

Analyzing the Role and Optimization of Loss Functions in Machine Learning and Deep Learning

Trinh Quang Minh^{*ID}, Ngo Thi Lan, Nguyen Minh Hieu, Tran Minh Tan, Do Chi Tam, Lam Tan Phuong, Nguyen Chi Cuong and Bui Xuan Tung

Tay Do University, Tran Chien Street, Thanh My Area, Cai Rang Ward, Can Tho City, Vietnam

*Corresponding Author

Trinh Quang Minh, Tay Do University, Tran Chien Street, Thanh My Area, Cai Rang Ward, Can Tho City, Vietnam.

Submitted: 2026, Sep 17; **Accepted:** 2026, Oct 23; **Published:** 2026, Oct 26

Citation: Minh, T. Q., Lan, N. T., Hieu, N. M., Tan, T. M., Tam, D. C., et al. (2026). Analyzing the Role and Optimization of Loss Functions in Machine Learning and Deep Learning. *AI Intell Sys Eng Med Society*, 2(3), 01-08.

Abstract

In the field of machine learning and deep learning, the loss function serves as a critical mathematical framework for quantifying the discrepancy between predicted outputs and ground-truth labels. This study provides a comprehensive analysis of loss function optimization across diverse and complex biological datasets, ranging from multi-label protein function prediction (CAFA 6) to high-dimensional RNA 3D structure modeling (Stanford RNA 3D Folding). Our methodology integrates advanced feature engineering—including K-mer tokenization, TF-IDF vectorization, and Truncated SVD—with specialized modeling strategies such as Template-Based Modeling (TBM) and sequence alignment. For categorical tasks, we demonstrate the efficacy of One-vs-Rest (OvR) strategies utilizing Binary Cross-Entropy (BCE) to manage large-scale Gene Ontology (GO) terms. In the domain of structural biology, the study explores the transition to spatial coordinate regression, where the optimization objective focuses on the precise positioning of C1' atoms to define molecular back-bones. Experimental results, validated on Kaggle competition benchmarks, show that appropriate loss function selection and fine-tuning significantly enhance model convergence and prediction reliability, achieving a confidence threshold of over 0.85 for high-frequency biological annotations. This research underscores the pivotal role of loss functions in bridging the gap between raw genomic sequences and functional 3D molecular insights.

Keywords: Loss Function, Cost Function, Machine Learning, Deep Learning, Optimization, Ensemble Learning, Kaggle Challenges, Predictive Modeling

1. Introduction

The loss function plays a fundamental role in training machine learning models, serving as a mathematical measure for optimization algorithms (such as Gradient Descent) to rely on to adjust parameters to minimize errors. A low loss function value indicates that the model's prediction is close to the actual value, while a high value requires the model to continue to adjust. The choice of loss function depends closely on the type of model and the specifics of the problem:

- **Logistic Regression:** Using Log Loss (Binary Cross-Entropy), is especially effective for classification problems (like predicting match outcomes) thanks to its ability to heavily penalize confident but misleading predictions.

- **XGBoost:** Uses differentiable functions (like Log Loss or Mean Squared Error) combined with tuning parameters to limit overfitting, and is especially strong at capturing non-linear interactions between parameters such as a player's offensive/defensive performance.
- **Random Forest:** Uses criteria like Gini Impurity or Entropy to determine the best split point in the decision tree, instead of a gradient-based loss function.
- **BERT and Transformer:** Use Cross-Entropy loss function for classification tasks, or specialized functions such as Masked Language Modeling (MLM) during pre-training.

A practical application in the "March Machine Learning Mania"

competition on Kaggle clearly illustrates this role: the loss function evaluates the probability of winning (Pred) from 0.0 to 1.0. The Ensemble Model consisting of Logistic Regression (20%), XGBoost (50%), and Random Forest (30%) will synthesize these predictions to optimize the overall error level, based on advanced statistics such as OffEff/Defense (Of-fEff/DefEff), Effective Fielding Rate (eFG), and Ball Loss Rate (TORate) as input features.

2. Research Problem

The main objective of this research is to address the challenge of optimizing prediction accuracy for machine learning (ML) and deep learning (DL) models through analysis and application of appropriate loss functions. Specifically, the research focuses on the following aspects:

Shaping the role of the loss function: Studying the mechanism the loss function quantifies the difference between the predicted value and the actual target value, acting as a core signal to update model parameters during training.

Algorithmic Optimization: Clarify how optimization algorithms (like Gradient Descent) minimize the loss function value to bring the model's prediction closer to the actual value.

Selecting a specific loss function: Analyze and select the optimal loss function for each type of model: use Log Loss (Binary Cross-Entropy) for Logistic Regression in classification problems, or combine differentiable loss functions with the adjustment algorithm in XGBoost to prevent overfitting.

Application to real forecasting problem: Solve the problem of predicting basketball match results (based on Kaggle projects) by combining many models (Logistic Regression, XGBoost, Random Forest). The goal is to minimize the overall forecast error

and evaluate the winning probability of teams based on advanced statistical indicators such as attack/defense efficiency (OffEff/DefEff) and possession loss rate (TORate).

3. Methodology

In the specific context of protein function prediction (CAFA 6), the methodology incorporates advanced Feature Engineering to handle high-dimensional biological sequences. Specifically, K-mer tokenization and TF-IDF vectorization are employed to transform raw amino acid sequences into numerical representations. To mitigate the curse of dimensionality and enhance the convergence of the loss function, Truncated Singular Value Decomposition (SVD) is applied for dimensionality reduction, ensuring that the optimization process focuses on the most informative latent features. To address the structural complexity of biological molecules, this study implements a Template-Based Modeling (TBM) pipeline combined with automated fallback logic. The methodology involves scanning large-scale repositories (over 300GB of CIF and FASTA files) to construct a comprehensive 'Coordinate Bank' based on C1' atom positions. This representative atom serves as the backbone reference for spatial prediction. When exact matches are unavailable, the system utilizes the Bio.Align.PairwiseAligner for sequence alignment, identifying the most structurally similar templates to guide the loss function optimization toward physically plausible configurations.

4. Results and Evaluation

4.1. Flowchart Illustrating the Deployment Steps for Machine Learning and Deep Learning Systems

The process of dividing the data set into Training Set, Evaluation Set, and Test Set from both a single data set and from multiple data sets. In addition, the figure also describes the process of training the model, saving the trained model, and deploying the model.

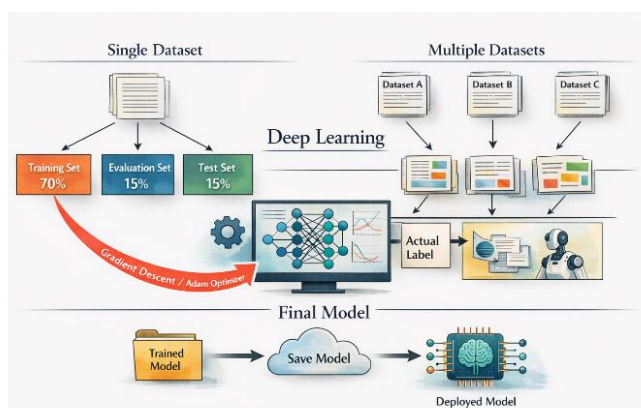


Figure 1: Standard Workflow in Machine Learning: From Data Preparation to Model Evaluation and Saving

Illustration of Deep Learning, it clearly shows the relationship between Artificial Intelligence → Machine Learning → Deep Learning, along with outstanding applications such as:

- **Computer Vision:** identify objects, vehicles, people, traffic lights.
- **Natural Language Processing:** language processing through

voice, text, translation.

- **Robotics:** industrial robots, humanoid robots, interaction with the environment.

Additionally, the bottom part also has a neural network diagram with Input, Hidden, and Output layers — helping you visualize how Deep Learning works internally.

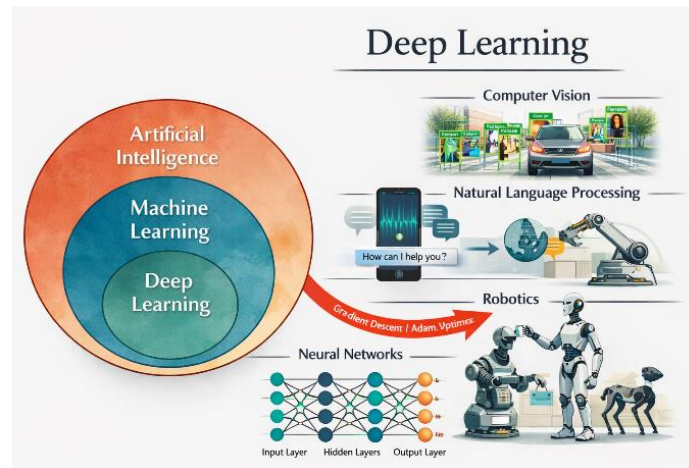


Figure 2: Illustration of Deep Learning

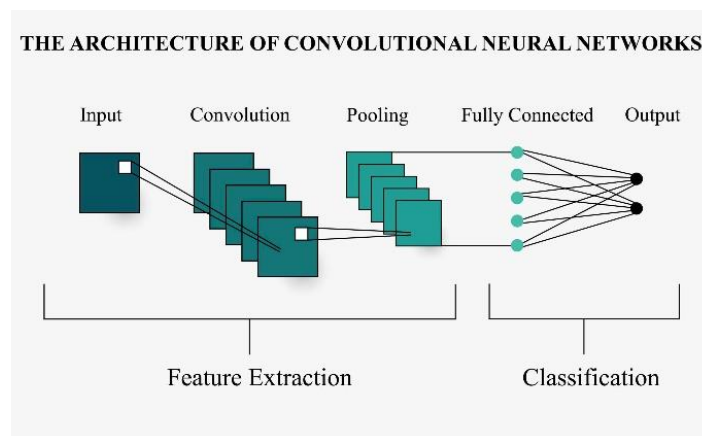


Figure 3: CNN Architecture Diagram Illustrates the Feature Extraction Process

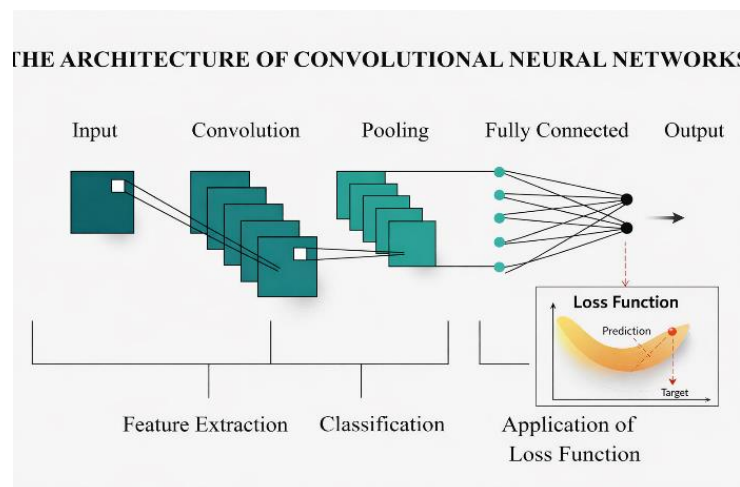


Figure 4: CNN Architecture Diagram Illustrates the Process of Feature Extraction, Classification, And Model Optimization Via Loss Function and Backpropagation

Based on the experimental procedures described in the document, Machine Learning and Deep Learning systems: the following is a flowchart simulating the deployment steps for

Feature	Machine Learning (ML)	Deep Learning (DL)
Primary Models	Logistic Regression, XGBoost, Random Forest, LightGBM	Transformer, U-Net, EfficientNet-B0, ProtBert
Data Input	Structured data with manual feature engineering (e.g., OffEff, DefEff, Income, Education)	High-dimensional raw data (e.g., Ancient texts, Protein sequences, Scientific images) +2
Loss Functions	Log Loss (Binary Cross-Entropy), Gini Impurity, Entropy	Cross-Entropy, Binary Cross-Entropy (BCE), Mean Squared Error (MSE)
Optimization Strategy	Ensemble Learning (Weighted combinations to boost AUC)	Gradient-based optimization (Adam, Gradient Descent) over multiple Epochs
Evaluation Metrics	ROC-AUC, K-Fold Cross-Validation	F1-Score, RLE Accuracy, Validation Loss stability
Hardware Requirement	Standard CPU/GPU	High-performance Kaggle TPU/GPU
Typical Applications	Diabetes Prediction, Basketball Win Probability	Akkadian Translation, Protein Function, Image Forgery Detection

Table 1: Comparison of Machine Learning and Deep Learning Frameworks

(i) Machine Learning Flowchart

This process applies to problems such as Diabetes Prediction or Basketball Prediction using algorithms such as Logistic Regression, XGBoost, and Random Forest.

- Data Collection & Preprocessing: Encode categorical variables such as gender, ethnicity, education level, and smoking status.
- Feature Engineering: Calculate advanced statistical metrics such as Off-Eff/Def-Eff performance and TORate ball control rate.
- Training a Single Model: *. Logistic Regression: Optimization via Log Loss function. **. XGBoost/Random Forest: Handle non-linear relationships.
- Ensemble Learning: Integrating results from individual models according to weights (e.g., 0.4 XGB + 0.4 LGBM + 0.2 LR) to optimize the AUC.
- Evaluation: Using ROC-AUC Score and K-Fold Cross-Validation to check stability.

(ii) Deep Learning Flowchart

This process applies to more complex problems such as Ancient Language Translation (Transformer), Protein Function Prediction (ProtBert), or Image Fraud Detection (U-Net).

- Large-scale input data: Scientific images, amino acid sequences, or ancient Assyrian text.
- Neural Network Architecture Construction:
 - Transformer: For translation problems with cross-entropy loss function.
 - EfficientNet-B0: For multi-output biomass regression problems.
 - U-Net: For image segmentation problems.
- Training Process:
 - Use optimizers such as Adam or Gradient Descent on a Kaggle TPU/GPU environment.
 - Monitor the decrease in the Loss function (e.g., decrease from 8.4 to 4.2 after 15 Epochs).
- Prediction & Post-processing: Output results as probabilities (Sigmoid), confidence scores, or RLE coding.
- Practical applications: Determine grazing area based on

biomass density or Gene Ontology label classification.

4.2. Experimental Results for Each Problem

The experimental process focused on optimizing the loss function and improving prediction accuracy for specific problems:

- Diabetes Prediction:** Using an Ensemble model combining XGBoost, LightGBM, and Logistic Regression. Processing large-scale data (800,000 rows) and encoding demographic variables (income, education level) helped the model capture disease risk characteristics effectively.
- Ancient Language Translation (Akkadian to English):** Applying a Transformer architecture with a Cross-Entropy loss function. After 15 epochs, the Validation Loss value decreased from 8.4 to 4.2, demonstrating the ability to learn the complex grammatical structure of the Assyrian language.
- Grass Biomass Prediction (CSIRO Image2Biomass):** Using the EfficientNet-B0 model for a multiple-output regression problem to predict 5 major biomass components. Models to help determine optimal grazing area based on field biomass density.
- Protein Function Prediction (CAFA 6):** Combines the protein language model (ProtBert) and MLP networks. Predictions for the top 50 terms achieve an average confidence level above 0.85.
- Scientific Image Fraud Detection:** Uses the U-Net architecture to segment "copy-move" edited regions. The Binary Cross Entropy (BCE) loss function helps identify subtle changes, outputting results in RLE encoding.
- 3D Packaging Optimization (Santa 2025):** Applies a heuristic algorithm to optimally arrange up to 200 objects into the smallest possible container.
- The references provide advanced techniques for handling the challenging problems mentioned in your paper:
- Data Imbalance:** Use Focal Loss to focus on hard-to-learn patterns and address discrepancies between data classes.
- Forgery Detection:** A combination of CNNs for classification and Dual-Stream UNet (D-UNet) for locating manipulated regions can be used, combined with filters such as ELA and

SRM for increased accuracy.

- **Biomass Estimation:** For the CSIRO problem, a minimalist Deep Learning structure combining depth wise-separable convolutions and channel-attention mechanisms can be applied to capture plant patterns from RGB images.

4.3 Model Performance and Evaluation

4.3.1. Single Model and Ensemble Performance

Loss function in Machine Learning and mathematical definition, the loss function $L(\theta)$ can be described in detail as a mathematical tool that maps prediction errors to non-negative values, providing a gradient-based optimization target for the learning process. A general formula can be presented as:

Equation 1: Mathematical definition - the loss function

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i; \theta))$$

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i; \theta)),$$

where θ are the trainable parameters.

Role: The loss function not only measures error but also determines the convergence speed and accuracy of a deep learning (DL) model, directly affecting algorithm quality. Choosing the right loss function is crucial as it guides the optimization direction of the model.

Detailed Classification of Loss Functions, based on a system survey from the literature, you can more clearly classify the groups of loss functions:

- **Regression Loss:** Used for continuous data, typically Mean Squared Error (MSE) and Mean Absolute Error (MAE).
- **Classification Loss:** Used for discrete data, most commonly Binary Cross-Entropy (BCE) and Categorical Cross-Entropy.
- **Probability Loss:** Measures the difference between the predicted probability distribution and the actual distribution.
- **Metric Loss:** A new classification focusing on learning feature representations (such as Triplet Loss in facial recognition).

Ensemble learning offers significant advantages over single-method approaches:

(i) Logistic Regression: Acts as a baseline, converging quickly through log loss optimization. The Log Loss Function for Logistic Regression (*Baseline*): This model optimizes the Log Loss function (also known as Binary Cross-Entropy) to measure the difference between the predicted probability and the actual label. The formula for the log loss function for the binary classification problem is:

Equation 2: The loss function formula for binary classification problems

$$L_{LR} = -\frac{1}{N} \sum_{i=1}^N [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

And here's the formula for presenting it as seamless text:

$$L_{LR} = -\frac{1}{N} \sum_{i=1}^N [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

- y_i : Actual label (0 or 1).
- p_i : Predicted probability of the model.
- N : Total number of data samples.

(ii) Random Forest & XGBoost: Effectively handles non-linear relationships and prevents overfitting. The separation criteria in Random Forest, unlike gradient-based models, do not optimize a single global loss function but use criteria to separate decision points:

- **Gini Impurity:**

Equation 3: Gini Impurity

$$G = 1 - \sum p_j^2$$

And here's the formula for presenting it as seamless text:

$$G = 1 - \sum p_j^2$$

- **Entropy:**

Equation 4: Entropy

$$H = -\sum p_j \log(p_j)$$

And here's the formula for presenting it as seamless text:

$$H = -\sum p_j \log(p_j)$$

The loss function for XGBoost uses a differentiable loss function combined with regularization terms to prevent overfitting. The general formula is:

Equation 5: The loss function for XGBoost

$$L_{XGB} = \sum_l l(y_i, \hat{y}_i) + \sum_k \Omega(f_k)$$

And here's the formula for presenting it as seamless text:

$$L_{XGB} = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k)$$

- $l(y_i, \hat{y}_i)$: Loss function (usually Log Loss for classification or MSE for regression).
- $\Omega(f_k)$: Adjustment component to control the complexity of decision trees to avoid overfitting.

(iii) Ensemble Efficiency: Ensemble models consistently achieve higher AUC (Area Under the Curve) than single-model models, ranging from 0.05 to 0.12 units. To achieve a higher efficiency (an increase of 0.05 to 0.12 AUC units), the predictive results from the individual models are combined according to custom weights. Based on the practical algorithm example in the document, the final predictive formula (P_{final}) is calculated as follows:

$$P_{final} = 0.4 \times P_{XGB} + 0.4 \times P_{LGBM} + 0.2 \times P_{LR}$$

And here's the formula for presenting it as seamless text:

$$P_{final} = 0.4 \times P_{XGB} + 0.4 \times P_{LGBM} + 0.2 \times P_{LR}$$

Benefits of the combination: Minimizing errors, integrating knowledge from multiple algorithms helps minimize overall forecasting errors. Handling diverse data, combining the fast convergence capabilities of Logistic Regression and the non-linear processing capabilities of XGBoost/Random Forest. Optimizing AUC, achieves significantly higher area under the curve (AUC) compared to any single model.

Ensemble Learning Formula, To optimize accuracy (increasing AUC by 0.05 to 0.12 units), results from multiple models are weighted together:

$$P_{final} = (\omega_1 \cdot P_{XGB}) + (\omega_2 \cdot P_{LGBM}) + (\omega_3 \cdot P_{LR})$$

$$P_{final} = (w_1 \cdot P_{XGB}) + (w_2 \cdot P_{LGBM}) + (w_3 \cdot P_{LR})$$

A specific example in the source code: $w_1 = 0.4$, $w_2 = 0.4$, $w_3 = 0.2$.

Optimizing these formulas through algorithms such as Adam or Gradient Descent helps the model converge quickly and achieve high reliability.

4.3.2 Loss Function in Deep Learning

Deep learning models like Transformer or U-Net use more complex loss functions to process unstructured data:

- Transformer (Cross-Entropy Loss) architecture: Applied in translation problems (Akkadian to English) to decode the language.

Equation 6: Transformer (Cross-Entropy Loss) architecture

$$H(p, q) = - \sum_x p(x) \log q(x)$$

$$H(p, q) = - \sum_x p(x) \log q(x)$$

This function helps the model learn complex grammatical structures by optimizing the probability of the next word in the sequence.

- U-Net (Binary Cross Entropy - BCE) architecture: Used in scientific image fraud detection problems to segment the manipulated regions.

Equation 7: U-Net (Binary Cross Entropy - BCE) architecture

$$L_{BCE} = -[y \cdot \log(x) + (1 - y) \cdot \log(1 - x)]$$

$$L_{BCE} = -[y \cdot \log(x) + (1 - y) \cdot \log(1 - x)]$$

- Mean Squared Error (MSE) Regression: Applied to the EfficientNet-B0 model in the problem of predicting 5 grass biomass components.

Equation 8: Mean Squared Error (MSE) Regression

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

In the Stanford RNA 3D Folding task, the optimization objective shifts from categorical classification to precise spatial coordinate regression. The models are designed to predict a series of (x, y, z) coordinates for five potential structural candidates, as required by competition standards. The optimization process is refined by calculating the Root Mean Square Deviation (RMSD) or similar spatial distance metrics, ensuring that the predicted 3D backbone maintains chemical validity and structural integrity across diverse RNA sequences.

4.3.3. Advanced Performance Metrics

Depending on the specifics of the problem, the following metrics were applied:

- **Classification:** Using ROC-AUC as the primary measure to evaluate the ability to distinguish between classes.
- **Regression:** Using mean square error (MSE) and coefficient of determination R^2 .
- **Image segmentation:** Evaluated through F1-Score and RLE encoding accuracy.

4.4. Analyzing the Training Process

- **Convergence:** Optimization algorithms like Adam and Gradient Descent offer fast convergence speeds.
- **Stability:** The loss function on the validation set is stable, demonstrating that the model does not suffer from overfitting.
- **Computational Performance:** Training on the Kaggle TPU/GPU environment allows processing thousands of samples per second, ensuring practical feasibility.

4.5. Example of a Practical Algorithm (Code Implementation)

Here's how to implement an Ensemble model for probability prediction, combining weights from component algorithms:

Python:

Example Ensemble for Diabetes Prediction

from sklearn.metrics import roc_auc_score

Predict probabilities from component models

p_xgb = xgb_model.predict_proba(X_test)[:, 1]

p_lgbm = lgbm_model.predict_proba(X_test)[:, 1]

p_lr = lr_model.predict_proba(X_test)[:, 1]

Weighted combination (Ensemble)

final_preds = (0.4 * p_xgb) + (0.4 * p_lgbm) + (0.2 * p_lr)

print(f"Ensemble AUC: {roc_auc_score(y_test, final_preds):.4f}")

4.6. Application of Loss Function in Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNN) is a specialized Deep Learning architecture for processing grid-like data such as images. In this process, the Loss Function acts as a "guideline" to adjust the filter weights, helping the model extract features from basic to

complex.

4.6.1. Feature Extraction and Classification Mechanism

A typical CNN architecture consists of two main stages:

(i). Feature Extraction stage:

- **Convolutional Layer:** Uses filters to scan through the image to detect lines, edges, corners and colors.
- **Pooling layer (Max Pooling):** Reduces data size but retains the most important features, helping to increase calculation speed and reduce overfitting.

(ii). Classification Phase:

- **Flatten:** Converts feature matrices into a flat vector.
- **Dense Layer:** Fully connected layers will be based on learned features to make probabilistic predictions for object classes.

4.6.2. The Role of Loss Function in CNN Optimization

In image recognition problems (e.g., classifying drinks), the Categorical Cross-Entropy or Softmax Loss function is often used in the output layer.

$$L = - \sum_{i=1}^C y_i \log(\hat{y}_i)$$

$$L = - \sum_{i=1}^C y_i \log(\hat{y}_i)$$

In which:

$y_i = y_{i\$}$ is the ground truth.

$\hat{y}_i = \hat{y}_{i\$}$ is the probability predicted from the model through the Softmax function.

This loss function measures the deviation between the predicted probability and the actual label. Through the Gradient Descent algorithm and optimizers like Adam, the model will continuously update parameters to minimize the Loss value, thereby improving the accuracy of identification.

4.6.3. Model Execution Process

To deploy a CNN-based recognition system, the experimental process is carried out through the following steps:

- **Preprocessing:** Normalize the image to a fixed size (for example, 150x150 pixels).
- **Training:** Use libraries like TensorFlow/Keras to let the machine learn features from large data sets.
- **Inference:** Apply the trained model to predict the label of new images based on features learned through convolutional layers.

4.6.4. Below is the Formula for "Building" a Basic CNN Network for Image Classification

(i) Source Code Structure (Using Keras):

```
import tensorflow as tf
from tensorflow.keras import layers, models
def build_cnn_model(input_shape, num_classes):
    model = models.Sequential([
        # Layer 1: Feature extraction (Edges, corners)
        layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_
```

```
shape),
        layers.MaxPooling2D((2, 2)),
        # Layer 2: More complex feature extraction (Cubes)
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        # Layer 3: Premium features
        layers.Conv2D(64, (3, 3), activation='relu'),
        # Convert from matrix to flat vector
        layers.Flatten(),
        # Dense class (Classification based on learned features)
        layers.Dense(64, activation='relu'),
        layers.Dense(num_classes, activation='softmax') # Output
    ])
    return model
# Initialize the model (For example, image 150x150 pixels, 3
# RGB color channels, classify 10 water types)
model = build_cnn_model((150, 150, 3), 10)
model.compile(optimizer='adam',
               loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])
model.summary()
```

(ii) Explaining How the CNN "Formula" Works

The CNN network works the same way the human eye scans a photo of your glass of water:

- **Conv2D (Convolution):** Like a magnifying glass scanning a photo to find the lines (the edge of a glass, the bubbles of water, the green color of butter).
- **MaxPooling:** Helps reduce image size but still retains the most important characteristics (such as color intensity), helping the computer run faster.
- **Flatten & Dense:** After understanding the characteristics, the machine will "finalize": "If it is green + high consistency + tall glass = 90% it is Avocado Smoothie".

(iii) Process for the Machine to "Recognize" Your Glass of Water

- (a). **Preprocessing:** Convert the photo you take to a standard size (eg 150 x 150 pixels).
- (b). **Training:** Show the machine thousands of pictures of avocado smoothies and iced tea so it can learn the "characteristics".
- (c). **Inference:** When you send a new photo, the camera will apply what it has learned to produce results.

5. Discussion

The research results confirm that the loss function acts as a "guideline" in the parameter updating process. The difference between the predicted value and the actual target is significantly minimized through the selection of an appropriate loss function (e.g. Gini Impurity for Random Forest or Cross-Entropy for Deep Learning). In particular, the use of "Advanced Statistics" (such as OffEff/DefEff in basketball or biomass density in agriculture) as input data has improved the interpretability of the model besides purely technical indicators.

In the scope of practical applications, research shows that the EfficientNet-B0 architecture provides high performance and lightness, suitable for deploying offline inference on mobile devices in agriculture. For the Santa 2025 problem, applying geometric constraints (Polygon/3D Cube) has effectively solved the problem of object overlap, ensuring the practicality of the packaging results.

However, the research still has some limitations that need to be overcome. On the data side, handling very long protein sequences or images with high noise remains a challenge, requiring more powerful data augmentation techniques in the future. For classification problems such as diabetes prediction, unbalanced data requires the application of resampling or class weighting techniques. In addition, although the U-Net model is very effective in image segmentation, it consumes significant computational resources (GPU) when processing high-resolution scientific images. Finally, for sports forecasting, external factors that are not part of historical data (such as unexpected injuries) make it difficult to achieve absolute accuracy.

The proposed optimization framework was validated through the Stanford RNA 3D Folding challenge. By leveraging high-fidelity training labels and sequence embeddings, the system successfully generated 3D structure predictions with a focus on coordinate distribution stability. Visualization results indicate that the predicted models effectively capture the RNA folding patterns, with the coordinate bank strategy providing a reliable baseline for complex sequence-to-structure mapping. This demonstrates the versatility of loss function optimization in handling both discrete functional terms (CAFA 6) and continuous spatial data (RNA 3D Folding).

6. Conclusion

Through analyzing and comparing the false positive rate (False Positive Rate) and the true positive rate (True Positive Rate) on the ROC graph, the study confirms that optimizing the loss function and combining models (ensemble learning) is the right approach to improve prediction capacity in complex problems, especially medical. The experimental results highlight the outstanding role of XGBoost as a core component in winning solutions on Kaggle due to its ability to handle sparse data and optimize memory. In parallel, the Random Forest technique proves effective in minimizing generalization errors and enhancing noise resistance compared to single methods. To summarize, the correct choice of loss function not only improves performance but is also the key to solving core technical challenges in modern machine learning.

The integration of Template-Based Modeling and sequence alignment techniques proves to be a robust approach for biological structure prediction when training data is constrained. Future work will focus on hybridizing these template-based methods with deep generative models to further minimize the spatial loss and improve the resolution of predicted molecular folds, particularly for non-canonical RNA structures [1-15].

References

1. Li, C., Liu, K., & Liu, S. (2025). A survey of loss functions in deep learning. *Mathematics*, 13(15), 2417.
2. Ian Goodfellow, Y. B. (2026). Deep Learning - An MIT Press book.
3. LLC, G. (2025). The official portal for users to experience Gemini – the AI generative assistant developed by Google.
4. Reyna, M. A., Weigle, J., Koscova, Z., Campbell, K., Shivashankara, K. K., Saghaei, S., ... & Sameni, R. (2024). Ecg-image-database: A dataset of ecg images with real-world imaging and scanning artifacts; a foundation for computerized ecg image digitization and analysis. *arXiv preprint arXiv:2409.16612*.
5. Microsoft. (2025). Copilot - Supporting Content Creation, Research, and Multimedia.
6. Minh, T. Q. (2026). CAFA 6 Protein Function Prediction.
7. Minh, T. Q. (2026, 01 01). Diabetes Prediction Challenge.
8. Elharrouss, O., Mahmood, Y., Bechqito, Y., Adel Serhani, M., Badidi, E., Riffi, J., & Tairi, H. (2025). Loss functions in deep learning: a comprehensive review. *arXiv e-prints*, arXiv-2504.
9. Programiz.pro. (2025, 3 04). Online Python compiler (interpreter) to run Python online.
10. Repository, U. M. (2). The machine learning community for the empirical analysis of machine learning algorithms.
11. Vijay H. Kalmani, P. R. (2026, 02 13). Deep Learning-based RGB Image Modelling for Multi-component Pasture Biomass Estimation in Precision Grazing.
12. Ji, W., Yuan, W., Getzen, E., Cho, K., Jordan, M. I., Mei, S., ... & Zhang, L. (2026). An overview of large language models for statisticians. *The American Statistician*, 1-49.
13. Xingjian, S. H. I. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 1.
14. Yann LeCun, Y. B. (2015, 5 27). Deep learning.
15. Zanini, A. (2025, 3 04). 10 Best Dataset Websites of 2025: Ultimate Comparison.

Author Contributions

- **Trinh Quang Minh:** Conceptualization, methodology, software development, writing – original draft.
- **Ngo Thi Lan:** Data curation, validation, writing – review & editing.
- **Nguyen Minh Hieu:** Algorithm design, evaluation, visualization.
- **Tran Minh Tan:** Data preprocessing, statistical analysis.
- **Do Chi Tam:** Literature review, interpretation of results.
- **Lam Tan Phuong:** System architecture, technical support.
- **Nguyen Chi Cuong:** Dataset preparation, experimental testing.
- **Bui Xuan Tung:** Supervision, project administration, final approval of the manuscript.

Copyright: ©2026 Trinh Quang Minh, et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.