```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report, confusion_matrix

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.utils import to_categorical


url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/penguins.csv"
df = pd.read_csv(url)
print(df.head())
```

```
   species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0  Adelie  Torgersen            39.1           18.7              181.0
1  Adelie  Torgersen            39.5           17.4              186.0
2  Adelie  Torgersen            40.3           18.0              195.0
3  Adelie  Torgersen             NaN            NaN                NaN
4  Adelie  Torgersen            36.7           19.3              193.0

   body_mass_g     sex
0       3750.0    MALE
1       3800.0  FEMALE
2       3250.0  FEMALE
3          NaN     NaN
4       3450.0  FEMALE
```

```
df.dropna(inplace=True)


le_species = LabelEncoder()
le_sex = LabelEncoder()
le_island = LabelEncoder()


df['species'] = le_species.fit_transform(df['species'])
df['sex'] = le_sex.fit_transform(df['sex'])
df['island'] = le_island.fit_transform(df['island'])


features = ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g', 'sex', 'island']
X = df[features]
y = df['species']


scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


X_cnn = X_scaled.reshape(-1, 2, 3, 1)
y_cnn = to_categorical(y, num_classes=3)


X_train, X_test, y_train, y_test = train_test_split(X_cnn, y_cnn, test_size=0.2, random_state=42)



model = Sequential([
    Conv2D(32, (2, 2), activation='relu', input_shape=(2, 3, 1)),
    MaxPooling2D(pool_size=(1, 1)),
    Flatten(),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(3, activation='softmax')
])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

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

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 1, 2, 32)	160
max_pooling2d (MaxPooling2D)	(None, 1, 2, 32)	0
flatten (Flatten)	(None, 64)	0
dense (Dense)	(None, 64)	4,160
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 3)	195

Total params: 4,515 (17.64 KB)
Trainable params: 4,515 (17.64 KB)
Non-trainable params: 0 (0.00 B)

```
history = model.fit(X_train, y_train, epochs=30, batch_size=16, validation_split=0.2)
```

```
Epoch 1/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 3s 34ms/step - accuracy: 0.4076 - loss: 1.0607 - val_accuracy: 0.8333 - val_loss: 0.8878
Epoch 2/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8263 - loss: 0.8430 - val_accuracy: 0.7778 - val_loss: 0.7068
Epoch 3/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8560 - loss: 0.6436 - val_accuracy: 0.7963 - val_loss: 0.5370
Epoch 4/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8855 - loss: 0.4534 - val_accuracy: 0.8519 - val_loss: 0.4030
Epoch 5/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8978 - loss: 0.3281 - val_accuracy: 0.8519 - val_loss: 0.3181
Epoch 6/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9137 - loss: 0.2586 - val_accuracy: 0.8889 - val_loss: 0.2433
Epoch 7/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9542 - loss: 0.2014 - val_accuracy: 0.9259 - val_loss: 0.1922
Epoch 8/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.9895 - loss: 0.1149 - val_accuracy: 0.9444 - val_loss: 0.1510
Epoch 9/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9826 - loss: 0.1235 - val_accuracy: 0.9815 - val_loss: 0.1160
Epoch 10/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9788 - loss: 0.1037 - val_accuracy: 0.9815 - val_loss: 0.0958
Epoch 11/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9987 - loss: 0.0573 - val_accuracy: 0.9815 - val_loss: 0.0822
Epoch 12/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9764 - loss: 0.0651 - val_accuracy: 0.9815 - val_loss: 0.0700
Epoch 13/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9880 - loss: 0.0525 - val_accuracy: 0.9815 - val_loss: 0.0571
Epoch 14/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9974 - loss: 0.0432 - val_accuracy: 0.9815 - val_loss: 0.0472
Epoch 15/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9903 - loss: 0.0484 - val_accuracy: 0.9815 - val_loss: 0.0427
Epoch 16/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9903 - loss: 0.0381 - val_accuracy: 0.9815 - val_loss: 0.0385
Epoch 17/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9948 - loss: 0.0315 - val_accuracy: 0.9815 - val_loss: 0.0404
Epoch 18/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9956 - loss: 0.0235 - val_accuracy: 0.9815 - val_loss: 0.0353
Epoch 19/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9948 - loss: 0.0260 - val_accuracy: 0.9815 - val_loss: 0.0302
Epoch 20/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 1.0000 - loss: 0.0159 - val_accuracy: 1.0000 - val_loss: 0.0279
Epoch 21/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 1.0000 - loss: 0.0135 - val_accuracy: 0.9815 - val_loss: 0.0274
Epoch 22/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9991 - loss: 0.0125 - val_accuracy: 0.9815 - val_loss: 0.0273
Epoch 23/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 1.0000 - loss: 0.0142 - val_accuracy: 0.9815 - val_loss: 0.0251
Epoch 24/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 1.0000 - loss: 0.0180 - val_accuracy: 0.9815 - val_loss: 0.0237
Epoch 25/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 1.0000 - loss: 0.0122 - val_accuracy: 0.9815 - val_loss: 0.0236
Epoch 26/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9956 - loss: 0.0134 - val_accuracy: 0.9815 - val_loss: 0.0225
Epoch 27/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 1.0000 - loss: 0.0124 - val_accuracy: 1.0000 - val_loss: 0.0201
Epoch 28/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 1.0000 - loss: 0.0076 - val_accuracy: 1.0000 - val_loss: 0.0200
Epoch 29/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9948 - loss: 0.0164 - val_accuracy: 1.0000 - val_loss: 0.0188
```

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy:.2f}")
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - accuracy: 1.0000 - loss: 0.0037
Test Accuracy: 1.00
```

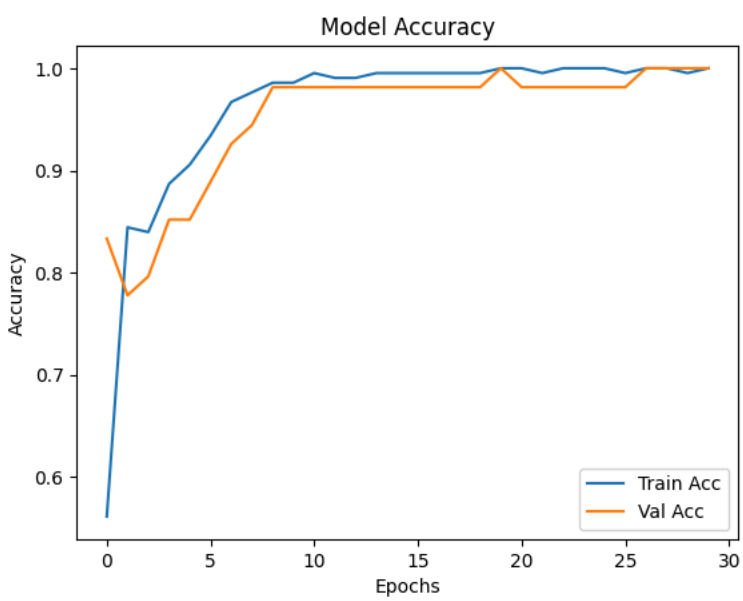
```
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
```

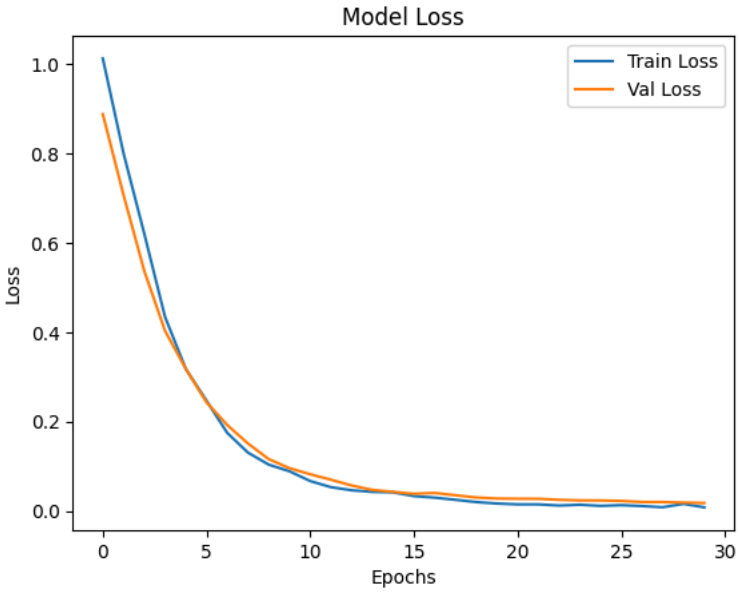
```
print("\nClassification Report:\n")
print(classification_report(y_true, y_pred_classes, target_names=le_species.classes_))
print("\nConfusion Matrix:\n")
print(confusion_matrix(y_true, y_pred_classes))
```

```
Classification Report:

              precision    recall  f1-score   support

      Adelie       1.00      1.00      1.00        31
   Chinstrap       1.00      1.00      1.00        13
      Gentoo       1.00      1.00      1.00        23

    accuracy                           1.00        67
   macro avg       1.00      1.00      1.00        67
weighted avg       1.00      1.00      1.00        67


Confusion Matrix:

[[31  0  0]
 [ 0 13  0]
 [ 0  0 23]]
```

```
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```




```
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



Start coding or generate with AI.