Plots 2D

May 3, 2025

1 Importing Necessary Libraries

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
import numpy as np
```

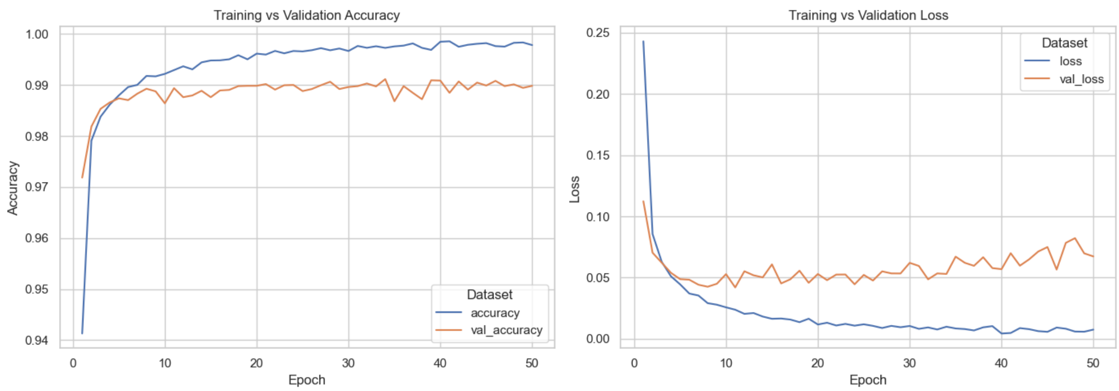
2 Loss and Accuracy Graph Plots

2.1 Loading the Model History DataFrame

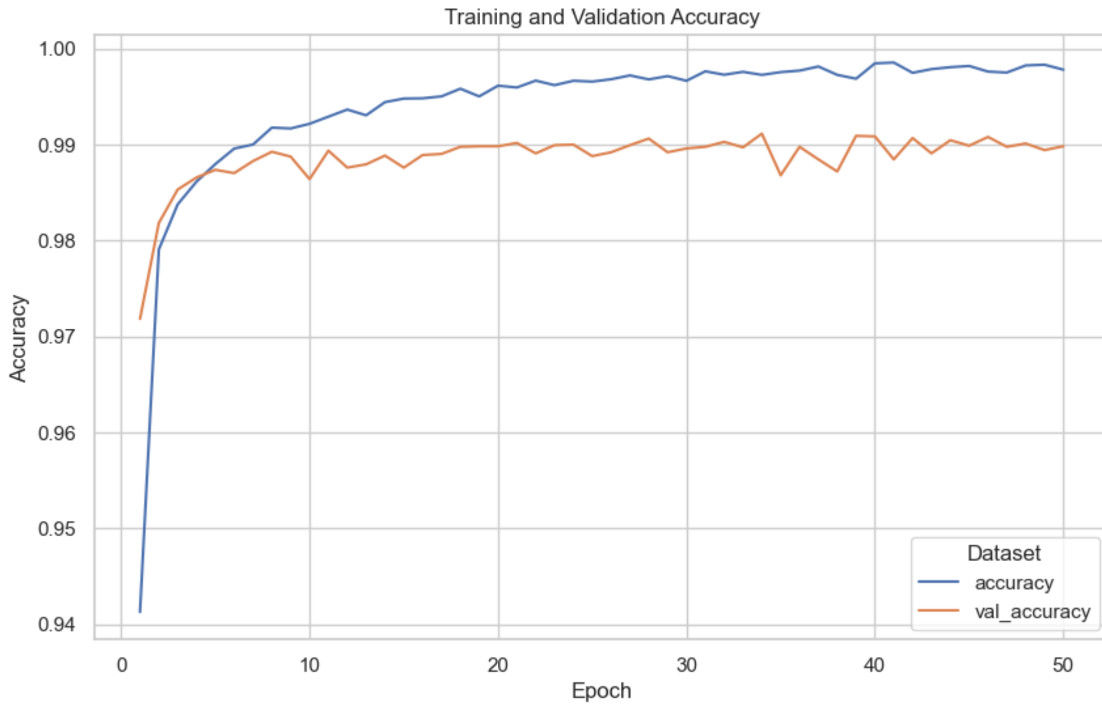
[6]:
```
hist_df = pd.read_csv("../History 2D.csv")
hist_df["epoch"] = hist_df.index + 1

accuracy_df = hist_df.melt(id_vars = "epoch", value_vars=["accuracy",␣
↪"val_accuracy"], var_name="Dataset", value_name="Accuracy")
loss_df = hist_df.melt(id_vars = "epoch", value_vars = ["loss", "val_loss"],␣
↪var_name = "Dataset", value_name=  "Loss")
```

[10]:
```
sns.set(style="whitegrid")

fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Accuracy plot
sns.lineplot(data=accuracy_df, x="epoch", y="Accuracy", hue="Dataset",␣
↪ax=axes[0])
axes[0].set_title("Training vs Validation Accuracy")
axes[0].set_xlabel("Epoch")
axes[0].set_ylabel("Accuracy")

# Loss plot
sns.lineplot(data=loss_df, x="epoch", y="Loss", hue="Dataset", ax=axes[1])
axes[1].set_title("Training vs Validation Loss")
axes[1].set_xlabel("Epoch")
```

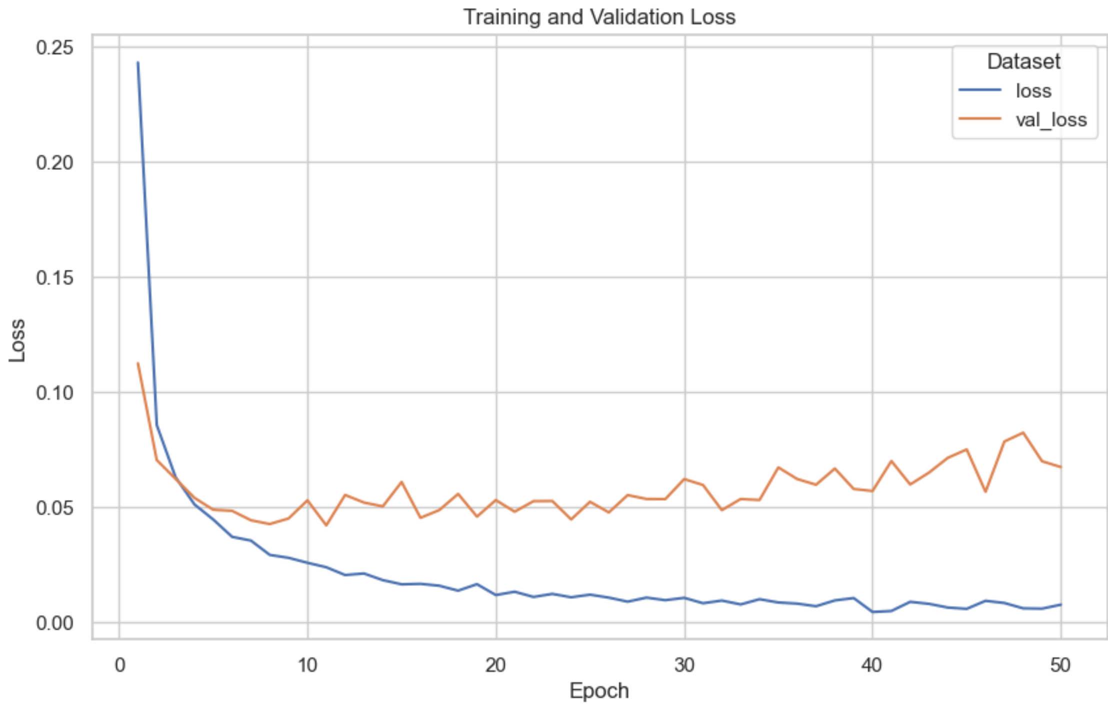
```
axes[1].set_ylabel("Loss")

plt.tight_layout()

plt.savefig("../Plots/2D/Combined Graphs 2D.png", dpi=300)  # High-res PNG
plt.show()
```



[12]:
```
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.lineplot(data=accuracy_df, x="epoch", y="Accuracy", hue="Dataset")
plt.title("Training and Validation Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(title="Dataset")
plt.savefig("../Plots/2D/Accuracy Graphs 2D.png", dpi=300)  # High-res PNG
plt.show()
```



[14]:
```
plt.figure(figsize=(10, 6))
sns.lineplot(data=loss_df, x="epoch", y="Loss", hue="Dataset")
plt.title("Training and Validation Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(title="Dataset")
plt.savefig("../Plots/2D/Loss Graphs 2D.png", dpi=300)
plt.show()
```



3 Confusion Matrix

3.1 Loading the Model and the Data

[18]:
```
model = tf.keras.models.load_model("../Models/Model 2D.h5")
```

[24]:
```
X = np.load("../Features.npy")
y = np.load("../Labels (Mutli Class).npy")
```

3.2 Splitting the Data and Performing Model Prediction

[27]:
```
_, X_test, _, y_test = train_test_split(X, y, shuffle = True, random_state =␣
↪10, test_size = 0.2)
```

[29]:
```
X_test.shape, y_test.shape
```

[29]:
```
((21884, 250, 2), (21884,))
```

[31]:
```
y_pred_probs = model.predict(X_test)
y_pred = y_pred_probs.argmax(axis = 1)
```

```
684/684 [==============================] - 3s 2ms/step
```

[32]:
```
y_true = y_test
```

3.2.1 Check for Missing Labels in y_true and y_pred

Carefully check the labels of both the True and the Predicted labels, then proceed to plotting the Confusion Matrix

[35]:
```python
values, _ = np.unique(y_true, return_counts = True)
```

[36]:
```python
values
```

[36]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13],
      dtype=int64)
```

[37]:
```python
np.unique(y_true, return_counts = True)
```

[37]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13],
       dtype=int64),
 array([ 1352,   501,    21,   149,    22,  1620, 15135,  1411,     1,
         1407,    28,     6,   191,    40], dtype=int64))
```

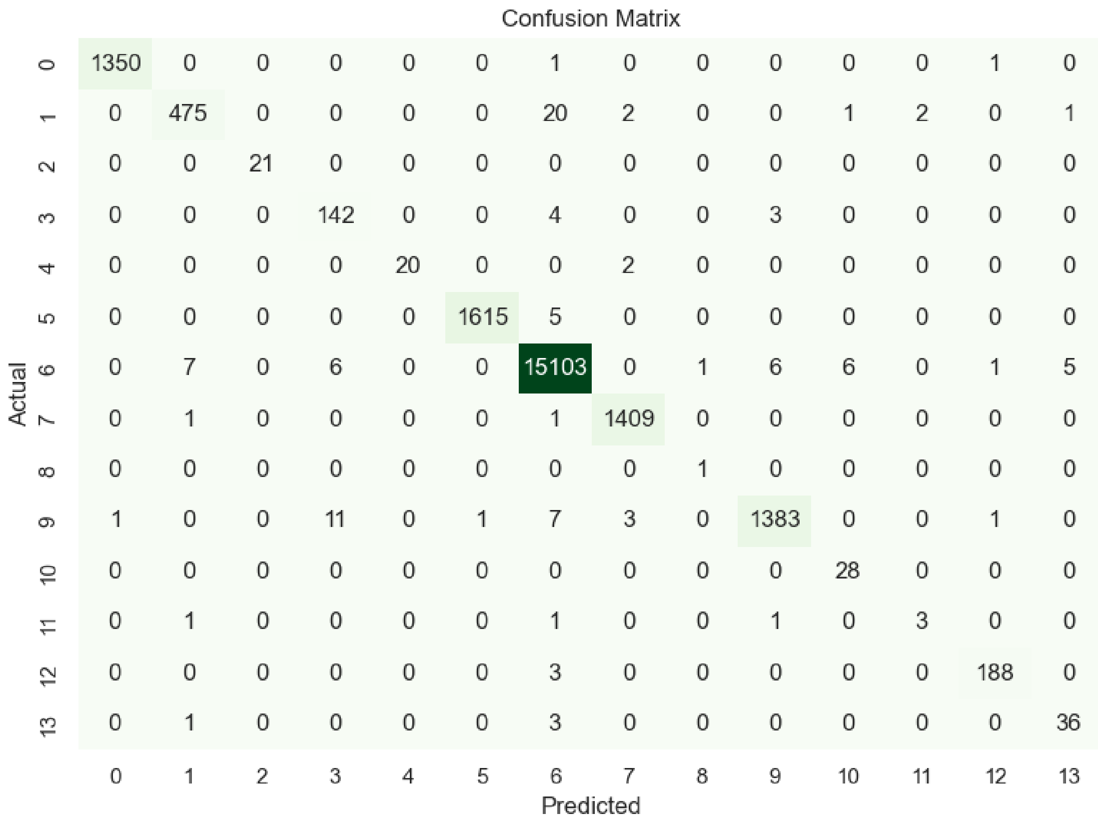
[38]:
```python
np.unique(y_pred, return_counts = True)
```

[38]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13],
       dtype=int64),
 array([ 1351,   485,    21,   159,    20,  1616, 15148,  1416,     2,
         1393,    35,     5,   191,    42], dtype=int64))
```

3.3 Plotting

[46]:
```python
cm = confusion_matrix(y_true, y_pred)
```

[48]:
```python
plt.figure(figsize = (8, 6))
sns.heatmap(cm, annot = True, fmt = "d", cmap = "Greens", cbar = False,␣
 ↪xticklabels = values, yticklabels = values)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.tight_layout()

# Save the plot
plt.savefig("../Plots/2D/Confusion Matrix 2D.png", dpi=300)
plt.show()
```



4 Classification Report

[53]:
```python
print(classification_report(y_true, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1352
           1       0.98      0.95      0.96       501
           2       1.00      1.00      1.00        21
           3       0.89      0.95      0.92       149
           4       1.00      0.91      0.95        22
           5       1.00      1.00      1.00      1620
           6       1.00      1.00      1.00     15135
           7       1.00      1.00      1.00      1411
           8       0.50      1.00      0.67         1
           9       0.99      0.98      0.99      1407
          10       0.80      1.00      0.89        28
          11       0.60      0.50      0.55         6
          12       0.98      0.98      0.98       191
          13       0.86      0.90      0.88        40
```

```
    accuracy                           0.99     21884
   macro avg       0.90      0.94      0.91     21884
weighted avg       1.00      0.99      0.99     21884
```