# Machine Learning: Skin Cancer: Benign and malignant

[Machine Learning: Skin Cancer: Benign and malignant 1](#_Toc161239962)

[Brief Introduction: 1](#_Toc161239963)

[Implement and document the ensemble learning model and the training algorithm. 3](#_Toc161239964)

[Process Data 3](#_Toc161239965)

[Split data set 4](#_Toc161239966)

[Flattening the images 4](#_Toc161239967)

[Hyper-parameter Tweaking 5](#_Toc161239968)

[Building an Artificial Neural Network (ANN) Model: 7](#_Toc161239969)

[Cost function Selection： 7](#_Toc161239970)

[Choose different learning rates: 8](#_Toc161239971)

[Select different Dense: 8](#_Toc161239972)

[Select Batch Size: 10](#_Toc161239973)

[Select Number of Hidden Units: 11](#_Toc161239974)

[Select Activation Function: 12](#_Toc161239975)

[Select Optimizer: 13](#_Toc161239976)

[The final choice 13](#_Toc161239977)

[Additional features the explainability/interpretability of the model and proposing new custom loss function 14](#_Toc161239978)

[New custom loss function 14](#_Toc161239979)

[LIME Explanation 15](#_Toc161239980)

[Conclusion 15](#_Toc161239981)

[Explain possible current limitations of solutions. 18](#_Toc161239982)

[Reference List: 19](#_Toc161239983)

## 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, 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, 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.

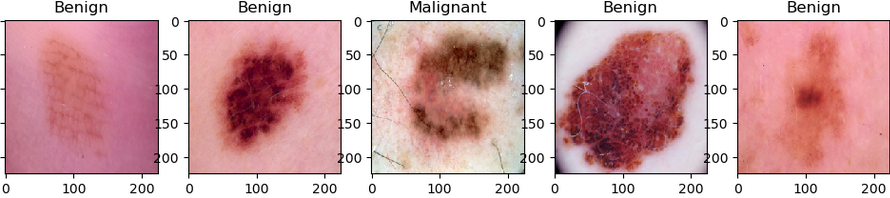
The relevance of this classification problem to the real world is demonstrated by the fact that skin cancer categories can even be determined by predictive tasks, reducing the number of manual annotations required to train medical imaging models, reducing the cost and time required for model development, 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.

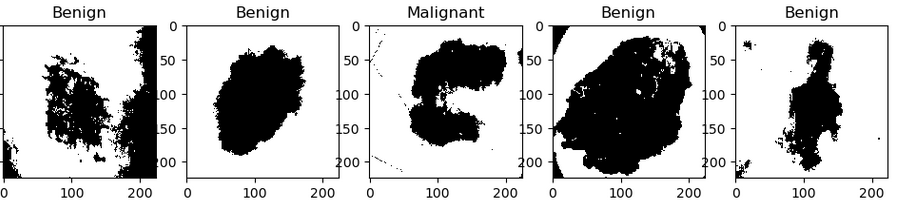
图示

描述已自动生成

### Process Data



(Diagram1: Data before Processing)



(Diagram2: Data after Processing)

### Split data set

图表, 条形图

描述已自动生成

(Diagram3:Number of Training Data and Testing Data)

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

### 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**：

First we choose the same model and then take different loss functions and finally get the optimal loss function

**custom\_loss\_v2**

**Binary Crossentropy**

**sparse\_categorical\_crossentropy**

图表, 折线图

描述已自动生成

(Diagram4: Cost function Selection**)**

According to the selection of different loss functions under the same model, it can be seen that Binary Crossentropy is the most suitable choice

#### Choose different learning rates:

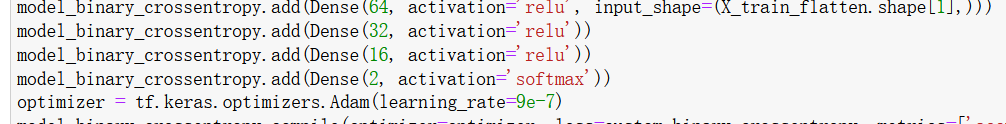
图表, 折线图

描述已自动生成

(Diagram5: Learning Rate of Accuracy)

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 different Dense:



(Diagram6: 2 hidden layers)

文本

描述已自动生成

(Diagram7: 3 hidden layers)

文本

描述已自动生成

(Diagram7: 4 hidden layers)

**Table 2 Layer selection**

|  |  |  |
| --- | --- | --- |
|  | | loss |
| 2 hidden layers | decline | |
| 3 hidden layers | decline | |
| 4 hidden layers | fixed | |

It can be seen from the table that the model is further optimized when 3 hidden layers are selected, but Moses' loss has not moved when 4 hidden layers are selected, which means that it is not advisable to choose 4 hidden layers

**Table 3 Epoch times selection**

|  |  |  |
| --- | --- | --- |
| Epoch times | | loss |
| 36  37 | 0.4284  0.4549 | |
| 38  39 | 0.4146  0.4430 | |
| 40 | 0.4491 | |

After further optimization operations, the loss function decreases continuously. this demonstrates that can continue for additional times. When I get 38 epochs, I observed that the loss function is no longer decrease, meaning is that the model has fully fitted the data. Therefore, I will choose the number of epochs as 38.

#### 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 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

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

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

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

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 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 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.

#### The final choice

**Table4** **The final choice**

|  |  |  |
| --- | --- | --- |
|  | | Selection |
| Cost function | Binary Cross entropy | |
| Learning rate | 1e-06(0.000001) | |
| Activation Function | tanh | |
| Epochs | 38 | |
| Cost Optimizer | Adam | |
| Number of Hidden Units | [128, 64, 32, 16, 8] | |
| Batch Size | 32 | |

## Additional features the explainability/interpretability of the model and proposing new custom loss function

### 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 Crossentropy 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.

### LIME Explanation

图表, 散点图

描述已自动生成窗体顶端

LIME (Local Interpretable Model-agnostic Explanations) is a technique used to explain the predictions of machine learning models. It helps us understand how models make predictions given input data.

In image processing, the application of LIME helps us understand the behavior of machine learning models in tasks such as image classification, object detection, and image segmentation. It reveals the model's perception of different features in the image, aiding in identifying potential biases or misconceptions and further improving the model's performance and interpretability.

## Conclusion

**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

Model Evaluation Metrics of custom\_binary\_crossentropy

图表, 折线图

描述已自动生成

Model Evaluation Metrics of custom\_loss\_v2

图表, 折线图

描述已自动生成

Model Evaluation Metrics of custom\_sparse\_categorical\_crossentropy not working.

custom\_binary\_crossentropy is a loss function used for binary classification tasks. According to the provided description, models trained with custom\_binary\_crossentropy perform better in terms of accuracy and recall. Accuracy measures the proportion of samples that are correctly classified by the model among all samples, while recall measures the proportion of actual positives that are correctly identified by the model among all actual positives. This indicates that models trained with custom\_binary\_crossentropy are better at overall sample classification and are more capable of correctly identifying positives.

On the other hand, custom\_loss\_v2 is a loss function with a regularization penalty term. According to the description, models trained with custom\_loss\_v2 perform better in terms of specificity and ROC-AUC. Specificity measures the proportion of actual negatives that are correctly identified by the model among all actual negatives, while ROC-AUC is the area under the receiver operating characteristic curve, representing the model's performance across different thresholds. This suggests that models trained with custom\_loss\_v2 perform better in correctly identifying negatives and exhibit better overall statistical properties.

It is noteworthy that neither custom\_binary\_crossentropy nor custom\_loss\_v2 mentions sensitivity. This may be because it was not mentioned in the description or not applicable to the specific problem. Sensitivity measures the proportion of actual positives that are correctly identified by the model among all actual positives and is typically synonymous with recall.

In summary, custom\_binary\_crossentropy and custom\_loss\_v2 have different advantages in terms of model performance. The choice of loss function depends on the specific requirements and optimization goals of the problem.

## Explain possible current limitations of solutions.

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:

Considering more suitable supervised learning methods for binary problems, such as logistic regression.

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>