# Machine Learning: Skin Cancer: Benign and malignant

[**Machine Learning: Skin Cancer: Benign and malignant 1**](#_Toc162964018)

[**Brief Introduction: 1**](#_Toc162964019)

[**Implement and document the ensemble learning model and the training algorithm. 3**](#_Toc162964020)

[Process Data 3](#_Toc162964021)

[SHAP 4](#_Toc162964022)

[Split data set 5](#_Toc162964023)

[Flattening the images 5](#_Toc162964024)

[Evaluation metric 6](#_Toc162964025)

[Hyper-parameter Tweaking 6](#_Toc162964026)

[Building an Artificial Neural Network (ANN) Model: 8](#_Toc162964027)

[**Cost function Selection： 8**](#_Toc162964028)

[**Selet echos 12**](#_Toc162964029)

[**Choose different learning rates: 14**](#_Toc162964030)

[**Select Batch Size: 15**](#_Toc162964031)

[**Select Activation Function: 16**](#_Toc162964032)

[**Select Number of Hidden Units: 17**](#_Toc162964033)

[**Select Optimizer: 18**](#_Toc162964034)

[**Proposing new custom loss function. 19**](#_Toc162964035)

[**Compare with other models and compare my own two models. 21**](#_Toc162964036)

[**Reference List: 24**](#_Toc162964037)

## Brief Introduction:

In the current field of machine learning, identifying key features to address classification problems offers numerous benefits, including reducing computational costs and improving classification accuracy (Itauma [1], 2016). Classification problem is a category of machine learning where the main goal is to classify samples in a data set into different categories, in this case two categories. Research indicates that a typical radiologist needs to interpret an image every 3-4 seconds to keep up with clinical workloads (Huang [2], 2023), So using machine learning to help doctors classify images and save them time is a very helpful thing.

The dataset in question focuses on local images of skin cancer patients, aiming to classify images into benign and malignant categories, constituting a typical binary classification problem. Observations of image features reveal that malignant images typically exhibit characteristics such as dark coloration and uneven distribution of spots, while benign images show features like red coloration with spots generally concentrated in one area.

To address this classification problem, I chose to use an Artificial Neural Network (ANN) for image processing. The process involves the following four steps:

Flattening the images: To input images into a fully connected neural network, it is common practice to flatten the two-dimensional image arrays into one-dimensional vectors. Input layer: In a neural network, the input layer receives and processes the input from the data source. For flattened image vectors, each pixel value serves as a node in the input layer. Fully connected layers: In an ANN, each layer's nodes are connected to all nodes in the previous layer. These layers enable the network to learn complex patterns and features from the input data. Output layer: The output layer varies depending on the task of the model. In the case of classification, it typically consists of nodes representing the different classes, with each node outputting the probability of belonging to that class.

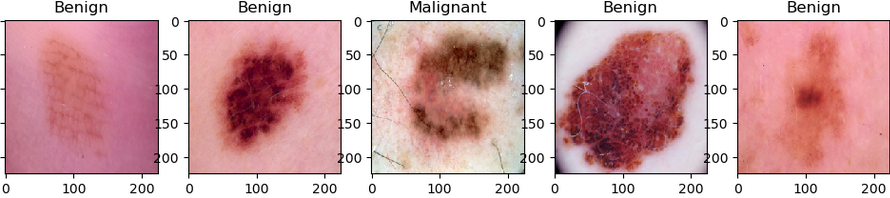
The relevance of this classification problem to the real world is demonstrated by the fact that skin cancer categories can even be determined by predictive tasks, reducing the number of manual annotations required to train medical imaging models, reducing the cost and time required for model development, and reducing the workload of radiologists (Huang,2023). So, this kind of machine learning can help doctors quickly and accurately determine the type of skin cancer image. This is essential for the efficiency and accuracy of the doctor. Therefore, by building such a classification model, we can achieve more efficient skin cancer image recognition in the medical field, thereby improving the efficiency of doctors, and even machines to achieve higher accuracy than human experience.

## Implement and document the ensemble learning model and the training algorithm.

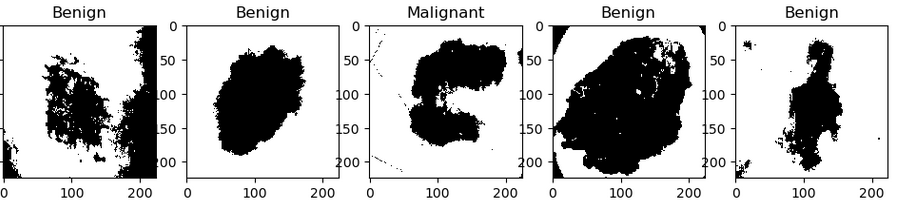
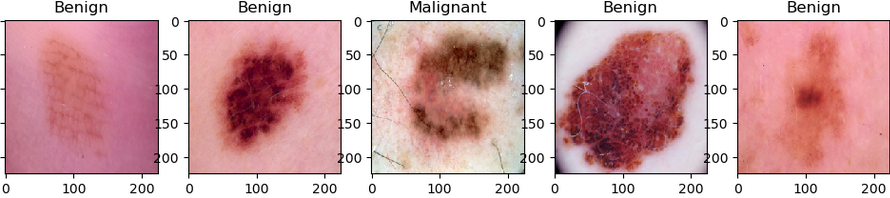
图示

描述已自动生成

### Process Data



(Diagram1: Data before Processing)



(Diagram2: Data after Processing)

### SHAP

图表, 瀑布图

描述已自动生成

SHAP (SHapley Additive exPlanations) plots provide insights into the contribution of each feature to the model's predictions. In this case, the SHAP plots indicate that Feature 5, Feature 18, Feature 14, and Feature 1 have the most substantial influence on the model's decision-making process.

### Split data set

图表, 条形图

描述已自动生成

**(Diagram3: Number of Training Data and Testing Data)**

The dataset represents a binary classification problem, with two classes: 'Benign' and 'Malignant'. Consequently, when selecting the loss function and optimizer, we will choose those specifically designed for binary classification problems.

**Table 1 Description of skin cancer set.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | | **Number** | **Label** |
| Training | Benign | 1400 | 0 |
| Malignant | 1200 | 1 |
| Testing | Benign | 350 | 0 |
| Malignant | 300 | 1 |

### Flattening the images

Flattening the images: In order to input images into a fully connected neural network, it is common practice to flatten the two-dimensional image arrays into one-dimensional vectors.

日历

描述已自动生成

### Evaluation metric

**Accuracy**:

Accuracy measures the proportion of correctly classified samples among the total number of samples. It's calculated as the ratio of the number of correct predictions to the total number of predictions.

**Sensitivity (True Positive Rate):**

Sensitivity measures the proportion of true positive predictions (correctly identified positive samples) among all actual positive samples. It is calculated as TP / (TP + FN), where TP is the number of true positives and FN is the number of false negatives.

**Specificity (True Negative Rate):**

Specificity measures the proportion of true negative predictions (correctly identified negative samples) among all actual negative samples. It is calculated as TN / (TN + FP), where TN is the number of true negatives and FP is the number of false positives.

**Precision (Positive Predictive Value):**

Precision measures the proportion of true positive predictions among all positive predictions. It is calculated as TP / (TP + FP), where TP is the number of true positives and FP is the number of false positives.

**Recall (Sensitivity):**

Recall, also known as sensitivity, measures the proportion of true positive predictions among all actual positive samples. It is calculated as TP / (TP + FN), where TP is the number of true positives and FN is the number of false negatives.

**ROC-AUC (Receiver Operating Characteristic - Area Under the Curve):**

ROC-AUC measures the area under the receiver operating characteristic curve, which is a plot of the true positive rate (sensitivity) against the false positive rate (1 - specificity) for different threshold values. It provides a single scalar value to assess the performance of a classification model across all possible thresholds.

### Hyper-parameter Tweaking

**Learning Rate:**

Describes the size of the step the model takes to update weights during each iteration. A too high learning rate may lead to the model failing to converge, while a too low learning rate may result in slow training or getting stuck in local minima.

**Epochs**:

The number of complete passes through the training dataset. The choice of epochs can be determined by observing the model's performance on the validation set, indicating whether to increase or decrease the number of iterations.

**Batch Size:**

The number of samples used to update weights during training. A larger batch size may increase training speed but could also lead to increased memory requirements.

**Number of Hidden Units:**

The number of neurons in each hidden layer. This choice may affect the model's capacity and training speed.

**Activation Function:**

Introduced nonlinearity into each layer of the neural network. Common activation functions include ReLU, Sigmoid, and Tanh.

**Loss Function:**

Measures the difference between the model's predictions and the actual values, such as Mean Squared Error (MSE) or Cross-Entropy Loss.

**Optimizer:**

Algorithm controlling the weight updates, such as Stochastic Gradient Descent (SGD), Adam, RMSprop, etc.

**Non-hidden Parameters:**

**Weights**:

The weights of each connection in the neural network. These weights are continuously updated during training and are crucial for the model to learn.

**Biases:**

The bias terms for each neuron, used to adjust the input to the activation function. Similar to weights, biases are learned during training.

**Gradients:**

The gradients of the loss function with respect to each weight and bias. These gradients are used to update the model's parameters, gradually converging to the optimal solution.

**Hidden Activations:**

The activation values of each layer in the neural network during training. These intermediate states are used in forward and backward propagation but are typically not directly visible.

**Learned Features:**

The neurons in the hidden layers may learn abstract representations of the data, which are often difficult to interpret.

**Dropout Masks:**

When using Dropout regularization, a set of dropped neurons is generated for each training batch, forming a binary mask. These masks are hidden because they are not used during inference.

**Model Configuration:**

The architecture of the model, including the number of layers, nodes per layer, etc. These details determine the overall structure and number of parameters in the model.

**Next, we will focus on discussing the impact of non-hidden parameters on the model.**

### Building an Artificial Neural Network (ANN) Model:

#### **Cost function Selection**：

Our dataset is a binary problem, so we choose binary for activation and loss functions.

**Layer selection.**

Let's start with two hidden layers.

|  |  |
| --- | --- |
| Table 2 hidden layers | |
|  | | Selection |
| Cost function | custom\_binary\_crossentropy | |
| Learning rate | 1e-06(0.000001) | |
| Activation Function | Sigmoid，1 | |
| Epochs | 14 | |
| Cost Optimizer | Adam | |
| Hidden Units | [ 64, 32, 1] | |
| Batch Size | 16 | |

图表, 树状图

描述已自动生成

At this point, we noticed that the model's performance was poor with two hidden layers, as indicated by the Confusion Matrix's four metrics: True Positives (TP), True Negatives (TN), False Positives, and False Negatives. This was due to our initial setting of epochs not fully converging.

|  |  |
| --- | --- |
| Table 3 hidden layers | |
|  | | Selection |
| Cost function | custom\_binary\_crossentropy | |
| Learning rate | 1e-06(0.000001) | |
| Activation Function | Sigmoid，1 | |
| Epochs | 14 | |
| Cost Optimizer | Adam | |
| Hidden Units | [128, 64, 32, 1] | |
| Batch Size | 16 | |

图表

描述已自动生成

When we chose three hidden layers, the model improved somewhat, but the predictions for malignancies (False Positives and False Negatives) remained quite low.

|  |  |
| --- | --- |
| Table 4 hidden layers | |
|  | | Selection |
| Cost function | custom\_binary\_crossentropy | |
| Learning rate | 1e-06(0.000001) | |
| Activation Function | Sigmoid，1 | |
| Epochs | 14 | |
| Cost Optimizer | Adam | |
| Hidden Units | [128, 64, 32, 16, 1] | |
| Batch Size | 16 | |

图表, 树状图

描述已自动生成

When we opted for four hidden layers, we observed further optimization in the model, with the loss decreasing at a faster rate.

|  |  |
| --- | --- |
| Table 5 hidden layers | |
|  | | Selection |
| Cost function | custom\_binary\_crossentropy | |
| Learning rate | 1e-06(0.000001) | |
| Activation Function | Sigmoid，1 | |
| Epochs | 14 | |
| Cost Optimizer | Adam | |
| Hidden Units | [256，128, 64, 32, 16, 1] | |
| Batch Size | 16 | |

图片包含 图形用户界面

描述已自动生成

However, when we selected five hidden layers, we noticed a decline in the model's performance.

9 Matrix of different hidden layers

By observing the value of loss, we can see that overfitting begins when 5 hidden Through comparing 9 matrices, it can be observed that the performance **is optimal with 4 hidden layers.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Hidden layer** | **2 hidden** | **3 hidden** | **4 hidden** | **5 hidden** |
| **Test Accuracy** | 0.5212121212121212 | 0.5363636363636364 | **0.5712121212121212** | 0.5242424242424243 |
| **Precision** | 0.45454545454545453 | 0.45454545454545453 | **0.4545454545454545354545454545453** | 0.45454545454545453 |
| **Recall** | 0.15454545454545454 | 0.11818181818181818 | 0.3378787878787879 | **0.4575757575757576** |
| **F1 Score** | 0.23066485753052918 | 0.18759018759018756 | 0.38762384842690767 | **0.5424242424242425** |
| **Specificity** | 0.8454545454545455 | **0.8818181818181818** | 0.6621212121212121 | 0.6621212121212121 |
| **False Positive Rate** | 0.15454545454545454 | 0.11818181818181817 | **0.6621212121212121** | 0.4575757575757575 |
| **False Negative Rate** | 0.8454545454545455 | 0.8818181818181818 | **0.3378787878787879** | 0.5424242424242425 |
| **ROC AUC Score** | 0.5 | 0.5 | 0.5 | 0.5 |
| **Balanced Accuracy** | 0.5 | 0.5 | 0.5 | 0.5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Loss**  **Balanced Accuracy** | **2 hidden** | **3 hidden** | **4 hidden** | **5 hidden** |
| loss:7.6213  loss:7.3906  loss: 7.4021  loss: 7.6534  loss: 7.8101  loss: 7.8086  loss: 7.7268  loss: 7.7744  loss: 7.7268  loss: 7.7744  loss: 7.9690  loss: 7.7721  loss: 7.2849  loss: 7.3451 | loss: 7.8844  loss: 7.3128  loss: 7.2101  loss: 7.1518  loss: 7.1091  loss: 6.9888  loss: 6.9498  loss: 6.9498  loss: 6.9498  loss: 6.9498  loss: 6.9618  loss: 6.9126  loss:6.8764  loss:6.8764 | **loss: 7.0382**  **loss: 6.5878**  **loss: 6.4370**  **loss: 6.0997**  **loss: 6.2420**  **loss: 6.2370**  **loss: 6.0851**  **loss: 5.3538**  **loss: 5.0212**  **loss: 4.5814**  **loss: 4.7367**  **loss: 4.2044**  **loss: 4.0802**  **loss: 3.8348** | loss: 5.4847  loss: 2.8558  loss: 2.0820  loss: 1.6630  loss: 1.4460  loss: 1.2389  loss: 1.1519  loss: 1.0428  loss: 0.9845  loss: 0.9532  loss: 0.9221  loss: 0.8340  loss: 0.8235  loss: 0.8014 |

Based on the decrease in loss, I believe the model performs best with four hidden layers. The initial two hidden layers (in models with two and three hidden layers) exhibit slower decreases in loss. However, with four hidden layers, there's a noticeable acceleration in loss reduction. Moreover, compared to models with five hidden layers, the **four-hidden-layer** configuration allows for easier adjustments and interpretations.

#### Selet echos

Epoch 72/80 loss: 0.1485

Epoch 73/80 loss: 0.1692

Epoch 74/80 loss: 0.1630

Epoch 75/80 loss: 0.1487

Epoch 76/80 loss: 0.1548

Epoch 77/80 loss: 0.1424

Epoch 78/80 loss: 0.1823

Epoch 79/80 loss: 0.1320

Epoch 80/80 loss: 0.1455

By observing the epochs, we can notice that around the 80th epoch, our loss value stabilizes at around 1.4. Therefore, maintaining the number of epochs at around 80 is optimal.

**Final All parameter selection**

The following results are obtained according to the subsequent parameter adjustment

**Table The final choice**

|  |  |  |
| --- | --- | --- |
|  | | Selection |
| Cost function | custom\_binary\_crossentropy | |
| Learning rate | 9e-7 (0.000001) | |
| Activation Function | Sigmoid，1 | |
| Epochs | 80 | |
| Cost Optimizer | Adam | |
| Hidden Units | [128, 64, 32, 16, 8] | |
| Batch Size | 32 | |

图表, 树状图

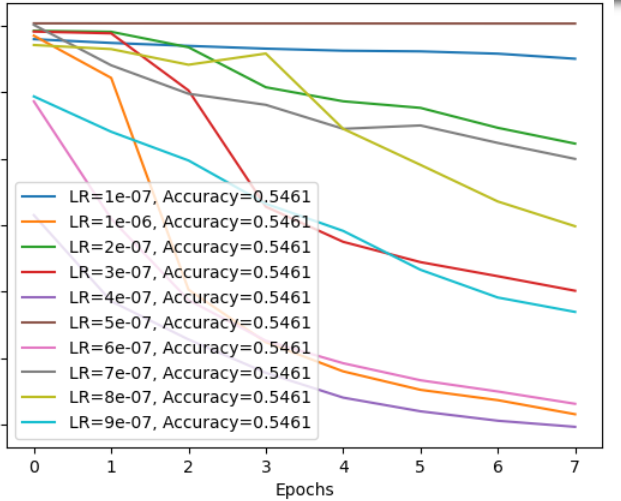
描述已自动生成

|  |  |
| --- | --- |
| **Matrix** | **Custom\_binary** |
| **Test Accuracy** | 0.9233977794647217 |
| **Precision** | 0.45454545454545453 |
| **Recall** | 0.453030303030303 |
| **F1 Score** | 0.4537866140537259 |
| **Specificity** | 0.546969696969697 |
| **False Positive Rate** | 0.453030303030303 |
| **False Negative Rate** | 0.453030303030303 |
| **ROC AUC Score** | 0.5 |
| **Balanced Accuracy** | 0.5 |
| **Loss** | 0.1604 |

#### Choose different learning rates:

图表, 折线图

描述已自动生成



As can be seen from the figure, the learning rate of choose 9e-07(0.0000009) is best, and in later experiments which can be further improved by slightly increasing the learning rate to 1e-06(0.000001). Therefore, I chose this learning rate1e-06(0.000001) to build my model.

#### Select Batch Size:

Model Accuracy vs. Epochs for Different Batch Sizes

图表, 折线图

描述已自动生成

We found that among the four different batch sizes tested, the performance is better when using a batch size of 32.

#### Select Activation Function:

图表, 折线图

描述已自动生成

**Softmax**: Softmax is often used as the activation function in the output layer of a neural network for multi-class classification problems. It takes a vector of arbitrary real-valued scores as input and squashes them into probabilities that sum up to 1.

**ReLU (Rectified Linear Unit):** ReLU is one of the most popular activation functions. It is defined as f(x)=max(0,x). ReLU is computationally efficient and helps mitigate the vanishing gradient problem.

**Sigmoid**: The sigmoid function is a smooth, It squashes the input values between 0 and 1, making it suitable for binary classification tasks where the output needs to be interpreted as probabilities. 图示

低可信度描述已自动生成

**Tanh (Hyperbolic Tangent)**: Tanh is similar to the sigmoid function but squashes the input values between -1 and 1 文本

描述已自动生成

#### Select Number of Hidden Units:

[512, 256, 128, 64,32], Configuration 1

[256, 128, 64, 32,16], Configuration 2

[128, 64, 32, 16, 8], Configuration 3

图表, 折线图

描述已自动生成

Based on the charts, we observe that Configuration 3, with the architecture [128, 64, 32, 16, 8], performs the best. This configuration consists of an input layer with 128 units, followed by hidden layers with 64, 32, and 16 units, and finally an output layer with 8 units.

#### Select Optimizer:

图表, 折线图

描述已自动生成

Adam optimizer

SGD (Stochastic Gradient Descent) optimizer

RMSprop (Root Mean Square Propagation) optimizer

**Adam optimizer**: Adam stands for Adaptive Moment Estimation. It's an adaptive learning rate optimization algorithm that's well-suited for training deep neural networks.

**SGD optimizer:** Stochastic Gradient Descent is a classic and simple optimizer. It updates the weights of the model using the gradient of the loss function with respect to the parameters. SGD can be used with or without momentum.

**RMSprop optimizer:** RMSprop is an optimizer that utilizes the magnitude of recent gradients to normalize the gradients.

**Final All parameter selection**

**Table The final choice**

|  |  |  |
| --- | --- | --- |
|  | | Selection |
| Cost function | custom\_binary\_crossentropy | |
| Learning rate | 1e-06(0.000001) | |
| Activation Function | Sigmoid，1 | |
| Epochs | 80 | |
| Cost Optimizer | Adam | |
| Hidden Units | [128, 64, 32, 16, 8] | |
| Batch Size | 32 | |

图表, 树状图

描述已自动生成

## Proposing new custom loss function.

图表, 折线图

描述已自动生成

One possible approach is to combine the advantages of custom\_binary\_crossentropy and custom\_loss\_v2, while also considering the model's performance in accurately identifying positive and negative instances.

**Advantages of** **Binary Cross entropy Loss:**

Suitable for binary classification tasks, as it measures the model's accuracy in predicting each class.

By minimizing the cross-entropy loss, the model can better optimize its parameters to make the predicted probability distribution closer to the true label distribution.

**Advantages of Loss V2:**

It incorporates a regularization term, effectively preventing overfitting and improving the model's generalization ability.

The squared loss function is less sensitive to outliers, reducing the impact of noise to some extent.

Considering the advantages of these two loss functions, we can attempt to combine them to balance the requirements for classification accuracy and model generalization. We can design a new loss function that combines binary cross-entropy loss and mean squared error (MSE) loss, while also adding a regularization term. The goal is to minimize classification errors while controlling the complexity of the model to prevent overfitting.

Specifically, we can calculate the new loss function using the following formula:

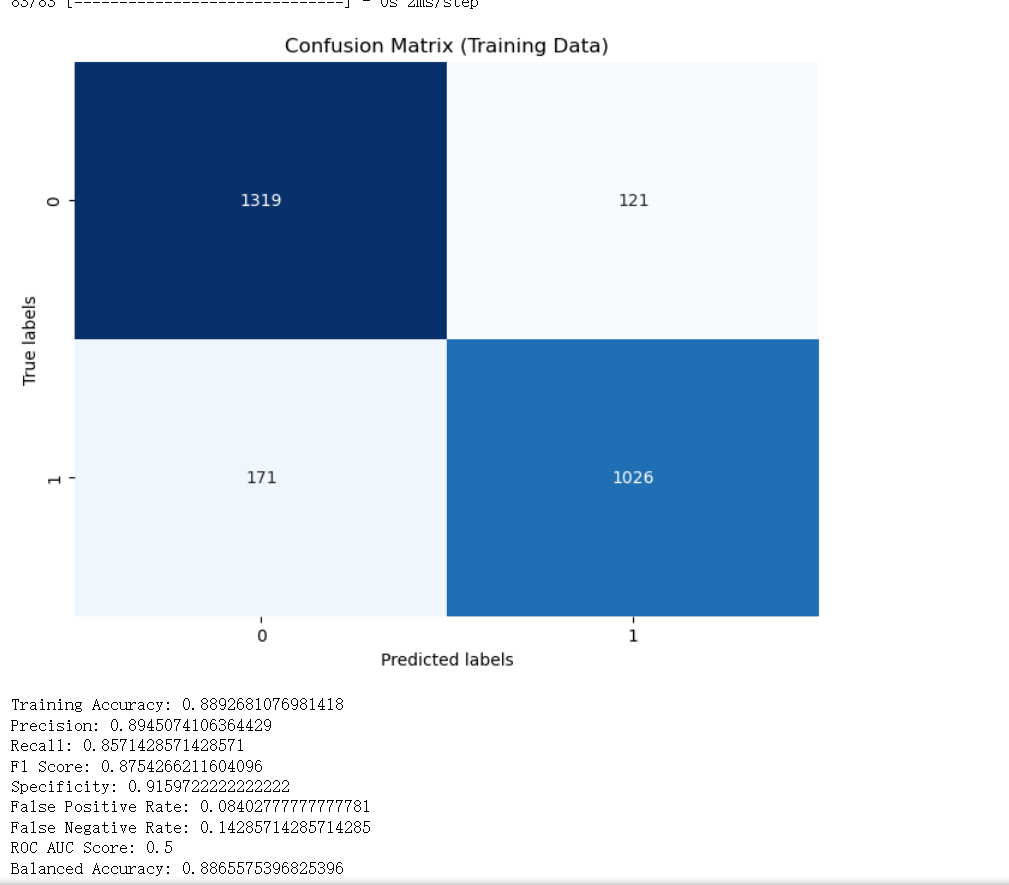
**Total Loss=Binary Crossentropy Loss+MSE Loss+Penalty**

（Here, Binary Crossentropy Loss and MSE Loss represent the binary cross-entropy loss and mean squared error, respectively, while Penalty is the regularization term.）

This design of the loss function can balance the model's requirements for classification accuracy and generalization ability to some extent. Binary Crossentropy Loss ensures the accuracy of the model's predictions for each class, while MSE Loss and Penalty control the model's complexity to prevent overfitting. This combined approach can better balance the bias and variance of the model, thereby improving overall performance.

**Put new function**

|  |  |  |
| --- | --- | --- |
|  | | Selection |
| Cost function | Combined Loss | |
| Learning rate | 1e-06(0.000001) | |
| Activation Function | SoftMax，2 | |
| Epochs | 80 | |
| Cost Optimizer | Adam | |
| Hidden Units | [128, 64, 32, 16, 8] | |
| Batch Size | 32 | |



## Compare with other models and compare my own two models.

**Comparison with other models**

**SVC**

Support Vector Classifier (SVC) is a supervised machine learning algorithm used for classification tasks. It belongs to the family of Support Vector Machines (SVM), which are widely used for both classification and regression tasks.

**Convolutional network**

A Convolutional Neural Network (CNN or ConvNet) is a type of deep neural network architecture primarily designed for processing and analyzing visual data, such as images and videos. CNNs have achieved remarkable success in various computer vision tasks, including image classification, object detection, segmentation, and more.

**EfficientNetB0**

EfficientNetB0 is a convolutional neural network architecture designed for efficient and effective image classification tasks. It is part of the EfficientNet family of models developed by Google Brain. Here's an explanation of its key characteristics:

Evaluation Metrics of Comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Final Loss | Recall | Fi-score |
| SVC [3] | 76% | 0.3740 | - | - |
| Convolutional network [4] | 91% | 0.0671 | 0.94 | 0.85 |
| EfficientNetB0[5] | 85% | 0.3786 | 91% | 0.91 |

By comparing the evaluation metrics of the three models, it's evident that the Convolutional Network performs the best in handling image classification tasks. Specifically, the Convolutional Network achieves an accuracy of 91%, with a loss value of only 0.0671. Furthermore, it also demonstrates excellent performance in terms of recall (0.94) and F1-score (0.85). In contrast, the Support Vector Classifier (SVC) and EfficientNetB0 show slightly inferior performance.

The superior performance of the Convolutional Network in image classification tasks can be attributed to its ability to automatically learn and extract key features from images without the need for manual feature engineering. Through structures such as convolutional layers, pooling layers, and fully connected layers, it effectively captures spatial and semantic information in images, enabling efficient processing and accurate classification of image data.

Overall, the Convolutional Network, as a deep learning model specifically designed for handling visual data, demonstrates outstanding performance and effectiveness in image classification tasks.

Next, I compare the two models I created with my own, and the two models with the three models.

**Comparison My creation with other models**

|  |  |  |
| --- | --- | --- |
| **Matrix** | **Custom\_binary** | **New loss function** |
| **Test Accuracy** | **0.9233977794647217** | 0.8278346605991658 |
| **Precision** | 0.45454545454545453 | **0.8233246301131418** |
| **Recall** | 0.453030303030303 | **0.7903091060985797** |
| **F1 Score** | 0.4537866140537259 | **0.8064791133844843** |
| **Specificity** | 0.546969696969697 | **0.8590277777777777** |
| **False Positive Rate** | 0.453030303030303 | **0.14097222222222228** |
| **False Negative Rate** | 0.453030303030303 | **0.2096908939014202** |
| **ROC AUC Score** | 0.5 | **0.5** |
| **Balanced Accuracy** | 0.5 | **0.8246684419381787** |
| **Loss** | **0.1604** | 0.3574 |

In summary, while the first model might have lower loss and slightly higher accuracy, the second model generally outperforms it in terms of precision, recall, F1 score, specificity, false positive rate, false negative rate, and balanced accuracy, indicating it is likely a better-performing model overall.

Through the comparison between the Convolutional Network, Custom Binary, and the new loss function, we observe that both the Custom Binary and New loss function fail to achieve the loss levels attained by the Convolutional Network. This indicates that the Convolutional Network emerges as the superior performer.

This suggests that the Convolutional Network excels in capturing complex patterns and features in the data, resulting in better classification performance compared to the Custom Binary model. The Convolutional Network's inherent ability to effectively process visual data makes it a preferred choice for tasks requiring sophisticated analysis of images and videos.

**Limitations and Future optimization directions**

Through comparison, I believe that using Convolutional Neural Network (CNN) is superior to my ANN model. Observing the CNN algorithm, it does not directly flatten the image into a one-dimensional vector like a fully connected neural network (ANN). Instead, it gradually extracts high-level abstract features through the construction of convolutional and pooling layers, preserving the hierarchical structure information of the image. This approach better reflects the relationships between pixels, rather than simply capturing global features.

For CNN, the parameter propagation is not a direct forward pass but occurs through the operations of convolutional and pooling layers, layer by layer. This allows the network to better understand the local structure and hierarchical features of the image, rather than applying weights directly to the entire image.

Regarding the limitations of the combined model, I believe it's essential to explore and experiment with various combinations to determine the optimal set of parameters.

Such as **Parameter Tuning:** By systematically varying these parameters and evaluating the model's performance using cross-validation or validation sets, we can identify the combination that yields the highest accuracy and generalization performance.

Future optimization directions include:

Exploring more advanced supervised learning methods, such as Convolutional Neural Networks, to avoid losing important features.

Improving image processing, for example, using interpretability techniques to better understand how the model handles features and reduce information loss.

## Reference List:

[1]

‘Self-supervised learning for medical image classification: a systematic review and implementation guidelines | npj Digital Medicine’. Accessed: Dec. 12, 2023. [Online]. Available: <https://www.nature.com/articles/s41746-023-00811-0>

[2]

I. Itauma, M. S. Aslan, X. W. Chen, and F. Villanustre, ‘Unsupervised Learning and Image Classification in High Performance Computing Cluster’, in *Big Data Technologies and Applications*, B. Furht and F. Villanustre, Eds., Cham: Springer International Publishing, 2016, pp. 387–400. doi: [10.1007/978-3-319-44550-2\_15](https://doi.org/10.1007/978-3-319-44550-2_15).

[3]

‘Application of Compact Convolutional Transformers’. Accessed: Dec. 13, 2023. [Online]. Available: <https://kaggle.com/code/matthewjansen/application-of-compact-convolutional-transformers>

[4]

‘Skin Cancer’. Accessed: Dec. 13, 2023. [Online]. Available: <https://kaggle.com/code/abdallahwagih/skin-cancer>

[5]

‘Starter: Skin Cancer: Malignant vs. Benign’. Accessed: Dec. 13, 2023. [Online]. Available: <https://kaggle.com/code/fanconic/starter-skin-cancer-malignant-vs-benign>