Crafting Deep Learning Models for Classifying ECG Paper Printouts

Damian Kucharski[1,*], Arkadiusz Czerwinski[1,*], Agata M Wijata[2], Jacek Kawa[2], Yalin Zheng[3,4], Gregory Y H Lip[3,5], Jakub Nalepa[1]

[1]Faculty of Automatic Control, Electronics and Computer Science, Department of Algorithmics and Software, Silesian University of Technology, Gliwice, Poland
[2]Faculty of Biomedical Engineering, Silesian University of Technology, Zabrze, Poland
[3]Liverpool Centre for Cardiovascular Science at University of Liverpool, Liverpool John Moores University and Liverpool Heart & Chest Hospital, Liverpool, United Kingdom
[4]Department of Eye and Vision Science, Institute of Life Course and Medical Sciences, University of Liverpool, Liverpool, United Kingdom
[5]Danish Center for Health Services Research, Department of Clinical Medicine, Aalborg University, Aalborg, Denmark
[*]*Equal contribution*

Abstract

As part of the George B. Moody PhysioNet Challenge 2024, we (the GIRAFFE team) built an approach based on InceptionV3 to classify the electrocardiogram (ECG) images. To deal with the class imbalance, we use the Generalized Extreme Value activation function and loss weighting. For the classification task, our best model received a macro F-measure of 0.652 over the hidden test data. Because we had not submitted any unofficial phase entry, we were not included in the official rankings.

1. Introduction

We took part in the 2024 George B. Moody PhysioNet Challenge, and developed a method for classifying electrocardiograms (ECGs) from images or paper printouts [1, 2]. While printouts are adopted, most algorithms rely on the time series data recorded from ECG devices to detect cardiac abnormalities. Our team, GIRAFFE, addresses this gap by applying loss weighting and the Generalized Extreme Value activation to deal with the class imbalance—the set was generated using ECG Image Kit [3, 4]. The availability of PTB-XL [5, 6] made the challenge possible.

2. Methods

2.1. Generated training data

We used the ECG Image Kit to create three datasets with different levels of augmentation. This tool generates images resembling scanned printouts from digital signal data. We fine-tuned its parameters:

- pi: pad_inches—it controls the white border padding.
- ph: print_header—it controls whether to add text from header file to the generated images.
- rr: random_resolution—it controls whether the resolution of the generated images should vary. If it is set, then the resolution is uniformly picked from $\langle 50, 200 \rangle$.
- rp: random_padding—it controls whether the size of the padding should vary and it should be chosen randomly between no padding and maximum padding of rr inches.
- cp: calibration_pulse—it is the probability of adding a calibration pulse to generated images.
- rgp: random_grid_present—it is the probability of generated images to have an ECG paper grid.
- rbw: random_bw—it controls a probability of a generated images to be black and white.
- rgc: random_grid_color—it controls whether the gridlines should be randomly chosen from five available colors, including brown, pink, blue, green and red (default).
- rot: rot—the max. angle for images.
- fr: fully_random—if it is set, each image has a 50% probability for each of the transformations (handwritten distortions, wrinkles and creases, augmentation, noise).
- se: se—it is a seed controlling all random parameters.

Other parameters were unchanged and set to their default values. We generated three sets (Table 1).

2.2. Dataset split

The dataset consisted of 21837 digitized signals from the PTB-XL dataset [5]—these digitized signals were in-

 ISSN: 2325-887X DOI: 10.22489/CinC.2024.210

Dataset ID	pi	ph	rr	rp	cp	rgp	rbw	rgc	rot	fr	se
D1	—	yes	—	—	—	—	—	—	—	—	42
D2	1	yes	yes	yes	0.8	0.95	0.4	yes	25	yes	42
D3	0	yes	yes	no	0.8	0.95	0.1	no	7	yes	42

Table 1. The datasets used in this study and generated using different hyperparameter values.

putted to the image generation procedures discussed in the previous section. Due to the nature of the challenge, the loaded samples were randomly split into the training and validation datasets, containing randomly selected 80% and 20% samples, respectively. We did not create a separate test set and instead relied on the hidden official phase validation set results reported after processing each submission. In our approach, we use the image resolution of 512×1024, therefore an additional resize operation was introduced before the training process is launched.

The resulting dataset, in a form identical to one described in [7], is multi-label, with most labels assigned infrequently, leading to significant class imbalance. To ensure similar distributions between the training and validation sets, we created a stratification factor—a binary vector indicating label presence. Unique vectors were mapped to categories, ensuring similar distribution of label combinations. The set characteristics are shown in Table 2.

2.3. Models and training process

Due to the relatively small amount of GPU VRAM to evaluate the models in the challenge, we used InceptionV3 [8] which offers flexibility for the input data and fits in memory. The degrees of freedom were:

- The selection of the synthesized dataset used for training (D1–D3, as summarized in Table 1).
- The use of the loss weighting (Section 2.3.1).
- The use the Sigmoid or GEV activation (Section 2.3.2).

The training was consistent for all models—they were trained for up to 30 epochs. The area under the ROC curve calculated for the validation set was used as the early stopping criterion. If the metric did not improve for 10 epochs, the training was stopped. The objective was to reduce the binary cross–entropy loss which was done using AdamW with $3e^{-4}$ learning rate and weight decay of $1e^{-4}$.

The deep learning networks were trained using NVidia A100 GPU, provided by PLGrid for academic purposes, using a batch size of 48 and the data was loaded using 8 worker processes to speed up the training procedure. For the inference (and for the training process as well) on the machines used to score the challenge submissions, the batch size was reduced to 8, and only one worker process was used to load the data. Finally, we also tried to replicate the "vanilla deep learning example" leaderboard results using the official code and training on the $D1$ and $D3$ datasets. No other modifications were made.

2.3.1. Loss weighting

We use the loss weighting to deal with class imbalance. Since each example can belong to multiple classes, let:

- $y = [y_1, y_2, \ldots, y_C] \in \{0, 1\}^C$ be the ground truth labels for a single example, where $y_i = 1$ if the example belongs to class i, and $y_i = 0$ otherwise.
- $\hat{y} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_C] \in [0, 1]^C$ be the predicted probabilities for each class, where $\hat{y}_i$ is the probability that the example belongs to class i.
- The vector of proportion weights, $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_C] \in \mathbb{R}^{\mathbb{C}}$, where λ_i is the proportion of class i.

The binary cross-entropy (BCE) loss for each class i is:

$$\text{BCE}_i(y_i, \hat{y}_i) = -\left(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)\right).$$

To incorporate the class proportion weights, the per-class loss vector for a single example is given by:

$$\mathbf{L} = [\mathbf{L_1}, \mathbf{L_2}, \ldots, \mathbf{L_C}] \quad \text{where} \quad \mathbf{L_i} = \lambda_\mathbf{i} \cdot \text{BCE}_\mathbf{i}(\mathbf{y_i}, \hat{\mathbf{y}}_\mathbf{i}).$$

The vector of class proportion weights λ becomes:

$$\lambda_i = \frac{\text{total\_samples}}{\text{num\_classes} \times \text{class\_counts}_i},$$

where class_counts_i is the sum of the binary labels for class i across all examples in the set, total_samples is the number of examples, and num_classes is the number of classes.

Given a batch of N examples, the loss matrix $\mathbf{M}$ is:

$$\mathbf{M} = \begin{bmatrix} L_{1,1} & L_{1,2} & \ldots & L_{1,C} \\ L_{2,1} & L_{2,2} & \ldots & L_{2,C} \\ \vdots & \vdots & \ddots & \vdots \\ L_{N,1} & L_{N,2} & \ldots & L_{N,C} \end{bmatrix},$$

where $L_{n,i} = \lambda_i \cdot \text{BCE}_i(y_{n,i}, \hat{y}_{n,i})$ represents the loss for class i for the n-th example in the batch. The loss across the batch is the average of all entries in the matrix $\mathbf{M}$.

2.3.2. Activation function

We assessed the impact of two activation functions on our models' performance: the Sigmoid function and the Generalized Extreme Value (GEV) function [9]. Sigmoid was used as it is a standard choice in various neural nets. Recognizing the challenges posed by highly imbalanced datasets, we explored the GEV activation. This function

Dataset↓	AFIB/AFL	Acute MI	BRADY	CD	HYP	NORM	Old MI	PAC	PVC	STTC	TACHY	Total
Training	1236	168	495	3922	2113	7621	4242	318	927	4154	687	17212
Validation	334	41	142	976	536	1893	1068	80	216	1081	180	4303

Table 2. The class distributions. AFIB/AFL—Atrial Fibrilation, MI— Myocardial Infarction, BRADY— Bradycardia, CD – Conduction Disturbance, HYP — Hypertrophy, NORM — Normal ECG, Old MI — Old Myocardian Infarcion, PAC — Atrial Premature Complex, PVC — Ventricular Premature Complex, STTC — ST/T change, TACHY — Tachycardia

is based on the Generalized Extreme Value distribution, which is used to model the distribution of the maximum (or minimum) of a large number of independent, identically distributed random variables. This distribution provides a flexible framework for modeling the tail data behavior, which is beneficial in handling skewed distributions.

Mathematically, the GEV function is defined as:

$$\mathrm{GEV}(x) = \begin{cases} \exp\left(-\left(1+\xi\cdot\frac{x-\mu}{\sigma}\right)^{-\frac{1}{\xi}}\right) & \text{if } \xi \neq 0, \\ \exp\left(-\exp\left(\frac{x-\mu}{\sigma}\right)\right) & \text{if } \xi = 0, \end{cases}$$

where x is the input to the activation function, and μ, σ, and ξ are the parameters controlling the location, scale, and shape of the distribution, respectively. Unlike typical hyperparameters, these parameters are learned during the training process through gradient descent, allowing the model to adapt the activation function to better fit the underlying data distribution. By comparing the results obtained with the Sigmoid and GEV activation functions, we aimed to determine whether the GEV function could improve our model's ability to generalize across all classes, particularly those that are less frequent in the dataset.

Task	Score	Rank
Digitization	—	—
Classification	F-measure: 0.652	Not ranked

Table 3. Main scores table. The table presents the best score obtained on hidden test data.

3. Results

We submitted 6 official phase entries for classification task, achieving 0.652 macro F-measure on the hidden test data for the best of them (Table 3). All configurations, along with the submission id and the results are summarized in Table 4. The second lowest hidden test set score was obtained for the first experiment (submission ID: 2163). We concluded that it was likely due to the very aggressive augmentation parameters for the $D2$. Thus, $D3$ was generated with the parameters that were less likely to generate such issues (and unrealistic ECG printouts, Figure 1) that we observed for $D2$. Using this set, along with introducing the loss weighting significantly improves the score (submission id: 2234). Changing the activation to GEV increases the performance by another 7.5% (id: 2322). This suggests that addressing the class imbalance problem is crucial to train well-performing models.

We submitted 3 additional entries to verify the effect of single modifications in the experiment settings. D1 is generated with no parameters that effect visual quality of the the images. The 17.5% improvement in the macro F-measure suggests that inclusion of some of these image operations helps model to generalize to real world data as the generated images are more similar to the real ECG printouts. The parameters should be chosen carefully as too "extreme" values make the classifier "useless" (id: 2163).

We tried to reproduce the "Vanilla convolutional neural network (CNN)" submission. The submission 2324 uses the unchanged code from the example along with the dataset generated as described by the organizers. We can observe that we obtained approx. 30% worse score than the official example. Also, the example was run on $D2$, and obtained the worst results. It shows that larger models may be less prone to deteriorate the performance when trained on the dataset with more complex augmentation. Therefore, combination of relatively large models and carefully selected, but allowing for complex data generation parameters, may allow for best generalization to the real world data. The challenge computational resources heavily limited the possible model size.

4. Discussion and Conclusions

4.1. Dataset

One of the characteristics of the challenge was that the training set was not the same for all teams. Thus, possible addition of real ECG printouts has a potential to increase the scores regardless of the method. It may grant advantage to the teams having access to the closed datasets and also make it challenging to compare the results obtained by different teams, as the difference may not come from the method, but also from the data. The lack of real examples in the set shared by the organizers also made it more challenging to find appropriate hyperparameters for generator. We identified issues with the sequence of generation operations, making them unrealistic.Some operations should be performed together, e.g., padding after rotation, as a real ECG scan would unlikely have obscured corners.

Submission ID	Architecture	Weighting	Activation Function	Dataset	Train. F-Measure	Hidden test set F-measure
2322	Inceptionv3	Yes	GEV	D3	0.725	0.652
2234	Inceptionv3	Yes	Sigmoid	D3	0.700	0.607
2321	Inceptionv3	Yes	GEV	D1	0.802	0.555
2324	Vanilla	No	Sigmoid	D1	0.471	0.227
2163	Inceptionv3	No	Sigmoid	D2	0.102	0.087
2323	Vanilla	Yes	Sigmoid	D3	0.056	0.056

Table 4. The results returned by the server (hidden test set. F-measure) for all submissions.

Figure 1. Unrealistically generated image from $D2$.

4.2. Technical and Resource Limitations

We faced technical challenges that prevented us from implementing some ideas. Limited GPU RAM made it impossible to submit entries with larger models. Our original idea was to exploit an approach for building ensemble models. However, due to the constrains on how the inference should be run, effectively any ensemble approach was difficult to utilize within the time limits posed by the challenge. This is because, in contrary, to the training code, the inference function expects a single example at a time. It means that for each single example, each base model would need to be loaded again to make the prediction unless they would fit in the GPU memory simultaneously. In this scenario, each base model would need to be loaded as many times as many there are examples to score. We were only able to focus on single-model approaches.

Acknowledgments

This work was supported by SUT through the grant for maintaining research potential and Excellence Initiative (02/080/SDU/10-21-01). We acknowledge Polish high-performance computing infrastructure PLGrid (ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2024/017393.

References

[1] Goldberger AL, Amaral LA, Glass L, Hausdorff JM, Ivanov PC, Mark RG, 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, Deepanshi, Weigle J, Koscova Z, Elola A, Seyedi S, et al. Digitization and Classification of ECG Images: The George B. Moody PhysioNet Challenge 2024. Computing in Cardiology 2024;51:1–4.

[3] Shivashankara KK, Deepanshi, Shervedani AM, Reyna MA, Clifford GD, Sameni R. ECG-Image-Kit: a synthetic image generation toolbox to facilitate deep learning-based electrocardiogram digitization. Physiological Measurement 2024; 45:055019.

[4] Deepanshi, Shivashankara KK, Clifford GD, Reyna MA, Sameni R. ECG-Image-Kit: A Toolkit for Synthesis, Analysis, and Digitization of Electrocardiogram Images, January 2024. Online at: `https://github.com/alphanumericslab/ecg-image-kit`.

[5] Wagner P, Strodthoff N, Bousseljot RD, Kreiseler D, Lunze FI, Samek W, et al. PTB-XL, a large publicly available electrocardiography dataset. Scientific Data 2020;7:154.

[6] Strodthoff N, Mehari T, Nagel C, Aston PJ, Sundar A, Graff C, et al. PTB-XL+, a comprehensive electrocardiographic feature dataset. Scientific Data 2023;10:279.

[7] Reyna MA, Deepanshi, Weigle J, Koscova Z, Campbell K, Shivashankara KK, 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. URL `https://arxiv.org/abs/2409.16612`.

[8] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. CoRR 2015;abs/1512.00567. URL `http://arxiv.org/abs/1512.00567`.

[9] Bridge J, Meng Y, Zhao Y, Du Y, Zhao M, Sun R, et al. Introducing the gev activation function for highly unbalanced data to develop covid-19 diagnostic models. IEEE Journal of Biomedical and Health Informatics 2020;24(10):2776–2786.

Address for correspondence:

Damian Kucharski
Akademicka 16, 44-100 Gliwice, Poland
Damian.Kucharski@polsl.pl