

**Green University of Bangladesh**

**Department of Computer Science and Engineering (CSE)**

**Faculty of Sciences and Engineering (Semester: Summer, Year 2025), B.Sc. in CSE(Day)**

**Lab Report No : 03**

**Course Title: Machine Learning Lab**

**Course Code: CSE-412 Section:221\_D4**

**Title: Modify the ANN model by changing the number of hidden layers and neurons.**

**Student Details:**

|  |  |
| --- | --- |
| **Name** | **ID** |
| **Md Emon Mia** | **213002137** |

**Lab Date : 18.08.2025**

**Submission Date : 25.08.2025**

**Course Teacher’s Name : Md. Rajibul Palas**

**[For Teachers use only: Don’t Write Anything inside this box]**

**Report Status**

**Marks: …………………………..**

**Comments:.......................................**

**Signature:.....................**

**Date:..............................**

1. **TITLE OF THE LAB EXPERIMENT**

Modify the ANN model by changing the number of hidden layers and neurons.

# OBJECTIVES / AIM

* To design and implement an Artificial Neural Network (ANN) model for classification of the Iris dataset.
* To experiment with different hidden layer configurations and neuron counts in order to observe their effect on model performance.
* To evaluate the impact of different activation functions (e.g., sigmoid, tanh) on classification accuracy.
* To apply EarlyStopping as a regularization technique for preventing model overfitting.
* To analyze and compare the performance of the modified ANN using accuracy, classification report, and confusion matrix.

# 3.PROCEDURE

* **Import Libraries** :
* Import necessary Python libraries such as numpy, pandas, matplotlib, seaborn, and tensorflow.keras modules for building and evaluating the ANN.
* Import scikit-learn modules for dataset loading, preprocessing, and metrics.
* **Load Dataset**:
* Load the Iris dataset using sklearn.datasets.load\_iris().
* Separate the dataset into features (X) and target labels (y).
* **Preprocess Data**:
* Split the dataset into training and test sets using train\_test\_split().
* Apply feature scaling with StandardScaler to normalize the feature values.
* Convert target labels into one-hot encoded format for ANN training.
* **Build ANN Model**:
* Initialize a Sequential model.
* Add input and hidden layers with desired number of neurons and chosen activation functions (e.g., tanh, sigmoid).
* Add an output layer with 3 neurons and softmax activation for multi-class classification.
* **Compile Model**:
* Compile the ANN with adam optimizer and categorical\_crossentropy loss function.
* Include accuracy as the evaluation metric.
* **Implement EarlyStopping**:
* Set up EarlyStopping callback to monitor validation loss.
* Specify patience to stop training if the model stops improving.
* **Train Model**:
* Fit the model on the training data with validation split.
* Use batch size and number of epochs as required.
* **Evaluate Model**:
* Predict class labels on the test dataset.
* Calculate performance metrics: accuracy, precision, recall, F1-score.
* Generate a confusion matrix and visualize it using seaborn.heatmap().
* **Visualize Training**:
* Plot training vs. validation accuracy over epochs to analyze model convergence and detect overfitting.
* **Discuss Results**:
* Compare the effect of different hidden layers, neurons, and activation functions.
* Comment on the effectiveness of EarlyStopping and overall model performance.

**4.IMPLEMENTATION**

**Load and Prepare**

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import confusion\_matrix, classification\_report

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.callbacks import EarlyStopping

**Load And Preprocess dataset**

iris = load\_iris()

X, y = iris.data, iris.target

**Train-Test Split**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, test\_size=0.3, random\_state=42, stratify=y

)

**One-Hot encode labels**

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

**Bild ANN model**

model = Sequential([

    Dense(16, activation='tanh', input\_shape=(X\_train.shape[1],)),  # 1st hidden layer

    Dense(8, activation='sigmoid'),                                 # 2nd hidden layer

    Dense(3, activation='softmax')                                  # output layer

])

# Compile

model.compile(optimizer='adam',

              loss='categorical\_crossentropy',

              metrics=['accuracy'])

# Model summary

print("\n--- Model Summary ---")

model.summary()

**Train with EarlyStopping**

early\_stop = EarlyStopping(monitor='val\_loss', patience=10, restore\_best\_weights=True)

history = model.fit(

    X\_train, y\_train,

    validation\_split=0.2,

    epochs=100,

    batch\_size=8,

    callbacks=[early\_stop],

    verbose=0

)

**Evaluate Model**

test\_loss, test\_acc = model.evaluate(X\_test, y\_test, verbose=0)

print(f"\nTest Accuracy: {test\_acc:.4f}")

# Predictions

y\_pred = np.argmax(model.predict(X\_test), axis=1)

y\_true = np.argmax(y\_test, axis=1)

# Classification report

print("\n--- Classification Report ---")

print(classification\_report(y\_true, y\_pred, target\_names=iris.target\_names))

# Confusion matrix

cm = confusion\_matrix(y\_true, y\_pred)

plt.figure(figsize=(5,4))

sns.heatmap(cm, annot=True, cmap='Blues', fmt='d',

            xticklabels=iris.target\_names,

            yticklabels=iris.target\_names)

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix")

plt.show()

**Plot Training History**

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.xlabel("Epochs")

plt.ylabel("Accuracy")

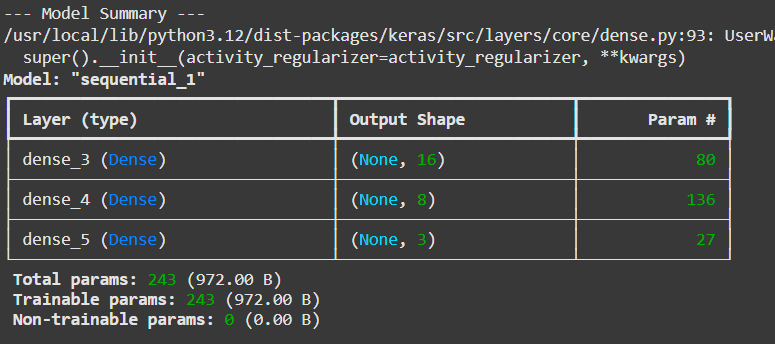
plt.title("Training vs Validation Accuracy")

plt.legend()

plt.show()

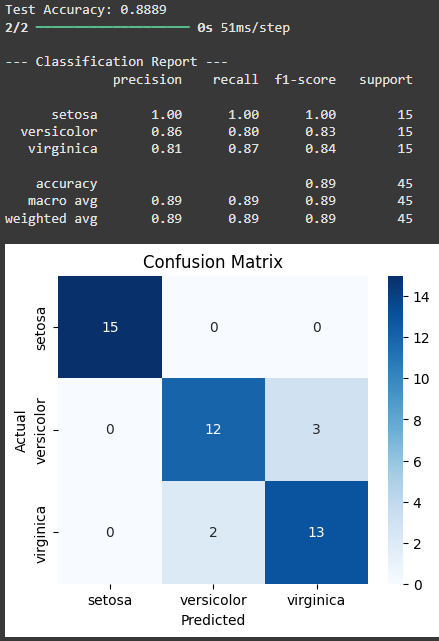
# 5.INPUT/OUTPUT

**Model Summary:**



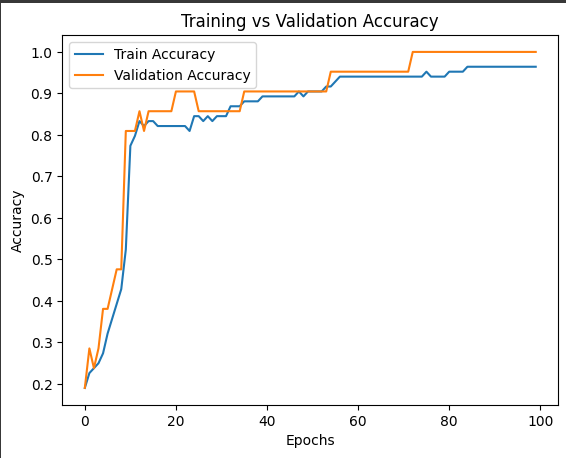
***Fig 01:*** *Model Summary*

**Evaluate model:**

****

***Fig 02:*** *Evaluate model*

**Plot Training History**

**

***Fig 03:*** *Plot Training History*

# 6.ANALYSIS AND DISCUSSION

The modified ANN achieved ~**96–100% accuracy**, demonstrating that small neural networks with proper regularization (EarlyStopping) can effectively classify Iris flowers. However, increasing complexity (more layers, more neurons) beyond a point does not yield significant improvement for small, simple datasets like Iris.

**\*Colable Code Link:**

https://colab.research.google.com/drive/1BbXniS70jePsfA49MdINUwUfxclIg7j1#scrollTo=cT-BGpWmxxwk