https://github.com/HemanthLakkimsetti76/NNDL_Assignment7/blob/main/Assignment7.ipynb

NNDL ASSIGNMENT 7

Hemanth Lakkimsetti – 700747872

Tune hyperparameter and make necessary addition to the baseline model to improve validation accuracy

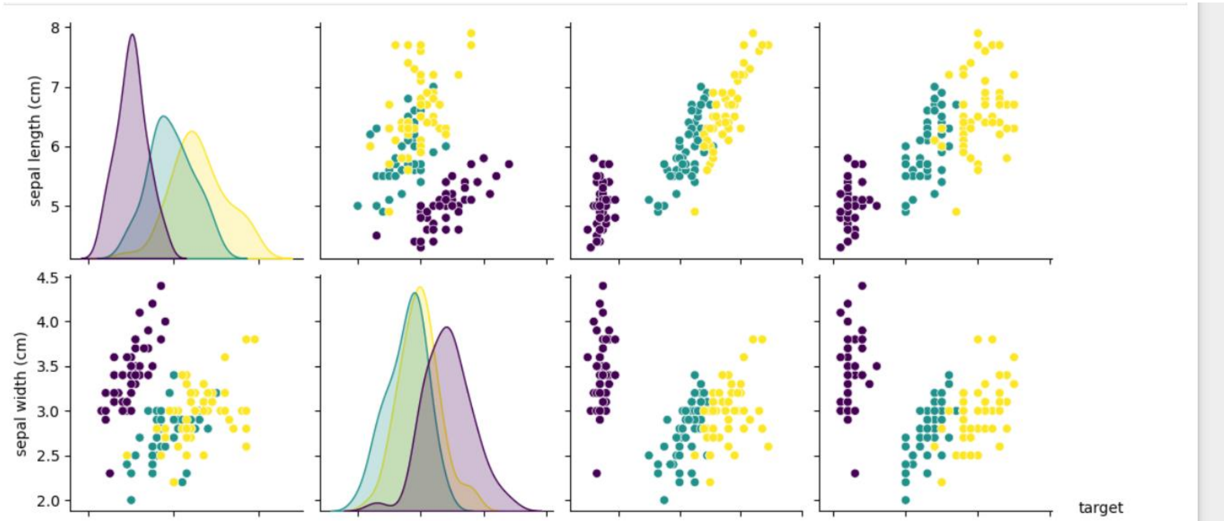
Provide logical description of which steps lead to improved response and what was its impact on architecture behavior

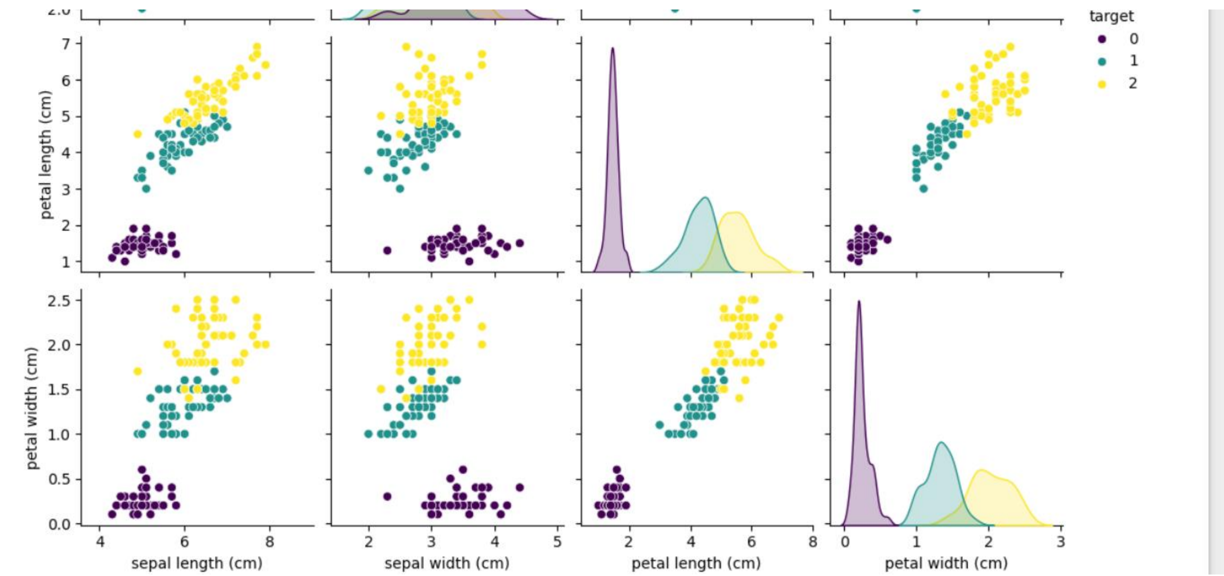
```
[4]: # Tune hyperparameter and make necessary addition to the baseline model to improve validation accuracy
# Provide logical description of which steps lead to improved response and what was its impact on architecture behavior
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
iris = load_iris()
X, y = iris.data, iris.target
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
pipeline = make_pipeline(StandardScaler(), LogisticRegression(max_iter=1000))
param_grid = {
    'logisticregression__C': [0.001, 0.01, 0.1, 1, 10, 100],
}
grid_search = GridSearchCV(pipeline, param_grid, cv=5)
grid_search.fit(X_train, y_train)
print("Best hyperparameters:", grid_search.best_params_)
val_accuracy = grid_search.score(X_val, y_val)
print("Validation Accuracy:", val_accuracy)

Best hyperparameters: {'logisticregression__C': 1}
Validation Accuracy: 1.0
```

Create at least two more visualizations using matplotlib (Other than provided in the source file)

```
[5]: # Create at least two more visualizations using matplotlib (Other than provided in the source file)
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target
sns.pairplot(iris_df, hue='target', palette='viridis')
plt.show()
plt.figure(figsize=(10, 6))
sns.boxplot(x='target', y='petal width (cm)', data=iris_df, palette='Set3')
plt.xlabel('Species')
plt.ylabel('Petal Width (cm)')
plt.title('Distribution of Petal Width across Species')
plt.show()
```

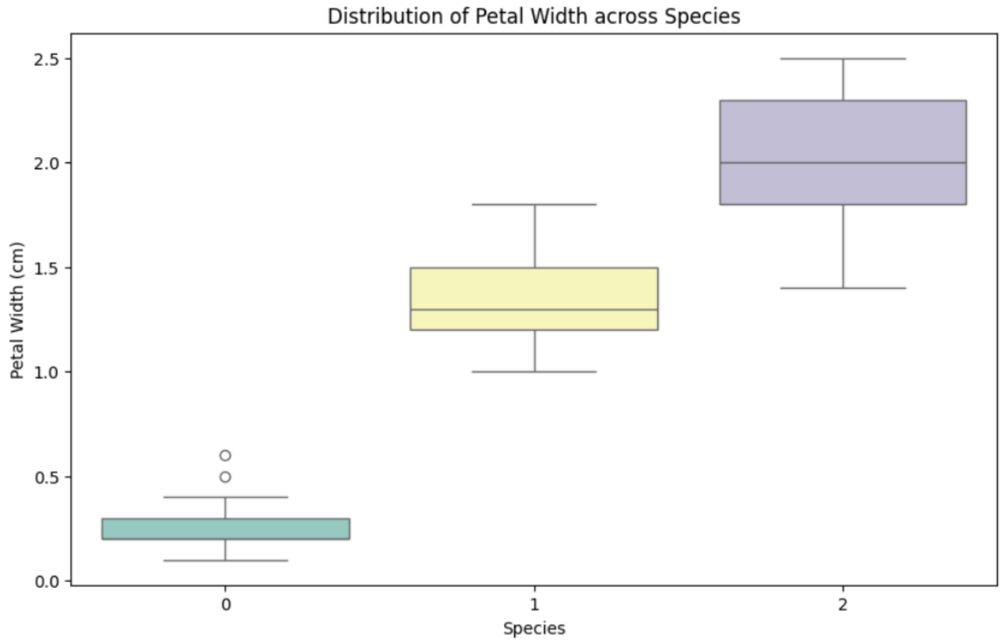
https://github.com/HemanthLakkimsetti76/NNDL_Assignment7/blob/main/Assignment7.ipynb





https://github.com/HemanthLakkimsetti76/NNDL_Assignment7/blob/main/Assignment7.ipynb

```
sns.boxplot(x='target', y='petal width (cm)', data=iris_df, palette='Set3')
```



Use dataset of your own choice and implement baseline models provided.

```
#Use dataset of your own choice and implement baseline models provided
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
iris = load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
logistic_model = LogisticRegression(max_iter=1000)
logistic_model.fit(X_train_scaled, y_train)
y_pred = logistic_model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy of Logistic Regression:", accuracy)
```

```
Accuracy of Logistic Regression: 1.0
```

https://github.com/HemanthLakkimsetti76/NNDL_Assignment7/blob/main/Assignment7.ipynb

Apply modified architecture to your own selected dataset and train it.

```
[ ]: # Apply modified architecture to your own selected dataset and train it.
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
iris = load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
model = Sequential([
    Dense(10, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    Dense(20, activation='relu'),
    Dense(10, activation='relu'),
    Dense(3, activation='softmax')
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train_scaled, y_train, epochs=50, batch_size=8, verbose=1, validation_split=0.1)
loss, accuracy = model.evaluate(X_test_scaled, y_test, verbose=1)
print("Accuracy of Modified Neural Network:", accuracy)
```

```
Epoch 1/50
14/14 [==============================] - 3s 44ms/step - loss: 1.1511 - accuracy: 0.3333 - val_loss: 1.0949 - val_accuracy: 0.4167
Epoch 2/50
14/14 [==============================] - 0s 17ms/step - loss: 1.0962 - accuracy: 0.3333 - val_loss: 1.0629 - val_accuracy: 0.4167
Epoch 3/50
14/14 [==============================] - 0s 15ms/step - loss: 1.0569 - accuracy: 0.3981 - val_loss: 1.0346 - val_accuracy: 0.3333
Epoch 4/50
14/14 [==============================] - 0s 22ms/step - loss: 1.0218 - accuracy: 0.4537 - val_loss: 1.0042 - val_accuracy: 0.5833
Epoch 5/50
14/14 [==============================] - 0s 24ms/step - loss: 0.9819 - accuracy: 0.5000 - val_loss: 0.9671 - val_accuracy: 0.6667
Epoch 6/50
14/14 [==============================] - 0s 18ms/step - loss: 0.9341 - accuracy: 0.5556 - val_loss: 0.9290 - val_accuracy: 0.9167
Epoch 7/50
14/14 [==============================] - 0s 23ms/step - loss: 0.8783 - accuracy: 0.6852 - val_loss: 0.8822 - val_accuracy: 0.8333
Epoch 8/50
14/14 [==============================] - 1s 43ms/step - loss: 0.8145 - accuracy: 0.7593 - val_loss: 0.8322 - val_accuracy: 0.9167
Epoch 9/50
14/14 [==============================] - 0s 35ms/step - loss: 0.7438 - accuracy: 0.7685 - val_loss: 0.7796 - val_accuracy: 0.9167
Epoch 10/50
14/14 [==============================] - 0s 11ms/step - loss: 0.6755 - accuracy: 0.7963 - val_loss: 0.7271 - val_accuracy: 0.9167
Epoch 11/50
14/14 [==============================] - 0s 11ms/step - loss: 0.6167 - accuracy: 0.7870 - val_loss: 0.6797 - val_accuracy: 0.9167
Epoch 12/50
14/14 [==============================] - 0s 11ms/step - loss: 0.5642 - accuracy: 0.8519 - val_loss: 0.6389 - val_accuracy: 0.9167
Epoch 13/50
14/14 [==============================] - 0s 9ms/step - loss: 0.5236 - accuracy: 0.8426 - val_loss: 0.6029 - val_accuracy: 0.9167
Epoch 14/50
14/14 [==============================] - 0s 8ms/step - loss: 0.4885 - accuracy: 0.8519 - val_loss: 0.5706 - val_accuracy: 0.9167
Epoch 15/50
14/14 [==============================] - 0s 8ms/step - loss: 0.4578 - accuracy: 0.8796 - val_loss: 0.5395 - val_accuracy: 0.9167
Epoch 16/50
14/14 [==============================] - 0s 7ms/step - loss: 0.4324 - accuracy: 0.8889 - val_loss: 0.5132 - val_accuracy: 0.9167
```

https://github.com/HemanthLakkimsetti76/NNDL_Assignment7/blob/main/Assignment7.ipynb

```
Epoch 17/50
14/14 [==============================] - 0s 8ms/step - loss: 0.4102 - accuracy: 0.8519 - val_loss: 0.4916 - val_accuracy: 0.9167
Epoch 18/50
14/14 [==============================] - 0s 9ms/step - loss: 0.3904 - accuracy: 0.8889 - val_loss: 0.4733 - val_accuracy: 0.9167
Epoch 19/50
14/14 [==============================] - 0s 7ms/step - loss: 0.3729 - accuracy: 0.8889 - val_loss: 0.4531 - val_accuracy: 0.9167
Epoch 20/50
14/14 [==============================] - 0s 8ms/step - loss: 0.3582 - accuracy: 0.8796 - val_loss: 0.4334 - val_accuracy: 0.9167
Epoch 21/50
14/14 [==============================] - 0s 9ms/step - loss: 0.3425 - accuracy: 0.8889 - val_loss: 0.4215 - val_accuracy: 0.9167
Epoch 22/50
14/14 [==============================] - 0s 7ms/step - loss: 0.3308 - accuracy: 0.8889 - val_loss: 0.4029 - val_accuracy: 0.9167
Epoch 23/50
14/14 [==============================] - 0s 9ms/step - loss: 0.3159 - accuracy: 0.8889 - val_loss: 0.3879 - val_accuracy: 0.9167
Epoch 24/50
14/14 [==============================] - 0s 8ms/step - loss: 0.3028 - accuracy: 0.8889 - val_loss: 0.3760 - val_accuracy: 0.9167
Epoch 25/50
14/14 [==============================] - 0s 9ms/step - loss: 0.2921 - accuracy: 0.8981 - val_loss: 0.3667 - val_accuracy: 0.9167
Epoch 26/50
14/14 [==============================] - 0s 7ms/step - loss: 0.2811 - accuracy: 0.8981 - val_loss: 0.3493 - val_accuracy: 0.9167
Epoch 27/50
14/14 [==============================] - 0s 8ms/step - loss: 0.2687 - accuracy: 0.8981 - val_loss: 0.3404 - val_accuracy: 0.9167
Epoch 28/50
14/14 [==============================] - 0s 8ms/step - loss: 0.2557 - accuracy: 0.9259 - val_loss: 0.3372 - val_accuracy: 0.9167
Epoch 29/50
14/14 [==============================] - 0s 8ms/step - loss: 0.2455 - accuracy: 0.9444 - val_loss: 0.3322 - val_accuracy: 0.9167
Epoch 30/50
14/14 [==============================] - 0s 5ms/step - loss: 0.2320 - accuracy: 0.9630 - val_loss: 0.3294 - val_accuracy: 0.9167
Epoch 31/50
14/14 [==============================] - 0s 4ms/step - loss: 0.2235 - accuracy: 0.9537 - val_loss: 0.3129 - val_accuracy: 0.9167
Epoch 32/50
14/14 [==============================] - 0s 5ms/step - loss: 0.2091 - accuracy: 0.9537 - val_loss: 0.3160 - val_accuracy: 0.9167
Epoch 33/50
14/14 [==============================] - 0s 4ms/step - loss: 0.2006 - accuracy: 0.9630 - val_loss: 0.3215 - val_accuracy: 0.9167
```

```
14/14 [==============================] - 0s 6ms/step - loss: 0.1889 - accuracy: 0.9630 - val_loss: 0.2968 - val_accuracy: 0.9167
Epoch 35/50
14/14 [==============================] - 0s 4ms/step - loss: 0.1798 - accuracy: 0.9444 - val_loss: 0.3020 - val_accuracy: 0.9167
Epoch 36/50
14/14 [==============================] - 0s 5ms/step - loss: 0.1700 - accuracy: 0.9537 - val_loss: 0.2988 - val_accuracy: 0.9167
Epoch 37/50
14/14 [==============================] - 0s 5ms/step - loss: 0.1599 - accuracy: 0.9537 - val_loss: 0.2912 - val_accuracy: 0.9167
Epoch 38/50
14/14 [==============================] - 0s 5ms/step - loss: 0.1506 - accuracy: 0.9537 - val_loss: 0.2668 - val_accuracy: 0.9167
Epoch 39/50
14/14 [==============================] - 0s 7ms/step - loss: 0.1430 - accuracy: 0.9630 - val_loss: 0.2724 - val_accuracy: 0.9167
Epoch 40/50
14/14 [==============================] - 0s 5ms/step - loss: 0.1402 - accuracy: 0.9722 - val_loss: 0.2785 - val_accuracy: 0.9167
Epoch 41/50
14/14 [==============================] - 0s 5ms/step - loss: 0.1316 - accuracy: 0.9630 - val_loss: 0.2386 - val_accuracy: 0.9167
Epoch 42/50
14/14 [==============================] - 0s 6ms/step - loss: 0.1260 - accuracy: 0.9630 - val_loss: 0.2658 - val_accuracy: 0.9167
Epoch 43/50
14/14 [==============================] - 0s 6ms/step - loss: 0.1162 - accuracy: 0.9630 - val_loss: 0.2474 - val_accuracy: 0.9167
Epoch 44/50
14/14 [==============================] - 0s 6ms/step - loss: 0.1126 - accuracy: 0.9630 - val_loss: 0.2275 - val_accuracy: 0.9167
Epoch 45/50
14/14 [==============================] - 0s 6ms/step - loss: 0.1070 - accuracy: 0.9630 - val_loss: 0.2421 - val_accuracy: 0.9167
Epoch 46/50
14/14 [==============================] - 0s 6ms/step - loss: 0.1035 - accuracy: 0.9630 - val_loss: 0.2629 - val_accuracy: 0.9167
Epoch 47/50
14/14 [==============================] - 0s 6ms/step - loss: 0.1018 - accuracy: 0.9630 - val_loss: 0.2445 - val_accuracy: 0.9167
Epoch 48/50
14/14 [==============================] - 0s 5ms/step - loss: 0.0991 - accuracy: 0.9630 - val_loss: 0.2208 - val_accuracy: 0.9167
Epoch 49/50
14/14 [==============================] - 0s 4ms/step - loss: 0.0940 - accuracy: 0.9722 - val_loss: 0.2465 - val_accuracy: 0.9167
Epoch 50/50
14/14 [==============================] - 0s 5ms/step - loss: 0.0921 - accuracy: 0.9630 - val_loss: 0.2325 - val_accuracy: 0.9167
1/1 [==============================] - 0s 31ms/step - loss: 0.0829 - accuracy: 1.0000
Accuracy of Modified Neural Network: 1.0
```

https://github.com/HemanthLakkimsetti76/NNDL_Assignment7/blob/main/Assignment7.ipynb

Evaluate the model on the testing set

```
Accuracy of Modified Neural Network: 1.0
```

```
[ ]: # Evaluate the model on the testing set
     loss, accuracy = model.evaluate(X_test_scaled, y_test, verbose=1)
     print("Accuracy on Testing Set:", accuracy)
```

```
1/1 [==============================] - 0s 52ms/step - loss: 0.0829 - accuracy: 1.0000
Accuracy on Testing Set: 1.0
```

Saving the the model and printing the first few predictions

```
Accuracy on Testing Set: 1.0
```

```
[ ]: # Saving the  the model and printing the first few predictions
     model.save("improved_iris_model.h5")
     from tensorflow.keras.models import load_model
     saved_model = load_model("improved_iris_model.h5")
     predictions = saved_model.predict(X_test_scaled)
     print("Predictions:")
     print(predictions[:5])
```

```
1/1 [==============================] - 0s 153ms/step
Predictions:
[[1.6918768e-03 9.3688971e-01 6.1418314e-02]
 [9.9732774e-01 2.6344496e-03 3.7800932e-05]
 [1.0534784e-07 7.2292364e-03 9.9277055e-01]
 [2.5175847e-03 8.1440175e-01 1.8308063e-01]
 [5.5927003e-04 7.9731899e-01 2.0212181e-01]]
```

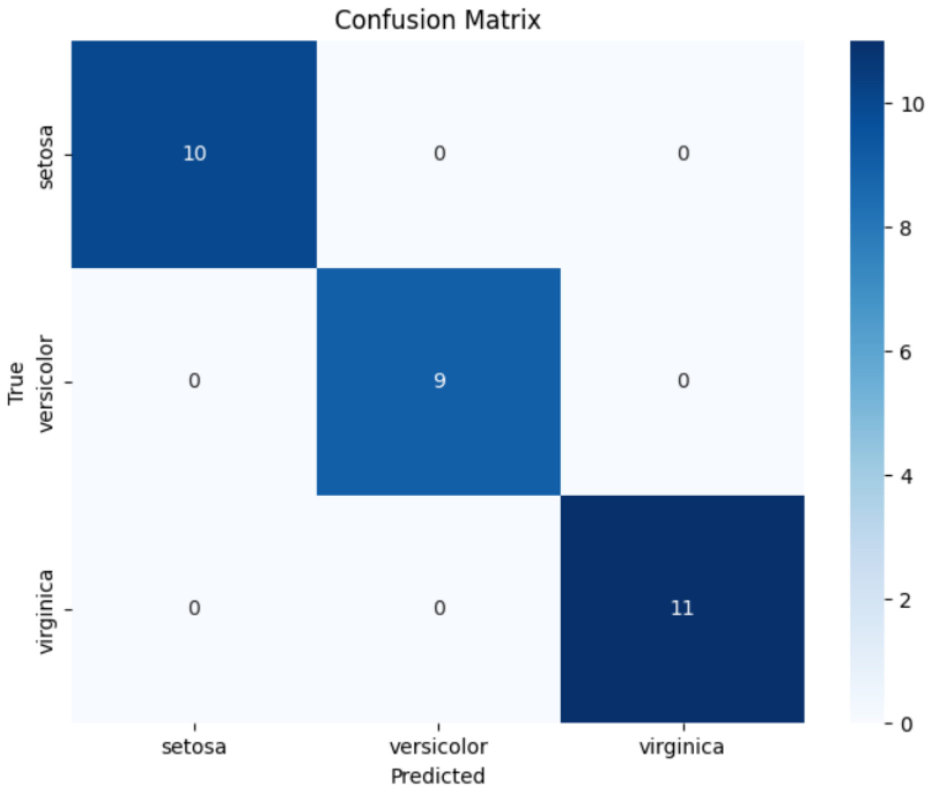
plot of confusion matric

```
[ ]: # plot of confusion matric
     import numpy as np
     import matplotlib.pyplot as plt
     from sklearn.metrics import confusion_matrix
     import seaborn as sns
     from tensorflow.keras.models import Sequential
     print(hasattr(model, 'predict_classes'))
     y_pred = model.predict(X_test_scaled).argmax(axis=1)
     cm = confusion_matrix(y_test, y_pred)
     class_names = iris.target_names
     plt.figure(figsize=(8, 6))
     sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=class_names, yticklabels=class_names)
     plt.title("Confusion Matrix")
     plt.xlabel("Predicted")
     plt.ylabel("True")
     plt.show()
```

```
False
1/1 [==============================] - 0s 110ms/step
```

https://github.com/HemanthLakkimsetti76/NNDL_Assignment7/blob/main/Assignment7.ipynb

```
1/1 [==============================] - 0s 110ms/step
```



Training and testing Loss and accuracy plots in one plot using subplot command and history object

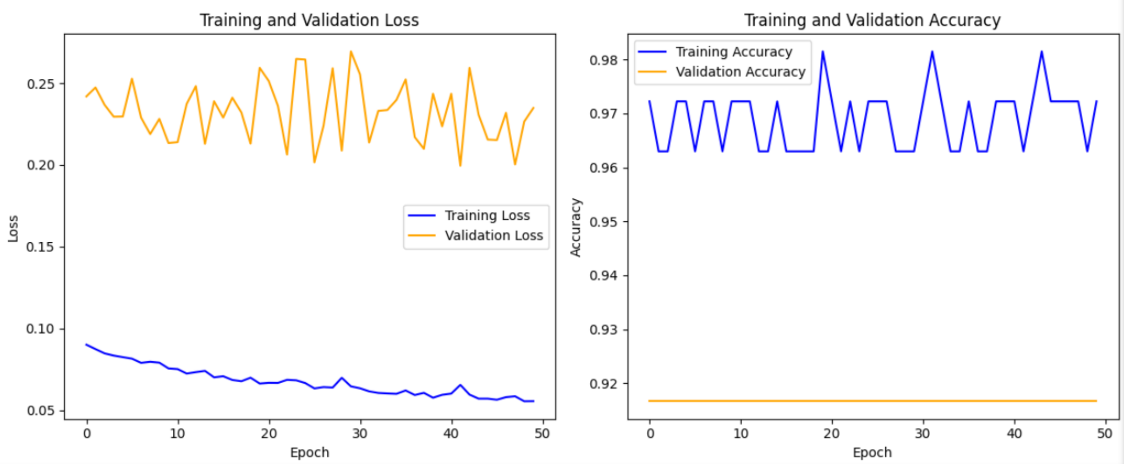
```
# Training and testing Loss and accuracy plots in one plot using subplot command and history object
history = model.fit(X_train_scaled, y_train, epochs=50, batch_size=8, verbose=1, validation_split=0.1)
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss', color='blue')
plt.plot(history.history['val_loss'], label='Validation Loss', color='orange')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy', color='blue')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy', color='orange')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```

https://github.com/HemanthLakkimsetti76/NNDL_Assignment7/blob/main/Assignment7.ipynb

```
plt.show()
```

```
14/14 [==============================] - 0s 5ms/step - loss: 0.0593 - accuracy: 0.9630 - val_loss: 0.2170 - val_accuracy: 0.9167
Epoch 38/50
14/14 [==============================] - 0s 5ms/step - loss: 0.0606 - accuracy: 0.9630 - val_loss: 0.2098 - val_accuracy: 0.9167
Epoch 39/50
14/14 [==============================] - 0s 5ms/step - loss: 0.0576 - accuracy: 0.9722 - val_loss: 0.2435 - val_accuracy: 0.9167
Epoch 40/50
14/14 [==============================] - 0s 5ms/step - loss: 0.0594 - accuracy: 0.9722 - val_loss: 0.2236 - val_accuracy: 0.9167
Epoch 41/50
14/14 [==============================] - 0s 5ms/step - loss: 0.0602 - accuracy: 0.9722 - val_loss: 0.2435 - val_accuracy: 0.9167
Epoch 42/50
14/14 [==============================] - 0s 6ms/step - loss: 0.0655 - accuracy: 0.9630 - val_loss: 0.1995 - val_accuracy: 0.9167
Epoch 43/50
14/14 [==============================] - 0s 5ms/step - loss: 0.0595 - accuracy: 0.9722 - val_loss: 0.2593 - val_accuracy: 0.9167
Epoch 44/50
14/14 [==============================] - 0s 5ms/step - loss: 0.0571 - accuracy: 0.9815 - val_loss: 0.2306 - val_accuracy: 0.9167
Epoch 45/50
14/14 [==============================] - 0s 6ms/step - loss: 0.0571 - accuracy: 0.9722 - val_loss: 0.2155 - val_accuracy: 0.9167
Epoch 46/50
14/14 [==============================] - 0s 5ms/step - loss: 0.0564 - accuracy: 0.9722 - val_loss: 0.2152 - val_accuracy: 0.9167
Epoch 47/50
14/14 [==============================] - 0s 8ms/step - loss: 0.0580 - accuracy: 0.9722 - val_loss: 0.2319 - val_accuracy: 0.9167
Epoch 48/50
14/14 [==============================] - 0s 9ms/step - loss: 0.0585 - accuracy: 0.9722 - val_loss: 0.2003 - val_accuracy: 0.9167
Epoch 49/50
14/14 [==============================] - 0s 6ms/step - loss: 0.0554 - accuracy: 0.9630 - val_loss: 0.2265 - val_accuracy: 0.9167
Epoch 50/50
14/14 [==============================] - 0s 9ms/step - loss: 0.0555 - accuracy: 0.9722 - val_loss: 0.2348 - val_accuracy: 0.9167
```

```
14/14 [==============================] - 0s 9ms/step - loss: 0.0555 - accuracy: 0.9722 - val_loss: 0.2348 - val_accuracy: 0.9167
```

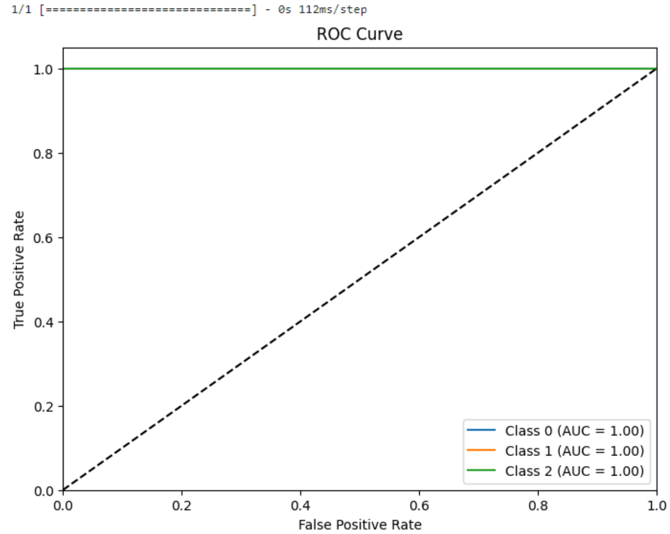


https://github.com/HemanthLakkimsetti76/NNDL_Assignment7/blob/main/Assignment7.ipynb

```
[ ]: import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
from sklearn.metrics import roc_auc_score
y_test_one_hot = label_binarize(y_test, classes=[0, 1, 2])
y_probs = model.predict(X_test_scaled)
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(3):
    fpr[i], tpr[i], _ = roc_curve(y_test_one_hot[:, i], y_probs[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
plt.figure(figsize=(8, 6))
for i in range(3):
    plt.plot(fpr[i], tpr[i], label=f'Class {i} (AUC = {roc_auc[i]:0.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")
plt.show()

first_layer_weights = model.layers[0].get_weights()[0]
importances = np.mean(np.abs(first_layer_weights), axis=1)
indices = np.argsort(importances)
plt.figure(figsize=(10, 6))
plt.title("Feature Importance")
plt.barh(range(X_train_scaled.shape[1]), importances[indices], align="center")
plt.yticks(range(X_train_scaled.shape[1]), [iris.feature_names[i] for i in indices])
plt.xlabel("Mean Absolute Weight")
plt.ylabel("Feature")
plt.show()
```

```
1/1 [==============================] - 0s 112ms/step
```

```
1/1 [==============================] - 0s 112ms/step
```



https://github.com/HemanthLakkimsetti76/NNDL_Assignment7/blob/main/Assignment7.ipynb