for clarity. The following subsections detail the statistical insights and visual patterns observed during the experiments.

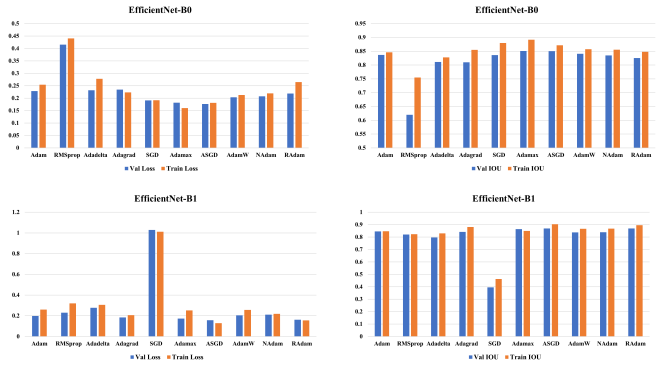

Fig. 4. Comparison of loss and mIoU over train and validation datasets of the model when incorporating the encoders EfficientNet-B0 and EfficientNet-B1 .

Figure 4 - Figure 8 present the performance evaluation of the U-Net model using different encoders in combination with ten different optimization algorithms. The figures reports the final training and validation loss values, along with the corresponding Intersection over Union (IoU) scores, for each optimizer. The results reveal noticeable variation in performance across optimizers, despite identical architectural settings.

Figure 4 illustrates the segmentation performance of the U-Net model using two lightweight backbones—EfficientNet-B0 and EfficientNet-B1. For EfficientNet-B0, the best validation IoU is achieved by Adamax (0.851), followed closely by ASGD (0.850) and SGD (0.836), with Adamax also attaining the highest training IoU (0.891). RMSprop yields the weakest performance in terms of both validation IoU (0.620) and training IoU (0.755). For EfficientNet-B1, the optimizers ASGD and RAdam clearly outperform others, achieving the highest validation IoUs of 0.869 and 0.869 respectively, and training IoUs above 0.89. In contrast, SGD performs poorly with the lowest validation and training IoUs. Overall, the results highlight that combining an EfficientNet encoder with advanced optimizers like ASGD, Adamax, or RAdam significantly enhances segmentation accuracy while maintaining model efficiency.

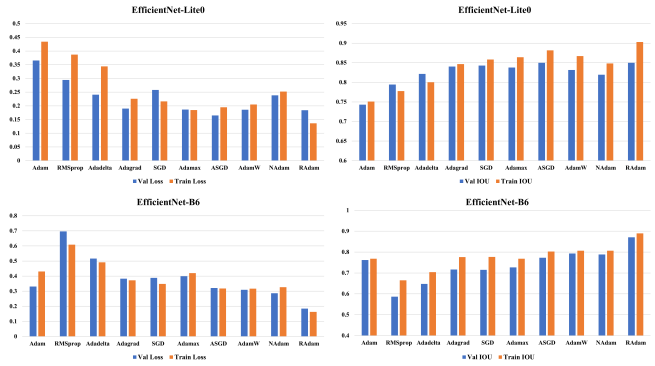

Fig. 5. Comparison of loss and mIoU over train and validation datasets of the model when incorporating the encoders EfficientNet-Lite0 and EfficientNet-B6 .

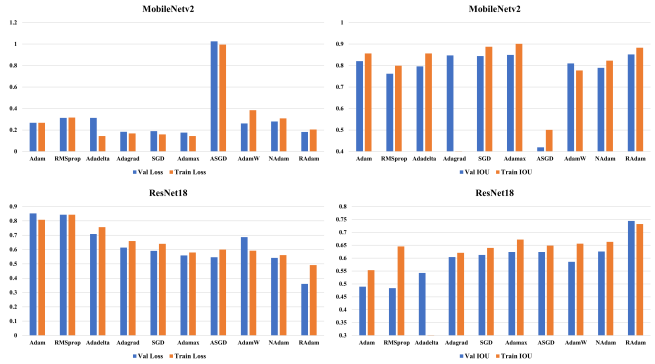

Fig. 6. Comparison of loss and mIoU over train and validation datasets of the model when incorporating the encoders MobileNetV2 and ResNet-18.

Figure 5 summarizes both EfficientNet-Lite0 and EfficientNet-B6 encoder configurations, pre-trained on ImageNet and evaluated with various optimizers, notable differences in performance were observed. For EfficientNet-Lite0 with 4 million parameters, RAdam achieved the best validation IoU (0.8498) and the highest training IoU (0.9028), accompanied by a low training loss of 0.1364. Other optimizers such as ASGD and Adamax also demonstrated strong performance, achieving validation IoUs above 0.83. In contrast, EfficientNet-B6 and its TIMM variant, despite having significantly more parameters (40 million), only showed superior performance when optimized with RAdam, reaching a higher validation IoU

of 0.8705 and a training IoU of 0.8895. While several other optimizers (e.g., AdamW, NAdam, and ASGD) also yielded validation IoUs above 0.79 with EfficientNet-B6, the increase in model complexity did not consistently translate to proportional gains in performance. This indicates that RAdam provided the most stable and effective optimization across both lightweight and large-scale models, suggesting its robustness in training deep encoder architectures for semantic segmentation tasks.

The Figure 6 presents the performance of MobileNetV2 and ResNet18 encoders, pretrained on ImageNet and integrated with ten different optimizers, was thoroughly evaluated. Among the MobileNetV2 configurations, RAdam achieved the highest validation IoU of 0.8522 and a corresponding low validation loss of 0.1821, followed closely by Adamax and Adagrad with validation IoUs of 0.8499 and 0.8476, respectively. Conversely, ASGD resulted in significantly lower segmentation performance with a validation IoU of just 0.4191. For ResNet18, the combination with RAdam again yielded the best result, producing a validation IoU of 0.7443 and the lowest validation loss of 0.3605, while ASGD, SGD, and Adamax also demonstrated moderate performance improvements. In general, the results indicate that RAdam and Adamax provide notable improvements in segmentation accuracy, especially when paired with lightweight encoders like MobileNetV2.

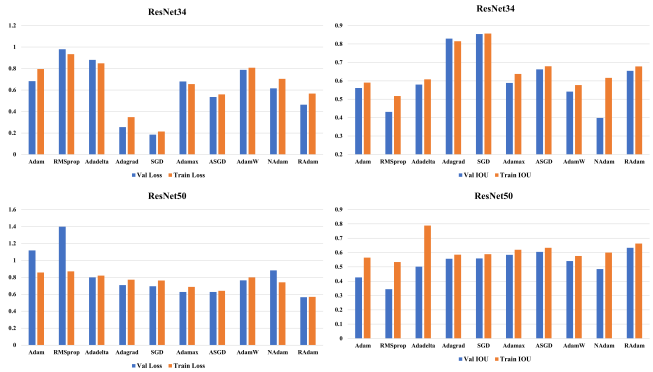

Fig. 7. Comparison of loss and mIoU over train and validation datasets of the model when incorporating the encoders ResNet-34 and ResNet-50.

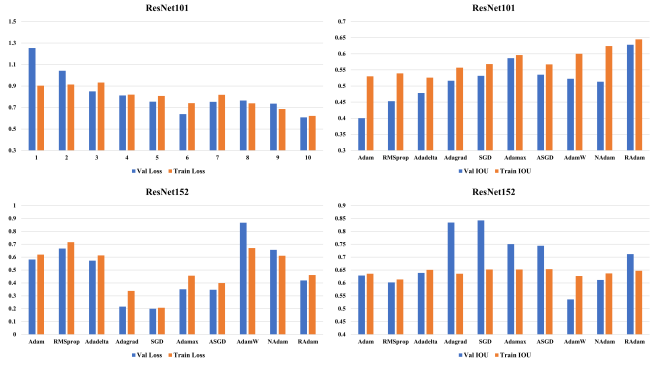

Fig. 8. Comparison of loss and mIoU over train and validation datasets of the model when incorporating the encoders ResNet-101 and ResNet-152.

Figure 7 shows the experimental results for ResNet34 and ResNet50 encoders. For ResNet34, the SGD optimizer achieved the highest validation IoU of 0.8545 with the lowest validation loss of 0.1851, indicating superior generalization. Adagrad and ASGD also performed well with validation IoUs of 0.8291 and 0.6619, respectively. In contrast, optimizers like NAdam and RMSprop yielded lower performance, with NAdam showing a validation IoU of just 0.3988. Similarly, for ResNet50, RAdam produced the best results with a validation IoU of 0.6337 and a validation loss of 0.5661, followed closely by ASGD and Adamax. While Adadelta and SGD maintained moderate effectiveness, RMSprop and NAdam exhibited the weakest segmentation accuracy. Overall, ResNet34 with SGD and ResNet50 with RAdam or ASGD emerge as the most promising combinations, highlighting the impact of optimizer choice on model performance.

The performance evaluation of the ResNet-101 and ResNet-152 encoders has been presented in Figure 8. Among the ResNet-101 experiments, RAdam achieved the best results with a validation loss of 0.6079 and the highest validation IoU of 0.6279, along with the lowest training loss (0.6227) and the highest training IoU (0.6449), indicating its strong generalization and fitting capabilities. Similarly, for ResNet-152, the SGD optimizer stood out by yielding the lowest validation loss of 0.1993 and the highest validation IoU of 0.8425, coupled with an exceptionally low training loss (0.2071) and strong training IoU (0.6523), outperforming other optimizers. In general, the ResNet-152 encoder consistently achieved superior segmentation performance compared to ResNet-101 across most optimizers, particularly with Adagrad, SGD, and Adadelta, suggesting that deeper architectures can leverage optimization techniques more effectively for semantic segmentation tasks.

Table I provides a detailed statistical comparison of the training and validation loss, along with the corresponding Intersection over Union (IoU) metrics, for different encoder architectures used within the U-Net model. The table includes results for several encoder families, namely EfficientNet (b0, b1, b6, and lite0), MobileNet, and ResNet (18, 34, 50, 101, and 152). For each encoder, descriptive statistics including mean, standard deviation, minimum, maximum, skewness, and kurtosis are reported for both loss and IoU on training and validation datasets. The findings indicate that EfficientNet variants, particularly EfficientNet-b0 and EfficientNet-lite0, generally achieve lower loss values and higher IoU scores, suggesting more effective and stable segmentation performance. MobileNet also demonstrates strong results with competitive IoU and relatively low loss. In contrast, deeper ResNet encoders such as ResNet-101 and ResNet-152 show higher validation loss and lower IoU on average, along with greater skewness and kurtosis in some cases, indicating less stable training behavior and potential overfitting. These results emphasize the effectiveness of lightweight and computationally efficient encoders in enhancing U-Net's segmentation capabilities while maintaining generalization performance.

TABLE I

COMPARISON OF TRAINING/VALIDATION LOSS AND IOU STATISTICS ACROSS DIFFERENT ENCODERS OF THE UNET MODEL

Encoder	Statistic	Train Loss	Val Loss	Train IOU	Val IOU	Encoder	Statistic	Train Loss	Val Loss	Train IOU	Val IOU
EfficientNet-b0	Mean	0.243	0.259	0.849	0.812	ResNet-18	Mean	0.653	0.630	0.670	0.594
	Std. Dev.	0.079	0.162	0.038	0.069		Std. Dev.	0.114	0.149	0.081	0.076
	Min	0.160	0.176	0.755	0.620		Min	0.492	0.360	0.553	0.484
	Max	0.441	0.716	0.891	0.851		Max	0.843	0.852	0.845	0.744
	Skewness	1.937	3.066	-1.862	-2.926		Skewness	0.549	-0.007	1.176	0.287
	Kurtosis	4.710	9.560	4.619	8.888		Kurtosis	-0.761	0.133	2.637	0.872
EfficientNet-b1	Mean	0.311	0.282	0.823	0.798	ResNet-34	Mean	0.643	0.706	0.658	0.610
	Std. Dev.	0.253	0.265	0.129	0.143		Std. Dev.	0.227	0.409	0.106	0.148
	Min	0.129	0.156	0.463	0.396		Min	0.214	0.185	0.518	0.399
	Max	1.012	1.029	0.903	0.869		Max	0.934	1.616	0.857	0.855
	Skewness	2.831	3.059	-2.923	-3.023		Skewness	-0.765	1.069	0.948	0.415
	Kurtosis	8.527	9.509	8.907	9.343		Kurtosis	-0.097	1.996	0.286	-0.289
EfficientNet-lite0	Mean	0.258	0.291	0.840	0.823	ResNet-50	Mean	0.752	0.819	0.615	0.524
	Std. Dev.	0.097	0.209	0.048	0.033		Std. Dev.	0.096	0.257	0.071	0.087
	Min	0.136	0.165	0.751	0.743		Min	0.570	0.566	0.534	0.344
	Max	0.434	0.858	0.903	0.850		Max	0.871	1.398	0.788	0.634
	Skewness	0.840	2.693	-0.792	-1.905		Skewness	-0.702	1.526	1.761	-0.962
	Kurtosis	-0.491	7.668	-0.296	3.832		Kurtosis	-0.125	1.991	3.937	0.683
EfficientNet-b6	Mean	0.380	0.382	0.776	0.738	ResNet-101	Mean	0.799	1.221	0.575	0.487
	Std. Dev.	0.119	0.140	0.061	0.080		Std. Dev.	0.102	1.422	0.040	0.144
	Min	0.163	0.185	0.665	0.586		Min	0.623	0.608	0.526	0.100
	Max	0.609	0.696	0.890	0.871		Max	0.933	5.254	0.645	0.628
	Skewness	0.228	1.229	-0.146	-0.424		Skewness	-0.271	3.121	0.452	-2.469
	Kurtosis	1.249	2.310	1.263	0.630		Kurtosis	-0.790	9.807	-0.835	7.089
MobileNet	Mean	0.309	0.319	0.810	0.779	ResNet-152	Mean	0.509	0.488	0.631	0.690
	Std. Dev.	0.255	0.254	0.123	0.130		Std. Dev.	0.163	0.216	0.019	0.103
	Min	0.144	0.176	0.501	0.419		Min	0.207	0.199	0.614	0.536
	Max	0.996	1.026	0.902	0.852		Max	0.716	0.867	0.651	0.843
	Skewness	2.567	2.896	-2.383	-2.854		Skewness	-0.562	0.232	0.841	0.227
	Kurtosis	7.204	8.778	6.233	8.547		Kurtosis	-0.581	-0.763	-	-1.062

TABLE II

COMPARISON OF TRAINING/VALIDATION LOSS AND IOU STATISTICS ACROSS DIFFERENT OPTIMIZERS IN UNET MODEL

Optimizer	Statistic	Train Loss	Val Loss	Train IOU	Val IOU	Optimizer	Statistic	Train Loss	Val Loss	Train IOU	Val IOU
Adam	Mean	0.563	0.988	0.701	0.621	Adamax	Mean	0.428	0.397	0.756	0.726
	Std. Dev.	0.264	1.529	0.139	0.236		Std. Dev.	0.232	0.213	0.125	0.121
	Min	0.255	0.199	0.530	0.100		Min	0.144	0.173	0.596	0.584
	Max	0.904	5.254	0.857	0.846		Max	0.741	0.679	0.902	0.864
	Skewness	0.037	2.944	-0.098	-1.241		Skewness	0.014	0.164	-0.086	-0.114
	Kurtosis	-1.956	8.952	-2.205	1.441		Kurtosis	-1.810	-1.988	-2.127	-2.055
RMSprop	Mean	0.635	0.718	0.687	0.590	ASGD	Mean	0.484	0.465	0.721	0.693
	Std. Dev.	0.253	0.371	0.129	0.164		Std. Dev.	0.289	0.286	0.148	0.150
	Min	0.315	0.229	0.518	0.344		Min	0.129	0.156	0.501	0.419
	Max	0.934	1.398	0.845	0.821		Max	0.996	1.026	0.903	0.869
	Skewness	-0.132	0.297	-0.203	0.109		Skewness	0.436	0.707	-0.059	-0.483
	Kurtosis	-1.950	-0.344	-1.869	-1.233		Kurtosis	-0.789	-0.089	-1.602	-0.658
Adadelta	Mean	0.554	0.539	0.732	0.661	AdamW	Mean	0.498	0.504	0.721	0.684
	Std. Dev.	0.279	0.261	0.116	0.135		Std. Dev.	0.248	0.291	0.125	0.148
	Min	0.144	0.232	0.526	0.478		Min	0.205	0.186	0.576	0.523
	Max	0.933	0.880	0.857	0.822		Max	0.807	0.867	0.867	0.841
	Skewness	-0.041	0.057	-0.734	-0.004		Skewness	0.058	0.030	0.040	-0.009
	Kurtosis	-1.645	-1.883	-0.777	-1.844		Kurtosis	-2.020	-2.331	-2.123	-2.450
Adagrad	Mean	0.413	0.378	0.742	0.740	NAdam	Mean	0.463	0.566	0.745	0.671
	Std. Dev.	0.245	0.242	0.132	0.132		Std. Dev.	0.217	0.443	0.116	0.165
	Min	0.168	0.183	0.557	0.516		Min	0.219	0.208	0.601	0.399
	Max	0.819	0.813	0.881	0.848		Max	0.742	1.616	0.868	0.839
	Skewness	0.851	0.988	-0.516	-0.880		Skewness	0.089	1.645	-0.256	-0.462
	Kurtosis	-0.978	-0.765	-1.931	-1.112		Kurtosis	-2.119	2.978	-2.293	-1.462
SGD	Mean	0.456	0.508	0.724	0.703	RAdam	Mean	0.364	0.335	0.793	0.764
	Std. Dev.	0.318	0.319	0.161	0.168		Std. Dev.	0.196	0.171	0.111	0.101
	Min	0.159	0.185	0.463	0.396		Min	0.136	0.162	0.645	0.628
	Max	1.012	1.029	0.888	0.855		Max	0.623	0.608	0.903	0.871
	Skewness	0.699	0.332	-0.455	-0.708		Skewness	0.068	0.505	-0.352	-0.304
	Kurtosis	-1.247	-1.492	-1.542	-0.925		Kurtosis	-2.099	-1.460	-2.156	-1.910

Table II presents a detailed statistical comparison of training and validation performance of the U-Net model using various optimization algorithms. The optimizers evaluated include Adam, Adamax, RMSprop, ASGD, Adadelta, AdamW, Adagrad, NAdam, SGD, and RAdam. For each optimizer, six statistical measures are reported for both the loss and the Intersection over Union (IoU) on the training and validation sets: mean, standard deviation, minimum, maximum, skewness, and kurtosis. The results reveal substantial performance differences among optimizers. Notably, RAdam and Adamax yield strong segmentation performance, achieving relatively low loss values and high IoU scores,

with RAdam showing the highest mean training IoU (0.793) and validation IoU (0.764). Conversely, the Adam optimizer, despite its popularity, exhibits higher variability and higher maximum validation loss, suggesting potential instability or sensitivity to learning dynamics in this setting. Adagrad and ASGD also perform competitively, balancing moderate training loss with strong IoU metrics. Metrics such as skewness and kurtosis further indicate the distributional behavior of each optimizer's output, where optimizers like RAdam show more centered and stable performance distributions, while Adam and NAdam display heavier tails or skewed patterns, especially in validation loss. These findings underscore the critical role of optimizer selection in achieving optimal convergence and segmentation quality in U-Net-based architectures.

TABLE III
ANOVA SUMMARY FOR ENCODERS: BETWEEN AND WITHIN GROUPS FOR LOSS AND IOU

Metric	Source	Sum Sq.	Mean Sq.	F	Sig.
Val Loss	Between Groups	8.619	0.958	3.759	0.000
	Within Groups	22.931	0.255		
Val IOU	Between Groups	1.389	0.154	13.278	0.000
	Within Groups	1.046	0.012		
Train Loss	Between Groups	4.062	0.451	16.704	0.000
	Within Groups	2.432	0.027		
Train IOU	Between Groups	0.910	0.101	14.880	0.000
	Within Groups	0.551	0.007		

Table III presents the results of a one-way Analysis of Variance (ANOVA) conducted to assess the statistical significance of differences in performance across different encoder architectures within the U-Net model. The analysis evaluates both training and validation metrics, including loss and Intersection over Union (IoU), by comparing the variance between encoder groups and within each group. For all four performance metrics—training loss, validation loss, training IoU, and validation IoU—the ANOVA results reveal statistically significant differences among the encoder groups, as indicated by p-values (Sig.) less than 0.001. Specifically, the F-values are highest for training loss (F = 16.704) and validation IoU (F = 13.278), demonstrating strong group-level effects. The between-group variance is consistently higher than the within-group variance, particularly for the validation metrics, suggesting that the choice of encoder has a substantial and measurable impact on the U-Net's segmentation performance. These findings confirm the relevance of encoder selection as a significant factor in model optimization and justify further analysis of individual encoder contributions.

Here, Figure 9 displays a predicted segmentation map generated by the trained model. Each region in the image corresponds to segmented parts of the human body or background, visualized using distinct grayscale intensities or color tones, depending on the encoding used. This visual result highlights the model with encoder EfficientNet-b0 and optimizer Adamax accurately delineate human boundaries and segment relevant regions with clear spatial consistency. The segmentation map serves as a qualitative assessment

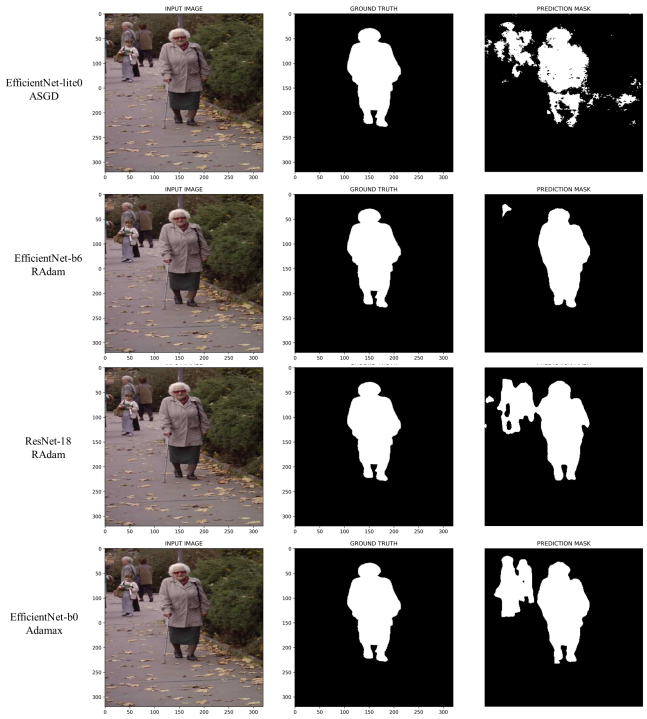

Fig. 9. Comparison of the predicted mask generated by the combination of encoders and optimizers.

of the model's performance and supports the quantitative evaluation presented in earlier sections.

TABLE IV
ANOVA RESULTS FOR OPTIMIZERS: BETWEEN AND WITHIN GROUPS FOR LOSS AND IOU

Metric	Source	Sum Sq.	Mean Sq.	F	Sig.
Val Loss	Between Groups	3.295	0.366	1.166	0.326
	Within Groups	28.255	0.314		
Val IOU	Between Groups	0.252	0.028	1.153	0.335
	Within Groups	2.184	0.024		
Train Loss	Between Groups	0.580	0.064	0.980	0.462
	Within Groups	5.914	0.066		
Train IOU	Between Groups	0.073	0.008	0.472	0.889
	Within Groups	1.388	0.017		

Table IV summarizes the results of a one-way ANOVA performed to examine whether different optimization algorithms lead to statistically significant differences in the U-Net model's performance, measured by training and validation loss and IoU. The analysis compares the variance between and within groups across ten optimizers. Unlike the encoder-based ANOVA results, none of the performance metrics show statistically significant differences between optimizers, as indicated by high p-values: 0.326 for validation loss, 0.335 for validation IoU, 0.462 for training loss, and 0.889 for training IoU. All F-values are below 1.2, suggesting minimal variance explained by optimizer choice relative to the within-group variability. These results indicate that, while different optimizers yield varying performance in descriptive statistics, these differences are not statistically significant at conventional thresholds. Therefore, optimizer selection

may have a more nuanced or context-dependent effect on segmentation performance, warranting further analysis or alternative evaluation methods beyond ANOVA.

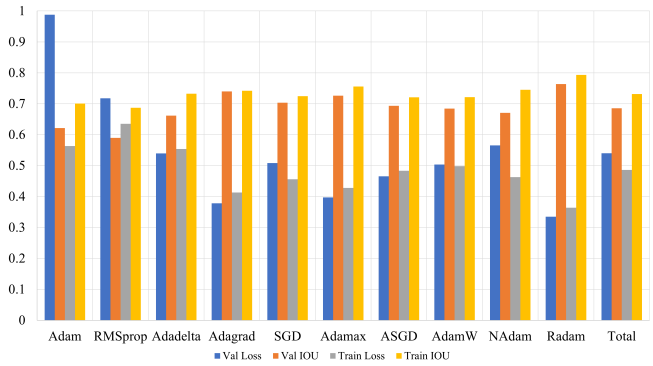

Fig. 10. Comparison of optimizers on loss and mIoU over train and validation datasets of the model.

Figure 10 reveals that among the evaluated encoders, timm-efficientnet-b0 achieved the best overall performance, yielding the lowest validation loss (0.2590) and a high validation IoU (0.8116), closely followed by EfficientNet-lite0, which attained the highest validation IoU (0.8231) and a comparably low training loss. These results demonstrate the strength of lightweight, modern EfficientNet variants in achieving high segmentation accuracy with efficient learning. In contrast, deeper ResNet models such as resnet101 and resnet50 exhibited notably higher validation losses (1.2213 and 0.8189, respectively) and lower IoUs (0.4866 and 0.5235), indicating weaker generalization and overfitting tendencies. Classic lightweight models like mobilenetv2 offered balanced performance, with a validation loss of 0.3189 and IoU of 0.7792, proving competitive despite their lower parameter count. Overall, EfficientNet-based encoders—particularly timm-efficientnet-b0 and lite0—outperformed conventional ResNet architectures, validating their suitability for efficient and accurate semantic segmentation tasks.

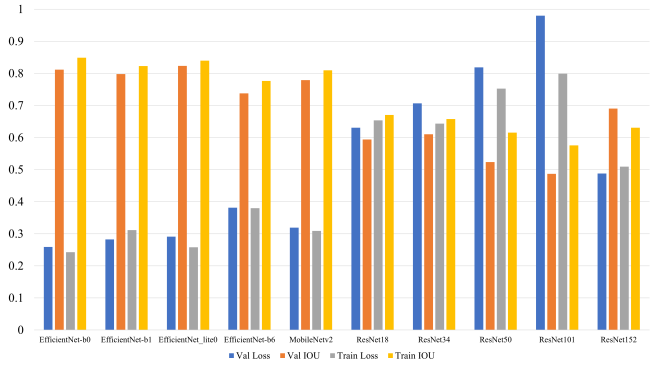

Fig. 11. Comparison of encoders on loss and mIoU over train and validation datasets of the model.

Figure 11 represents the results across all encoders evaluated with ten distinct optimizers, RAdam consistently demonstrated the most effective performance, achieving the lowest mean validation loss of 0.3349 and the highest validation IoU of 0.7638. This was closely followed by Adagrad and Adamax, which also yielded competitive validation IoUs of 0.7397 and 0.7263, respectively, with relatively low training losses and high training IoUs, indicating stable convergence and strong generalization. In contrast, Adam and RMSprop showed higher validation losses (0.9880 and 0.7177, respectively) and lower validation IoUs, suggesting suboptimal performance in segmentation tasks under this configuration. Overall, optimizers such as RAdam, Adagrad, and Adamax consistently outperformed others in enhancing the segmentation accuracy of both lightweight and deep encoders, as reflected in the aggregated mean performance across all models.

V. Conclusion

In this research, we systematically evaluated the performance of various encoder-optimizer combinations within a U-Net segmentation framework on a human segmentation dataset. The findings reveal that lightweight and efficient encoders, particularly EfficientNet variants, consistently outperformed deeper ResNet architectures in both training and validation phases. EfficientNet-B0 with Adamax and EfficientNet-Lite0 with RAdam yielded the highest validation IoU scores, indicating their effectiveness in capturing human contours with minimal computational complexity. Although optimizers influenced performance to a lesser extent, Adamax, RAdam, and ASGD emerged as the most effective choices. ANOVA results substantiated the statistical significance of encoder choice while suggesting minimal impact from the choice of optimizer. These insights advocate for the integration of efficient encoder architectures in semantic segmentation pipelines, especially for resource-constrained or real-time applications.

References

[1] S. Hao, Y. Zhou, and Y. Guo, "A brief survey on semantic segmentation with deep learning," *Neurocomputing*, vol. 406, pp. 302–321, 2020.

[2] K. Aitken, V. Ramasesh, Y. Cao, and N. Maheswaranathan, "Understanding how encoder-decoder architectures attend," *Advances in Neural Information Processing Systems*, vol. 34, pp. 22 184–22 195, 2021.

[3] M. Krithika Alias AnbuDevi and K. Suganthi, "Review of semantic segmentation of medical images using modified architectures of unet," *Diagnostics*, vol. 12, no. 12, p. 3064, 2022.

[4] 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. 97 457–97 465, 2021.

[5] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[6] Z.-L. Ni, G.-B. Bian, Z.-G. Hou, X.-H. Zhou, X.-L. Xie, and Z. Li, "Attention-guided lightweight network for real-time segmentation of robotic surgical instruments," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 9939–9945.

[7] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation," in *Proceedings of the european conference on computer vision (ECCV)*, 2018, pp. 552–568.

[8] D. You, S. Wang, F. Wang, Y. Zhou, Z. Wang, J. Wang, and Y. Xiong, "Efficientunet+: a building extraction method for emergency shelters based on deep learning," *Remote Sensing*, vol. 14, no. 9, p. 2207, 2022.

[9] Z. Yang, C. Wu, Z. Zhou, X. Zhang, X. Wang, and Y. Liu, "Mobility increases localizability: A survey on wireless indoor localization using inertial sensors," *ACM Computing Surveys (Csur)*, vol. 47, no. 3, pp. 1–34, 2015.

[10] P. Yakubovskiy, "Segmentation models pytorch," 2020, https://github.com/qubvel/segmentation_models.pytorch.

[11] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[12] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017, pp. 2881–2890.

[13] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[14] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning (ICML)*, 2019, pp. 6105–6114.

[15] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015, pp. 234–241.

[16] M. T. Islam and Y. Zhang, "Efficient u-net for real-time semantic segmentation," *Sensors*, vol. 20, no. 7, p. 1911, 2020.

[17] H. Abdi and L. J. Williams, "Tukey's honestly significant difference (hsd) test," in *Encyclopedia of Research Design*, N. J. Salkind, Ed. Sage, 2010, pp. 1–5.

[18] P. Mishra, C. M. Pandey, U. Singh, A. Gupta, C. Sahu, and A. Keshri, "Descriptive statistics and normality tests for statistical data," *Annals of Cardiac Anaesthesia*, vol. 22, no. 1, pp. 67–72, 2019.

[19] P. Rasti, S. T. Seydi, and O. Ozcan, "Statistical analysis and performance evaluation of deep learning-based segmentation methods," *Neural Processing Letters*, vol. 53, no. 2, pp. 1727–1748, 2021.

[20] V. Badrinarayanan, A. Handa, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling," *arXiv preprint arXiv:1505.07293*, 2015.

[21] R. Ankareddy and R. Delhibabu, "Dense segmentation techniques using deep learning for urban scene parsing: a review," *IEEE Access*, 2025.

[22] S. Manzoor, E.-J. Kim, S.-H. Joo, S.-H. Bae, G.-G. In, K.-J. Joo, J.-H. Choi, and T.-Y. Kuc, "Edge deployment framework of guardbot for optimized face mask recognition with real-time inference using deep learning," *Ieee Access*, vol. 10, pp. 77 898–77 921, 2022.

[23] P. Xu, T. M. Hospedales, Q. Yin, Y.-Z. Song, T. Xiang, and L. Wang, "Deep learning for free-hand sketch: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 1, pp. 285–312, 2022.

[24] P. Yadav, N. Menon, V. Ravi, S. Vishvanathan, and T. D. Pham, "Efficientnet convolutional neural networks-based android malware detection," *Computers & Security*, vol. 115, p. 102622, 2022.

[25] J. Shang, K. Zhang, Z. Zhang, C. Li, and H. Liu, "A high-performance convolution block oriented accelerator for mbconv-based cnns," *Integration*, vol. 88, pp. 298–312, 2023.

[26] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[27] Z. Daquan, "Efficient architecture design for deep neural networks," Ph.D. dissertation, National University of Singapore (Singapore), 2022.

[28] X. Liu, H. Peng, N. Zheng, Y. Yang, H. Hu, and Y. Yuan, "Efficientvit: Memory efficient vision transformer with cascaded group attention," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 14 420–14 430.

[29] D. Hernandez and T. B. Brown, "Measuring the algorithmic efficiency of neural networks," *arXiv preprint arXiv:2005.04305*, 2020.

[30] S. Cahyawijaya, "Greenformers: Improving computation and memory efficiency in transformer models via low-rank approximation," *arXiv preprint arXiv:2108.10808*, 2021.

[31] Z. Zhao, G. Gao, Z. Zhang, H. Huang, and H. Liu, "An individual identification method for cetaceans based on efficientnet," in *Proceedings of the 2024 7th International Conference on Machine Learning and Machine Intelligence (MLMI)*, 2024, pp. 198–203.

[32] S. Rokhva and B. Teimourpour, "A novel method for accurate & real-time food classification: The synergistic integration of efficientnetb7, cbam, transfer learning, and data augmentation," *arXiv preprint arXiv:2410.02304*, 2024.

[33] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[34] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[36] X. Ou, P. Yan, Y. Zhang, B. Tu, G. Zhang, J. Wu, and W. Li, "Moving object detection method via resnet-18 with encoder–decoder structure in complex scenes," *IEEE Access*, vol. 7, pp. 108 152–108 160, 2019.

[37] X. Wang, S. Wang, P. Shao, B. Jiang, L. Zhu, and Y. Tian, "Event stream based human action recognition: a high-definition benchmark dataset and algorithms," *arXiv preprint arXiv:2408.09764*, 2024.

[38] O. Elharrouss, Y. Akbari, N. Almaadeed, and S. Al-Maadeed, "Backbones-review: Feature extraction networks for deep learning and deep reinforcement learning approaches," *arXiv preprint arXiv:2206.08016*, 2022.

[39] B. Koonce, "Resnet 50," in *Convolutional neural networks with swift for tensorflow: image recognition and dataset categorization*. Springer, 2021, pp. 63–72.

[40] ——, "Resnet 50," in *Convolutional neural networks with swift for tensorflow: image recognition and dataset categorization*. Springer, 2021, pp. 63–72.

[41] Q. Zhang, "A novel resnet101 model based on dense dilated convolution for image classification," *SN Applied Sciences*, vol. 4, no. 1, p. 9, 2022.

[42] P. Ghosal, L. Nandanwar, S. Kanchan, A. Bhadra, J. Chakraborty, and D. Nandi, "Brain tumor classification using resnet-101 based squeeze and excitation deep neural network," in *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*. IEEE, 2019, pp. 1–6.

[43] O. Elharrouss, Y. Akbari, N. Almaadeed, and S. Al-Maadeed, "Backbones-review: Feature extraction networks for deep learning and deep reinforcement learning approaches," *arXiv preprint arXiv:2206.08016*, 2022.

[44] M. K. Panda, B. N. Subudhi, T. Veerakumar, and V. Jakhetiya, "Modified resnet-152 network with hybrid pyramidal pooling for local change detection," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 4, pp. 1599–1612, 2023.

[45] M. G. Abdolrasol, S. S. Hussain, T. S. Ustun, M. R. Sarker, M. A. Hannan, R. Mohamed, J. A. Ali, S. Mekhilef, and A. Milad, "Artificial neural networks based optimization techniques: A review," *Electronics*, vol. 10, no. 21, p. 2689, 2021.

[46] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," *Advances in neural information processing systems*, vol. 31, 2018.

[47] J. Zhang, F. Hu, L. Li, X. Xu, Z. Yang, and Y. Chen, "An adaptive mechanism to achieve learning rate dynamically," *Neural Computing and Applications*, vol. 31, no. 10, pp. 6685–6698, 2019.

[48] M. Daun and C. Sandberg, "Cloudy with a chance of segmentation: Evaluating semantic segmentation models," 2025.