Automated Digitization of Paper ECG Records Using Convolutional Networks: a Faster R-CNN and U-Net Approach

Haoliang Shang[1], Clemens Hutter[1], Yani Zhang[1]

[1] Department of Information Technology and Electrical Engineering, ETH Zurich, Switzerland

Abstract

As part of the George B. Moody PhysioNet Challenge 2024, we developed a deep learning model based on detection and segmentation to recover electrocardiogram (ECG) time series from ECG record printouts. Our team, mins-eth, designed a hybrid pipeline of convolutional neural networks (CNNs) that leverages the strength of Faster Region-based Convolutional Neural Network (Faster R-CNN) for precise detection of the signals and that of U-Net for pixel-level accurate segmentation. Our model can handle a variety of distortions present in scanned ECG records, including rotation, cropping, creases, as well as text artifacts, and efficiently identifies and extracts ECG waveforms. For the digitization task, our model received an SNR of 0.893 (ranked 6/16) on the hidden test set.

1. Introduction

ECG records are vital tools in the diagnosis and monitoring of cardiac conditions. Despite the increasing shift toward electronic health records, a significant volume of ECG data remains in paper form, particularly within older records. Existing digitization methods often involve manual input and lack the robustness to handle the wide variability in ECG paper formats and annotations. We participate in the 2024 George B. Moody PhysioNet Challenge [1–3], which invites teams to develop automated and open-source [1] tools for digitizing ECG paper printouts.

2. Method

2.1. Assumptions

For this task, we make the following assumption: The ECG scan adheres to the standard format with 12 short leads and 1 long lead [4]. Deviations from this format may result in inaccurate read-out results. This issue could be addressed by incorporating an optical character recognition (OCR) module prior to the reading-out step.

[1]Code available at `github.com/MINSMoody/ECG_Paper_Digitization_via_Faster_R-CNN_and_U-Net`

2.2. Data generation and preprocessing

We are provided with ECG records in WFDB format [5, 6]. We perform two main steps to transform the ECG signals into a 2-D image before feeding it into the model.

Image generation and augmentation. Given digital ECG signals, we generate synthetic ECG images using ECG-Image-Kit [7]. Specifically, we introduce different types of noise and distortions to mimic the artifacts induced by the process of printing and scanning in the real world. A summary of the randomly added noise can be found in Table 1.

Category	Details	Probability
Paper format	**Paper margins:** randomly added to the top, bottom, left or right.	0.2
	Calibration pulse: randomly added at the beginning of each row.	0.7
Grids artifacts	**Intersection artifacts:** grid intersection points randomly marked with dots.	0.3
	Gridline Removal: vertical and horizontal gridlines are removed completely.	0.3
Noise	**Gaussian Noise:** per-pixel Gaussian noise added to simulate natural noise.	0.2
	Poisson noise (shot noise): added to mimic the quality loss during photo shooting.	0.3
Wrinkles and streaks	**Paper wrinkles:** randomly added to the paper surface to simulate wear and tear.	0.4

Table 1. List of artifacts.

Preprocessing. Following the CoCo-format [10], which is the standard format for object detection and instance segmentation tasks, we convert the ground truth masks, i.e., which pixels in the generated images stand for signals, for the ECG signals into Run Length Encoding (RLE). Cropped single lead images and masks are randomly saved for the training of the segmentation network. The triggering of the saving step follows a Bernoulli distribution with a probability of 0.1, which ensures fair sampling across all samples.

2.3. Model architecture

We break down the ECG digitization process into the following steps: signal detection, signal pixel segmentation, and reading-out. There is a plethora of neural network architectures for the detection and segmentation tasks in the literature, see e.g. [11, 12]. We design our pipeline based on Faster R-CNN [8] and U-Net [9] due to their superior performance on the ECG digitization task, as detailed in this subsection.

An overview of our pipeline is demonstrated in Figure 1. After the input data is transformed into the standard format as described in Section 2.2, it is passed to the detection model based on Faster R-CNN [8], which crops out single ECG leads. The cropped-out leads are then processed by the segmentation model based on U-Net [9] to remove the gridlines and any noise in the background. Since the detection model is trained on the entire ECG scans and the segmentation model is trained on single crop-outs, the two models can be trained in parallel, enabling efficient and specialized learning.

Faster R-CNN [8]. The Faster R-CNN model is a deep learning architecture designed for object detection. In our pipeline, Faster R-CNN is used to detect and localize regions that contain all waveform signals within the entire ECG scans. The model begins by extracting feature maps from the input ECG scans using a convolutional backbone network—ResNet101 [13] in our implementation. These feature maps are then processed through a Region Proposal Network (RPN) [8] that generates candidate regions likely to contain the ECG signal. For each proposed region, the network predicts the class label and refines the bounding box coordinates, allowing Faster R-CNN to precisely isolate the ECG waveforms even in the presence of noise, distortions, and other extraneous markings on the paper scans.

U-Net [9]. The U-Net architecture is characterized by its symmetric "U" shape, consisting of a contracting path—the socalled encoder—and an expansive path—the socalled decoder. This model is a convolutional neural network specifically designed for pixel-level image segmentation tasks and is particularly well-suited for tasks involving ill-defined objects with little area and elongated shape. In our pipeline, we employ U-Net to segment the ECG waveforms from the crops generated by the Faster R-CNN [8] model. The encoder compresses the input image into a low-dimensional feature representation by successively applying convolutional layers followed by max-pooling operations. This process captures high-level features and reduces spatial dimensions. The decoder then upsamples these compressed features back to the original resolution, using a series of up-convolutions and concatenations with corresponding layers from the encoder. This skip connection mechanism allows the model to retain fine-grained high-level spatial information that is crucial for accurate segmentation.

For the ECG digitization task, by leveraging both the contracting and expansive paths, U-Net effectively learns to differentiate between the ECG waveform pixels and the background, producing a precise binary mask that outlines the waveform. This segmentation step is essential for generating a clean, noise-free digital representation of the ECG signals, ready for the extraction of 1-D time series.

2.4. Reading-out and postprocessing

Sorting. Since the bounding box prediction result from the detection model is random in relative positions, we sort the bounding boxes into the standard format as mentioned in Section 2.1. Algorithm 1 summarizes our sorting procedure. A bounding box box is a list of the form $[\ell, t, r, b]$, representing the left, top, right, and bottom coordinates of the boundary. Each row $\boldsymbol{B}_i$, for $i \in 1, 2, 3, 4$, is of shape $n \times 4$, where $n = 4$ if the detection model successfully detects all four leads in a row, otherwise $n < 4$. Algorithm 1 fills up potential missing bounding boxes by aligning with other leads and outputs in total 13 bounding boxes.

Reading-Out. To transform 2-D masks generated by the detection model into 1-D time series, we vertically scan each column of the mask to search for the coordinates of the signal pixels. Specifically, on a mask that represents the ground truth, within each column, white pixels stand for ECG signals while black pixels are the background. The masks produced by our segmentation model contain noise pixels aside from the ECG signals, see Figure. 2 for an example. To mitigate the effect of such noise introduced in the segmentation step, we scan through each column and take the median of the coordinates of all white pixels to compute the signal level in mV.

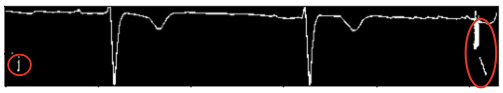

Figure 2. A segmentation example generated by our model. Artifacts are marked in red.

Postprocessing. We apply a series of postprocessing steps to remove noise introduced in the reading-out

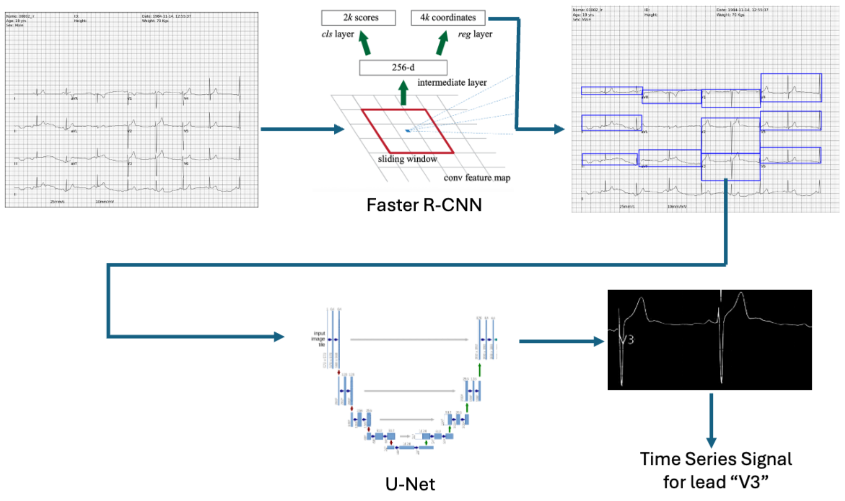

Figure 1. Pipeline overview. Illustrations of Faster R-CNN and U-Net are taken from [8] and [9], respectively.

step. Specifically, we apply Cubic Spline interpolation of NaN values [14] to correct missing data, Wavelet decomposition [15] for frequency-based denoising, and resampling [14] to ensure consistency with standard formats.

Algorithm 1 Bounding Boxes Sorting Algorithm

```
Input: lists of bounding boxes $\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3, \mathbf{B}_4$
  Step 1: Calculate the left and right boundaries
  $I \leftarrow$ indices of rows with 4 bounding boxes
  $\ell \leftarrow \mathit{median}\{\mathbf{B}_i[0,0] : i \in I\}$
  $r \leftarrow \mathit{median}\{\mathbf{B}_i[2,3] : i \in I\}$
  $m \leftarrow \frac{r-\ell}{2}, \quad m\ell \leftarrow \frac{\ell+m}{2}, \quad mr \leftarrow \frac{r+m}{2}$
  Step 2: Adjust the left and right boundaries of each bounding box
  for $i \in I$ do
      $\boldsymbol{B}_i[0,0] \leftarrow \ell$, $\boldsymbol{B}_i[0,2] \leftarrow m\ell$, $\boldsymbol{B}_i[1,0] \leftarrow m\ell$
      $\boldsymbol{B}_i[1,2] \leftarrow m$, $\boldsymbol{B}_i[2,0] \leftarrow m$
      $\boldsymbol{B}_i[2,2] \leftarrow mr$, $\boldsymbol{B}_i[3,0] \leftarrow mr$, $\boldsymbol{B}_i[3,2] \leftarrow r$
  for $i \in \{1,2,3\} \setminus I$ do
      for $box \in \boldsymbol{B}_i$ do
          if $box[0] < m\ell$ then
              $box[0,2] \leftarrow \ell, m\ell$
          else if $box[0] < m$ then
              $box[0,2] \leftarrow m\ell, m$
          else if $box[0] < mr$ then
              $box[0,2] \leftarrow m, mr$
          else
              $box[0,2] \leftarrow mr, r$
  $\boldsymbol{B}_4[0,0] \leftarrow \min\{\boldsymbol{B}_4[0,0], \ell\}$
  $\boldsymbol{B}_4[0,2] \leftarrow \max\{\boldsymbol{B}_4[0,2], r\}$
Output: lists of adjusted bounding boxes $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_4$
```

3. Results

We evaluate the model using 5-fold cross-validation on the training set. Table 2 summarizes our final score and ranking. Before calculating the SNR metric, the estimated signal is shifted horizontally up to ± 0.5 ms and vertically up to ± 1 mV to better match the ground-truth signal. See [2] for a detailed description of the SNR metric as well as the hidden validation and test set.

We test our method on a synthetic set of noise-free ECG images generated using [7] without adding noise, as well as a real-world set of ECG images generated by [7], printed and scanned with a photoscanner. The results are summarized in Table 3-4.

Task	Score	Rank
Digitization	SNR: 0.893	6/16
Classification	–	–

Table 2. SNR and our ranking on the test set.

Training on noise-free dataset	Validation on held-out dataset	Test
8.736 ± 1.272	7.927	–

Table 3. SNR on the synthetic noise-free training dataset (1400 images) and held-out subset of the training set (100 images).

Training on noise-augmented dataset	Validation on real-world dataset	Test
1.762 ± 0.216	0.704	–

Table 4. SNR on the real-scanned, noise-free training dataset (1400 images), validation on held-out real-scanned set (50 images).

4. Discussion

The gap between SNR scores on the synthetic noise-free dataset and the real-world dataset. As shown in Table 3, our pipeline performs better on noise-free datasets, demonstrating its effectiveness in handling high-quality data. However, a significant drop in performance, as observed in Table 4, reveals that our pipeline struggles to generalize to unseen data and is sensitive to noise. This is a direct consequence of U-Net's instability, which leads to overfitting on noise and training data [16].

Why not use the Mask R-CNN [17] for signal pixel segmentation? Faster R-CNN [8], as a popular instance segmentation model that extends Faster R-CNN [8] with an additional segmentation branch, can predict mask labels at the pixel level. However, after experimenting with various hyperparameter settings, we reach the conclusion that Mask R-CNN [17] does not segment signals well and has a strong bias toward segmenting signals as a simple horizontal line. We think the reason for this is that its ResNet [13] backbone, with its pooling layers, leads to many detailed low-level spatial features being permanently lost. Whereas in U-Net [9], those features are retained through skip connections between the encoder and decoder modules.

References

[1] Goldberger AL, et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation 2000;101(23):e215–e220.

[2] Reyna MA, et al. Digitization and Classification of ECG Images: The George B. Moody PhysioNet Challenge 2024. Computing in Cardiology 2024;51:1–4.

[3] Reyna MA, et al. ECG-Image-Database: A dataset of ECG images with real-world imaging and scanning artifacts; a foundation for computerized ECG image digitization and analysis, 2024.

[4] Hampton JR, Adlam D. The ECG Made Easy. 9th edition. Philadelphia: Elsevier, 2019. Figure 2.22, p. 57.

[5] Wagner P, et al. PTB-XL, a large publicly available electrocardiography dataset. Scientific Data 2020;7:154.

[6] Strodthoff N, et al. PTB-XL+, a comprehensive electrocardiographic feature dataset. Scientific Data 2023;10:279.

[7] Shivashankara KK, et al. ECG-Image-Kit: a synthetic image generation toolbox to facilitate deep learning-based electrocardiogram digitization. Physiological Measurement 2024;45:055019.

[8] Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 2017;39(6):1137–1149.

[9] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention (MICCAI). Springer, 2015; 234–241.

[10] Lin TY, et al. Microsoft COCO: Common objects in context. In European conference on computer vision. Springer, 2014; 740–755.

[11] Zou Z, Chen K, Shi Z, Guo Y, Ye J. Object detection in 20 years: A survey. Proceedings of the IEEE 2023; 111(3):257–276.

[12] Trivedi A, Udagawa T, Merler M, Panda R, El-Kurdi Y, Bhattacharjee B. Neural architecture search for effective teacher-student knowledge transfer in language models, 2023. URL `https://arxiv.org/abs/2303.09639`.

[13] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016; 770–778.

[14] Virtanen P, et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods 2020;17:261–272.

[15] Lee GR, et al. PyWavelets: A python package for wavelet analysis. Journal of Open Source Software 2019; 4(36):1237.

[16] Mosinska A, et al. Beyond the pixel-wise loss for topology-aware delineation. In Proceedings of the IEEE conference on computer vision and pattern recognition. 2018; 3136–3145.

[17] He K, Gkioxari G, Dollár P, Girshick R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision. 2017; 2961–2969.

Address for correspondence:

Haoliang Shang
Rämistrasse 101, 8092 Zürich, Switzerland
hshang@ethz.ch