Benchmarking ECG Delineation using Deep Neural Network-based Semantic Segmentation Models

JaeHo Park WOGH2012@MEDICALAI.COM
TaeJun Park SKDF1020@MEDICALAI.COM
Gyurin Kim GYU_LL@MEDICALAI.COM
JiHyun Lee RHDWN082@MEDICALAI.COM
Jeong Min Son JMSON@MEDICALAI.COM
Joon-myoung Kwon CTO@MEDICALAI.COM
Yong-Yeon Jo * YY.JO@MEDICALAI.COM
Medical AI Co., Ltd. Seoul, Republic of Korea

Abstract

Accurate electrocardiogram (ECG) delineation is essential for automated cardiac diagnosis, enabling the precise identification of key waveforms such as the P wave, QRS complex, and T wave. This study presents the first comprehensive benchmarking of neural network-based semantic segmentation models for ECG delineation, evaluating their accuracy, resource efficiency, and robustness across both public and private datasets. Our results demonstrate that convolutional neural network (CNN)-based approaches consistently achieve superior accuracy compared to Transformer-based approaches. Additionally, we observed the presence of fragmented segments in the delineation results. To address this issue, we explored post-processing techniques to consolidate or eliminate fragmented segments using an optimal configuration, leading to performance improvements. Furthermore, by analyzing performance variations across different waveform labels, we provide critical insights into key considerations for ECG segmentation tasks. Notably, our findings also reveal that larger model sizes do not necessarily correlate with better performance. Based on our findings, we propose a set of practical guidelines for leveraging segmentation models in ECG delineation, offering valuable direction for future research and clinical applications.

Data and Code Availability For the data, we use both publicly available and private ECG datasets. Specifically, we employ the Lobachevsky University Electrocardiography Database (LUDB) from PhysioNet (Goldberger et al., 2000 (June 13)) and a private dataset annotated by in-house medical experts to compare the performance of different approaches. To ensure a fair benchmark on the LUDB dataset, we carefully stratify the data and provide information on its partitioning into training, validation, and test sets while maintaining a representative distribution of ECG waveforms. For the code, we provide semantic segmentation network implementations specifically designed for ECG delineation. Additionally, we offer a prototype webpage for demonstration purposes.

Due to anonymization requirements, all resources mentioned above are included as supplementary material before publication. Once the paper is accepted, the code and relevant resources will be publicly released via a dedicated webpage.

1. Introduction

Electrocardiograms (ECG) are critical signals that record the electrical activity of the heart, playing an essential role in diagnosing various cardiac conditions. An ECG waveform consists of three primary components: the P wave, QRS complex, T wave, and etc., which collectively represent the electrical signals corresponding to the heart's movements. Accurately segmenting these components, known as `ECG delineation`, aids in diagnosing heart diseases, enabling timely interventions and effective treatment planning Zimetbaum and Josephson (2003).

In the field of computer vision, a diverse array of semantic segmentation models has emerged, with early developments predominantly centered around CNN-based architectures. Beginning with models

* Correspondence

such as Fully Convolutional Networks (FCN) Long et al. (2015) and U-Net Ronneberger et al. (2015), segmentation techniques have undergone continuous refinement through iterative enhancements and competitive benchmarks. This ongoing process has led to the creation of increasingly sophisticated models Long et al. (2015); Huang et al. (2020); Chen et al. (2018); Zhao et al. (2017); Wang et al. (2020). In addition, Transformer-based architectures Dosovitskiy et al. (2021); Zheng et al. (2021); Wang et al. (2021); Liu et al. (2021); Chu et al. (2021); Xie et al. (2021) have established a new paradigm in semantic segmentation by leveraging self-attention mechanisms.

These advancements have not only enhanced performance in traditional image-based tasks but have also been applied to the ECG delineation task Moskalenko et al. (2020); Kuvaev and Khudorozhkov (2020); Chen et al. (2020); Peimankar and Puthusserypady (2021); Joung et al. (2024). However, the question of which approaches are most effective remains unanswered due to the lack of systematic validation and benchmarking in current research. To address this question, we have developed a robust benchmarking framework to evaluate the effectiveness of various deep neural network–based segmentation approaches for ECG delineation. This framework encompasses the careful refinement and stratification of datasets, the implementation of various segmentation methods, and comprehensive hyperparameter tuning to optimize performance. Furthermore, we ensure transparency by publicly releasing the data distributions and implementations employed, thereby facilitating benchmarking efforts through the utilization of our resources. [1].

To summarize the benchmarking results, CNN-based architectures consistently demonstrate strong and reliable performance, exhibiting high accuracy, robustness, and efficiency across diverse ECG datasets. In contrast, Transformer-based approaches show greater instability and limited generalizability, particularly when handling subtle waveform variations. Additionally, our analysis identified fragmented segments in the delineation results. To address this, we explored post-processing techniques that consolidate fragmented segments using an optimal configuration, leading to improved overall performance and reduced discontinuities in predicted segment boundaries. Furthermore, by analyzing performance variations across different ECG diagnostic categories, we identified specific challenges associated with certain ECG components. A crucial insight from our findings is that larger model sizes do not necessarily correlate with better performance. This underscores the need for efficient architectures that balance accuracy, computational efficiency, and generalization capability.

By providing a comprehensive evaluation, our study offers valuable direction for future research and clinical applications in automated ECG analysis.

Our contributions are summarized as follows:

1. We identified the lack of benchmarking efforts in ECG delineation and conducted the first comprehensive study to address this gap, establishing a robust foundation for systematic evaluation in the field.

2. We developed specialized semantic segmentation models tailored for ECG delineation and carefully refined the dataset for benchmarking. To promote reproducibility and facilitate future research, we publicly release these resources along with the implemented networks.

3. We performed extensive benchmarking experiments, providing practical guidelines for model selection and optimization through a detailed performance analysis across multiple dimensions, including accuracy, robustness, efficiency, and generalization capability.

2. Related Work

This section introduces the ECG delineation task and presents widely used semantic segmentation methods that leverage deep learning, particularly CNN and Transformer-based architectures.

2.1. ECG Delineation

ECG delineation refers to the process of identifying and marking key points or segments within an ECG waveform. These key points correspond to distinct phases of the cardiac cycle, including the P wave, QRS complex, and T wave. The task involves detecting the onset and offset of these waves, which are crucial for analyzing cardiac function. By extracting features such as the duration of the PR interval, QRS complex, or QT interval, clinicians can gain valuable insights into heart health Zimetbaum and Josephson

1. https://huggingface.co/spaces/MedicalAI-DP/ECG_Delineation

(2003). Figure 1 shows sample results of ECG delineation, highlighting the onsets and offsets of the P wave, QRS complex, and T wave.

2.2. CNN-based Approach

We have investigated the most popular CNN-based segmentation methods Long et al. (2015); Huang et al. (2020); Chen et al. (2018); Zhao et al. (2017); Wang et al. (2020). These methods can be categorized based on three key techniques: (1) multi-scale feature fusion Long et al. (2015), (2) global average pooling Zhao et al. (2017), and (3) dilated convolutions Chen et al. (2018).

Multi-scale feature fusion integrates outputs from multiple layers or scales within the network to produce the final segmentation result; by combining features from various depths, the model can leverage both high-level semantic information and low-level spatial details. Global average pooling aggregates features across spatial dimensions using average pooling, helping to reduce the spatial dimensions of feature maps while retaining essential information. Dilated convolutions utilize dilation in the convolutional kernels to expand the receptive field without increasing parameters or computational cost. In summary, these techniques have evolved to efficiently extract and fuse multi-scale feature information, ultimately contributing to precise boundary delineation of each object.

FCN Long et al. (2015), the UNet series Ronneberger et al. (2015); Zhou et al. (2018); Huang et al. (2020), and HRNet Wang et al. (2020) are fundamentally designed to leverage multi-scale feature fusion. However, even with the same technique, each approach differs in how and to what extent it is utilized. FCN extracts low-resolution feature maps through progressive downsampling across layers and restores them to the original size through upsampling, fusing the feature maps in the process. The UNet series refines this concept with an encoder-decoder architecture, directly passing encoder features to the decoder to better preserve information during reconstruction. As the version of UNet series increases, they have employed more diverse methods of integrating encoder features into the decoder. HRNet generates features at multiple scales and fuses them iteratively while maintaining high-resolution features throughout the network, effectively preserving high-resolution details throughout the process.

PSPNet Zhao et al. (2017) is an approach that adopts two key techniques: multi-scale feature fusion and global average pooling. It introduces a Pyramid Pooling Module utilizing global average pooling, enabling effective multi-scale feature fusion without depending solely on skip connections or downsampling. This design significantly expands the receptive field for each pixel, enhancing segmentation accuracy.

The DeepLab series Chen et al. (2016, 2017a,b, 2018) represents a family of models that effectively integrates multiple widely-used segmentation techniques, encompassing multi-scale feature fusion, global average pooling, and dilated convolutions. A defining characteristic of this series is its use of dilated convolutions, which expand the receptive field without the need for downsampling, enabling precise feature extraction from complex spatial patterns. Furthermore, the incorporation of depthwise separable convolutions enhances computational efficiency while maintaining a broad receptive field.

2.3. Transformer-based Approach

Beyond CNNs, many segmentation methods have emerged that utilize self-attention-based Transformer architectures. These approaches blend the strengths of traditional CNN-based methods with the unique capabilities of Transformers, such as capturing long-range dependencies and modeling global relationships effectively.

SETR Zheng et al. (2021) employs Vision Transformer (ViT) Dosovitskiy et al. (2021) as its backbone to capture relationships between image patches through Transformer blocks. This model explores various decoder architectures to determine the optimal configuration for semantic segmentation tasks. It was the first to apply Transformer to the segmentation task and demonstrated its effectiveness.

SegFormer Xie et al. (2021) is a lightweight and efficient framework designed for multi-scale feature fusion. It introduces an overlapped patch merging technique that maintains local continuity between patches, preserving spatial details and improving segmentation accuracy. This design strikes a balance between performance and computational efficiency, making it suitable for real-world applications.

3. Method

This section presents the datasets used in our study, along with methods for implementing and optimiz-

ing ECG delineation models. We also describe preprocessing steps applied to raw ECG signals and the metrics used to evaluate model performance.

3.1. Dataset

We utilized (1) the publicly available LUDB and (2) a private dataset annotated by in-house medical experts.

The LUDB dataset includes 200 ECG recordings, each sampled at 500 Hz and spanning 10 seconds across 12 leads, resulting in 2400 individual lead waveforms. Each lead is annotated with the onset and offset points of the P wave, QRS complex, and T wave, making the dataset highly suitable for ECG delineation tasks. They include diagnostic labels for heart rhythms and the electrical axis of the heart. In addition to these labels, we incorporated an additional label indicating the presence or absence of a P wave—a critical feature for diagnosing conditions such as atrial fibrillation, as identified by our medical staff. We stratified and divided the dataset into training, validation, and test sets in an 8:1:1 ratio, ensuring that each ECG sample and its associated 12 leads were exclusively assigned to a single subset. However, due to the rarity of certain labels, some samples were necessarily confined to one or two subsets. A detailed description of the sample and label distribution is provided in Appendix A.

Our private dataset, used for external validation, consists of 10,298 annotated leads from 1,285 patients.[2] Furthermore, our dataset primarily comprises a diverse range of ECGs reflecting various medical abnormalities. Expert medical staff meticulously annotated the onset and offset points of the P wave, QRS complex, and T wave. These annotations can be seen as a reflection of the diverse diseases and challenging-to-diagnose ECG cases encountered in real-world scenarios.

3.2. Pre-Processing

A 12-lead ECG device does not display the raw sampled signal directly to the user but applies filtering before outputting the signal. Based on consultations with our internal medical experts, we applied filtering to remove as much noise as possible while preserving the identity of the P, QRS, and T waves. Specifically, we employed a Butterworth filter with a 0.05 Hz high-pass filter to eliminate baseline wander and a 150 Hz low-pass filter to reduce noise. These filtered signals were then used for model training.

3.3. Performance Metric

Intersection over Union (IoU) is a widely used metric in segmentation tasks for evaluating how well predicted regions align with ground truth annotations. It measures accuracy by computing the ratio of the intersection area to the total combined area of the prediction and ground truth:

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

In the context of the ECG delineation task, models predict the P wave, QRS complex, and T wave on a point-wise basis. To compute the IoU for each component, we calculate the intersection and union between the predicted segments and the ground truth. To provide an overall assessment of the model's performance, we calculate the mean Intersection over Union (**mIoU**), which is the average of the IoU values for the P wave, QRS complex, and T wave.

3.4. Implementation and Optimization

We implemented networks, including FCN Long et al. (2015), UNet 3+ Huang et al. (2020), PSPNet Zhao et al. (2017), DeepLabv3+ Chen et al. (2018), HRNet Wang et al. (2020), SETR Zheng et al. (2021), and SegFormer Xie et al. (2021), as outlined in Section 2. [3]. Originally designed for image semantic segmentation, these networks were modified to take one-dimensional ECG waveforms as input and output three components of ECG. However, their core architectures remained unchanged, preserving the novel contributions proposed by the original authors. For example, UNet 3+ employs a hybrid loss function that combines Binary Cross Entropy (BCE), Multi-Scale Structural Similarity (MS-SSIM), and Dice loss functions. In contrast, all other models were trained using BCE loss.

While retaining the structural designs of these models, we conducted extensive hyperparameter tuning to optimize their performance for ECG delineation. Our implementation supports a broad range of configurations, allowing not only fine-tuning of details such as selecting the appropriate interpolation

2. Not all 12 leads were labeled, and some leads were excluded during the collection and validation process.

3. https://huggingface.co/spaces/MedicalAI-DP/ECG_Delineation/tree/main/res/impl

mode but also reinforcing core architectural elements, such as adjusting the number of convolutional layers and network depth. Detailed descriptions of the hyperparameters and configurable options for each model are provided in Appendix B.

Due to (1) limitations in computing resources and (2) the vast number of potential hyperparameter combinations, identifying an optimal sampling strategy becomes highly challenging. For the first limitation, we carefully set the hyperparameter ranges to fully leverage the computational capabilities of the NVIDIA DGX A100 system.[4] For the second difficulty, we conduct following two stages to narrow down the number of candidate hyperparameter combinations. In this process, we employed the Asynchronous Successive Halving Algorithm Li et al. (2020) as a hyperparameter optimization scheduler to efficiently guide the selection process.

To select the model, we use mean Intersection over Union (mIoU) as the performance metric, calculating the average IoU values for the P wave, QRS complex, and T wave.

Stage 1. Model Structure Screening: This stage exclusively focuses on the model structure. We performed a grid search on hyperparameters that influence the model structure. During this stage, other hyperparameters unrelated to the model structure are fixed as follows: the SGD optimizer Sutskever et al. (2013) is employed with a learning rate of 0.01 and a momentum of 0.99; no learning rate scheduler; the batch size is set to either 64 or 128, depending on the model size and available GPU memory; and training was conducted in a constrained manner, limited to a maximum of 50 epochs. To select the best-performing hyperparameters from the training results, we first listed the top 5% of experiments based on mIoU. For each hyperparameter, we identified the two most frequently seen values. Additionally, the highest-performing hyperparameter was always included. As a result, each hyperparameter had a minimum of two and a maximum of three selected values, which were carried forward to the next stage.

Stage 2. Model Fine-Tuning: This stage focuses on refining the model by tuning all hyperparameters based on the structural hyperparameters identified in Stage 1. Specifically, we tuned the model using the SGD optimizer with the following settings: a learning rate sampled from a log-uniform distribution between 1×10^{-4} and 1×10^{-1}, momentum set to 0.9, weight decay sampled from a log-uniform distribution between 5×10^{-6} and 5×10^{-4}, Polynomial scheduler where the power is selected from values between 0 and 1, with a step size of 0.2[5] a batch size of 64, and a maximum of 500 epochs. The hyperparameter optimization process utilized a sampling size of 256 combinations to explore the most effective configurations. Additionally, to address performance bias introduced by random seeds, we repeated the process across *five different random seeds*, ensuring robustness and minimizing variability in the final configurations.

4. 320GB GPU memory(8x NVIDIA A100 40GB GPUs) and 1TB system memory

5. We used a common configuration that encompasses the settings introduced in the experiments of the various approaches referenced.

4. Evaluation

This section presents a comprehensive comparison of the performance of all implemented approaches.

4.1. Performance Comparison

We evaluated the overall performance of various approaches, focusing on their ability to accurately segment the P wave, QRS complex, and T wave in ECG waveform. Table 1 presents the mIoU scores for each component and overall. The boldface text indicates the best-performing approach in each column.

On the LUDB dataset, CNN-based approaches demonstrate superior performance. Specifically, UNet 3+ achieves the best mIoU of 0.854, excelling in segmenting the P wave with an mIoU of 0.814. FCN records the best performance in the QRS complex (0.882) and T wave (0.870), though its overall mIoU is slightly lower than that of UNet 3+. HRNetV2 maintains balanced performance across all components, with an overall mIoU of 0.846. DeepLabv3+ and PSPNet show decent performance on the QRS complex and T wave, but their performance significantly drops on the P wave, resulting in lower overall performance compared to the top-performing approaches. Transformer-based approaches, such as SegFormer and SETR, exhibit relatively weaker performance, particularly in the P wave and T wave segments.

On the private dataset, which has a distribution dominated by disease cases, CNN-based approaches also demonstrate superior performance. Interestingly, FCN emerges as the top performer with an

Table 1: Comparison of ECG delineation performance based on the mIoU of individual components and overall performance on LUDB and a private dataset.

Approach	LUDB				Private			
	P	QRS	T	Overall	P	QRS	T	Overall
UNet 3+	**0.814**	0.881	0.868	**0.854**	0.733	**0.845**	0.771	0.783
FCN	0.809	**0.882**	**0.870**	0.853	**0.741**	0.841	**0.773**	**0.785**
HRNetV2	0.803	0.875	0.860	0.846	0.724	0.837	0.767	0.776
SegFormer	0.771	0.869	0.841	0.827	0.682	0.843	0.765	0.763
PSPNet	0.755	0.849	0.851	0.819	0.705	0.815	0.765	0.762
DeepLabv3+	0.748	0.867	0.841	0.819	0.650	0.827	0.746	0.741
SETR	0.620	0.800	0.770	0.730	0.570	0.732	0.669	0.657

Table 2: Over-segmentation ratio on LUDB and private datasets

Approach	LUDB				Private			
	P	QRS	T	Overall	P	QRS	T	Overall
SETR	114.5	99.7	100.6	104.2	101.0	107.6	110.3	106.4
PSPNet	111.8	101.2	103.3	104.9	110.4	107.0	113.3	110.2
HRNetV2	110.0	102.2	105.6	105.6	108.8	108.1	118.7	111.9
DeepLabv3+	120.5	101.7	110.4	110.0	**131.6**	113.8	**133.4**	**126.0**
UNet 3+	116.8	**106.7**	109.3	110.4	116.1	**115.5**	119.0	116.9
SegFormer	120.5	105.0	114.5	112.6	126.2	110.4	124.6	120.2
FCN	**121.3**	106.1	**117.4**	**114.2**	122.5	113.9	131.1	122.4

overall mIoU of 0.785, demonstrating better generalizability to unseen data. UNet 3+ also delivers strong overall results with an mIoU of 0.783 but is slightly outperformed by FCN in this setting. HRNetV2 demonstrated strong generalizability with an mIoU of 0.776, following FCN and UNet 3+. Similarly to the results on LUDB, DeepLabv3+ and PSPNet demonstrate lower segmentation accuracy. Transformer-based approaches show a further decline in performance on the private dataset.

The results indicate that determining the best approach for ECG delineation is context-dependent, as performance trends vary significantly between the LUDB and private datasets. CNN-based approaches, particularly UNet 3+ and FCN, exhibit strong performance across different datasets, with FCN showing better generalizability on disease-dominated data. Similar to the results on LUDB, DeepLabv3+, PSPNet, and Transformer-based approaches exhibit similar trends, showing lower segmentation accuracy and indicating poor generalizability to unseen data.

4.2. Over Segmentation

We examined the qualitative performance by comparing each approach's predicted ECG waveforms to the ground truth, as illustrated in Figure 1. The region above the baseline corresponds to the model's segmented output for the P wave, QRS complex, and T wave, while the region below displays the annotated gold standard. In normal ECGs (Figures 1(a) and 1(b)), this process generally yields accurate delineations of the primary wave components—clear P waves, distinctly bounded QRS complexes, and well-defined T waves—indicating that the models adapt successfully to regular cardiac patterns.

However, as shown in Figure 1(c), many approaches struggle with pathological samples, where atypical morphologies can cause *over-segmentation*, which refers not only to splitting a single wave into multiple disjoint segments but also to falsely identifying ambiguous regions as separate components. UNet 3+ exemplifies this by occasionally predicting multiple small segments for one QRS complex or T wave. Figure 1(d) shows that most approaches tend to noisily detect a P wave even when none is actually present, especially in pathologically abnormal ECGs.

To quantify over-segmentation, we computed a ratio that indicates how much the predicted segments deviate from a one-to-one correspondence with the ground truth. Table 2 presents this ratio for various approaches on both the LUDB and private datasets. The boldface tetxt highlights the highest

over-segmentation ratio for each category, indicating the most cohesive segmentation. A ratio of 100% signifies perfect alignment with the ground truth

On the LUDB dataset, FCN exhibits a high over-segmentation ratio, whereas DeepLabv3+ shows similar behavior on the private dataset. Notably, across both datasets, UNet 3+ tends to produce elevated over-segmentation ratios, particularly for the QRS complex. In contrast, SETR achieves the lowest ratios—suggesting a more contiguous segmentation result—even though its overall performance may not always be the highest. These findings emphasize that over-segmentation is not strictly detrimental if the approach simultaneously captures the essential boundaries of the P wave, QRS complex, and T wave. Additionally, we analyze segmentation behavior across individual ECG leads (Appendix C) and confirm a similar trend.

4.3. Post-Hoc Consolidation and Elimination of Over Segmentation

In Section 4.2, we observed that certain pathological signals show the over-segmentation, causing a single wave to appear as multiple disjoint segments and leading to the misclassification of noisy signals as primary components. This phenomenon can degrade performance by failing to capture the wave as a coherent entity. A common practice for mitigating over-segmentation is to apply post-processing that merges adjacent predicted segments Joung et al. (2024). In this study, we investigate how merging such adjacent segments and removing short, fragmented segments affects the delineation across both LUDB and private datasets. To define adjacent segments, we specify a *gap threshold*—the maximum allowable time (in milliseconds) between two segments for merging—selected from $[10, 30, 50]$. Additionally, a *duration threshold*, chosen from $[10, 30, 50, 70]$, removes any segments that fail to meet a minimum required duration.

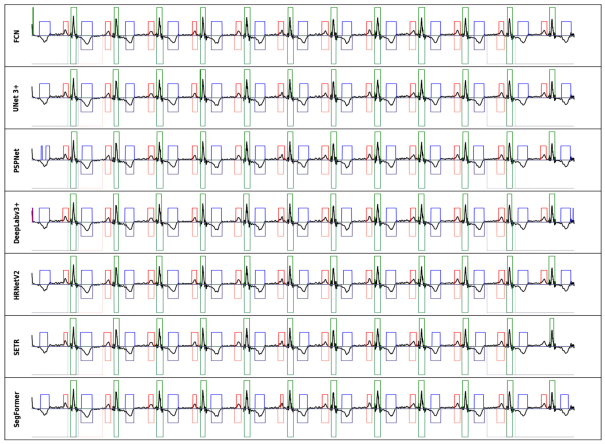
Table 3 summarizes the performance of each approach under three distinct configurations. *Base* refers to the original results without any post-processing (no segment consolidation and elimination); *Best* corresponds to the parameter setting that yields the highest mIoU; and *Worst* represents the configuration that causes the largest performance drop relative to the Base. For each wave type (P, QRS, T), the two listed values denote the gap and duration thresholds applied to the LUDB and private datasets, respectively. The boldface text indicates the best performance for each dataset.

The overall effect of post-processing indicates that most approaches achieve modest to moderate improvements under their best configurations, with carefully tuned gap and duration thresholds effectively reducing over-segmentation and refining segmentation boundaries. For example, FCN sees gains of up to +0.005 mIoU on the LUDB dataset and +0.002 on the private dataset. In contrast, poorly chosen thresholds can lead to significant performance declines, as demonstrated by DeepLabv3+ (a -0.008 drop in LUDB mIoU under its worst configuration), showing how overly aggressive segment merging or removal can distort crucial wave features.

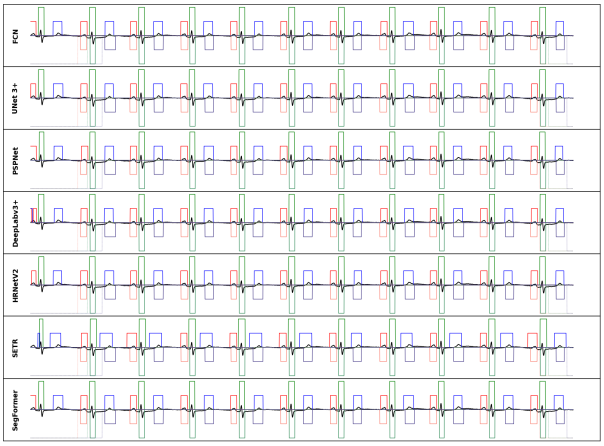
Figures 1(*e*) and 1(*f*) further demonstrate the impact of meticulously refined post-processing. A direct comparison with Figures 1(*c*) and 1(*d*), respectively, reveals that previously disjoint segments have been consolidated into single, continuous segments, and overly short segments have been removed—thereby yielding cleaner, more cohesive delineations of the underlying ECG waveforms.

From a wave-centric perspective, it was challenging to identify distinct and consistent trends of thresholds across all models and datasets. When examining the best-case configurations, the P wave threshold varies tendencies between the LUDB and private datasets, with many models showing divergent behaviors. This suggests that P wave segmentation is more sensitive to dataset characteristics, possibly due to differences in pathological distributions or signal quality. In contrast, QRS complex threshold displays consistent patterns across models, regardless of the dataset, highlighting its robustness to post-processing thresholds. For the worst-case configurations, the models generally show similar tendencies across all components. This uniformity can be attributed to similar thresholds for merging or removing segments for each component, which inadvertently leads to deviations from the ground truth. These findings underline the importance of carefully tuning gap and duration thresholds to align with the specific characteristics of each wave component and dataset.

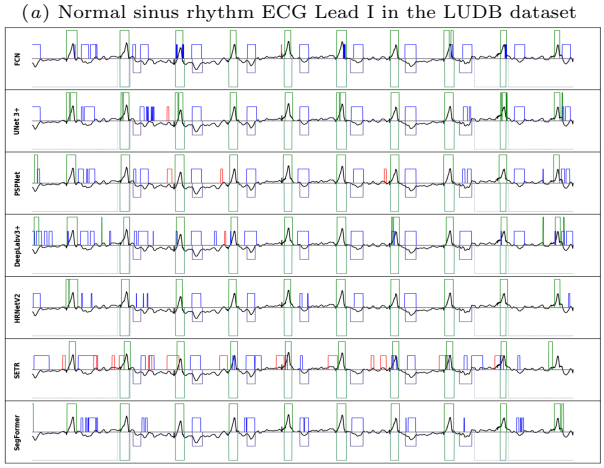
From a model-centric perspective, CNN-based approaches such as UNet 3+ and FCN often show equal or superior performance once segment consolidation and elimination is applied, indicating that post-processing can effectively boost or maintain baseline accuracy. This trend is observed on both LUDB and private datasets. Notably, FCN narrows its perfor-



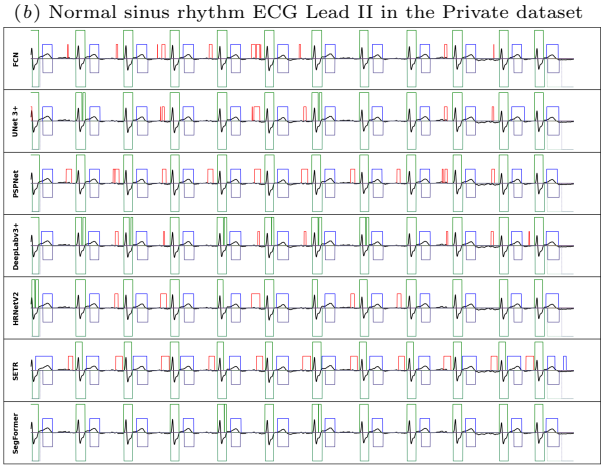
(*a*) Normal sinus rhythm ECG Lead I in the LUDB dataset



(*b*) Normal sinus rhythm ECG Lead II in the Private dataset

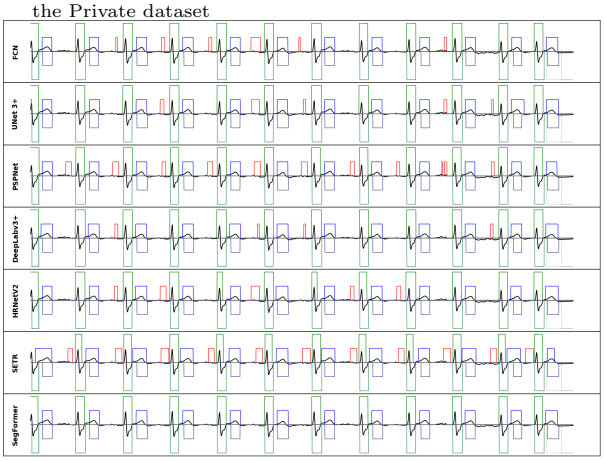


(*c*) Abnormal ECG Lead V1 with Atrial Fibrillation in the LUDB dataset



(*d*) Abnormal ECG Lead I with Right Bundle Branch Block in the Private dataset

(*e*) Post-processed segmentation result of Figure 1(*c*)



(*f*) Post-processed segmentation result of Figure 1(*d*)

Figure 1: Segmentation results of various ECG samples. The black line represents the ECG signal. The area above the baseline shows the predicted P wave (red), QRS complex (green), and T wave (blue), while the area below displays the ground truth, indicating the labeled boundaries for the P wave (salmon), QRS complex (seagreen), and T wave (darkslateblue).

Table 3: Performance evaluation on the postprocessing

Approach	Type	P		QRS		T		LUDB	Private
		Gap	Dur.	Gap	Dur.	Gap	Dur.	mIoU (± against Base)	
UNet 3+	Base			N/A				0.854	0.783
	Best	10 / 50	50 / 10	50 / 50	50 / 50	10 / 50	70 / 50	**0.858** (+0.004)	0.785 (+0.002)
	Worst	50 / 10	70 / 70	10 / 10	70 / 70	50 / 10	10 / 70	0.850 (−0.004)	0.773 (−0.010)
FCN	Base			N/A				0.853	0.785
	Best	10 / 30	50 / 10	50 / 50	50 / 50	50 / 50	70 / 70	**0.858** (+0.005)	**0.787** (+0.002)
	Worst	50 / 10	70 / 70	10 / 10	70 / 70	10 / 10	10 / 70	0.851 (−0.002)	0.778 (−0.007)
HRNetV2	Base			N/A				0.846	0.776
	Best	10 / 50	50 / 10	50 / 50	50 / 30	30 / 50	50 / 50	0.849 (+0.003)	0.777 (+0.001)
	Worst	10 / 10	70 / 70	10 / 10	70 / 70	50 / 10	70 / 70	0.841 (−0.005)	0.769 (−0.007)
SegFormer	Base			N/A				0.827	0.763
	Best	10 / 30	50 / 10	50 / 30	50 / 30	50 / 50	30 / 50	0.830 (+0.003)	0.765 (+0.002)
	Worst	50 / 10	70 / 70	10 / 10	70 / 70	10 / 10	70 / 70	0.820 (−0.007)	0.754 (−0.009)
DeepLabv3+	Base			N/A				0.819	0.741
	Best	50 / 50	50 / 30	50 / 50	30 / 30	30 / 50	50 / 50	0.823 (+0.004)	0.745 (+0.004)
	Worst	10 / 10	70 / 70	10 / 10	70 / 70	10 / 10	10 / 70	0.811 (−0.008)	0.728 (−0.013)
PSPNet	Base			N/A				0.819	0.762
	Best	10 / 30	50 / 10	50 / 30	70 / 10	30 / 50	50 / 50	0.822 (+0.003)	0.762 (-)
	Worst	50 / 10	70 / 70	10 / 10	10 / 70	50 / 10	10 / 70	0.814 (−0.005)	0.753 (−0.009)
SETR	Base			N/A				0.730	0.657
	Best	10 / 10	50 / 10	50 / 10	10 / 30	50 / 10	70 / 70	0.731 (+0.001)	0.657 (-)
	Worst	50 / 50	70 / 70	30 / 50	70 / 70	30 / 50	10 / 10	0.726 (−0.004)	0.651 (−0.006)

Note: *Base* refers to the configuration with no post-processing applied, *Best* represents the configuration with optimal thresholds, and *Worst* indicates the configuration with the thresholds that most deteriorate performance. *Gap* specifies the maximum allowable temporal distance (in milliseconds) between two segments for merging, while *Dur.* defines the minimum duration (in milliseconds) required to retain a segment after merging. Threshold values are separated using '/' for LUDB and the Private dataset.

mance gap with UNet 3+ on the LUDB dataset after post-processing, suggesting that this strategy can reduce the performance margin among top-performing models in real-world scenarios. For HRNetV2 and SETR, as shown in Table 2, the over-segmentation ratio was initially small, so the performance improvement effect of post-processing was not significant. Although post-processing led to performance improvements in SegFormer, DeepLabv3+, and PSPNet, it was not enough to surpass the top-performing approaches like UNet 3+ and FCN.

4.4. Performance Evaluation on Labels

Each LUDB sample is categorized by its rhythm and electric axis as summarized in Table 6 in the Appendix. Since the dataset includes varying proportions of labels, from the more prevalent Sinus rhythm to rarer labels like Biventricular pacing and Ventricular extrasystole, the evaluation focuses on how well each model manages these distribution imbalances. This label-wise breakdown highlights the extent to which each approach can maintain stable segmentation quality across heterogeneous ECG patterns.

Table 4 provides a consolidated view of the performance achieved by each approach across various rhythm and electric axis labels on the LUDB dataset. The left side of the table focuses on rhythm classes, whereas the right side presents performance on electric axis categories. Each cell indicates the ECG delineation performance for a specific label.

FCN and UNet 3+ consistently achieve strong performance, retaining relatively high accuracy in common categories such as Sinus rhythm and Normal electric axis. However, both models experience noticeable drops in underrepresented classes, such as Atrial fibrillation and Ventricular extrasystole, pointing to the need for additional data augmentation or refined architectures. HRNetV2 performed slightly worse than the top two approaches overall, achieving the highest performance on Ventricular extrasystole, but still experiencing a drop in performance on other rare labels. SegFormer, DeepLabv3+, and PSPNet exhibit moderate performance declines compared to top-performing approaches across both common and rare labels. SETR, on the other hand, shows broader difficulties across both common and rare labels, reflecting limited robustness to heterogeneous data.

Across all approaches, Sinus rhythm shows relatively high segmentation accuracy, underscoring the benefit of clearer P wave delineation and a more abundant training set. Conversely, Atrial fibrillation

Table 4: Performance evaluation with mIoU scores depending on labels

Approach	Rhythm			Electric Axis						
	Sinus rhythm	Sinus brady.	Atrial fib.	Norm.	Vert.	R. axis dev.	L. axis dev.	Hori.	Vent. extra.	Bivent. pacing
FCN	**0.869**	0.866	0.554	0.895	0.880	**0.873**	**0.848**	0.821	0.570	0.533
UNet 3+	0.868	**0.868**	**0.561**	**0.896**	**0.881**	0.869	0.847	**0.823**	0.549	**0.543**
HRNetV2	0.860	0.856	0.545	0.887	0.864	0.859	0.836	0.822	**0.571**	0.514
SegFormer	0.836	0.844	0.541	0.868	0.838	0.842	0.822	0.810	0.521	0.512
DeepLabv3+	0.829	0.837	0.539	0.864	0.833	0.835	0.805	0.801	0.552	0.521
PSPNet	0.832	0.840	0.540	0.861	0.836	0.839	0.819	0.784	0.537	0.520
SETR	0.754	0.774	0.470	0.793	0.768	0.750	0.756	0.707	0.414	0.415
Avg.	0.835	0.841	0.536	0.880	0.850	0.838	0.833	0.795	0.531	0.508

Note: Sinus brady. stands for Sinus bradycardia. Atrial fib. stands for Atrial fibrillation. Norm. stands for Normal. Vert. stands for Vertical. R. axis dev. stands for Right axis deviation. L. axis dev. stands for Left axis deviation. Hori. stands for Horizontal. Vent. extra. stands for Ventricular extrasystole. Bivent. pacing stands for Biventricular pacing

Table 5: Model size and performance across random seeds

Seed	UNet 3+	FCN	HRNetV2	SegFormer	DeepLabv3+	PSPNet	SETR
1	40 / 0.850	**715** / **0.856**	5 / 0.854	362 / 0.833	29 / 0.814	16 / 0.829	4 / 0.703
2	**240** / 0.856	229 / **0.857**	2 / 0.844	264 / 0.830	42 / 0.817	1 / 0.825	5 / 0.726
3	40 / **0.854**	**79** / 0.853	3 / 0.842	48 / 0.826	47 / 0.823	4 / 0.813	7 / 0.725
4	**240** / **0.860**	62 / 0.854	3 / 0.847	41 / 0.821	56 / 0.820	3 / 0.809	2 / 0.755
5	**115** / **0.853**	72 / 0.847	20 / 0.844	93 / 0.825	60 / 0.819	12 / 0.816	4 / 0.741
Avg.	135.0 / **0.854**	**231.4** / 0.853	6.6 / 0.846	161.6 / 0.827	46.8 / 0.819	7.2 / 0.819	4.4 / 0.730

Each cell is formatted as the parameter count (in millions) / mIoU score, respectively.

is problematic for every approach, mainly due to the absence of a distinct P wave that hampers boundary learning. Similar trends appear in electric axis labels: Normal and Vertical axis are more consistently segmented, while rare and clinically intricate conditions (Biventricular pacing, Ventricular extrasystole) remain difficult to delineate.

4.5. Comparison across Random Seeds

To ensure fairness in performance evaluation, five final models for each approach were selected using different random seeds, as described in Section 3.4. Each random seed determines the initialization, training, and resulting model structure. By evaluating the relationship between model size and segmentation performance, this investigation aims to identify trends, trade-offs between computational demands and performance, and the stability of various architectures. Table 5 summarizes the parameter count in millions and the segmentation performance ($mIoU$) for five random seeds.

Overall, larger models tend to achieve better performance, but this is not always the case. In some approaches, smaller models were selected during hyperparameter tuning because they provided superior results compared to their larger counterparts. This highlights that optimal model selection depends not only on size but also on tuning strategies and dataset characteristics.

UNet 3+ strikes a strong balance between accuracy and computational efficiency, making it a reliable choice for ECG delineation across various settings. FCN consistently achieves high performance, but its large and variable model size poses a potential computational burden in resource-constrained environments. HRNetV2 delivers competitive accuracy with a compact size, making it well-suited for real-time applications or environments with limited computational resources. SegFormer demonstrates moderate accuracy, but its high variability in model size across random seeds suggests that careful hyperparameter tuning is essential to achieve consistent performance. DeepLabv3+ achieves decent segmentation accuracy, but its performance lags behind other CNN-based architectures. Despite its relatively small model size, it does not offer a strong balance between efficiency and accuracy. PSPNet, while compact, exhibits lower performance compared to other architectures, making it a less competitive option for ECG delineation. SETR, which relies heavily on Transformer blocks, suffers from low segmentation accuracy and instability.

5. Conclusion

This study conducted a comprehensive benchmarking of CNN-based and Transformer-based semantic segmentation approaches for ECG delineation. CNN architectures such as UNet 3+ and FCN consistently achieved high accuracy and displayed strong generalizability across different datasets. In contrast, Transformer-based approaches exhibited greater variability, hinting at the need for further optimization tailored to biomedical signal processing. Meanwhile, compact architectures like HRNetV2 demonstrated a favorable balance between segmentation accuracy and computational efficiency, making them especially valuable for real-time or resource-constrained deployments. Across all methods, post-processing proved integral to consolidating fragmented segments or eliminating overly short segments, highlighting its importance in generating clinically reliable delineation outputs.

There are some limitations to this study. First, our work primarily focuses on CNN-based and early Transformer approaches. Consequently, we did not explore RNN-based or fully hybrid designs, nor did we incorporate the latest CNN-and Transformer-based approaches. Second, we only considered well-established Transformer variants, leaving other promising configurations unexplored. Third, the limited size and diversity of our datasets may restrict the generalization of findings to broader clinical populations. Finally, resource constraints restricted the scope of hyperparameter tuning, potentially impacting inter-model comparisons. For future work, we aim to develop an ECG delineation approach that is specialized for end-to-end processing—not only applying traditional segmentation methods but also integrating post-processing within the model itself. This holistic approach could improve accuracy and consistency by optimizing both segmentation and post-processing in a unified framework.

Institutional Review Board (IRB) This work was approved by the IRBs of Incheon Sejong Hospital and Bucheon Sejong Hospital in Republic of Korea (ISH 2024-09-001 and BSH 2024-09-003), and all procedures were conducted in accordance with the Declaration of Helsinki.

Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by Ministry of Science and ICT (MSIT) in the Korea government (RS-2024-00444014, Global AI-ECG-based Software Medical Device Approval and Commercialization International Collaboration)

References

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs, 2016. URL https://arxiv.org/abs/1412.7062.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017a.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017b. URL https://arxiv.org/abs/1706.05587.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

Ming Chen, Guijin Wang, Hui Chen, and Zijian Ding. Adaptive region aggregation network: Unsupervised domain adaptation with adversarial training for ecg delineation. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1274–1278, 2020. doi: 10.1109/ICASSP40776.2020.9053244.

Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in neural information processing systems*, 34:9355–9366, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani,

Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000 (June 13). Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

Huimin Huang, Lanfen Lin, Ruofeng Tong, Hongjie Hu, Qiaowei Zhang, Yutaro Iwamoto, Xianhua Han, Yen-Wei Chen, and Jian Wu. Unet 3+: A full-scale connected unet for medical image segmentation. In *ICASSP 2020-2020 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 1055–1059. IEEE, 2020.

Chankyu Joung, Mijin Kim, Taejin Paik, Seong-Ho Kong, Seung-Young Oh, Won Kyeong Jeon, Jaehu Jeon, Joong-Sik Hong, Wan-Joong Kim, Woong Kook, et al. Deep learning based ecg segmentation for delineation of diverse arrhythmias. *PloS one*, 19(6):e0303178, 2024.

Alexander Kuvaev and Roman Khudorozhkov. An attention-based cnn for ecg classification. In Kohei Arai and Supriya Kapoor, editors, *Advances in Computer Vision*, pages 671–677, Cham, 2020. Springer International Publishing. ISBN 978-3-030-17795-9.

Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-Tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. *Proceedings of Machine Learning and Systems*, 2:230–246, 2020.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

Viktor Moskalenko, Nikolai Zolotykh, and Grigory Osipov. Deep learning for ecg segmentation. In *Advances in Neural Computation, Machine Learning, and Cognitive Research III: Selected Papers from the XXI International Conference on Neuroinformatics, October 7-11, 2019, Dolgoprudny, Moscow Region, Russia*, pages 246–254. Springer, 2020.

Abdolrahman Peimankar and Sadasivan Puthusserypady. Dens-ecg: A deep learning approach for ecg signal delineation. *Expert systems with applications*, 165:113911, 2021.

Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters–improve semantic segmentation by global convolutional network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4353–4361, 2017.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578, 2021.

Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34:12077–12090, 2021.

Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6881–6890, 2021.

Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4*, pages 3–11. Springer, 2018.

Peter J Zimetbaum and Mark E Josephson. Use of the electrocardiogram in acute myocardial infarction. *New England Journal of Medicine*, 348(10): 933–940, 2003.

Appendix A. Details for LUDB Dataset

To ensure balanced representation and stratification of the LUDB dataset during its division for model training and evaluation, we utilized detailed labels to maintain consistent distributions across training, validation, and test datasets. Table 6 provides a comprehensive breakdown of the distribution for each category. Each of the 200 ECG samples is assigned a single label per category.

`no_p` indicates the presence (0) or absence (1) of the P wave in the ECG signal. This label was introduced by our team and is not part of the original LUDB dataset. `Rhythm` classifies the type of heart rhythm, including categories such as Sinus rhythm, Atrial fibrillation, and Bradycardia. `Electric axis of the heart` specifies the position of the electrical axis, with categories such as Normal, Left axis deviation, and Right axis deviation. Additionally, the dataset contains an `Etc` category, which encompasses diverse and unique labels that do not align with the primary categories. Due to the complexity and variability of the `Etc` category, it was excluded from the stratification process and is not represented.

Table 6: Distribution of labels for each category

Category	Number of samples
no_p	0: 176 1: 24
Rhythm	Sinus rhythm: 143 Sinus bradycardia: 25 Atrial fibrillation: 15 Sinus arrhythmia: 8 Sinus tachycardia: 4 Atrial flutter, typical: 3 Irregular sinus rhythm: 2
Electric axis of the heart	normal: 73 left axis deviation: 65 vertical: 26 horizontal: 19 III degree AV-block: 5 Wandering atrial pacemaker: 3 right axis deviation: 3 Undefined ischemia/scar/supp. NSTEMI: anterior wall: 1 Ventricular extrasystole, localisation: RVOT, anterior wall: 1 Incomplete right bundle branch block: 1 Biventricular pacing: 1 Left ventricular hypertrophy: 1 Aberrant conduction: 1

To prepare the data and perform a stratified split, we first combined labels from different categories into a single tuple for each ECG sample, as outlined in Table 7. These combined tuples served as the basis for stratification. For tuples with at least 10 occurrences, we distributed them proportionally across the dataset subsets. For tuples with fewer than 10 occurrences, we allocated them randomly to ensure representation.

This approach ensured that the proportional representation of key labels, such as `no_p`, `Rhythm`, and `Electric axis of the heart`, was preserved across all subsets. At the same time, it effectively addressed the sparsity of less common label combinations, maintaining a balanced dataset for model training and evaluation.

Table 8 presents the label composition across the training, validation, and test sets. The constructed dataset will be made publicly available, providing a transparent and standardized benchmark dataset for future research and fair performance comparisons.

Appendix B. Ablation Study: Model Performance Based on Hyperparameters

In `Stage 1: Model Structure Screening`, as described in Section 3.4, structural hyperparameters related to the model architecture were screened. In the previous section, only hyperparameters that recorded top performance were selected and used, while in this section, we aim to analyze the performance trends across all possible hyperparameter combinations.

The results below focus on IoU values obtained from the validation dataset. For each approach, IoU scores were evaluated across different hyperparameter combinations, with both mean and maximum scores considered in the analysis.

B.1. Fully convolution network (FCN)

Table 9 presents the hyperparameters influencing the architecture of the Fully Convolutional Network (FCN) and outlines the range of settings we employed. The original FCN study introduced the *FCN-32s* and *FCN-8s* architectures, distinguished by their upsampling strategies and use of skip connections. FCN-32s relies on a single upsampling step, enlarging the feature map by a factor of 32, often leading to coarser segmentation results. By contrast, FCN-8s employs skip connections to combine details from

Table 7: Distribution of labels combined by categories

Combined labels	Number of samples
(0, Sinus rhythm., normal.)	55
(0, Sinus rhythm., left axis deviation.)	49
(0, Sinus rhythm., vertical.)	17
(0, Sinus rhythm., horizontal.)	12
(0, Sinus bradycardia., normal.)	12
(1, Atrial fibrillation., left axis deviation.)	7
(0, Sinus bradycardia., left axis deviation.)	6
(1, Sinus rhythm., III degree AV-block.)	5
(0, Sinus bradycardia., vertical.)	3
(0, Sinus bradycardia., horizontal.)	3
(0, Sinus arrhythmia., normal.)	3
(0, Sinus arrhythmia., vertical.)	2
(0, Sinus rhythm., right axis deviation.)	2
(1, Atrial flutter, typical., vertical.)	2
(1, Atrial fibrillation., normal.)	2
(0, Irregular sinus rhythm., left axis deviation.)	2
(1, Atrial fibrillation., Undefined ischemia/scar/supp.NSTEMI: anterior wall)	1
(0, Sinus arrhythmia., horizontal.)	1
(0, Sinus rhythm., Incomplete right bundle branch block.)	1
(0, Sinus tachycardia., normal.)	1
(1, Sinus rhythm., Ventricular extrasystole, localisation: RVOT, anterior wall.)	1
(1, Atrial fibrillation., Left ventricular hypertrophy.)	1
(0, Sinus arrhythmia., right axis deviation.)	1
(1, Atrial fibrillation., vertical.)	1
(1, Atrial flutter, typical., horizontal.)	1
(1, Atrial fibrillation., Biventricular pacing.)	1
(1, Atrial fibrillation., Aberrant conduction.)	1
(0, Sinus tachycardia., left axis deviation.)	1
(0, Sinus bradycardia., Wandering atrial pacemaker.)	1
(0, Sinus tachycardia., vertical.)	1
(0, Sinus rhythm., Wandering atrial pacemaker.)	1
(1, Atrial fibrillation., horizontal.)	1
(0, Sinus arrhythmia., Wandering atrial pacemaker.)	1
(0, Sinus tachycardia., horizontal.)	1

Table 8: Distribution of labels in subsets

	Category	Number of samples
Training	no_p	0: 141 1: 19
	Rhythm	Sinus rhythm.: 115 Sinus bradycardia.: 18 Atrial fibrillation.: 11 Sinus arrhythmia.: 8 Atrial flutter, typical.: 3 Sinus tachycardia.: 3 Irregular sinus rhythm.: 2
	Electric axis of the heart	normal.: 58 left axis deviation.: 53 vertical.: 22 horizontal.: 14 III degree AV-block.: 5 Wandering atrial pacemaker.: 3 right axis deviation.: 2 Undefined ischemia/scar/supp.NSTEMI: anterior wall: 1 Incomplete right bundle branch block.: 1 Aberrant conduction.: 1
Validation	no_p	0: 18 1: 2
	Rhythm	Sinus rhythm.: 13 Sinus bradycardia.: 4 Atrial fibrillation.: 2 Sinus tachycardia.: 1
	Electric axis of the heart	normal.: 9 left axis deviation.: 6 horizontal.: 2 vertical.: 2 Left ventricular hypertrophy.: 1
Test	no_p	0: 17 1: 3
	Rhythm	Sinus rhythm.: 15 Sinus bradycardia.: 3 Atrial fibrillation.: 2
	Electric axis of the heart	left axis deviation.: 6 normal.: 6 horizontal.: 3 vertical.: 2 Ventricular extrasystole, localisation: RVOT, anterior wall.: 1 right axis deviation.: 1 Biventricular pacing.: 1

earlier layers, followed by upsampling by a factor of 8, which significantly improves segmentation precision.Consequently, FCN-8s demonstrated superior performance and was selected as the representative model Long et al. (2015).

Our experiments similarly validated the efficacy of combining features at intermediate resolution levels. Like FCN-8s, combining features from the 3rd or 4th layers (corresponding to 8x or 16x downsampled features) achieved the best results. While it was expected that leveraging features downsampled by 64x through 6 layers would result in better performance, no improvement was observed in (Figure 2(a). Also, combining overly early high-resolution features caused a decline in performance, emphasizing the importance of selecting optimal resolution layers for combination.

Additionally, we observed trends related to the *kernel size* and *dilation* hyperparameters. Larger values for both parameters consistently enhanced performance (Figure 2(b)), underscoring the critical role

Table 9: Details of FCN's hyperparameters

Name	Descriptions	Value
kernel_size	The kernel size used in the convolutions of each layer, excluding the last one.	`[3, 5, 7]`
last_layer_kernel_size	The kernel size of the first of two convolutions in the last layer of the network. The kernel size of the second one is fixed at 1.	`[5, 7, 9]`
inplanes	The scaling factor of the channels increases with each layer, excluding the last one. The output channels set with each layer as $inplanes \times (2^{\text{index of layer}})$. For example, when the inplanes is 32, the input channels and output channels in the second layer would be 32 and 64, respectively.	`[32, 64, 128]`
combine_conf	A hyperparameter that defines the total number of layers and determines up to which layer the combination of the convolution results from the next layer and the pooling results from the previous layer should be performed. For example, the following settings represent FCN8s `{ "num_layers": 5, "combine_until": 2 }`.	`[{"num_layers": 4, "combine_until": 0 }, { "num_layers": 4, "combine_until": 1 }, { "num_layers": 4, "combine_until": 2 }, { "num_layers": 4, "combine_until": 3 }, { "num_layers": 5, "combine_until": 0 }, { "num_layers": 5, "combine_until": 1 }, { "num_layers": 5, "combine_until": 2 }, { "num_layers": 5, "combine_until": 3 }, { "num_layers": 5, "combine_until": 4 }, { "num_layers": 6, "combine_until": 0 }, { "num_layers": 6, "combine_until": 1 }, { "num_layers": 6, "combine_until": 2 }, { "num_layers": 6, "combine_until": 3 }, { "num_layers": 6, "combine_until": 4 }, { "num_layers": 6, "combine_until": 5 }]`
num_convs	This refers to the number of convolutions in each layer, excluding the last one, where each layer consists of at least one convolution and one pooling operation.	`[2, 4, 6]`
dilation	The dilation rate used in the convolutions of each layer, except for the last layer.	`[1, 2, 4]`

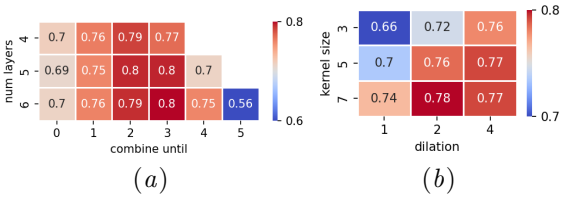


Figure 2: In FCN, The correlation between the mean IoU and both *num layers*, which refers to the depth of the network, and *combine until*, which defines up to which layer's features will be combined (Figure 2(*a*)). The mean IoU trend with respect to *kernel size* and *dilation* (Figure 2(*b*)).

of receptive field in ECG delineation. These results align with findings in CNN-based image semantic segmentation, where larger receptive field are crucial for capturing global context Peng et al. (2017).

B.2. UNet 3+

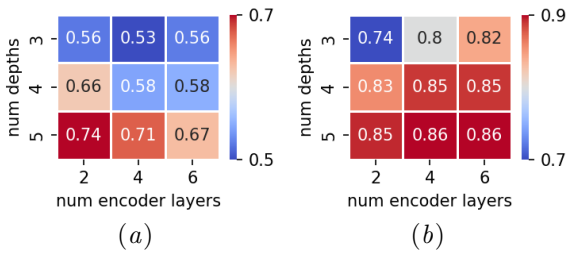


Figure 3: In UNet 3+, The correlation between the mean(3(*a*)), maximum(3(*b*)) IoU and both *num depths*, which refers to the depth of the network, and *num encoder layers*, which refers to the number of convolutional layers performed at each depth in the encoder part.

Table 10 highlights the hyperparameters that influence the architecture of UNet 3+ and the corresponding ranges employed. Our experiments revealed that increasing the network depth generally enhanced performance, with a depth of five layers yielding the highest mean and maximum IoU scores (Figure 3).

Table 10: Details of UNet 3+'s hyperparameters

Name	Descriptions	Value
kernel_size	The kernel size used in the convolutions of each layer.	[3, 5, 7]
inplanes	The scaling factor of the channels increases with each depth. The output channels set with each depth as $inplanes \times (2^{\text{index of depth}})$. For example, when the inplanes is 32, the input channels and output channels in the second depth would be 32 and 64, respectively.	[32, 64, 128]
num_depths	The number of stages where the resolution is halved while extracting higher-dimensional features.	[3, 4, 5]
num_encoder_layers	The number of times to perform the [Convolution, (Batch Norm), ReLU] combination at each depth. Batch norm may be omitted depending on the `encoder_batchnorm` variable below.	[2, 4, 6]
encoder_batchnorm	Indicates whether to perform Batch Norm in the layers of the encoder. A value of 0 means it is not used, while a value of 1 means it is used.	[0, 1]
interpolate_mode	The mode used for interpolation in the decoder.	["linear", "nearest"]
use_cgm	Indicates whether to use the Classification Guided Module (CGM). A value of 0 means it is not used, while a value of 1 means it is used.	[0, 1]
loss_fns_ratio	UNet 3+ uses a hybrid loss function. The values represent the contribution ratio of each loss to the total loss, in the order of Binary Cross Entropy Loss, MSSSIM Loss, and Dice Loss, with the sum totaling 1.	[[1.0, 0.0, 0.0], [0.0, 1.0, 0.0], [0.0, 0.0, 1.0], [0.5, 0.5, 0.0], [0.5, 0.0, 0.5], [0.0, 0.5, 0.5], [0.3333, 0.3333, 0.3333]]

This underscores the importance of leveraging features compressed by a factor of 32 for effective ECG delineation. However, performing too many convolutional layers at each depth resulted in diminishing performance improvements and occasionally led to a reduction in mean IoU. This suggests that excessively complex architectures may hinder the model's ability to effectively optimize weights across layers, leading to suboptimal results. Notably, UNet 3+ exhibited training instabilities, such as convergence failures or divergence, particularly when too many convolutional layers were used at each depth—occurring more frequently than with other approaches.

For the *loss function ratios*, the analysis demonstrated nuanced impacts on performance. Dice, MS-SSIM, and BCE losses sequentially contributed to higher mean IoU scores. However, achieving the highest maximum IoU emphasized the critical role of MS-SSIM loss, indicating a need for further refinement of the loss ratios. As noted by the authors of UNet 3+ Huang et al. (2020), the use of hybrid loss functions proved effective in boosting performance (Table 11).

These findings emphasize the need for judicious selection of network depth and optimization of loss function ratios to balance feature extraction capabilities and maintain training stability in ECG delineation tasks.

Table 11: In UNet 3+, Trends in mean IoU and maximum IoU according to the contribution ratio of each loss function. The *loss fns ratio* is in list form, and in order, it represents Binary Cross Entropy (BCE) loss, Multi-Scale Structural Similarity (MS-SSIM) loss, and Dice loss

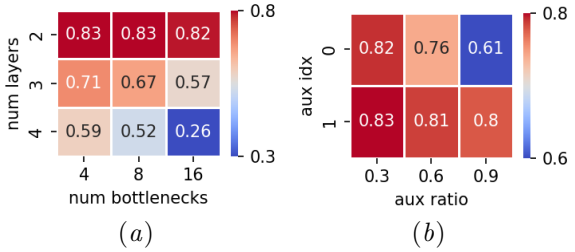
Loss fns ratio	MeanIoU	Loss fns ratio	MaxIoU
[0.0, 0.0, 1.0]	0.650	[0.333, 0.333, 0.333]	0.856
[0.0, 0.5, 0.5]	0.645	[0.5, 0.5, 0.0]	0.856
[0.333, 0.333, 0.333]	0.643	[0.0, 0.5, 0.5]	0.854
[0.5, 0.0, 0.5]	0.631	[0.0, 0.0, 1.0]	0.850
[0.5, 0.5, 0.0]	0.626	[0.5, 0.0, 0.5]	0.850
[0.0, 1.0, 0.0]	0.600	[1.0, 0.0, 0.0]	0.844
[1.0, 0.0, 0.0]	0.548	[0.0, 1.0, 0.0]	0.828

B.3. PSPNet

Table 12 presents the hyperparameters influencing the architecture of PSPNet and the corresponding ranges we employed. In our experiments, as depicted in Figure 4(*a*), utilizing features compressed by a factor of 16 in the backbone network yielded the highest IoU scores. Further compression led to decreased performance. This outcome differs slightly

Table 12: Details of PSPNet's hyperparameters

Name	Descriptions	Value
kernel_size	The kernel size used in the convolutions throughout the network.	`[3, 5, 7]`
expansion	PSPNet is based on ResNet and incorporates a bottleneck structure. This variable represents the expansion in the bottleneck structure.	`[2, 3, 4]`
inplanes	The scaling factor of the channels increases with each layer.	`[32, 64, 128]`
num_layers	The number of layers after the stem stage that include one downsampling operation and at least one bottleneck structure.	`[2, 3, 4]`
num_bottlenecks	The number of bottlenecks in each layer. By default, one bottleneck is included. If this value is set to 4, a total of 5 bottlenecks are performed within a single layer.	`[4, 8, 16]`
interpolate_mode	The mode used for upsampling in the Pyramid Pooling Module (PPM). This mode is also used for upsampling to match the output with the input length.	`["linear", "nearest"]`
dilation	The dilation rate used in the convolutions of each layer, except for the last layer.	`[1, 2, 4]`
ppm_bins	The list of bin sizes to be used in the PPM.	`[[1, 2, 3, 6], [1, 2, 4, 8]]`
aux_idx	The index of the layer used to calculate the auxiliary loss. Therefore, this value must be less than `num_layers`.	`[0, 1]`
aux_ratio	The ratio at which the auxiliary loss is incorporated into the total loss.	`[0.3, 0.6, 0.9]`



(a) (b)

Figure 4: In PSPNet, Figure 4(*a*) represents the comparison of the changes in maximum IoU with variations in *num layers*, which refers to the number of layers that include downsampling and at least one BottleNeckResidualBlock after the stem of the ResNet backbone, and *num bottlenecks*, which refers to the number of BottleNeckResidualBlocks within each layer. Figure 4(*b*) shows the comparison of the changes in maximum IoU with variations in *aux idx*, which indicates the index of the layer where auxiliary loss is calculated, and *aux ratio*, which refers to the contribution ratio of the auxiliary loss to the total loss.

from the results observed with FCN and UNet 3+, suggesting that the Pyramid Pooling Module (PPM) in PSPNet effectively leverages highly compressed features, significantly impacting performance. Additionally, incorporating an excessive number of BottleNeck Residual Blocks appeared to impede convergence to an optimal model.

Regarding auxiliary loss, our findings align with those reported by the authors of PSPNet Zhao et al. (2017), as shown in Figure 4(*b*). Calculating auxiliary loss using features compressed by a factor of 16, rather than 8, contributed to achieving higher IoU scores. Moreover, integrating auxiliary loss into the total loss with a ratio of 0.3 resulted in optimal performance.

These insights highlight the importance of appropriately selecting feature compression levels and auxiliary loss configurations to enhance the effectiveness of PSPNet in segmentation tasks.

B.4. DeepLabv3+

Table 13 presents the hyperparameters influencing the architecture of DeepLabv3+ and the corresponding ranges we employed. In our experiments, as shown in Figure 5, we observed that smaller kernel sizes and dilation values led to higher IoU scores. This trend contrasts with findings from FCN, likely due to DeepLabv3+'s Atrous Spatial Pyramid Pooling (ASPP) module, which already provides a sufficiently large receptive field. It suggests that beyond a certain point, increasing the receptive field may introduce excessive information for a single output prediction, resulting in performance degradation. Fur-

Table 13: Details of DeepLabv3+'s hyperparameters.

Name	Descriptions	Value
output_stride	DeepLabv3+ uses Xception as the encoder. After passing through Xception, the output size can be adjusted, and the output will be downsampled by the specified factor.	`[8, 16]`
middle_block_rate	The dilation rate to be used in the middle block of Xception.	`[1, 2, 3]`
middle_repeat	The number of middle blocks in Xception.	`[8, 16, 24]`
exit_block_rates	The dilation rate to be used in the final part of convolution in Xception.	`[[1, 2], [2, 4], [4, 8]]`
kernel_size	The kernel size to be used in both the Xception and Atrous Spatial Pyramid Pooling (ASPP).	`[3, 5, 7]`
interpolate_mode	The mode used for interpolation when resizing the final output to match the input size.	`["linear", "nearest"]`
aspp_channel	The output channels in the ASPP.	`[128, 256]`
aspp_rate	The dilation rates to be used in the ASPP.	`[[1, 9, 18, 27], [1, 12, 24, 36], [1, 15, 30, 45]]`

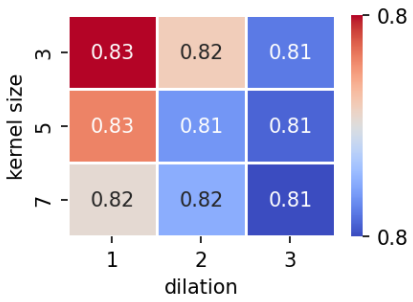


Figure 5: In DeepLabv3+, Trends in maximum IoU based on variations in the *kernel size* of the convolutional layers in Xception and Atrous Spatial Pyramid Pooling(ASPP), and the *dilation* values of the middle blocks in Xception.

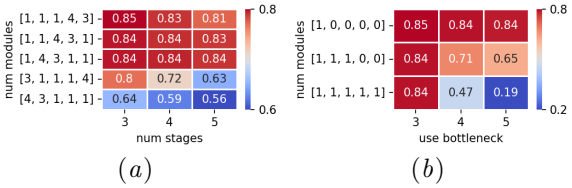


Figure 6: In HRNetV2, Figure 6(a) represents the trend of maximum IoU with changes in *num stages*, which refers to the number of stages, and *num modules*, which refers to the number of HRModules at each stage. Figure 6(b) shows the trend of maximum IoU with changes in *num stages* and *use bottleneck*, which indicates whether to use BottleNeckResidualBlock(1) or BasicResidualBlock(0) in the HRModule.

thermore, modifying the dilations and channel configurations within the ASPP module did not yield significant performance changes.

These insights highlight the importance of appropriately selecting kernel sizes and dilation rates to optimize the balance between receptive field size and segmentation accuracy in DeepLabv3+.

B.5. HRNetV2

Table 14 presents the hyperparameters influencing the architecture of HRNet and the corresponding ranges we employed. Our findings emphasize that simplicity often yields superior results. As shown in Figure 6(a), utilizing features compressed up to stage 3 (16x compression) resulted in better performance compared to using features compressed to 32x. This outcome contrasts with results from FCN and UNet 3+, where higher compression (32x) improved performance.

Excessive stacking of HRModules in each stage adversely affected the ability to learn optimal weights. This performance degradation was more pronounced when a large number of HRModules were used in high-resolution stages (Figure 6(a)). Additionally, in the later stages of the network, substituting the more complex BottleNeckResidualBlock with the simpler BasicResidualBlock contributed to improved IoU scores (Figure 6(b)).

These findings underscore the importance of balancing architectural complexity and simplicity to

Table 14: Details of HRNetV2+'s hyperparameters

Name	Descriptions	Value
kernel_size	The kernel size used in the convolutions throughout the network.	`[3, 5, 7]`
dilation	The dilation rate used in the convolutions throughout the network.	`[1, 2, 4]`
num_stages	The number of stages in HRNet where the number of branches increases by one.	`[3, 4, 5]`

num_modules	A list specifying the number of HRModules in each stage. Each HRModule consists of `num_block` blocks (such as ResNet's BottleNeckBlock or BasicBlock) and one fusion layer. The length of the list must be greater than or equal to the value of `num_stages`.	`[[1, 1, 4, 3, 1],`
		`[1, 1, 1, 4, 3],`
		`[3, 1, 1, 1, 4],`
		`[4, 3, 1, 1, 1],`
		`[1, 4, 3, 1, 1]]`

use_bottleneck	A list specifying whether to use BottleNeckBlock in the HRModules of each stage. A value of 0 in the list indicates the use of BasicBlock, while a value of 1 indicates the use of BottleNeckBlock. The length of the list must be greater than or equal to the value of `num_stages`.	`[[1, 0, 0, 0, 0],`
		`[1, 1, 1, 0, 0],`
		`[1, 1, 1, 1, 1]]`

num_blocks	The number of repetitions of blocks within a HRModule.	`[4, 6]`
stage1_channels	The number of channels to amplify in the stem stage.	`[64, 128]`
num_channels_init	The input channel of the first branch starting from stage 2, with each subsequent branch having twice the number of channels as the previous one. In the original paper, this is denoted as C.	`[32, 48]`
interpolate_mode	The mode used for upsampling during the fusion stage of the HRModule and at the end of the network to resize the output to the input size.	`["linear", "nearest"]`

maximize HRNet's performance for ECG delineation tasks.

B.6. SETR

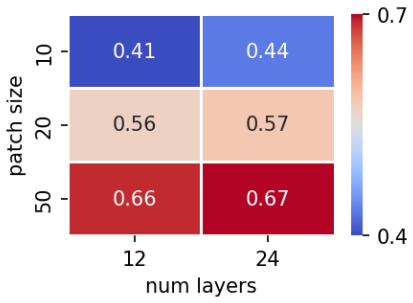


Figure 7: In SETR, Trends in maximum IoU based on variations in patch size, which represents the length of points embedded into a single patch in the ViT encoder, and num layers, which refers to the number of transformer blocks.

Table 15 presents the hyperparameters influencing the architecture of SETR and the corresponding ranges we employed. Key hyperparameters affecting SETR's performance include patch size, the number of transformer layers, and the decoder type. As shown in Figure 7, extremely small patch sizes resulted in a marked decline in performance, likely due to insufficient feature extraction. While increasing the number of transformer layers improved results, excessively large patch sizes (greater than 100) also negatively impacted performance, suggesting a trade-off between patch size and the model's ability to generalize effectively.

The authors of SETR proposed three decoder types: `naive`, `pup`, and `mla`, with `pup` and `mla` demonstrating superior performance to `naive` on various datasets Zheng et al. (2021). In our experiments on ECG delineation, `pup` consistently outperformed both `mla` and `naive` (Table 16). However, the performance of `mla` appeared highly sensitive to specific configurations, suggesting that further fine-tuning could enable it to rival or surpass `pup`.

These results emphasize the importance of hyperparameter selection and fine-tuning to maximize SETR's potential in ECG delineation tasks, with further performance improvements requiring more intricate and extensive efforts.

B.7. SegFormer

Table 15: Details of SETR's hyperparameters

Name	Descriptions	Value
embed_dim	The embedding dimension in the patch embedding stage.	`[1024, 2048]`
patch_size	The patch size in the patch embedding stage.	`[10, 20, 50, 100, 200, 250]`
patch_bias	The bias of convolution in the patch embedding stage.	`[0, 1]`
num_layers	The number of the transformer blocks.	`[24, 48]`
num_attn_heads	The number of attention heads in the transformer block.	`[8, 16]`
attn_head_dim	The dimension per attention head in the transformer block.	`[64, 128]`
mlp_dim	The dimension of the feedforward in the transformer block.	`[2048, 4096]`
interpolate_mode	The mode used for interpolation when resizing the final output to match the input size.	`["linear", "nearest"]`

dec_conf	This represents the decoder mode(`Naive`, `Progressive UPsampling(PUP)`, `Multi-Level feature Aggregation(MLA)`) and its corresponding configurations.	`[{ "naive": {} },`
		`{`
		`  "pup": {`
		`  "kernel_size": [1, 3],`
		`  "channels": [256, 512],`
		`  "num_convs_by_layer": [3, 6],`
		`  "up_scale": [4, 2]`
		`  }`
		`},`
		`{`
		`  "mla": {`
		`  "output_step": [12, 6],`
		`  "kernel_size": 1,`
		`  "channels": [128, 64],`
		`  "num_convs_by_layer": [2, 3],`
		`  "up_scale": 4`
		`  }`
		`}]`

Table 16: In SETR, The maximum IoU changes based on the type of decoder and its detailed configurations. *Progressive UPsampling (pup)*, *Multi-Level feature Aggregation (mla)*, and *Naive upsampling (naive)* all have the same meanings as explained by the authors of SETR Zheng et al. (2021). Due to space constraints, options with lower performance have been omitted.

Decoder type	**MaxIoU**
`{"pup": {"channels":256, "kernel_size":3, "num_convs_by_layer":3, "up_scale":4}}`	0.673
`{"pup": {"channels":128, "kernel_size":3, "num_convs_by_layer":3, "up_scale":4}}`	0.663
`{"mla": {"channels":128, "kernel_size":1, "num_convs_by_layer":2, "output_step":6, "up_scale":4}}`	0.450
`{"naive":{}}`	0.449
`{"mla": {"channels":128, "kernel_size":1, "num_convs_by_layer":2, "output_step":4, "up_scale":4}}`	0.446

Table 17 presents the hyperparameters influencing the architecture of SegFormer and the corresponding ranges we employed. SegFormer utilizes overlapped patches with an initial 4x compression, progressively increasing compression scales in subsequent stages (e.g., 8x, 16x) to process transformer blocks. Our experiments, detailed in Table 18, involved varying both the number of stages and the number of transformer blocks per stage. The results indicate that employing patches compressed by factors of 32x or 64x, and extracting substantial information from these highly compressed patches, led to improved performance.

These findings suggest that deeper hierarchical structures with higher compression rates enable SegFormer to capture both local and global features effectively, enhancing its performance in ECG Delineation tasks.

B.8. Performance Comparison by Interpolation Mode

Each approaches employs a resizing process to align the dimensions of the input and output data after passing through the network. FCN handles resizing by padding the input to closely match the output size, followed by cropping the excess. Other approaches, however, rely on interpolation methods, typically using either the `linear` or `nearest` mode for 1D data.

To evaluate the impact of interpolation modes on segmentation performance, we analyzed the maximum IoU achieved by different algorithms under both linear and nearest modes (Figure 8). Contrary to expectations that interpolation mode would have minimal influence, significant differences were observed in specific cases. Notably, PSPNet demonstrated superior performance with the linear mode compared to the nearest mode, achieving a maximum IoU of 0.83 with linear interpolation versus 0.79 with nearest interpolation. Meanwhile, most other approaches, such as UNet 3+, DeepLabv3+ and HRNet, maintained consistent performance across interpolation modes.

These findings underscore the importance of even seemingly minor hyperparameters, such as interpolation mode, in achieving optimal performance. While the choice of interpolation mode may not drastically affect all algorithms, its impact on approaches like PSPNet highlights the need for careful tuning, particularly in ECG delineation tasks.

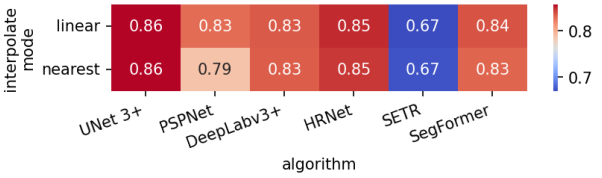


Figure 8: Comparison of maximum IoU values for each approaches based on different *interpolate modes*.

Table 17: Details of SegFormer's hyperparameters

Name	Descriptions	Value

num_blocks	Here, we define a "stage" as the unit where transformer blocks are executed at each step, with progressively increasing patch sizes of 4, 8, 16, 32, and so on. This variable represents both the number of stages and the number of transformer blocks performed at each stage. The configuration `[9, 9, 9, 9]` defines a network architecture consisting of 4 stages, each comprising 9 transformer blocks.	`// num_stages: 3`
		`[15, 15, 15],`
		`[5, 10, 15],`
		`[5, 15, 10],`
		`// num_stages: 4`
		`[9, 9, 9, 9],`
		`[2, 4, 6, 8],`
		`[2, 4, 8, 6],`
		`[2, 4, 8, 16],`
		`[2, 4, 16, 8],`
		`[3, 6, 9, 12],`
		`[3, 6, 12, 9],`
		`[3, 6, 12, 24],`
		`[3, 6, 24, 12],`
		`[4, 8, 12, 16],`
		`[4, 8, 16, 12],`
		`// num_stages: 5`
		`[7, 7, 7, 7, 7],`
		`[2, 4, 6, 8, 10],`
		`[2, 4, 6, 10, 8],`
		`[2, 4, 10, 8, 6],`
		`// num_stages: 6`
		`[4, 4, 4, 4, 4, 4],`
		`// num_stage: 7`
		`[2, 2, 2, 2, 2, 2, 2]`

embed_dim	The dimension of the patch embedding is initialized using this variable. In each stage, the embedding dimension is calculated as the product of `embed_dim` and the number of heads in the Multi-Head Attention mechanism.	`[32, 64]`

num_heads	This variable is used to calculate the number of heads in the Multi-Head Attention mechanism for each stage. To accommodate the randomly selected number of stages during training, the calculation is based on the following parameters: `["coefficient", "scale factor", "rounding method"]`. For example, with `[1, 2.1, "floor"]` and 5 stages, the `num_heads` is recalculated as `[1, 2, 4, 9, 19]`($[1,\ 1 \times floor(2.1),\ 1 \times floor((2.1)^2),\ 1 \times floor((2.1)^3),\ 1 \times floor((2.1)^4)]$) and used accordingly in the model.	`[[1, 2, "ceil"],`
		`[1, 2.1, "floor"],`
		`[1, 2.1, "ceil"]]`

sr_ratios	The ratio used for Efficient Self-Attention in each stage follows the same structure and implementation as the `num_heads` variable in the code.	`[1, 1, "ceil"], [1, 2, "ceil"]`
mlp_ratio	The dimension in each stage's Mix-FFN is calculated using the formula: `mlp_ratio` $\times$ `embed_dim[stage]`	`[4, 8]`
decoder_channels	The variable that adjusts the output channel of the transformer block in each stage is denoted as C in the original paper.	`[256, 512]`
interpolate_mode	The mode used for interpolation when matching the length of the MLP Layer outputs for concatenation and adjusting the final output size to match the input size.	`["linear", "nearest"]`

Table 18: The correlation between *num blocks*, which represents the number of transformer blocks in each stage, and maximum IoU in SegFormer

Num Blocks	**MaxIoU**	**Num Blocks**	**MaxIoU**
[2, 4, 6, 8, 10]	0.836	[3, 6, 12, 9]	0.830
[2, 4, 6, 10, 8]	0.835	[4, 8, 16, 12]	0.830
[2, 4, 10, 8, 6]	0.834	[4, 8, 12, 16]	0.829
[2, 4, 8, 6]	0.833	[3, 6, 9, 12]	0.826
[2, 4, 6, 8]	0.833	[2, 2, 2, 2, 2, 2, 2]	0.825
[4, 4, 4, 4, 4, 4]	0.832	[9, 9, 9, 9]	0.818
[2, 4, 8, 16]	0.832	[3, 6, 24, 12]	0.818
[3, 6, 12, 24]	0.832	[5, 10, 15]	0.810
[7, 7, 7, 7, 7]	0.832	[15, 15, 15]	0.807
[2, 4, 16, 8]	0.831	[5, 15, 10]	0.791

Appendix C. Correlation between Performance and Over Segmentation Ratios Across Leads

Table 19 and Table 20 present performance and over-segmentation ratio across leads. By analyzing the contradictory relationship between performance and over-segmentation through these tables, insights into it were gained.

There are some interesting observations regarding FCN and SETR. FCN demonstrates top-level segmentation performance across most leads, but it also shows top-level over-segmentation ratios. Many of the small, fragmented predicted segments appear to cover a large portion of the ground truth. SETR, on the other hand, shows the complete opposite pattern, where it predicts fewer but cohesive segments, which, however, seem to deviate from the ground truth.

UNet 3+ demonstrates stable and high performance across most leads, with over-segmentation ratios appearing at an average level. This indicates its robustness in segmenting ECG waveforms consistently, showing outstanding performance on leads such as III, aVL, and V1, where other approaches struggle. Similarly, HRNetV2 demonstrates balanced and top-tier performance across most leads, with over-segmentation ratios significantly lower than those of UNet 3+. This result suggests that HRNetV2 is a highly reliable choice when cleaner segmentation outputs are desired.

These findings emphasize the importance of evaluating segmentation models not only on overall metrics but also on their consistency across individual leads. Models like UNet 3+ and HRNetV2 stand out for their stable performance, making them suitable for practical applications where uniform segmentation across all leads is critical.

Table 19: Performance comparison for each lead with mIoU scores on LUDB

Approach	**I**	**II**	**III**	**aVR**	**aVL**	**aVF**	**V1**	**V2**	**V3**	**V4**	**V5**	**V6**
FCN	0.840	0.876	0.827	0.857	**0.789**	**0.838**	0.788	**0.857**	0.894	**0.884**	**0.895**	**0.874**
UNet 3+	**0.841**	**0.881**	**0.832**	**0.866**	0.787	0.836	**0.790**	**0.857**	**0.896**	0.882	0.888	**0.874**
PSPNet	0.818	0.845	0.790	0.828	0.746	0.813	0.749	0.819	0.854	0.844	0.855	0.836
DeepLabv3+	0.814	0.860	0.781	0.827	0.748	0.802	0.749	0.812	0.850	0.852	0.856	0.844
HRNetV2	0.835	0.869	0.823	0.853	0.772	0.838	0.776	0.845	0.885	0.876	0.884	0.871
SETR	**0.733**	**0.781**	**0.680**	**0.736**	**0.645**	**0.695**	**0.680**	**0.734**	**0.758**	**0.763**	**0.779**	**0.763**
SegFormer	0.816	0.859	0.784	0.820	0.765	0.819	0.772	0.826	0.860	0.859	0.865	0.856
Avg.	0.814	0.853	0.788	0.827	0.750	0.806	0.758	0.821	0.857	0.851	0.860	0.845

Table 20: Over-segmentation ratio on LUDB by lead

Approach	**I**	**II**	**III**	**aVR**	**aVL**	**aVF**	**V1**	**V2**	**V3**	**V4**	**V5**	**V6**
FCN	**117.9**	**111.2**	117.0	**118.0**	**120.9**	113.2	**123.5**	**109.8**	**107.3**	**108.7**	109.2	**114.1**
UNet 3+	117.7	107.5	112.3	116.0	116.5	108.6	113.6	108.8	103.7	105.2	107.2	108.2
PSPNet	107.2	**104.7**	105.0	105.9	105.0	**103.3**	104.9	103.2	104.0	104.7	105.4	106.0
DeepLabv3+	114.8	107.3	112.7	109.0	114.3	109.5	110.8	107.0	106.4	108.5	**109.9**	109.8
HRNetV2	107.7	104.9	107.1	107.4	107.2	105.3	107.9	103.1	**102.2**	103.4	**105.2**	105.4
SETR	**103.6**	105.5	**101.3**	**105.7**	**104.0**	104.9	**104.4**	**103.0**	105.0	**102.9**	105.4	**104.9**
SegFormer	116.3	109.7	**118.4**	115.1	119.7	**114.4**	116.7	107.5	106.7	108.5	109.5	109.2
Avg.	112.2	107.3	110.5	111.0	112.5	108.2	111.7	106.1	105.0	106.0	107.4	108.2