DL_3

April 6, 2025

[1]:
```
# Step 1: Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
WARNING:tensorflow:From C:\Users\sanya\AppData\Roaming\Python\Python311\site-
packages\keras\src\losses.py:2976: The name
tf.losses.sparse_softmax_cross_entropy is deprecated. Please use
tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

[2]:
```
# Step 2: Load the dataset from UCI URL
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/
 ↪letter-recognition/letter-recognition.data"
```

[3]:
```
# Column names based on dataset description
columns = ['letter', 'x-box', 'y-box', 'width', 'height', 'onpix', 'x-bar',␣
 ↪'y-bar',
           'x2bar', 'y2bar', 'xybar', 'x2ybr', 'xy2br', 'x-ege', 'xegvy',␣
 ↪'y-ege', 'yegvx']
```

[4]:
```
# Load into a DataFrame
df = pd.read_csv(url, header=None, names=columns)
```

[5]:
```
# Display first 5 rows
print("Sample data:")
print(df.head())
```

```
Sample data:
  letter  x-box  y-box  width  height  onpix  x-bar  y-bar  x2bar  y2bar  \
0      T      2      8      3       5      1      8     13      0      6
1      I      5     12      3       7      2     10      5      5      4
2      D      4     11      6       8      6     10      6      2      6
3      N      7     11      6       6      3      5      9      4      6
```

```
4      G      2      1      3      1      1      8      6      6      6

   xybar  x2ybr  xy2br  x-ege  xegvy  y-ege  yegvx
0      6     10      8      0      8      0      8
1     13      3      9      2      8      4     10
2     10      3      7      3      7      3      9
3      4      4     10      6     10      2      8
4      6      5      9      1      7      5     10
```

[6]:
```
# Step 3: Preprocess the data
X = df.drop('letter', axis=1)
y = df['letter']
```

[7]:
```
# Convert letter labels to numeric using LabelEncoder
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
```

[8]:
```
# One-hot encode the target labels for multiclass classification
from tensorflow.keras.utils import to_categorical
y_onehot = to_categorical(y_encoded)
```

[9]:
```
# Normalize feature values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[10]:
```
# Step 4: Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_onehot,␣
 ↪test_size=0.2, random_state=42)
```

[11]:
```
# Step 5: Build the DNN model
model = Sequential([
    Dense(64, input_shape=(16,), activation='relu'),
    Dense(64, activation='relu'),
    Dense(26, activation='softmax')  # 26 output classes (A-Z)
])
```

```
WARNING:tensorflow:From C:\Users\sanya\AppData\Roaming\Python\Python311\site-
packages\keras\src\backend.py:873: The name tf.get_default_graph is deprecated.
Please use tf.compat.v1.get_default_graph instead.
```

[12]:
```
# Step 6: Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
```

```
WARNING:tensorflow:From C:\Users\sanya\AppData\Roaming\Python\Python311\site-
packages\keras\src\optimizers\__init__.py:309: The name tf.train.Optimizer is
deprecated. Please use tf.compat.v1.train.Optimizer instead.
```

[13]:
```
# Step 7: Train the model
history = model.fit(X_train, y_train, epochs=20, batch_size=32,␣
↪validation_split=0.1)
```

```
Epoch 1/20
WARNING:tensorflow:From C:\Users\sanya\AppData\Roaming\Python\Python311\site-
packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue
is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\sanya\AppData\Roaming\Python\Python311\site-
packages\keras\src\engine\base_layer_utils.py:384: The name
tf.executing_eagerly_outside_functions is deprecated. Please use
tf.compat.v1.executing_eagerly_outside_functions instead.

450/450 [==============================] - 4s 4ms/step - loss: 1.6841 -
accuracy: 0.5484 - val_loss: 0.9379 - val_accuracy: 0.7406
Epoch 2/20
450/450 [==============================] - 2s 3ms/step - loss: 0.7794 -
accuracy: 0.7808 - val_loss: 0.6791 - val_accuracy: 0.8000
Epoch 3/20
450/450 [==============================] - 1s 3ms/step - loss: 0.5999 -
accuracy: 0.8230 - val_loss: 0.5486 - val_accuracy: 0.8350
Epoch 4/20
450/450 [==============================] - 1s 3ms/step - loss: 0.4962 -
accuracy: 0.8557 - val_loss: 0.4620 - val_accuracy: 0.8606
Epoch 5/20
450/450 [==============================] - 2s 3ms/step - loss: 0.4270 -
accuracy: 0.8729 - val_loss: 0.4085 - val_accuracy: 0.8769
Epoch 6/20
450/450 [==============================] - 1s 3ms/step - loss: 0.3755 -
accuracy: 0.8874 - val_loss: 0.3768 - val_accuracy: 0.8806
Epoch 7/20
450/450 [==============================] - 2s 3ms/step - loss: 0.3357 -
accuracy: 0.8985 - val_loss: 0.3456 - val_accuracy: 0.9006
Epoch 8/20
450/450 [==============================] - 1s 3ms/step - loss: 0.3036 -
accuracy: 0.9074 - val_loss: 0.3079 - val_accuracy: 0.9112
Epoch 9/20
450/450 [==============================] - 2s 3ms/step - loss: 0.2765 -
accuracy: 0.9151 - val_loss: 0.2894 - val_accuracy: 0.9162
Epoch 10/20
450/450 [==============================] - 1s 3ms/step - loss: 0.2533 -
accuracy: 0.9233 - val_loss: 0.2780 - val_accuracy: 0.9194
Epoch 11/20
450/450 [==============================] - 1s 3ms/step - loss: 0.2328 -
accuracy: 0.9295 - val_loss: 0.2573 - val_accuracy: 0.9200
Epoch 12/20
450/450 [==============================] - 2s 3ms/step - loss: 0.2166 -
```

```
accuracy: 0.9334 - val_loss: 0.2304 - val_accuracy: 0.9337
Epoch 13/20
450/450 [==============================] - 2s 3ms/step - loss: 0.1989 -
accuracy: 0.9390 - val_loss: 0.2364 - val_accuracy: 0.9281
Epoch 14/20
450/450 [==============================] - 1s 3ms/step - loss: 0.1881 -
accuracy: 0.9415 - val_loss: 0.2249 - val_accuracy: 0.9350
Epoch 15/20
450/450 [==============================] - 1s 3ms/step - loss: 0.1752 -
accuracy: 0.9491 - val_loss: 0.2073 - val_accuracy: 0.9388
Epoch 16/20
450/450 [==============================] - 2s 3ms/step - loss: 0.1649 -
accuracy: 0.9490 - val_loss: 0.2012 - val_accuracy: 0.9450
Epoch 17/20
450/450 [==============================] - 2s 3ms/step - loss: 0.1548 -
accuracy: 0.9513 - val_loss: 0.1902 - val_accuracy: 0.9469
Epoch 18/20
450/450 [==============================] - 2s 3ms/step - loss: 0.1459 -
accuracy: 0.9552 - val_loss: 0.1921 - val_accuracy: 0.9444
Epoch 19/20
450/450 [==============================] - 2s 3ms/step - loss: 0.1378 -
accuracy: 0.9570 - val_loss: 0.1767 - val_accuracy: 0.9450
Epoch 20/20
450/450 [==============================] - 2s 4ms/step - loss: 0.1314 -
accuracy: 0.9595 - val_loss: 0.1797 - val_accuracy: 0.9463
```

[14]:
```python
# Step 8: Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f"\nTest Accuracy: {accuracy:.2f}")
```

```
125/125 [==============================] - 0s 3ms/step - loss: 0.1928 -
accuracy: 0.9405

Test Accuracy: 0.94
```

[15]:
```python
# Step 9: Predictions
y_pred_probs = model.predict(X_test)
y_pred = np.argmax(y_pred_probs, axis=1)
y_true = np.argmax(y_test, axis=1)
```

```
125/125 [==============================] - 1s 3ms/step
```

[16]:
```python
# Decode predictions and actual labels
y_pred_labels = label_encoder.inverse_transform(y_pred)
y_true_labels = label_encoder.inverse_transform(y_true)
```

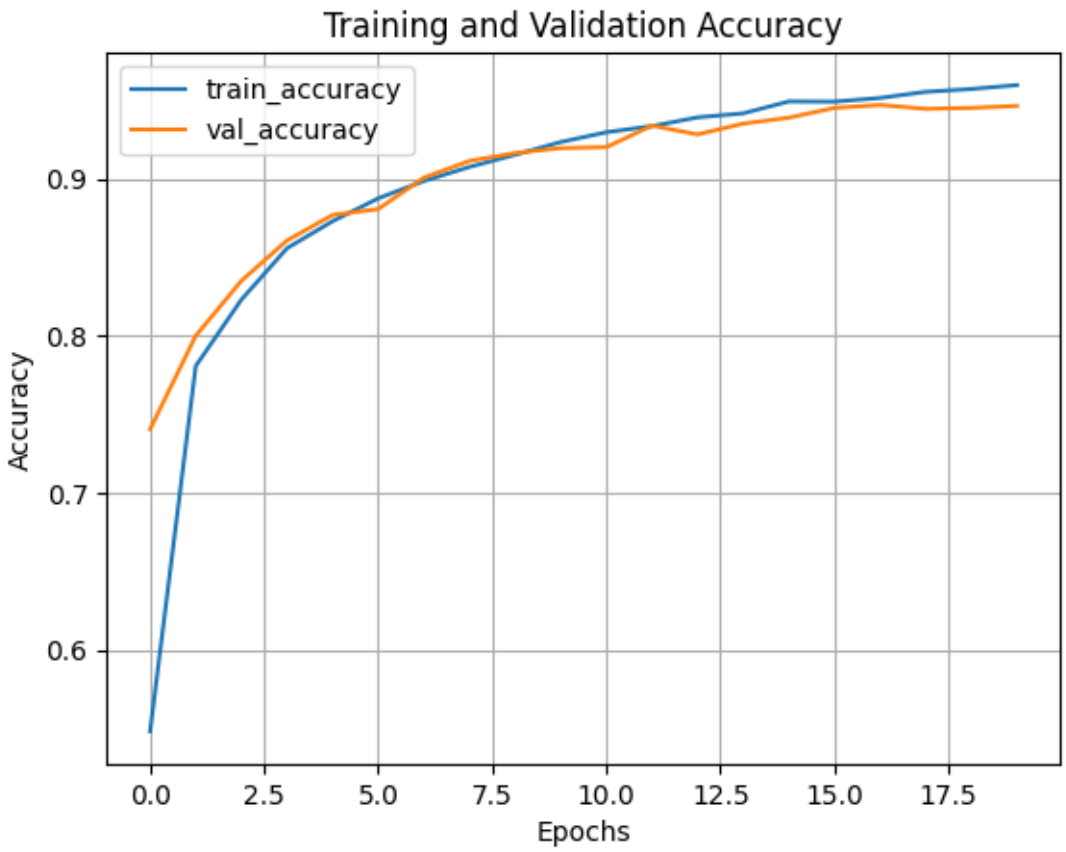
[17]:
```python
# Step 10: Classification report
print("\nClassification Report:")
print(classification_report(y_true_labels, y_pred_labels))
```

```
Classification Report:
              precision    recall  f1-score   support

           A       0.97      0.97      0.97       149
           B       0.87      0.95      0.91       153
           C       0.98      0.93      0.95       137
           D       0.95      0.88      0.91       156
           E       0.93      0.94      0.93       141
           F       0.85      0.94      0.89       140
           G       0.95      0.96      0.95       160
           H       0.89      0.86      0.88       144
           I       0.97      0.90      0.94       146
           J       0.91      0.97      0.94       149
           K       0.88      0.93      0.90       130
           L       0.99      0.97      0.98       155
           M       0.93      0.99      0.96       168
           N       0.96      0.92      0.94       151
           O       0.91      0.94      0.93       145
           P       0.97      0.88      0.92       173
           Q       0.95      0.98      0.96       166
           R       0.93      0.89      0.91       160
           S       0.98      0.97      0.97       171
           T       0.89      0.98      0.93       163
           U       0.96      0.96      0.96       183
           V       0.97      0.96      0.96       158
           W       0.97      0.97      0.97       148
           X       0.96      0.99      0.97       154
           Y       0.97      0.88      0.93       168
           Z       0.98      0.93      0.95       132

    accuracy                           0.94      4000
   macro avg       0.94      0.94      0.94      4000
weighted avg       0.94      0.94      0.94      4000
```

[18]:
```python
# Step 11: Plot accuracy over epochs
plt.plot(history.history['accuracy'], label='train_accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title("Training and Validation Accuracy")
plt.legend()
plt.grid(True)
plt.show()
```


Training and Validation Accuracy
train_accuracy
val_accuracy
Accuracy
0.6
0.7
0.8
0.9
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs