Impact of Optimizer: Comparative Analysis of Loss Functions in Different Models

Abdulvahit Karail[1][0009000387188906] and Yasin Ortakcı[2][0000-0002-0683-2049]

[1] Karabuk University, Vocational School of Information Technologies, Karabük, Türkiye
[2] Karabuk University, Dept. of Computer Engineering, Karabük, Türkiye
abdulvahitkarail@karabuk.edu.tr

Abstract. This study compares the performance of different artificial neural network models for data classification problems. The models used include Multilayer Perceptron (MLP), Kolmogorov–Arnold Network (KAN), and Liquid Time-Constant Networks (LTC). Model training was performed on the MITBIH ECG dataset, and modern optimization strategies with different loss functions were systematically compared. Evaluation metrics such as accuracy, macro-F1, precision, and recall were assessed. The findings show that KAN models provide high accuracy, while LTC models operate more slowly due to their time-step based calculations.

Keywords: Optimizers, loss functions, Kolmogorov–Arnold Network, Liquid Time-Constant Networks, Multilayer Perceptron.

1 Introduction

A significant gap in the current literature is the absence of an evidence-based implementation guide that evaluates the interactions between loss functions and optimizer for artificial neural network (ANN), particularly when processing unstable health data such as ECGs. This study was conducted because determining the compatibility of a particular loss function and optimizer with the applied model is a challenging task.

Determining the optimal fit between a given loss function, an optimizer, and the implemented model remains a challenging task. To overcome this challenge, we conducted a systematic and reproducible experimental study on the MIT-BIH Arrhythmia dataset. This study comparatively examines modern artificial neural network models for solving the data classification problem. Three different model families were selected for the experimental analysis. These models are Multilayer Perceptron (MLP), Kolmogorov Arnold Network (KAN), and Liquid Time Constant Networks (LTC). A comprehensive experimental design was implemented using various loss functions such as Cross Entropy (CE), Focal Loss (FL) and Label Smoothing Cross Entropy (LSCE), optimizer such as Adam, AdamW, Stochastic

Gradient Descent (SGD), Adagrad and RMSProp. Each model was tested with 3 different loss functions and 5 different optimizers. Thus, the study revealed which architecture was more suitable in terms of both performance and computational cost. The results we have obtained are practice-oriented guidance.

In comparative experiments, CE/LSCE consistently provided high accuracy in imbalanced ECG data when used in conjunction with Adam/AdamW. KAN showed competitive performance with MLP and under the evaluated representation and time step settings, LTC was observed to train more slowly. These observations shed light on the suitability of the model training strategy under data unstable and computational constraints.

The geometry of the loss surface, local minima, saddles, straight directions, multiple fractal structure, and large-scale connectivity, fundamentally shapes the optimization trajectories, convergence speed, and generalization performance across model families. Therefore, a comparative study aiming to reveal how different loss functions interact with the optimization landscapes of KAN, MLP, and LTC should utilize diagnostics that capture the curvature, modality, and dynamic signatures of the optimization, and select loss targets and optimization algorithms whose interactions with the landscape structure are well-defined in literature [1-4].

Understanding the interaction between the choice of loss function and the resulting optimizers is crucial for predicting training dynamics and the ability to generalize across model families [3], [4], [5].

This study fills a gap in the literature by systematically comparing KAN, MLP, and LTC architectures on the MIT-BIH Arrhythmia dataset under a single documented and reproducible protocol using three loss-based and five optimizers. The evaluation focuses not only on accuracy but also on practical metrics such as macro-F1.

The rest of the article is structured as follows. Related work in section 2 summarizes previous work and identifies research gaps. Section 3 examines preprocessing and details the architecture of ANN, loss functions and optimizers. Section 4 examines the methodology. Section 5 describes experimental studies and analyzes the results. Section 6 summarizes the conclusions.

2 Related Work

While prior work covers architecture, loss functions, and optimizers, joint comparisons of MLP/KAN/LTC, CE/Focal Loss/LSCE, SGD/Adam/AdamW/RMSProp/Adagrad under a single, documented protocol on MIT-BIH, with accuracy, macro-F1 reported together, remain limited. Our study addresses this gap and offers practice oriented guidance for ECG classification under class imbalance and compute constraints.

Prior work provides comprehensive surveys of MLP architecture, training, regularization/generalization, design choices, variants and hybrids, and application domains. The synthesis integrates empirical findings and methodological recommendations reported in applied and methodological studies, creating a

consistent guide for researchers and practitioners who design, train, evaluate or compare MLPs in applied settings [6-13].

KAN variants have been used for rolling bearing fault diagnosis, EEG seizure detection, and ECG arrhythmia classification, reporting improved robustness and interpretability in those settings [14], [15].

Training LTCs benefits from principled loss selection and from using adaptive gradient optimizers together with solver–optimizer co design to stabilize gradients and improve convergence [16-22].

Cross entropy is a canonical supervised classification loss that compares a predicted probability distribution to a target distribution and yields gradients suitable for gradient based training. Once a model produces a differentiable output, a differentiable loss such as cross entropy may be used to train the network by gradient based procedures. Wunderlich & Pehle show that exact gradients can be computed for a general loss function, which includes cross entropy as a standard differentiable classification loss. Wunderlich and Pehle show that exact gradients can be obtained for a general differentiable loss and because Yin et al. demonstrate training schemes for liquid neurons compatible with differentiable losses, any differentiable variant of cross entropy is theoretically trainable in LTC/spiking frameworks using the methods described in [23], [24].

Multiple empirical studies document that adaptive optimizers such as Adam, RMSProp, Adagrad typically enable faster decrease of training loss and faster wall clock progression during early epochs, whereas SGD with momentum tends to produce solutions with better generalization when carefully tuned, this tradeoff is widely reported in both survey and empirical benchmark studies [25], [26], [28], [29], [30].

Modern optimization algorithm selection requires striking a pragmatic balance between speed, stability, and generalization. Adaptive optimizer such as Adam and RMSProp should be used when rapid development and ease of adjustment are important, AdamW should be used when weight reduction adjustment is required, Adagrad should be preferred for sparse problems, and SGD or a hybrid adaptive SGD strategy should be used when ultimate generalization performance is prioritized. The aforementioned survey, robustness, and comparative studies provide both the theoretical framework and empirical evidence that motivate these recommendations [25], [26], [27], [28], [30].

Precision penalizes false positives while Recall penalizes false negatives. The operating decision threshold on continuous prediction scores controls this trade-off, and practitioners inspect Precision–Recall pairs and the confusion matrix to select thresholds appropriate to their risk profile, many medical and detection studies report sensitivity and specificity alongside precision and accuracy to characterize clinically relevant trade-offs [32], [33], [34].

F1-Score balances precision and recall and is particularly useful when a single composite metric is required and the costs of false positives and false negatives are comparable. Applied classification studies often report both per-class F1 and per-class precision/recall so that readers can examine asymmetric error patterns [31], [32].

3 Background

Loss functions form the training dynamics of neural networks. A loss function maps the predicted outputs and the actual value to a scalar that quantifies the model error and minimizing the expected loss on the data is the fundamental optimization goal of learning.
Optimizer uses the gradients of this objective to update parameters to reduce loss on the training data. Optimizer are methods for finding the minimum or maximum values of objective functions. The selection and configuration of an optimization algorithm greatly influence the quality of the solution, the speed of convergence, the ability to generalize, and the computational cost.

3.1 Loss Functions

To analyze the effects of the KAN, LTC and MLP models on optimization, three different loss functions were compared. Cross-Entropy Loss, Weighted Focal Loss and Label Smoothing Cross-Entropy were used as loss functions.
In classification problems, the CE standard approach minimizes the difference between the predicted probability distribution and the actual labels. The multi-class CE loss used in this study is defined in Eq. (1):

$$L_{CE} = -\sum_{c=1}^{C} y_c \log(p_c) \tag{1}$$

To reduce class imbalance and the dominance of easy examples, Focal loss is designed to down weight examples and focus learning on hard examples. It is formulated by adding a modulation factor. The formulation of Focal Loss is given in Eq. (2):

$$L_{FL} = -(1 - p_t)^{\gamma} log(p_t) \tag{2}$$

Here, p_t is the probability that the model predicts for the correct class. It is commonly expressed as a modulated version of cross entropy with a tunable focusing parameter γ and an optional class weighting factor α to address class imbalance.

Label Smoothing is a function used in ANN models to prevent overconfidence and improve the model's generalization ability. To break the model's overconfidence and increase its generalization ability, softened labels are used instead of hard labels. The smoothed target obtained by label smoothing is defined in Eq. (3):

$$y_c^{LS} = (1 - \epsilon) y_c + \frac{\epsilon}{C} \tag{3}$$

Here, ϵ is smoothing factor.

3.2 Optimizer

The general structure of an optimizer consists of the basic steps of gradient

calculation, internal state update, direction and magnitude determination, and parameter update. It is expressed as a mathematical formula as follow Eq. (4):

$$w_{t+1} = w_t - \eta \cdot \phi(g_t, M_t) \tag{4}$$

Here, w is model parameters, weights, η is learning rate, g_t is current gradient, M_t is optimizer's memory and ϕ is optimizer's decision function.

Adam is a popular optimizer that combines both momentum and adaptive learning rate methods. For each parameter, the average and the average of the squares of past gradients are maintained. Parameter update for Adam is given as Eq. (5):

$$\theta_{t+1} = \theta_t - \alpha \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}}+\varepsilon} \tag{5}$$

Here, α is learning rate, $\widehat{m_t}$ is first moment of gradients, v_t is second moment of gradient squares, ε is small constant.

AdamW is a variant of the Adam algorithm that implements weight decay more accurately. While weight decay in classical Adam is implemented by adding it to gradients, AdamW performs it separately from parameter updates. Parameter update for AdamW is given as Eq. (6):

$$\theta_{t+1} = \theta_t - \eta \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}}+\varepsilon} - \eta \lambda \theta_t \tag{6}$$

Here, η is learning rate, λ is weight decay constant.

SGD is one of the most basic optimizer. It calculates the gradient and updates parameters using a small subset of data in each iteration. Parameter update for SGD is given as Eq. (7):

$$\theta_{t+1,i} = \theta_{t,i} - \eta\, g_{t,i} \tag{7}$$

Here, θ_t is the vector of all parameters in t-th iteration, $\theta_{t,i}$ is the scalar value of the i-th parameter at the t-th iteration, $g_{t,i}$ is the i-th component of the gradient.

RMSProp is an algorithm that provides an adaptive learning rate. By keeping the exponential mean of the squares of the gradients, it determines a different learning rate for each parameter. Parameter update for RMSProp is given as Eq. (8):

$$\theta_{t+1} = \theta_t - \eta \frac{g_t}{\sqrt{E[g_t^2]}+\varepsilon} \tag{8}$$

Here, ρ is decay rate, θ_t is parameters in step t, $g_t = \nabla_{\theta_t}\mathcal{L}_t$ is the gradient of that step.

Adagrad is an algorithm that adapts the learning rate based on the sum of the squares of the past gradients for each parameter. It reduces the learning rate for frequently updated parameters while keeping the learning rate high for infrequently updated parameters. Parameter update each component for Adagrad is given as Eq.

(9):

$$\theta_{t+1,i} = \theta_{t,i} - \eta \frac{g_{t,i}}{\sqrt{G_{t,i}+\varepsilon}} \quad (9)$$

Here, i is parameter index and $G_{(t,i)}$ is sum of the squared gradients of the parameter.

3.3 Architecture of ANN Models

3.3.1 Multilayer Perceptron

MLP is a classic artificial neural network consisting of fully connected layers. It is one of the most fundamental architectures of artificial neural networks and is composed of fully connected layers. In each layer, the input data undergoes linear transformation, followed by the application of nonlinear activation functions. This structure is suitable for learning relationships between features and is widely used, particularly in data classification. Its simplicity and rapid training make it a preferred choice in many applications. Figure 1 shows the structure of MLP.

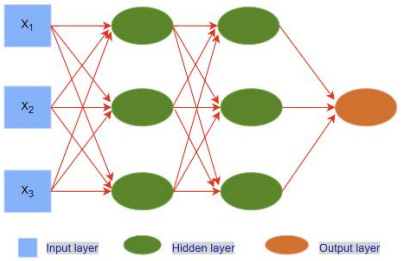

Fig. 1. Structure of MLP

3.3.2 Kolmogorov Arnold Network

The Kolmogorov Arnold Network is a new artificial neural network architecture introduced in 2024 by researchers at MIT, Caltech, and others, offering a radical alternative to MLPs, which have been the standard in deep learning for the past 10-15 years. KAN is an architecture that provides functional flexibility through a kernel-based spline approach. KAN is an architecture developed based on the Kolmogorov–Arnold representation theorem. The overall architecture of the Kolmogorov Arnold Networks is shown in Figure 2.

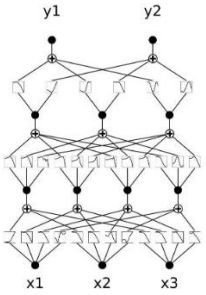

Fig. 2. The overall architecture of the Kolmogorov-Arnold Networks [35]

3.3.3 Liquid Time-Constant Networks

LTC is a continuous-time dynamic Recurrent Neural Network (RNN)-based model designed for time series modeling. Unlike traditional RNNs, LTC cells are defined by differential equations, and the state of each neuron is continuously updated over time. This feature makes LTC a more flexible and biologically inspired model for time series data. Figure 3 shows the general architecture of Liquid Time Constant Networks.

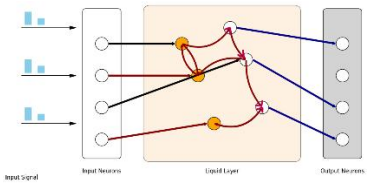

Fig. 3. The overall architecture of the Liquid Time-Constant Networks

4 Methodology

Our study involved testing three different ANN models using three different loss functions and five different optimizers, resulting in a detailed comparison. Figure 4 shows an overview of the methodology.

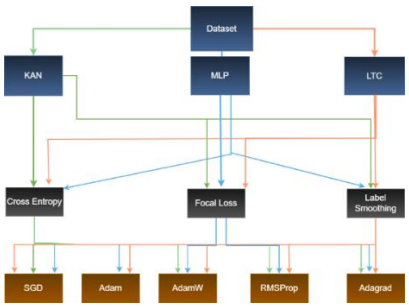

Fig. 4. The overview of the methodology

Figure 4 illustrates the end to end experimental stages of the study. Samples from the MIT-BIH Arrhythmia dataset are fed into three different architectures via separate streams after a common preprocessing step. For each architecture, training was conducted using three different loss functions such as Cross-Entropy, Focal Loss, and Label-Smoothing CE and each configuration was run under the same training protocol with five optimizer. This resulted in 45 different outcomes, all combinations of which were evaluated in a comparable and reproducible manner.

In all experiments, data is processed using the same train/validation/test partitioning and Z-score normalization. Hyperparameters such as batch size, epoch, and initial learning rate are kept constant at the baseline level and gradient clipping is applied using the same rules. Class weights for Weighted CE were derived from the training distribution, and alpha and gamma values were reported for Focal Loss. Each architecture-loss-optimizer combination was run with at least one fixed seed, and the results were reported as averages. Thus, convergence speed, stability, and generalization behavior can be isolated from the effects of optimizer and loss selection.

The output of each experiment, accuracy, macro-F1, and precision/recall was aggregated. Colored arrows visually distinguish which architecture was used for training, along with which loss and optimizer. Finally, tables and training validation curves summarize the performance-cost profile of each combination.

Features were scaled using Z-score normalization. For architectures trained with PyTorch, normalization was applied on a sample by sample basis within the dataset. The aim is to prevent data leakage and ensure scale consistency.

4.1 Structure of Models

The study compares three main architectures. These are KAN, MLP, and LTC. The MLP architecture used in this study consists of 3 hidden layers by default. Each hidden layer comprises Linear, BatchNorm1d, ReLU, and Dropout blocks, respectively. The network terminates with a linear output layer for classification. The default hidden dimension is 128, and the depth is set to 3, these hyperparameters can be adjusted according to the experimental scenario. The KAN architecture features two KAN layers between the input and output. Each KAN layer applies learnable piecewise linear transformations on a fixed grid for each input feature, and the resulting transformations are summed up along with the bias to produce the output. GELU activation and dropout are used in the intermediate layer. The default settings are grid size 16 and hidden size 64, which can be changed according to requirements. The LTC architecture is built on an LTC cell created using AutoNCP wiring from the NCPs library. The default number of NCP neurons is 12, and hyperparameters such as time step, integrator, and number of neurons can be reconfigured for different setups.

5 Experimental studies

In this study, hyperparameters were kept constant to ensure fair and reproducible comparisons. The batch size was set at 128, the initial learning rate at 1×10^{-3}, and the training duration at 15 epochs. Additionally, the number of NCP neurons for the LTC architecture was 12. To prevent overlearning during the training process, gradient clipping was applied, and the learning rate scheduler was optionally tested. All experiments were performed in Visual Studio Code using Python.

5.1 Data Set

This study utilized the MITBIH Arrhythmia ECG dataset. The dataset consists of time-series data representing different arrhythmia types, commonly used for heart rhythm classification. Each sample represents a heartbeat signal of a specific length and is presented with class labels. With Z-score normalization, the mean and standard deviation were calculated for each feature, thus normalizing the data.

In this study, MIT-BIH Arrhythmia data was loaded from two separate files. These are mitbih train file and mitbih test file. With data splitting, the dataset was divided into 68% training, 12% validation and 20% testing ratios. All samples in the test file were used as the test set. Thus, the data flow was standardized in the form of training, then validation, and finally testing.

5.2 Evaluation metrics

In addition to accuracy, macro-F1 was reported as the primary metric to accurately reflect the effect of class imbalance, precision and recall were also measured. Macro F1 was used because it calculates the F1 Score separately for each class in the dataset and then takes the arithmetic mean of these scores, and because the dataset is unbalanced. The primary reason for using these metrics is to evaluate model performance from different perspectives and to avoid the misleading nature of relying on a single measure, such as accuracy, especially in datasets with unbalanced classes.

Accuracy is the ratio of correctly predicted samples to the total sample size. Eq. (10) defines accuracy.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total number of samples}} \tag{10}$$

Precision focuses on reducing false positives. It is critical for lowering the number of false alarms in healthcare applications. Eq. (11) defines precision.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive + False Positive}} \tag{11}$$

Recall indicates the rate at which true positives are detected. Since missed cases in disease detection can have serious consequences, recall must be high. It shows how

many true positive samples are correctly predicted. Eq. (12) defines Recall.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{12}$$

In unbalanced datasets, F1 is more meaningful than accuracy. It is the harmonic mean of precision and recall. Eq. (13) defines F1-Score.

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{13}$$

5.3 Results

Comparisons of training duration and convergence curves with uniform axes/styles Figures 5. Figures 5 show the validation accuracy graphs for the KAN, LTC, MLP.

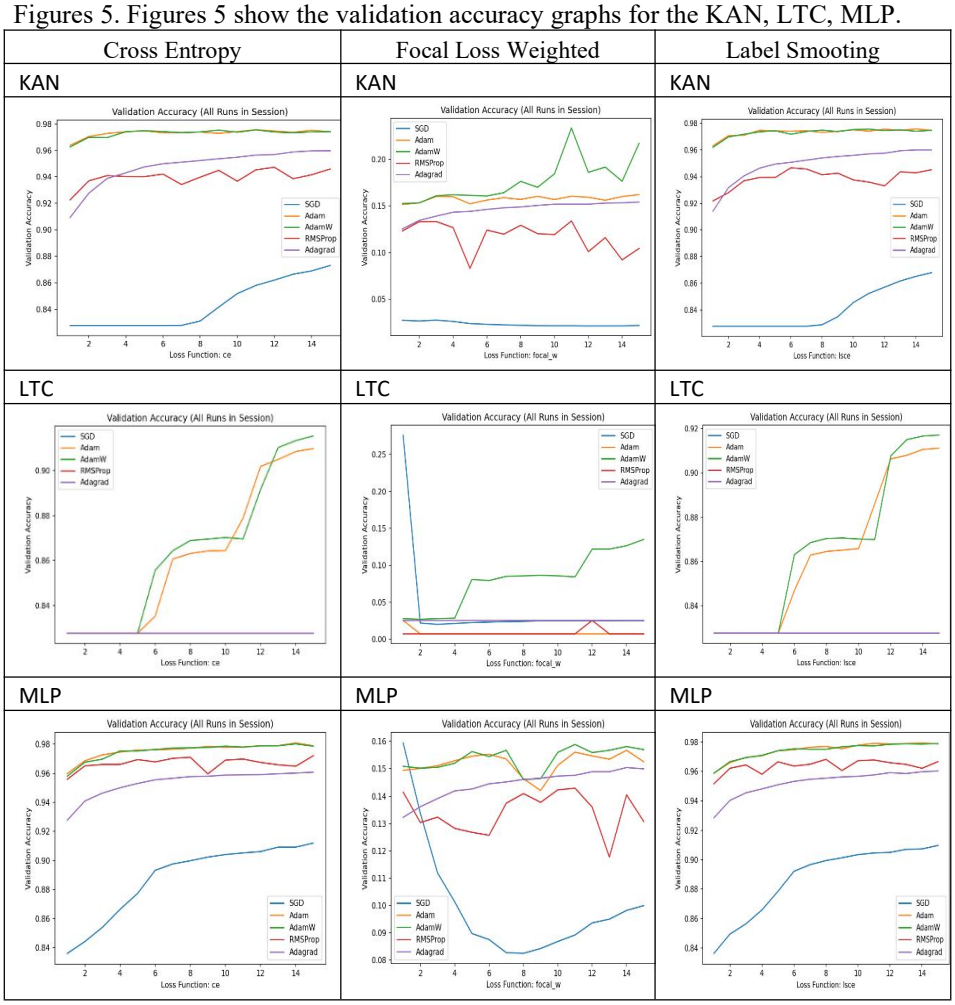

Fig. 5. Validation Accuracy Results

When using the KAN model with CE or LSCE and the Adam/AdamW optimizer, fast convergence, high test accuracy, and high macro F1 were achieved. Adagrad, used with the KAN model, was stable but had low minority class performance. RMSProp was choppy and generally lower. SGD was slow and low.

Experiments conducted for MLP show that the choice of optimization is crucial. With Adam/AdamW, both Cross Entropy and LSCE losses converged rapidly within the first few epochs, and validation accuracy reached the 97.5–98% range, this was confirmed in testing with 97.7–97.8% accuracy and approximately 0.87–0.88 macro-F1. RMSProp/Adagrad reached lower plateau values, while SGD showed slow and low convergence without a scheduler/momentum. In Focal Loss configurations, learning did not practically begin due to insufficient α–γ settings, and validation accuracy remained in the 10–16% range. Therefore, the most reliable choice in this dataset is the combination of CE or LSCE and Adam/AdamW.

LSCE and Adam/AdamW yielded stable and high validation accuracy for LTC, label smoothing reduced overconfidence and improved calibration. Learning did not begin with SGD/RMSProp/Adagrad. Results with CE and Adam/AdamW were very close to LSCE, the optimizer was the primary determinant in LTC.

Adam and AdamW have demonstrated the highest accuracy, fastest convergence, and highest stability in all successful scenarios. They are the only working solutions, especially in architectures with complex dynamics like LTC. Adagrad has been a safe but slow alternative to MLP and KAN, but has failed in LTC. RMSProp, on the other hand, has exhibited high volatility and has generally lagged behind the Adam family. SGD proved insufficient for this dataset and architectures within the examined hyperparameter space. The convergence speed is very low, and it failed to overcome local minima.

The results show that F1 values are critical, especially due to class imbalance. Table 1 shows the best configuration such as accuracy, macro-F1 for each model. The best combination results of the models are shown in the appendix Table 1.

Table 1. Summary table for the best combinations of models.

Model	Optimizer	Loss	Test Accuracy	Macro F1
MLP	Adam	CE	0.978	0.879
KAN	Adam	CE	0.974	0.874
LTC	AdamW	LSCE	0.918	0.493

When dealing with unbalanced data, accuracy alone is not enough, Macro-F1 shows how well you capture minority groups. The wide difference between Accuracy and Macro-F1 indicates that the model is missing minority groups, therefore Macro-F1 has been reported as the primary benchmark. Under the same conditions, Adam and CE, along with MLP and KAN, achieve high accuracy and macro-F1, while LTC, AdamW, and LSCE exhibit a large accuracy-macro-F1 gap, indicating poor minority-class performance. MLP and KAN demonstrate balanced performance with high accuracy and high Macro-F1.

For the MLP model, Adam/AdamW and CE yield the highest performance,

weighted Focal Loss configurations show significantly lower performance. For the KAN model, Adam/AdamW and CE/LSCE with KAN provide performance very close to MLP, SGD and weighted Focal Loss cause a significant decrease. The best performance for LTC was obtained using the AdamW optimizer and the Cross Entropy Loss function. With SGD and RMSProp, the performance of LTC was found to be approximately 82%. MLP and KAN achieved the highest success with Adam/AdamW optimization. Accuracy is greater than 97%, while Macro F1 is approximately 0.87–0.88. Focal Loss, while theoretically suitable for unbalanced classes, produced very low accuracy and F1 scores in practice. SGD proved insufficient in complex models, Adam and AdamW yielded better results.

Provided in the appendix, Table 2 shows the accuracy and Macro F1 results for MLP, KAN, and LTC.

6 Conclusion

This study compares different deep learning architectures, MLP, KAN, LTC, and various loss functions and optimization techniques on the MITBIH ECG dataset. Experimental results show that model performance is directly related to optimizer selection and loss function. MLP and KAN architectures provided the highest accuracy and balanced class performance when using Adam/AdamW optimization along with CE and LSCE losses. In contrast, Focal Loss configurations were insufficient for this dataset, and learning performance decreased significantly due to class imbalance. Although the LTC model offers theoretical advantages for time series modeling thanks to its continuous-time structure, it did not perform as well as MLP and KAN in practice.

These findings reveal that simple and powerful optimization strategies, such as Adam/AdamW and classical loss functions CE and LSCE, remain the most reliable choice for unbalanced health data. Future research suggests investigating methods such as class-weighted losses, sampling techniques, and hyperparameter optimization for Focal Loss to more effectively manage class imbalance. Furthermore, more advanced regularization and optimization strategies can be evaluated to improve the performance of continuous-time architectures like LTC.

Experimental analyses revealed that for the problem set under consideration, configuring the MLP or KAN architectures with the AdamW optimization algorithm and Label Smoothing Cross Entropy and CE loss functions yielded the most optimal result, approximately 0.98. The LTC architecture, however, requires a more selective hyperparameter optimization.

Table 2. Accuracy and Macro-F1 results for MLP, KAN, and LTC.

Model	Loss	Optimizer	Test Accuracy	Macro F1
MLP	CE	Adagrad	0.958	0.723
MLP	CE	Adam	0.978	0.879
MLP	CE	Adamw	0.978	0.878

MLP	CE	RMSProp	0.969	0.811
MLP	CE	SGD	0.912	0.484
MLP	Focal_w	Adagrad	0.148	0.285
MLP	Focal_w	Adam	0.155	0.288
MLP	Focal_w	Adamw	0.156	0.286
MLP	Focal_w	RMSProp	0.141	0.304
MLP	Focal_w	SGD	0.164	0.166
MLP	LSCE	Adagrad	0.957	0.730
MLP	LSCE	Adam	0.978	0.876
MLP	LSCE	AdamW	0.978	0.873
MLP	LSCE	RMSProp	0.967	0.825
MLP	LSCE	SGD	0.909	0.475
KAN	CE	Adagrad	0.956	0.684
KAN	CE	Adam	0.974	0.874
KAN	CE	Adamw	0.973	0.874
KAN	CE	RMSProp	0.944	0.696
KAN	CE	SGD	0.872	0.350
KAN	Focal_w	Adagrad	0.151	0.284
KAN	Focal_w	Adam	0.161	0.270
KAN	Focal_w	Adamw	0.230	0.296
KAN	Focal_w	RMSProp	0.130	0.210
KAN	Focal_w	SGD	0.027	0.019
KAN	LSCE	Adagrad	0.958	0.697
KAN	LSCE	Adam	0.973	0.865
KAN	LSCE	Adamw	0.974	0.867
KAN	LSCE	RMSProp	0.943	0.660
KAN	LSCE	SGD	0.866	0.331
LTC	CE	Adagrad	0.828	0.181
LTC	CE	Adam	0.908	0.466
LTC	CE	Adamw	0.914	0.484
LTC	CE	RMSProp	0.828	0.181
LTC	CE	SGD	0.828	0.181
LTC	Focal_w	Adagrad	0.025	0.009
LTC	Focal_w	Adam	0.025	0.009
LTC	Focal_w	Adamw	0.135	0.221
LTC	Focal_w	RMSProp	0.025	0.009
LTC	Focal_w	SGD	0.278	0.102
LTC	LSCE	Adagrad	0.828	0.181
LTC	LSCE	Adam	0.911	0.473
LTC	LSCE	Adamw	0.918	0.493
LTC	LSCE	RMSProp	0.828	0.181
LTC	LSCE	SGD	0.828	0.181

References

1. Verpoort, P., Lee, A., & Wales, D. (2020). Archetypal landscapes for deep neural networks. Proceedings of the National Academy of Sciences, 117(36), 21857-21864. https://doi.org/10.1073/pnas.1919995117
2. Geiger, M., Spigler, S., d'Ascoli, S., Sagun, L., Baity-Jesi, M., Biroli, G. & Wyart, M. (2019). Jamming transition as a paradigm to understand the loss landscape of deep neural networks. Physical Review E, 100(1). https://doi.org/10.1103/physreve.100.012115
3. Baity-Jesi, M., Sagun, L., Geiger, M., Spigler, S., Arous, G., Cammarota, C., … & Biroli, G. (2019). Comparing dynamics: deep neural networks versus glassy systems. Journal of Statistical Mechanics Theory and Experiment, 2019(12), 124013. https://doi.org/10.1088/1742-5468/ab3281
4. Mehta, D., Zhao, X., Bernal, E., & Wales, D. (2018). Loss surface of XOR artificial neural networks. Physical Review E, 97(5). https://doi.org/10.1103/physreve.97.052307
5. Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., … & Zecchina, R. (2019). Entropy-SGD: biasing gradient descent into wide valleys*. Journal of Statistical Mechanics Theory and Experiment, 2019(12), 124018. https://doi.org/10.1088/1742-5468/ab39d9
6. Güler, İ. and Übeylı, E. (2005). An expert system for detection of electrocardiographic changes in patients with partial epilepsy using wavelet-based neural networks. Expert Systems,22(2),6271. https://doi.org/10.1111/j.1468-0394.2005.00295.x
7. Özesmi, S., Tan, C., & Özesmi, U. (2006). Methodological issues in building, training, and testing artificial neural networks in ecological applications. Ecological Modelling, 195(1-2), 83-93. https://doi.org/10.1016/j.ecolmodel.2005.11.012
8. Lopes, M., Rocha, B., Vieira, A., Sá, J., Rolim, P., & Silva, A. (2019). Artificial neural networks approaches for predicting the potential for hydropower generation: a case study for Amazon region. Journal of Intelligent & Fuzzy Systems, 36(6), 5757-5772. https://doi.org/10.3233/jifs-181604
9. Ahmad, M., Kumar, N., & Singh, B. (2016). Fast multilayer perceptron neural network-based control algorithm for shunt compensator in distribution systems. Iet Generation Transmission & Distribution, 10(15), 3824-3833. https://doi.org/10.1049/iet-gtd.2016.0328
10. Sharma, A., Jain, A., Gupta, P., & Chowdary, V. (2021). Machine Learning Applications for Precision Agriculture: A Comprehensive Review. Ieee Access, 9, 4843-4873. https://doi.org/10.1109/access.2020.3048415
11. Sladojević, S., Arsenović, M., Anderla, A., Ćulibrk, D., & Stefanović, D. (2016). Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification. Computational Intelligence and Neuroscience, 2016, 1-11. https://doi.org/10.1155/2016/3289801
12. Varjovi, M., Rahmanpour, M., Khosravi, M., Majdi, A., & Le, B. (2024). Performance Evaluation of Artificial Neural Networks and Support Vector Regression in Tunneling-Induced Settlement Prediction Incorporating Umbrella Arch Method Characteristics. International Journal of Engineering, 37(8), 1510-1521. https://doi.org/10.5829/ije.2024.37.08b.05
13. Rabiei, S., Jalilvand, E., & Tajrishy, M. (2021). A Method to Estimate Surface Soil Moisture and Map the Irrigated Cropland Area Using Sentinel-1 and Sentinel-2 Data. Sustainability, 13(20), 11355. https://doi.org/10.3390/su132011355

14. Patuwo, E., Hu, M., & Hung, M. (1993). Two-Group Classification Using Neural Networks*. Decision Sciences, 24(4), 825-845. https://doi.org/10.1111/j.1540-5915.1993.tb00491.x
15. Bousqaoui, H., Slimani, I., & Achchab, S. (2021). Comparative analysis of short-term demand predicting models using ARIMA and deep learning. International Journal of Electrical and Computer Engineering (Ijece), 11(4), 3319. https://doi.org/10.11591/ijece.v11i4.pp3319-3328
16. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444. https://doi.org/10.1038/nature14539
17. Fei, R., Yao, Q., Zhu, Y., Xu, Q., Li, A., Wu, H., … & Hu, B. (2020). Deep Learning Structure for Cross-Domain Sentiment Classification Based on Improved Cross Entropy and Weight. Scientific Programming, 2020, 1-20. https://doi.org/10.1155/2020/3810261
18. Huang, Q., Le, J., Joshi, S., Mendes, J., Adluru, G., & DiBella, E. (2024). Arterial Input Function (AIF) Correction Using AIF Plus Tissue Inputs with a Bi-LSTM Network. Tomography, 10(5), 660-673. https://doi.org/10.3390/tomography10050051
19. Huang, J., Niu, G., Guan, H., & Song, S. (2023). Ultra-Short-Term Wind Power Prediction Based on LSTM with Loss Shrinkage Adam. Energies, 16(9), 3789. https://doi.org/10.3390/en16093789
20. Li, Y., Ren, X., Zhao, F., & Yang, S. (2021). A Zeroth-Order Adaptive Learning Rate Method to Reduce Cost of Hyperparameter Tuning for Deep Learning. Applied Sciences, 11(21), 10184. https://doi.org/10.3390/app112110184
21. Jia, P., Liu, H., Wang, S., & Wang, P. (2020). Research on a Mine Gas Concentration Forecasting Model Based on a GRU Network. Ieee Access, 8, 38023-38031. https://doi.org/10.1109/access.2020.2975257
22. Yu, X. (2023). The influence of regional tourism economy development on carbon neutrality for environmental protection using improved recurrent neural network. Frontiers in Ecology and Evolution, 11. https://doi.org/10.3389/fevo.2023.1146887
23. Wunderlich, T. and Pehle, C. (2021). Event-based backpropagation can compute exact gradients for spiking neural networks. Scientific Reports, 11(1). https://doi.org/10.1038/s41598-021-91786-z
24. Yin, B., Corradi, F., & Bohté, S. (2023). Accurate online training of dynamical spiking neural networks through Forward Propagation Through Time. Nature Machine Intelligence, 5(5), 518-527. https://doi.org/10.1038/s42256-023-00650-4
25. Liao, X., Sahran, S., Abdullah, A., & Shukor, S. (2022). AdaCB: An Adaptive Gradient Method with Convergence Range Bound of Learning Rate. Applied Sciences, 12(18), 9389. https://doi.org/10.3390/app12189389
26. Tian, Y., Zhang, Y., & Zhang, H. (2023). Recent Advances in Stochastic Gradient Descent in Deep Learning. Mathematics, 11(3), 682. https://doi.org/10.3390/math11030682
27. Wang, Y., Zhou, P., & Zhong, W. (2018). An Optimization Strategy Based on Hybrid Algorithm of Adam and SGD. Matec Web of Conferences, 232, 03007. https://doi.org/10.1051/matecconf/201823203007
28. Chaudhury, S. and Yamasaki, T. (2021). Robustness of Adaptive Neural Network Optimization Under Training Noise. Ieee Access, 9, 37039-37053. https://doi.org/10.1109/access.2021.3062990
29. Albayati, A., Altaie, S., Al-Obaydy, W., & Alkhalid, F. (2024). Performance analysis of optimization algorithms for convolutional neural network-based handwritten digit recognition. Iaes International Journal of Artificial Intelligence (Ij-Ai), 13(1), 563. https://doi.org/10.11591/ijai.v13.i1.pp563-571

30. Liang, D., Ma, F., & Li, W. (2020). New Gradient-Weighted Adaptive Gradient Methods with Dynamic Constraints. Ieee Access, 8, 110929-110942. https://doi.org/10.1109/access.2020.3002590
31. Barkhordari, M. and Massone, L. (2022). Failure Mode Detection of Reinforced Concrete Shear Walls Using Ensemble Deep Neural Networks. International Journal of Concrete Structures and Materials, 16(1). https://doi.org/10.1186/s40069-022-00522-y
32. He, Y., Li, W., Zhang, W., Zhang, S., Pi, X., & Liu, H. (2021). Research on Segmentation and Classification of Heart Sound Signals Based on Deep Learning. Applied Sciences, 11(2), 651. https://doi.org/10.3390/app11020651
33. M.G, S. and Venkatesan, C. (2025). SwinDFU-Net: Deep learning transformer network for infection identification in diabetic foot ulcer. Technology and Health Care, 33(1), 601-618. https://doi.org/10.3233/thc-241444
34. Huang, T., Yang, R., Shen, L., Feng, A., Li, L., He, N., … & Lyu, J. (2022). Deep transfer learning to quantify pleural effusion severity in chest X-rays. BMC Medical Imaging, 22(1). https://doi.org/10.1186/s12880-022-00827-0
35. J. Xu and X. Fang, "ATT-BLKAN: A Hybrid Deep Learning Model Combining Attention is Used to Enhance Business Process Prediction," in IEEE Access, vol. 13, pp. 36175-36189, 2025, https://doi.org/10.1109/ACCESS.2025.3545071.