DL_3

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns


url = "https://archive.ics.uci.edu/ml/machine-learning-databases/letter-recognition/letter-recognition.data"
columns = ['letter', 'x-box', 'y-box', 'width', 'high', 'onpix', 'x-bar', 'y-bar', 'x2bar',
           'y2bar', 'xybar', 'x2ybr', 'xy2br', 'x-ege', 'xegvy', 'y-ege', 'yegvx']
df = pd.read_csv(url, names=columns)


X = df.drop('letter', axis=1).values
y = df['letter'].values


label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)


X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)


scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


y_train_cat = keras.utils.to_categorical(y_train, num_classes=26)
y_test_cat = keras.utils.to_categorical(y_test, num_classes=26)


model = keras.Sequential([
    layers.Input(shape=(16,)),
    layers.Dense(128, activation='relu'),
    layers.Dense(64, activation='relu'),
    layers.Dense(26, activation='softmax')  # 26 classes for letters A-Z
])


model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])


history = model.fit(X_train_scaled, y_train_cat, epochs=20, batch_size=64, validation_split=0.2, verbose=1)
```

```
Epoch 1/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - accuracy: 0.3225 - loss: 2.5130 - val_accuracy: 0.6894 - val_loss: 1.1274
Epoch 2/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7262 - loss: 0.9919 - val_accuracy: 0.7716 - val_loss: 0.8070
Epoch 3/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7959 - loss: 0.7228 - val_accuracy: 0.8166 - val_loss: 0.6461
Epoch 4/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8335 - loss: 0.5890 - val_accuracy: 0.8416 - val_loss: 0.5549
Epoch 5/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8639 - loss: 0.4919 - val_accuracy: 0.8559 - val_loss: 0.4975
Epoch 6/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.8784 - loss: 0.4263 - val_accuracy: 0.8744 - val_loss: 0.4309
Epoch 7/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8849 - loss: 0.3872 - val_accuracy: 0.8891 - val_loss: 0.3891
Epoch 8/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9031 - loss: 0.3397 - val_accuracy: 0.8928 - val_loss: 0.3548
Epoch 9/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9067 - loss: 0.3166 - val_accuracy: 0.9003 - val_loss: 0.3352
Epoch 10/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9168 - loss: 0.2740 - val_accuracy: 0.9097 - val_loss: 0.3072
Epoch 11/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9218 - loss: 0.2653 - val_accuracy: 0.9147 - val_loss: 0.2926
Epoch 12/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9301 - loss: 0.2358 - val_accuracy: 0.9141 - val_loss: 0.2825
Epoch 13/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9352 - loss: 0.2200 - val_accuracy: 0.9184 - val_loss: 0.2608
Epoch 14/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9411 - loss: 0.2061 - val_accuracy: 0.9212 - val_loss: 0.2568
Epoch 15/20
```

```
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9480 - loss: 0.1873 - val_accuracy: 0.9350 - val_loss: 0.2370
Epoch 16/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9455 - loss: 0.1752 - val_accuracy: 0.9359 - val_loss: 0.2254
Epoch 17/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9517 - loss: 0.1662 - val_accuracy: 0.9337 - val_loss: 0.2230
Epoch 18/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9594 - loss: 0.1492 - val_accuracy: 0.9366 - val_loss: 0.2066
Epoch 19/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9563 - loss: 0.1446 - val_accuracy: 0.9362 - val_loss: 0.2081
Epoch 20/20
200/200 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9599 - loss: 0.1360 - val_accuracy: 0.9422 - val_loss: 0.1986
```

```
test_loss, test_acc = model.evaluate(X_test_scaled, y_test_cat, verbose=0)
print(f"\nTest Accuracy: {test_acc:.2f}")
```

```

Test Accuracy: 0.94
```

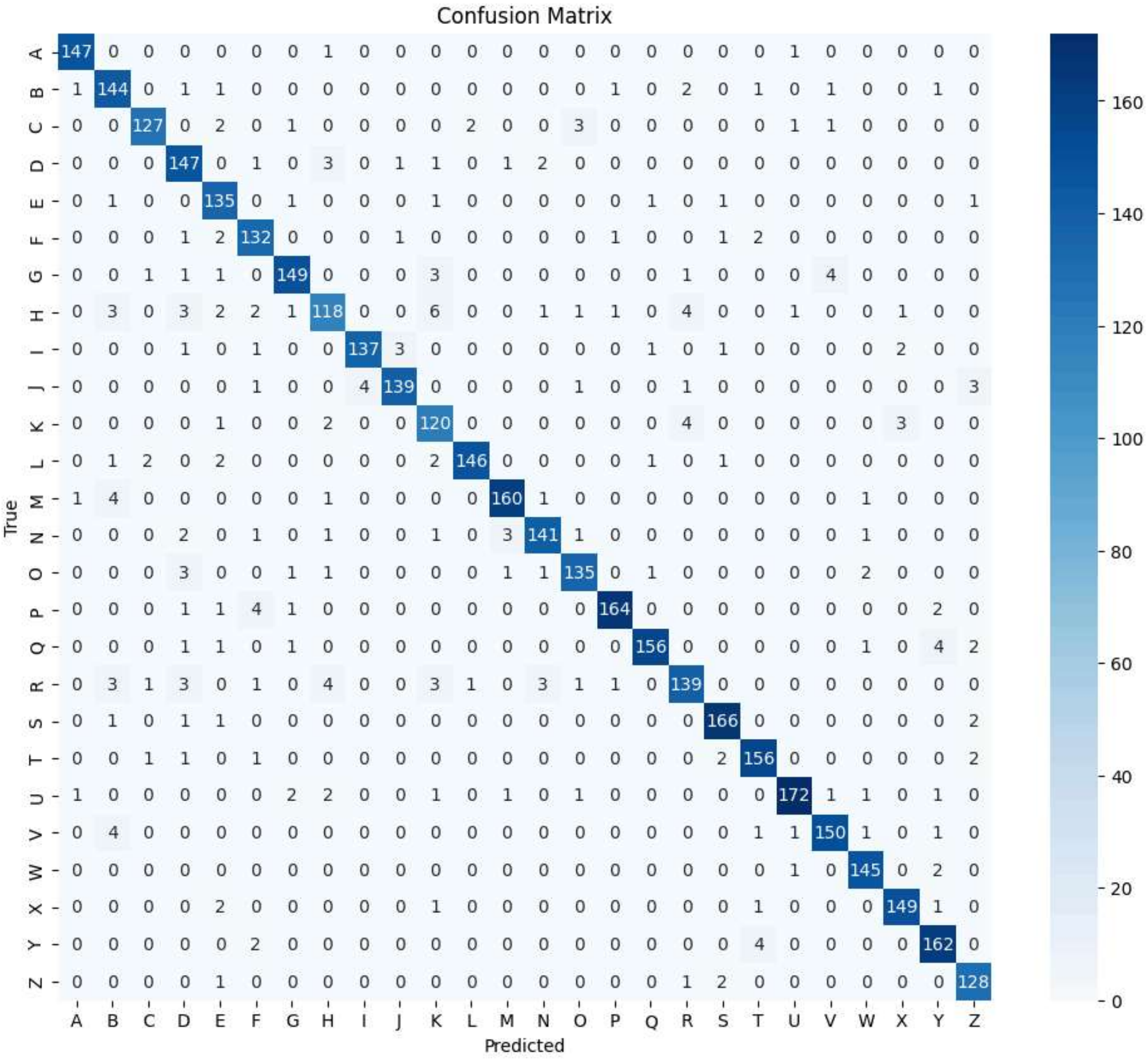
```
y_pred_probs = model.predict(X_test_scaled)
y_pred = np.argmax(y_pred_probs, axis=1)
```

```
125/125 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

```
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=label_encoder.classes_))
```

```

Classification Report:
              precision    recall  f1-score   support

           A       0.98      0.99      0.98       149
           B       0.89      0.94      0.92       153
           C       0.96      0.93      0.94       137
           D       0.89      0.94      0.91       156
           E       0.89      0.96      0.92       141
           F       0.90      0.94      0.92       140
           G       0.95      0.93      0.94       160
           H       0.89      0.82      0.85       144
           I       0.97      0.94      0.95       146
           J       0.97      0.93      0.95       149
           K       0.86      0.92      0.89       130
           L       0.98      0.94      0.96       155
           M       0.96      0.95      0.96       168
           N       0.95      0.93      0.94       151
           O       0.94      0.93      0.94       145
           P       0.98      0.95      0.96       173
           Q       0.97      0.94      0.96       166
           R       0.91      0.87      0.89       160
           S       0.95      0.97      0.96       171
           T       0.95      0.96      0.95       163
           U       0.97      0.94      0.96       183
           V       0.96      0.95      0.95       158
           W       0.95      0.98      0.97       148
           X       0.96      0.97      0.96       154
           Y       0.93      0.96      0.95       168
           Z       0.93      0.97      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

```

```
plt.figure(figsize=(12, 10))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues',
            xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	147	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
B	1	144	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	2	0	1	0	1	0	0	1	0
C	0	0	127	0	2	0	1	0	0	0	0	2	0	0	3	0	0	0	0	0	1	1	0	0	0	0
D	0	0	0	147	0	1	0	3	0	1	1	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0
E	0	1	0	0	135	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1
F	0	0	0	1	2	132	0	0	0	1	0	0	0	0	0	1	0	0	1	2	0	0	0	0	0	0
G	0	0	1	1	1	0	149	0	0	0	3	0	0	0	0	0	0	1	0	0	0	4	0	0	0	0
H	0	3	0	3	2	2	1	118	0	0	6	0	0	1	1	1	0	4	0	0	1	0	0	1	0	0
I	0	0	0	1	0	1	0	0	137	3	0	0	0	0	0	0	1	0	1	0	0	0	0	2	0	0
J	0	0	0	0	0	1	0	0	4	139	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	3
K	0	0	0	0	1	0	0	2	0	0	120	0	0	0	0	0	0	4	0	0	0	0	0	3	0	0
L	0	1	2	0	2	0	0	0	0	0	2	146	0	0	0	0	1	0	1	0	0	0	0	0	0	0
M	1	4	0	0	0	0	0	1	0	0	0	0	160	1	0	0	0	0	0	0	0	0	1	0	0	0
N	0	0	0	2	0	1	0	1	0	0	1	0	3	141	1	0	0	0	0	0	0	0	1	0	0	0
O	0	0	0	3	0	0	1	1	0	0	0	0	1	1	135	0	1	0	0	0	0	0	2	0	0	0
P	0	0	0	1	1	4	1	0	0	0	0	0	0	0	0	164	0	0	0	0	0	0	0	0	2	0
Q	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	156	0	0	0	0	0	1	0	4	2
R	0	3	1	3	0	1	0	4	0	0	3	1	0	3	1	1	0	139	0	0	0	0	0	0	0	0
S	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	166	0	0	0	0	0	0	2
T	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2	156	0	0	0	0	0	2
U	1	0	0	0	0	0	2	2	0	0	1	0	1	0	1	0	0	0	0	0	172	1	1	0	1	0
V	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	150	1	0	1	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	145	0	2	0
X	0	0	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	149	1	0
Y	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	162	0
Z	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	128