Utilizing Deep Learning for Domain Knowledge Classification of Heart Disease

Kevin Sean Hans Lopulalan[1], Mochamad Fadlan Adhari[2]

Mathematics Department, Faculty of Mathematics and Natural Sciences, Institut Teknologi Bandung

e-mail: kevinsean.lopulalan@gmail.com[1], fadlanadhari20@gmail.com [2]

Abstract

Heart disorders are a common occurrence among individuals due to various factors. To diagnose heart disease, an assessment of the symptoms manifesting in the body is crucial. Accurate prediction of these symptoms can be achieved by processing the detected symptoms. One approach involves utilizing historical data of heart disease patients to predict these symptoms using Deep Learning (DL). DL is a subfield of artificial intelligence inspired by the human brain. It enables pattern recognition and prediction using historical data. This article delves into a comparative analysis of DL models for patient data classification, specifically focusing on MLP and CNN models. For MLP, three different packages are employed to construct the model, while for CNN, two feature dimension reduction techniques are utilized. The variations considered include layers, epochs, features, and learning rate. Through simulations, it is concluded that MLP outperforms CNN due to its comprehensive utilization of data, preventing information loss. Nevertheless, all the developed models exhibit remarkable accuracy, exceeding 70%.

Keywords: Deep Learning, MLP, CNN, Classification

1. Introduction

Deep Learning is a method within the field of artificial intelligence, inspired by the structure and function of the human brain. Deep Learning models are capable of recognizing complex patterns across various types of data, such as images, text, audio, and other forms of data, allowing for highly accurate predictions. One of the main advantages of Deep Learning is its ability to perform classification based on the data provided.

In recent literature, a growing number of methods have emerged as applications of Deep Learning, including *Multi-Layer Perceptron (MLP), Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM).* Each of these models serves distinct purposes. For classification tasks in particular, MLP and CNN are two models that are highly suitable and widely used.

There are numerous ways to utilize the architectures of CNN and MLP, especially in the healthcare sector. In this field, these models can be employed to classify disease-related issues in patients. This is feasible due to the availability of extensive historical patient data, which can be used to predict or classify diseases in patients with similar conditions. As such, these models are particularly valuable for conducting preliminary diagnoses before further medical action is taken, especially in the context of *heart disease*.

The heart is a vital organ for human life. Its primary function is to pump blood throughout the body and to receive it back after it has been oxygenated by the lungs. Despite its critical role, many individuals suffer from heart abnormalities, often caused by unhealthy lifestyle habits. Various symptoms may begin to appear as signs of heart-related issues. To make a diagnosis, physicians typically rely on patient data such as blood sugar levels, cholesterol, age, gender, and other relevant factors. This information is essential in identifying the presence of heart disease. Therefore, a model that can process patient data to predict heart disease is needed.

This article aims to classify heart disease based on a given dataset. The dataset contains information about patient conditions along with labels indicating whether the patient has a

normal or abnormal heart condition. The classification will be conducted using CNN and MLP models, and the results of both approaches will be compared.

2. Methodology

Untuk keperluan analisis, digunakan data pasien penyakit jantung yang diambil dari For the purpose of analysis, heart disease patient data was obtained from https://kaggle.com, comprising approximately 650 patient records. The dataset includes the following features:

- Age
- Gender (Male or Female)
- Chest Pain Type (with four possible values: suspected Typical Angina, Atypical Angina, Non-anginal Chest Pain, or Asymptomatic)
- Resting Blood Pressure
- Serum Cholesterol Level (in mg/dL)
- Fasting Blood Sugar
- Resting Electrocardiographic Results
    - Note: 0 = normal, 1 = ST-T wave abnormality, 2 = left ventricular hypertrophy or other abnormalities
- Maximum Heart Rate Achieved
- Exercise-Induced Angina
- Oldpeak: ST depression induced by exercise relative to rest
- Slope of the Peak Exercise ST Segment
- Number of Major Vessels Colored by Fluoroscopy (ranging from 0 to 3)
- Thallium Stress Test Result
    - Note: 0 = normal, 1 = fixed defect, 2 = reversible defect
- Target Variable
    - Note: 0 = no heart disease, 1 = presence of heart disease

Based on this dataset, classification models will be developed using Deep Learning methods. The models used in this study include:

- MLP implemented with NumPy
- MLP implemented with TensorFlow

- CNN with dimensionality reduction using PCA (Principal Component Analysis)
- CNN with dimensionality reduction using feature selection techniques

The architectural details of each model are outlined in the following section.

2.1 *Multi Layer Perceptron* (MLP)

Multi-Layer Perceptron (MLP) is one of the models or architectures in Deep Learning and represents a modern construction of the *Feedforward Neural Network* (FNN). This model consists of fully connected neurons equipped with nonlinear activation functions.

In this study, three types of MLP implementations will be developed using NumPy, TensorFlow, and Scikit-learn (Sklearn) in Python.

All three models share the same basic architecture, which includes an *input layer*, one or more *hidden layers*, and an *output layer*. The detailed architecture of each implementation will be explained in the subsequent sections.

2.2 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) is a Deep Learning model commonly used for *image recognition*, *image classification*, and other tasks involving *spatial data*. In this study, the dataset consists of numerical data with 13 different variables.

To adapt this data for use in a CNN, it must be transformed into an image-like format (i.e., pixels) and reshaped into a square matrix. Therefore, the number of features will be reduced to the nearest *perfect square* (e.g., 4 or 9).

Two dimensionality reduction techniques will be applied for this purpose, namely:

2.2.1. *Principal Component Analysis* (PCA)

Principal Component Analysis (PCA) is a dimensionality reduction technique that preserves as much information as possible from the original data. The following are the steps involved in applying PCA:

1. Standardize the data: Subtract the mean from each feature and scale to unit

variance.

2. Compute the covariance matrix: The covariance matrix summarizes the pairwise relationships between features, calculated from the standardized data.
3. Calculate eigenvalues and eigenvectors: The eigenvalues and eigenvectors of the covariance matrix represent the magnitude and direction of the principal components. These are typically computed using *Singular Value Decomposition* (SVD).
4. Select the principal components: Sort the eigenvalues in descending order and select the top *k* eigenvectors corresponding to the *k* largest eigenvalues.
5. Project the data onto the principal components: Transform the original data by projecting it onto the subspace formed by the selected principal components.

This projected data retains the essential information from the original dataset and is therefore well-suited for dimensionality reduction.

2.2.2. Metode Seleksi (fitur *scikit-learn*)

Scikit-learn provides a feature selection tool in Python called *'SelectKBest'*. Similar to PCA, this feature is used to perform dimensionality reduction on a given dataset. However, unlike PCA, this method does **not** retain information from the features that are excluded. In other words, data columns deemed to have low relevance are completely discarded. As a result, this technique may lead to some loss of information from the original dataset.

Scikit-learn offers several built-in scoring functions for feature selection, including:

Terdapat beberapa fungsi skor bawaan, yaitu :

1. `f_regression – Used for linear regression problems and calculates the F-value
2. `mutual_info_regression` – Used for regression tasks and measures the mutual information between two random variables.
3. `f_classif`– Used for classification problems and computes the ANOVA F-value.
4. `mutual_info_classif`– Used for classification tasks and measures the mutual information between two variables.
5. `chi2` – Used for classification problems and computes the chi-squared statistic.
6. SelectPercentile – Used to select a specific percentile of features based on a

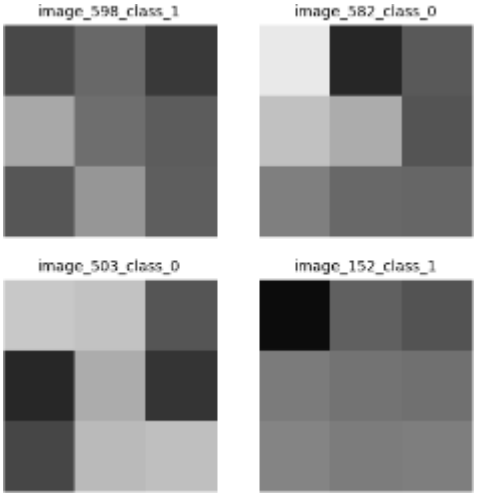
scoring function.

In this study, since the objective is classification and the aim is to reduce non-correlated variables, the mutual_info_classif scoring function is used.

After transforming the original data into a set of variables whose count is a perfect square, the data is ready for classification using a Convolutional Neural Network (CNN). The CNN modeling process consists of several stages: transforming the data into image form, feeding it into the CNN architecture, and performing evaluation.

From the selected variables, each row of data is converted into an image. In this process, both the selected variables and their order can influence prediction results, as they directly affect the structure and content of the resulting images. Each generated image is then assigned a corresponding label based on the dataset's target values.

These labeled images are then fed into the CNN architecture that will be constructed. Below is an example of the generated image.



Once the images have been prepared, a CNN architecture is constructed to process them. The CNN model used in this study is based on the architecture proposed in the paper *"Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach."* The architecture consists of **nine layers**, including:

- Input Layer: ($m \times m$)
- Two Convolutional Layers: ($m \times m \times 32$) and ($m \times m \times 64$)

- Max Pooling Layer: (7 × 7 × 64)
- Two Dropout Layers: with dropout rates of 0.25 and 0.50
- Fully Connected Layer: 128 neurons
- Output Layer: 2 neurons (for binary classification)

The **Convolutional Layers** apply convolution operations, which are used to extract spatial features from the input image. A simplified explanation of the convolution operation is as follows:

$$s(t) = (x \cdot w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

Since the operation is performed on two-dimensional images, the convolution operation is extended to:

$$s(i,j) = (I \cdot K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

Max Pooling, which is applied after the Convolutional Layers, aims to build the structure of the Deep Neural Network. Afterward, the *Fully Connected Layers* are used to calculate the weights that need to be learned. Finally, to generate the output, the *softmax function* is used, which is suitable for the CNN architecture.

This CNN architecture is implemented using Keras and TensorFlow, with varying epochs to achieve accurate predictions.

Once the model has undergone training and testing, the results of the CNN architecture are evaluated. Evaluation is performed based on computational metrics, which can be measured using **accuracy** and **F1 score** for each target label (0 and 1).

3. Result

Based on the constructed framework, several simulations are conducted using different models. Below is a snippet of the data used.

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

A total of 1025 data points are used in this simulation, with the columns from 'age' to 'thal' serving as features, and the 'target' column being used as the value to be predicted. The data is then applied to the models that have been built. The results from the different models will be compared. Once the models are formed, a test will be conducted using a single data point, as shown below.

```
age          50.0
sex           0.0
cp            1.0
trestbps    120.0
chol        244.0
fbs           0.0
restecg       1.0
thalach     162.0
exang         0.0
oldpeak       1.1
slope         2.0
ca            0.0
thal          2.0
target        1.0
```

3.1 Simulation Using MLP NumPy

In this simulation, the data is processed using an MLP with the NumPy package. Several simulations are conducted by varying the hyperparameters to compare the results.

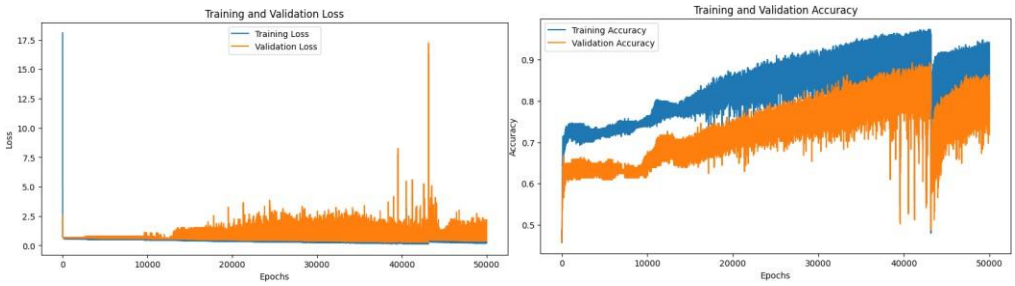
- First Simulation :

  The optimization algorithm used is *Adam* with a learning rate of 0.01, the loss function is binary cross-entropy, the random state is set to 999, and the architecture configuration is as follows:

Layer	**Neuron**	**Activation Function**
Input Layer	13	Relu
Hidden Layer 1	128	Relu

Hidden Layer 2	64	Sigmoid
Output Layer	1	-

The model achieved a training accuracy of 0.8975, a validation accuracy of 0.8292, and a test accuracy of 0.7902 after 50,000 epochs. The learning process is as follows:

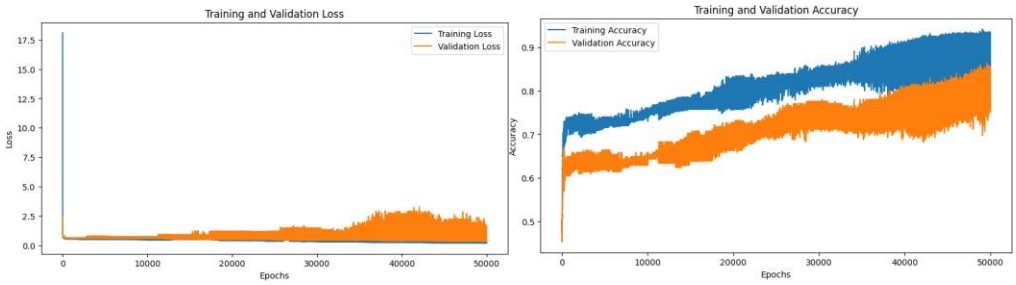


- Second Simulation :

  The optimization algorithm used is *Adam* with a learning rate of 0.01, the loss function is binary cross-entropy, the random state is set to 999, and the architecture configuration is as follows:

Layer	**Neuron**	**Activation Function**
Input Layer	13	Relu
Hidden Layer 1	64	Relu
Hidden Layer 2	32	Sigmoid
Output Layer	1	-

The model achieved a training accuracy of 0.9170, a validation accuracy of 0.8243, and a test accuracy of 0.7804 after 50,000 epochs. The learning process is as follows:



3.2 Simulation Using MLP TensorFlow

In this simulation, the data is processed using an *MLP* with the *TensorFlow* package.

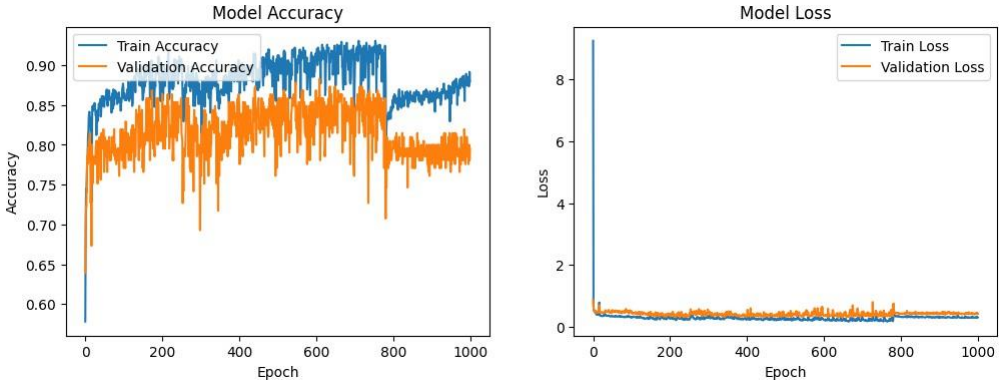
Several simulations are conducted by varying the hyperparameters and the layers to compare the results.

- First Simulation :

  The optimization algorithm used is *Adam* with a learning rate of 0.01, the loss function is binary cross-entropy, the random state is set to 999, and the architecture configuration is as follows:

Layer	**Neuron**	**Activation Function**
Input Layer	13	Relu
Hidden Layer 1	128	Relu
Hidden Layer 2	64	Sigmoid
Output Layer	1	-

  The model achieved an accuracy of 0.8818 and a validation accuracy of 0.7951 after 1000 epochs. The learning process is as follows:



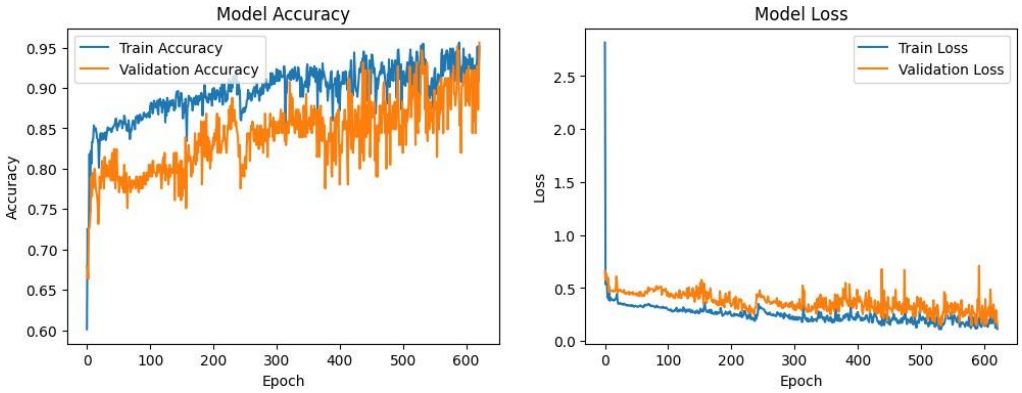
- Second Simulation :

  The optimization algorithm used is *Adam* with a learning rate of 0.01, the loss function is binary cross-entropy, the random state is set to 999, and the architecture configuration is as follows:

Layer	**Neuron**	**Activation Function**
Input Layer	13	Relu
Hidden Layer 1	64	Relu
Hidden Layer 2	32	Sigmoid

Output Layer	1	-

The model achieved an accuracy of 0.9445 and a validation accuracy of 0.9561 after 622 epochs. The learning process is as follows:



3.3 Simulation Using MLP Sklearn

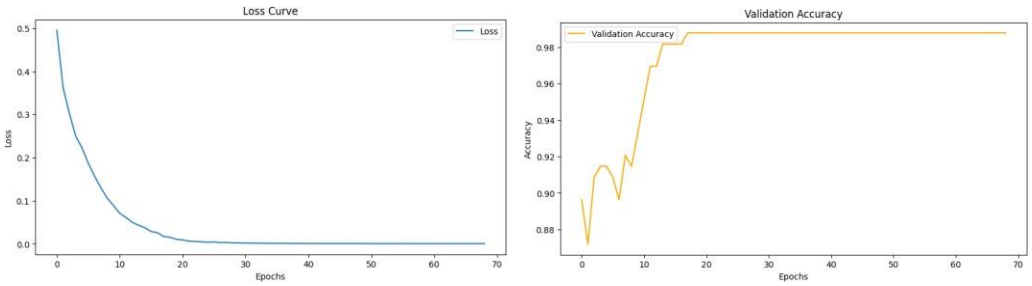
In this simulation, the data is processed using an *MLP* with the *sklearn* package. Several simulations are conducted by varying the hyperparameters and the layers to compare the results.

- First Simulation :

  The optimization algorithm used is *Adam* with a learning rate of 0.01, the loss function is cross-entropy, the random state is set to 999, and the architecture configuration is as follows:

Layer	**Neuron**	**Activation Function**
Input Layer	13	Relu
Hidden Layer 1	128	Relu
Hidden Layer 2	64	Sigmoid
Output Layer	1	-

The model achieved a training accuracy of 0.9878 and a validation accuracy of 0.9951 after 69 epochs. The learning process is as follows:

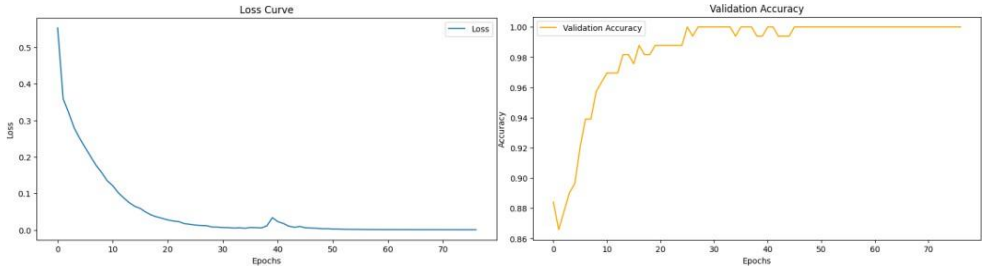


- Second Simulation :

  The optimization algorithm used is *Adam* with a learning rate of 0.01, the loss function is cross-entropy, the random state is set to 999, and the architecture configuration is as follows:

Layer	**Neuron**	**Activation Function**
Input Layer	13	Relu
Hidden Layer 1	64	Relu
Hidden Layer 2	32	Sigmoid
Output Layer	1	-

The model achieved a training accuracy of 1.0 and a validation accuracy of 1.0 after 77 epochs. The learning process is as follows:



3.4 Simulasi Using CNN with PCA Reduction Technique

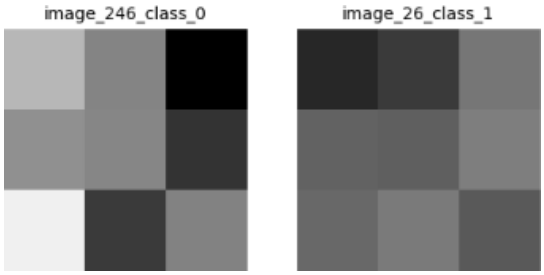
In this simulation, a CNN model with PCA data reduction technique is used. The dimensions of the data will be transformed into a perfect square to be converted into square-shaped images. Below are the results from the simulations that have been conducted.

- First Simulation : Learning Rate 0.001

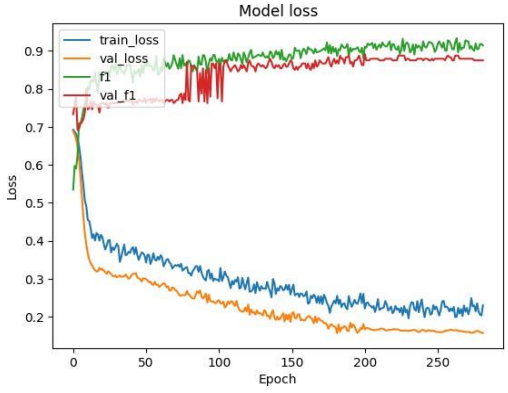
From the 13 features available, PCA is performed to reduce the data to 9 dimensions. After this step, the following results were obtained:

	0	1	2	3	4	5	6	7	8	target
0	-0.522556	-1.112803	0.956816	-1.149198	-0.559252	-1.505052	0.071292	0.049732	0.872570	0
1	2.590381	-0.533162	1.467315	1.536614	1.345335	1.524630	1.469460	0.594801	-0.127561	0
2	3.042352	-1.327521	-0.424765	1.567204	0.283814	-0.738182	0.378211	-1.397097	-0.836844	0
3	-0.492522	-0.276720	0.801442	-0.984277	-0.487587	-1.438634	0.385833	-1.566671	0.085219	0
4	2.187464	1.951477	-0.385539	0.295793	-2.386144	-0.563839	1.022689	1.682067	0.451377	0
...	...	...	...	...	...	...	...	...	...	...
1020	-0.762321	-0.512273	0.046672	-0.308011	-0.415454	0.083938	0.957084	-1.517729	-0.519319	1
1021	2.374273	-0.940859	0.182370	-0.628459	0.822406	0.373266	-0.493855	0.086828	-0.050143	0
1022	1.245073	-1.457356	-0.473873	-0.645240	-0.271197	1.786723	-0.623980	0.614532	-0.084198	0
1023	-1.620053	0.124443	-1.327956	-1.196804	-0.224913	1.263473	-0.498017	0.265437	0.345972	1
1024	0.934169	-1.778549	-0.005882	0.353372	-0.743381	-1.034172	-0.484996	-0.130881	-0.428642	0

These 9 dimensions are considered to represent the entire original data. After that, a split of 80% for training and 20% for testing is applied. For each data row, these 9 features are transformed into the following image:



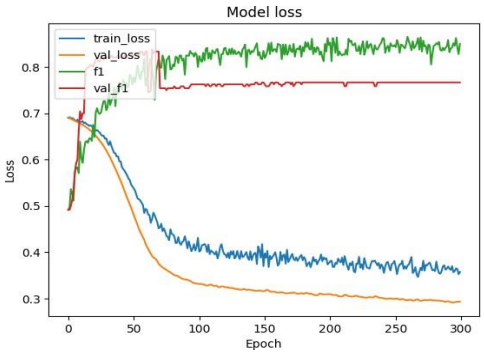
Thus, the image class with target 0 is on the left, and the image class with target 1 is on the right. These images are then processed through the CNN model that has been built, with 300 epochs used. The results obtained are as follows:



It was found that in each epoch iteration, the loss function decreased rapidly, and the accuracy increased. In general, an evaluation was performed, and it was obtained that the precision score for class 0 was 0.96, and the precision score for class 1 was 0.9. Thus, the average precision of this model is 0.93.

- Second Simulation : Learning Rate 0.0001

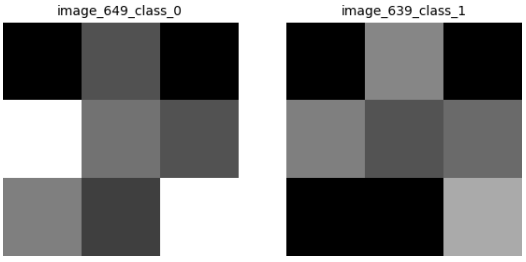
  For a learning rate of 0.0001, or reduced from the previous value, the following results were obtained:



With an evaluation, the precision for class 0 was 0.74 and for class 1 was 0.8. Thus, the precision of this model is 0.77.

3.5 Simulation Using CNN with Selection Reduction Technique

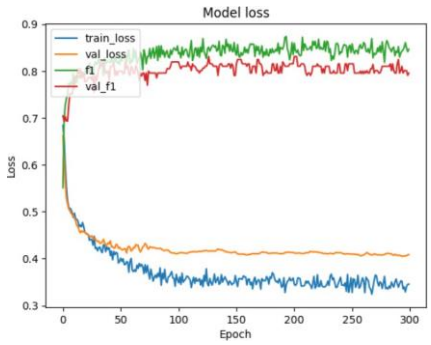
In this simulation, the data dimensions are reduced to a perfect square using the selection method. This selection method ensures that the original data remains intact, but only the variables deemed important are selected. From the 13 available features, 9 features are selected, allowing the data to be transformed into a 3 × 3 image, which will be processed by the CNN. The resulting image is as follows:



After that, the image is processed using the CNN model that has been built. Two simulations were conducted with different numbers of epochs:

- First Simulation : Epochs 300

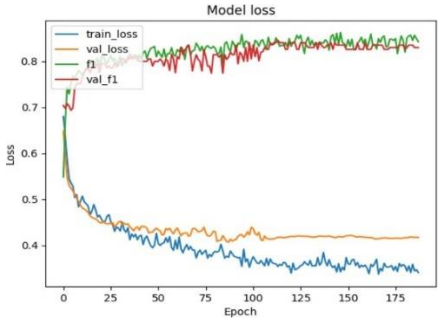
When 300 epochs were used, the following results were obtained.



It was found that the precision for class 0 was 0.84 and the precision for class 1 was 0.78. As a result, the overall precision of the model was 0.81.

- Second Simulation : Epochs 500

  When 500 epochs were used, the following results were obtained.



It can be observed that the epoch stopped at 188 because the model was no longer able to improve its performance. The precision for class 0 was 0.84, and the precision for class 1 was 0.78. As a result, the overall precision of the model was 0.81.

4. Discussion

From the four simulation results above, the following summary was obtained:

Aspect	MLP NP		MLP TF		MLP SK		CNN PCA		CNN SL	
	1	2	1	2	1	2	1	2	1	2
Epochs	50.000	50.000	1000	622	69	77	300	300	300	500
LR	0.01	0.01	0.01	0.01	0.01	0.001	0.0001	0.001	0.001	0.01
Train Acc	0.8975	0.9170	0.8818	0.9445	0.9878	1	0.93	0.77	0.81	0.81

It can be seen that the accuracy of all the models built is quite good, with accuracy values above 50%. The model with the highest accuracy is the MLP model using TensorFlow or Sklearn. This means that these models are sufficiently reliable for classifying new data. By comparing the architectures of the same model, we can observe how each library performs in the MLP algorithm. For example, with TensorFlow, we obtain a performance of around 80% after hundreds of epochs. Next, for SK Learn, we achieve an accuracy of over 90% with around 60-80 epochs. For the manually implemented numpy model, we obtain results up to 0.9 with epochs reaching tens of thousands. This indicates that the convergence of the numpy model is weaker compared to TensorFlow and SK Learn. Additionally, learning with numpy is much more fluctuating than with SK Learn and TensorFlow.

The prediction results made by each model are as follows:

Aspect	MLP NP		MLP TF		MLP SK		CNN PCA		CNN SL	
	1	2	1	2	1	2	1	2	1	2
Target	1	1	1	1	1	1	1	1	1	1

It can be seen that all models made accurate predictions in line with the data. Based on the results obtained, an analysis and comparison of each model were conducted.

For the MLP model, although it was observed during the training phase that the numpy library has weaker convergence compared to the SK Learn and TensorFlow libraries, all MLP models that were tested were generally able to capture the patterns present in the data. This is demonstrated by the prediction experiments, which show that the predictions yielded a value of 1.

On the other hand, two CNN models were built. From the accuracy results, it can be observed that the highest accuracy was achieved by the CNN model using the PCA reduction technique. This indicates that PCA was able to summarize all the original data into a reduced dimension form without losing important information from the data. The CNN model using the selection technique performed quite well but with lower accuracy, which could be due to the fact that the selected data did not capture all the information. A lot of data or information was left out, causing the model to struggle with fitting.

For both CNN and MLP models, it can be concluded that the MLP model has higher accuracy compared to the CNN model. This is because the MLP model uses the entire dataset. As a result, the data is used in its entirety (original), and no information is lost. The CNN architecture requires data with a number of features that is a perfect square, which leads to the higher accuracy of the MLP model compared to the CNN model.

5. Conclusion

The analysis of the various models—MLP, CNN with PCA, and CNN with selection techniques—reveals some key insights into their performance and suitability for predicting heart disease based on patient data.

1. **MLP Models**: The MLP models, regardless of the library used (NumPy, TensorFlow, or Sklearn), demonstrated strong predictive capabilities. The models with TensorFlow and Sklearn showed higher accuracy, which is likely due to their efficient training and better convergence properties. Although NumPy-based models had slower convergence and more fluctuations during training, they still managed to capture the patterns in the data, achieving relatively good performance.
2. **CNN Models**: The CNN models, especially the one utilizing PCA for dimensionality reduction, showed promising results with the highest accuracy. PCA effectively condensed the data into a lower-dimensional space without losing critical information, enhancing the model's performance. The CNN model using the selection technique, while still performing reasonably well, had slightly lower accuracy. This is likely because the selection method excluded certain features, which could have impacted the model's ability to fully understand the data.
3. **Comparing MLP and CNN**: Overall, MLP models outperformed CNN models in terms of accuracy. This can be attributed to the fact that MLPs utilize the entire set of data features, without any reduction or exclusion, which ensures that all the available information is leveraged. In contrast, the CNN architecture, while effective in some scenarios, is constrained by the need for data that fits specific dimensional requirements (e.g., perfect square features for image-like inputs). This constraint made CNNs less effective than MLPs in this case.
4. **Practical Implications**: For future applications, if the goal is to achieve higher

accuracy, MLP models should be prioritized. They are better suited to handle the full range of data features without requiring dimensional reduction or selection. However, CNN models with PCA reduction could be useful in cases where the data size is large, and dimensionality needs to be controlled to improve training efficiency. Nonetheless, any CNN model would need careful tuning and may not always outperform well-optimized MLP models for classification tasks like heart disease prediction.

In conclusion, while both MLP and CNN models have their strengths, MLP models are more reliable for this specific dataset and task. Further optimization, especially of the CNN models, could still lead to improved performance, but MLP remains the preferred choice for high accuracy in heart disease prediction.

6. Reference

Sezer, O. B., & Ozbayoglu, A. M. (2018). Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. Applied Soft Computing, 70, 525-538.

7. Contribution

The following are the contributions made by each group member:

- Mochamad Fadlan Adhari: Collected data, created CNN syntax, prepared the report
- Kevin Sean Hans Lopulalan: Created MLP syntax using NumPy, TensorFlow, and Scikit-learn