LAB Assignment No. 9

Convolution Neural Network

Open Ended LAB

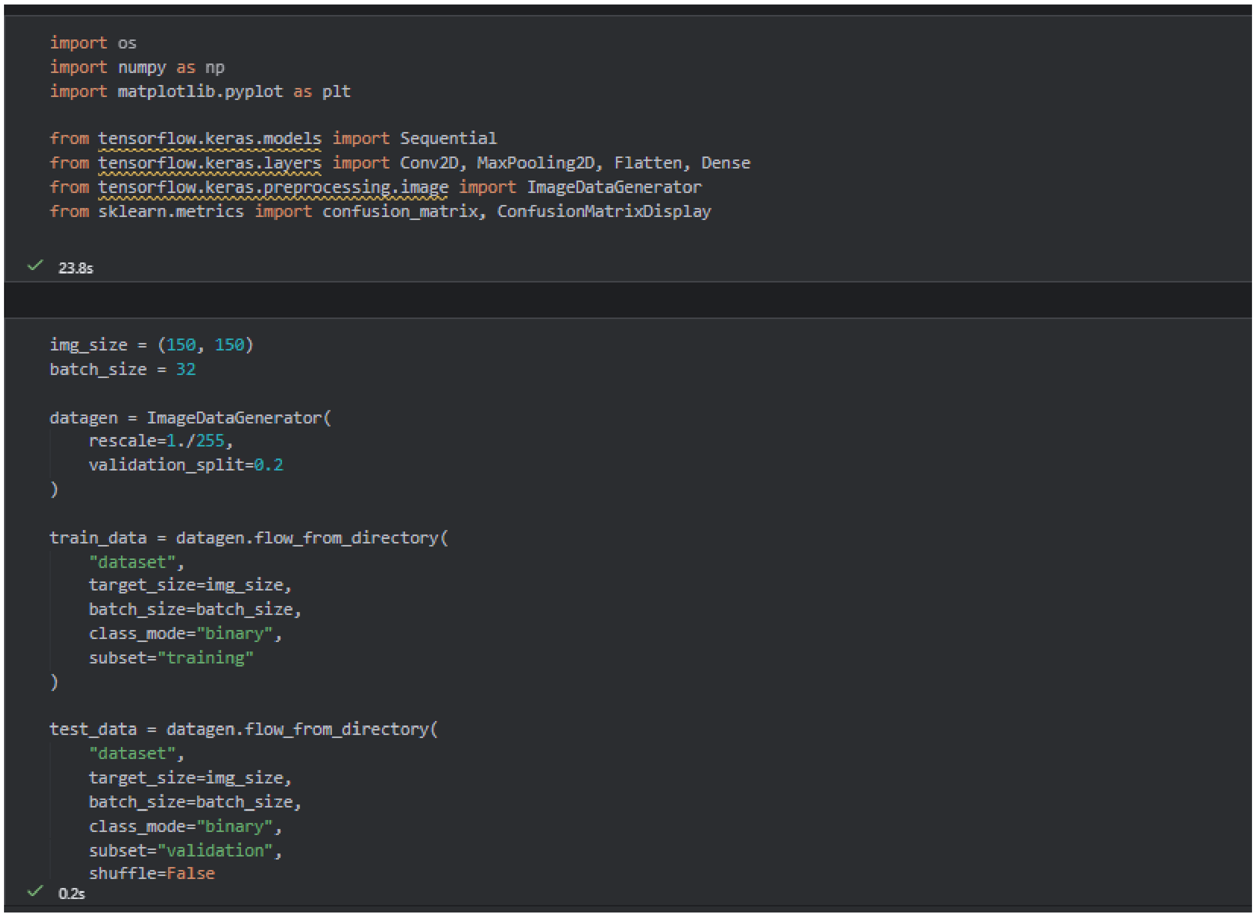
Build a Convolutional Neural Network (CNN) using Python and TensorFlow/Keras to classify images of Cats and Dogs.

Instructions:

1. Collect or download a dataset containing:
    - **500 images of Cats**
    - **500 images of Dogs**
2. Organize the dataset as:

```
dataset/
          cats/
           dogs/
```

3. Write Python code to:
    - Load and preprocess images (resize to 150×150)
    - Split into training (80%) and testing (20%)
    - Build a CNN model
    - Train the model for 10–20 epochs
    - Plot training & validation accuracy and loss
    - Evaluate model performance using a confusion matrix
    - Predict whether a new input image is Cat or Dog
4. At the end of the program, show the prediction result for a test image:
    - "This image is a CAT"
    - or
      "This image is a DOG"
5. Submit your Python code, dataset, graphs, and output screenshots.

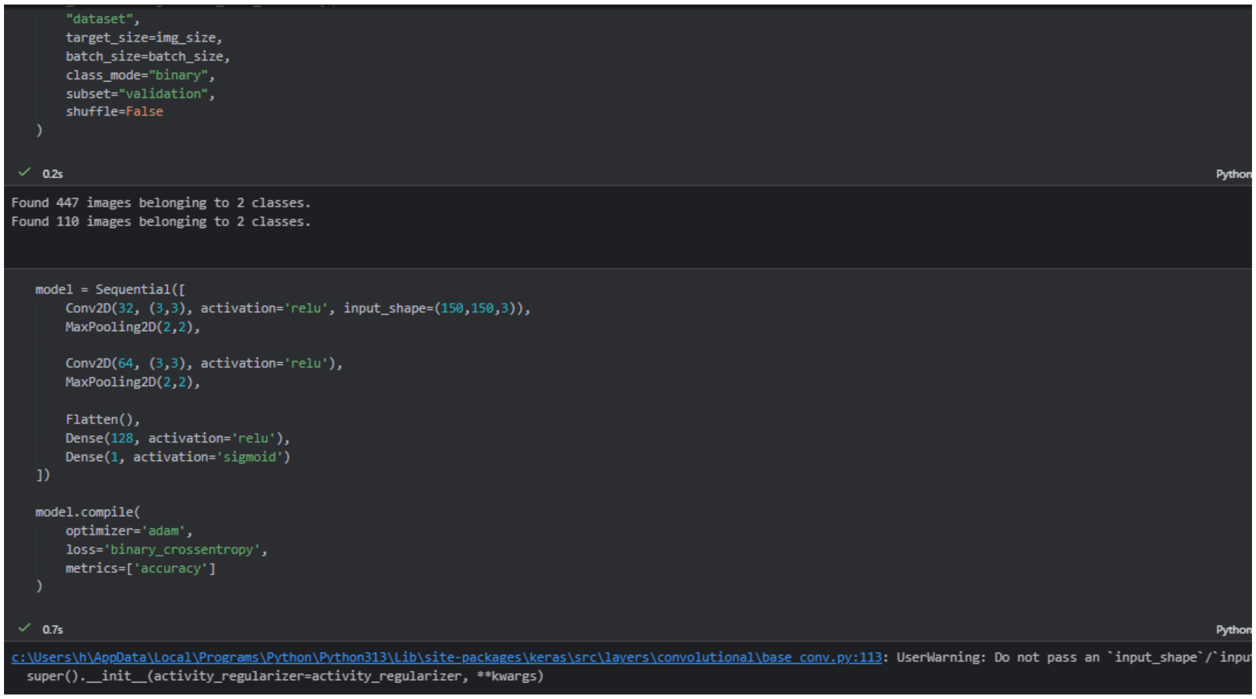


```python
import os
import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

✓ 23.8s

```python
img_size = (150, 150)
batch_size = 32

datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
)

train_data = datagen.flow_from_directory(
    "dataset",
    target_size=img_size,
    batch_size=batch_size,
    class_mode="binary",
    subset="training"
)

test_data = datagen.flow_from_directory(
    "dataset",
    target_size=img_size,
    batch_size=batch_size,
    class_mode="binary",
    subset="validation",
    shuffle=False
```

✓ 0.2s



```python
    "dataset",
    target_size=img_size,
    batch_size=batch_size,
    class_mode="binary",
    subset="validation",
    shuffle=False
)
```

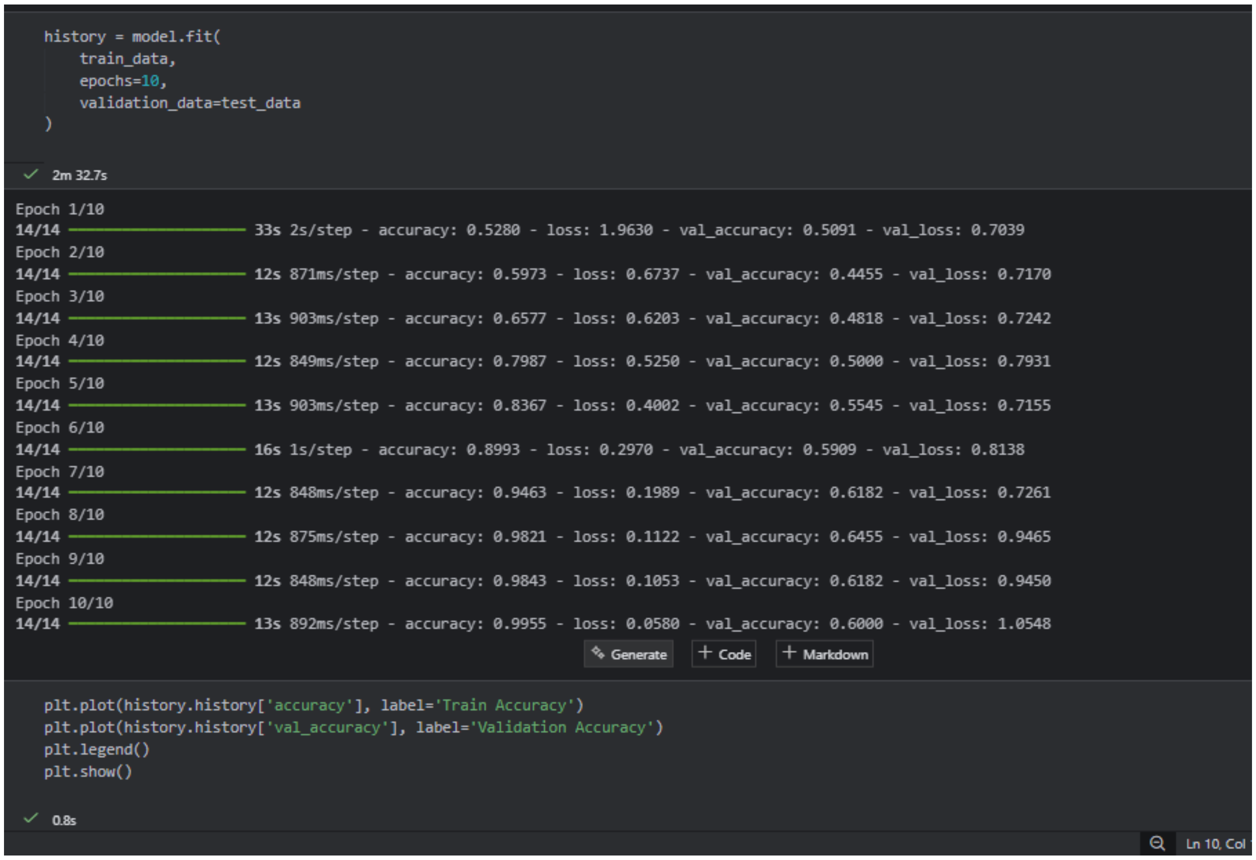
✓ 0.2s Python

```
Found 447 images belonging to 2 classes.
Found 110 images belonging to 2 classes.
```

```python
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(150,150,3)),
    MaxPooling2D(2,2),

    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),

    Flatten(),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')
])

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

✓ 0.7s Python

```
c:\Users\h\AppData\Local\Programs\Python\Python313\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:113: UserWarning: Do not pass an `input_shape`/`inpu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```



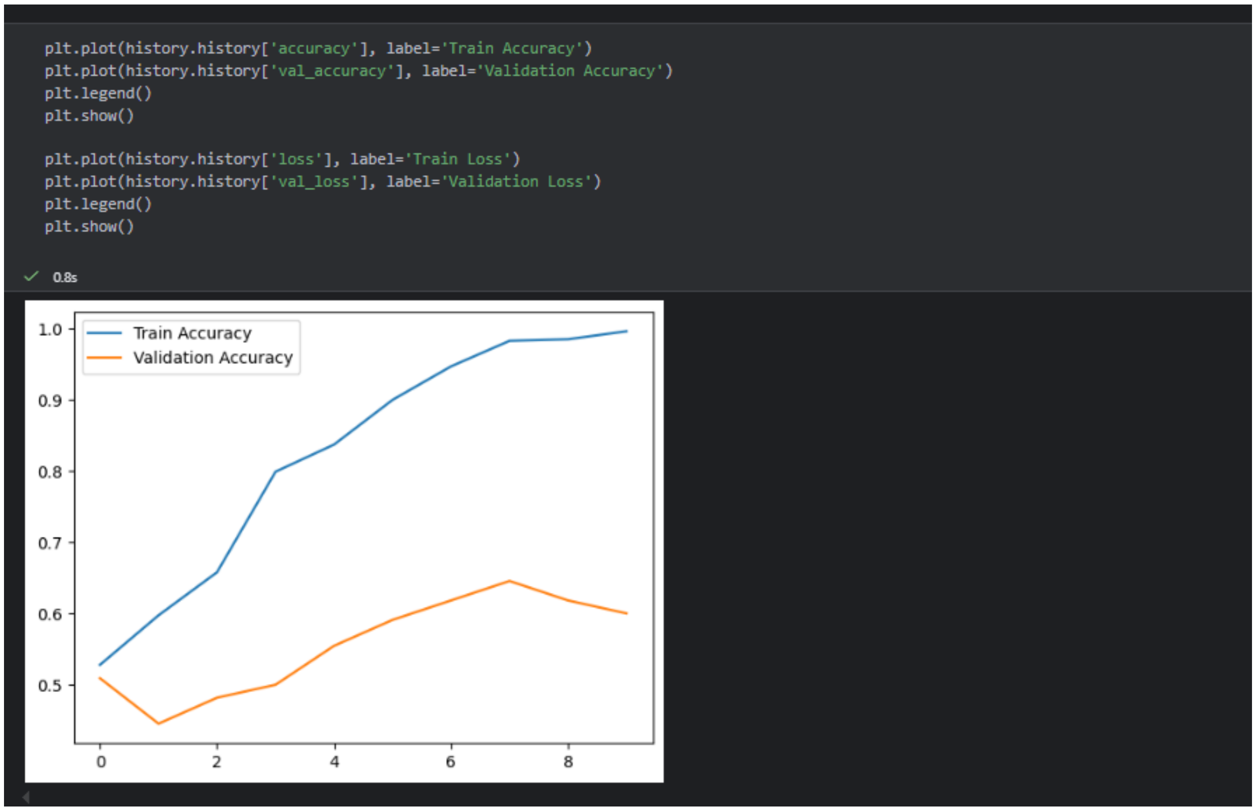
```
history = model.fit(
    train_data,
    epochs=10,
    validation_data=test_data
)
```

✓ 2m 32.7s

```
Epoch 1/10
14/14 ━━━━━━━━━━━━━━━━━━━━ 33s 2s/step - accuracy: 0.5280 - loss: 1.9630 - val_accuracy: 0.5091 - val_loss: 0.7039
Epoch 2/10
14/14 ━━━━━━━━━━━━━━━━━━━━ 12s 871ms/step - accuracy: 0.5973 - loss: 0.6737 - val_accuracy: 0.4455 - val_loss: 0.7170
Epoch 3/10
14/14 ━━━━━━━━━━━━━━━━━━━━ 13s 903ms/step - accuracy: 0.6577 - loss: 0.6203 - val_accuracy: 0.4818 - val_loss: 0.7242
Epoch 4/10
14/14 ━━━━━━━━━━━━━━━━━━━━ 12s 849ms/step - accuracy: 0.7987 - loss: 0.5250 - val_accuracy: 0.5000 - val_loss: 0.7931
Epoch 5/10
14/14 ━━━━━━━━━━━━━━━━━━━━ 13s 903ms/step - accuracy: 0.8367 - loss: 0.4002 - val_accuracy: 0.5545 - val_loss: 0.7155
Epoch 6/10
14/14 ━━━━━━━━━━━━━━━━━━━━ 16s 1s/step - accuracy: 0.8993 - loss: 0.2970 - val_accuracy: 0.5909 - val_loss: 0.8138
Epoch 7/10
14/14 ━━━━━━━━━━━━━━━━━━━━ 12s 848ms/step - accuracy: 0.9463 - loss: 0.1989 - val_accuracy: 0.6182 - val_loss: 0.7261
Epoch 8/10
14/14 ━━━━━━━━━━━━━━━━━━━━ 12s 875ms/step - accuracy: 0.9821 - loss: 0.1122 - val_accuracy: 0.6455 - val_loss: 0.9465
Epoch 9/10
14/14 ━━━━━━━━━━━━━━━━━━━━ 12s 848ms/step - accuracy: 0.9843 - loss: 0.1053 - val_accuracy: 0.6182 - val_loss: 0.9450
Epoch 10/10
14/14 ━━━━━━━━━━━━━━━━━━━━ 13s 892ms/step - accuracy: 0.9955 - loss: 0.0580 - val_accuracy: 0.6000 - val_loss: 1.0548
```

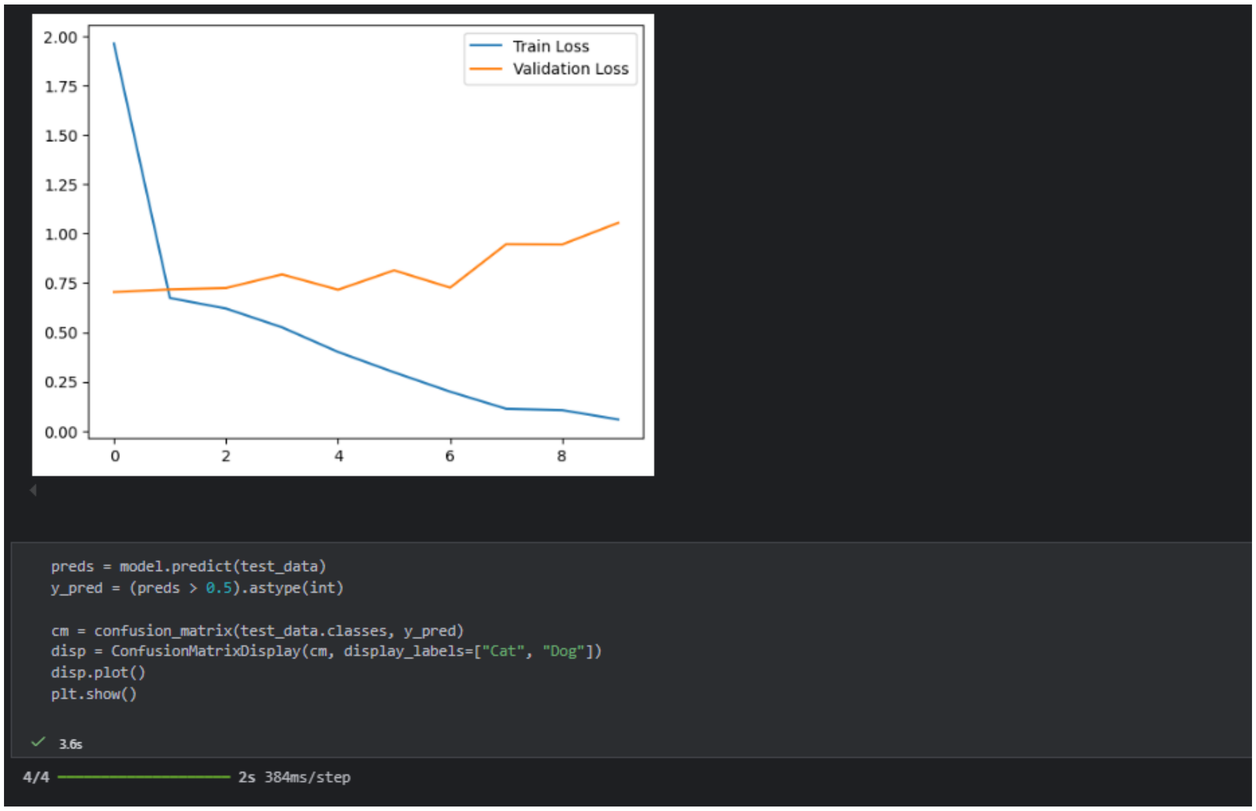
```
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.show()
```

✓ 0.8s



```
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.show()
```

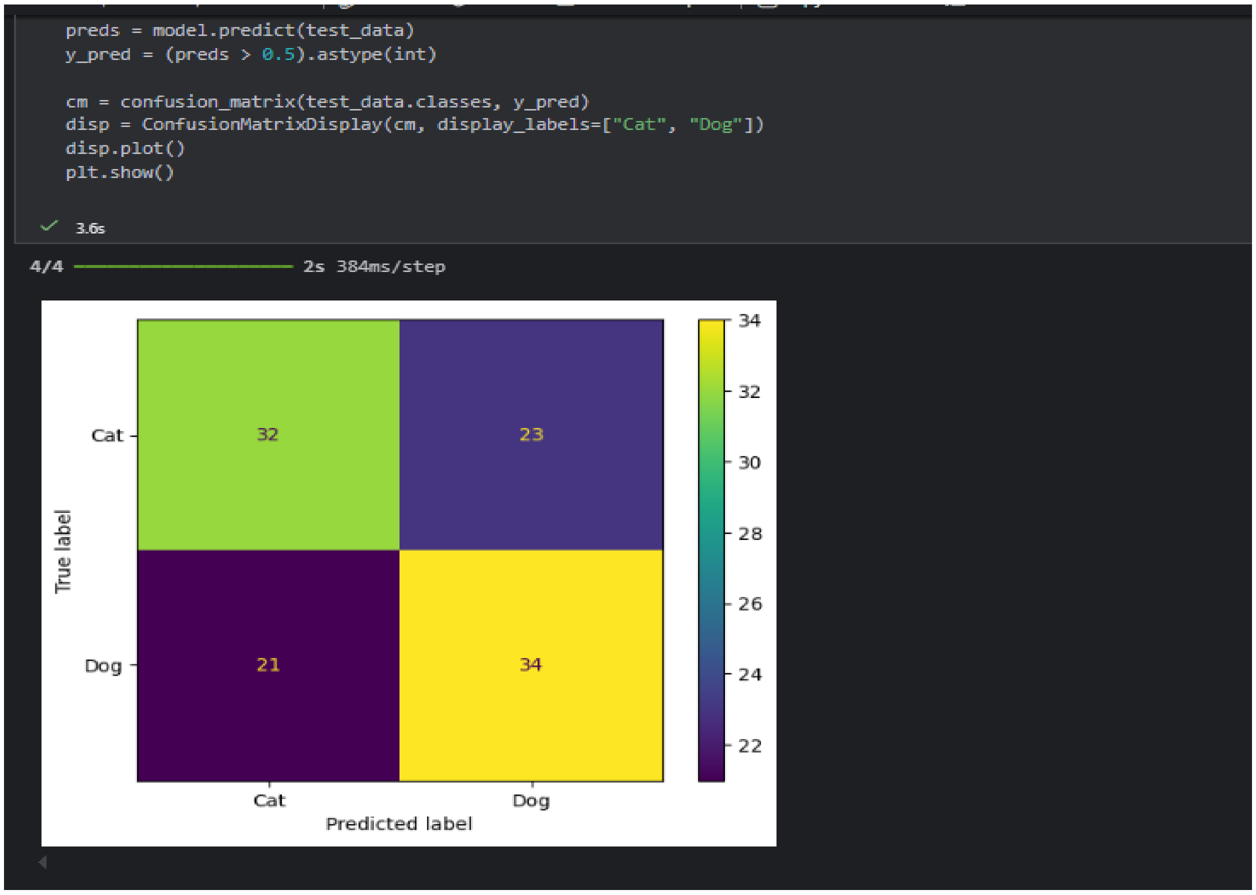
✓ 0.8s



```
preds = model.predict(test_data)
y_pred = (preds > 0.5).astype(int)

cm = confusion_matrix(test_data.classes, y_pred)
disp = ConfusionMatrixDisplay(cm, display_labels=["Cat", "Dog"])
disp.plot()
plt.show()
```

✓ 3.6s

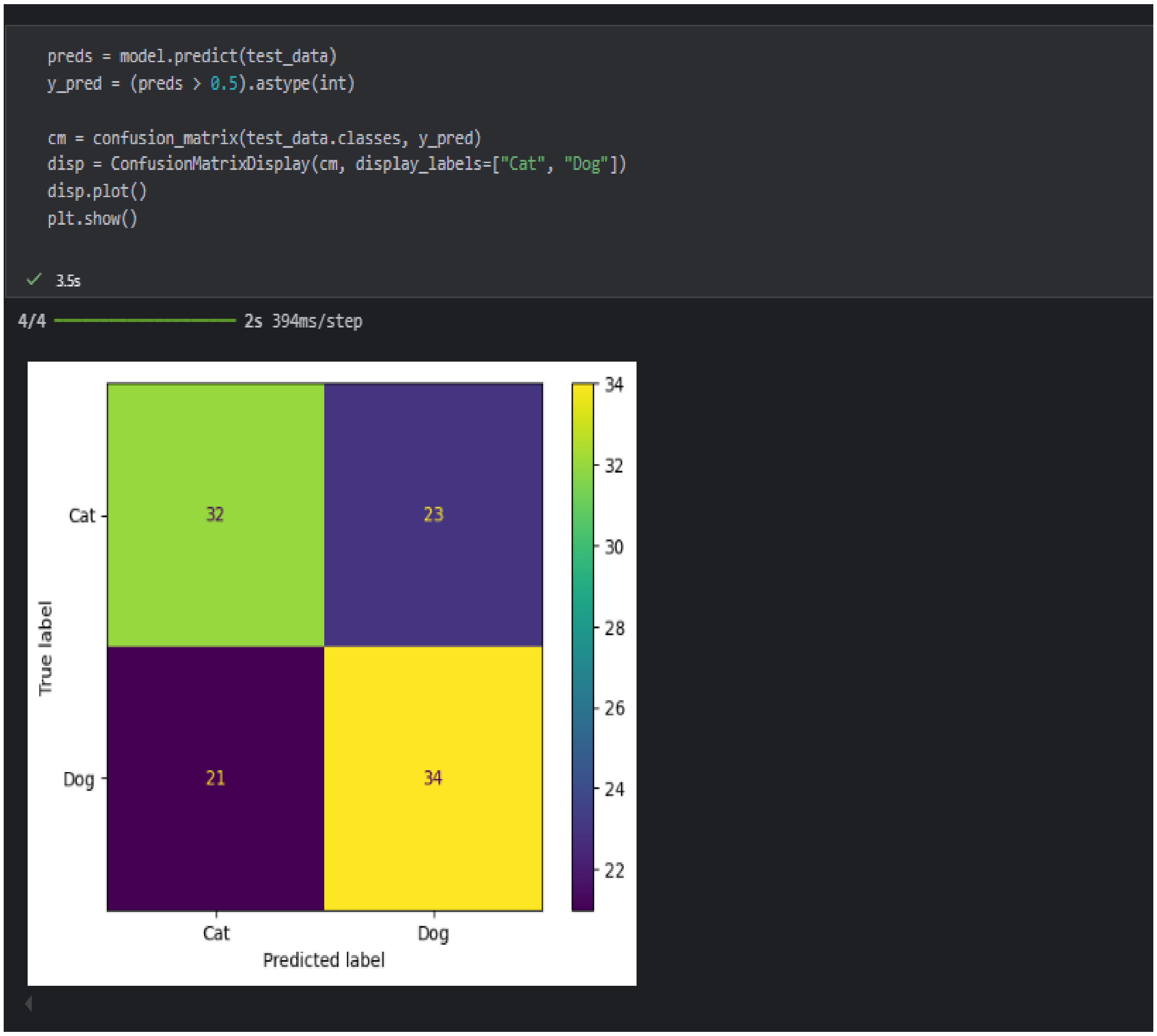
```
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 384ms/step
```



```
preds = model.predict(test_data)
y_pred = (preds > 0.5).astype(int)

cm = confusion_matrix(test_data.classes, y_pred)
disp = ConfusionMatrixDisplay(cm, display_labels=["Cat", "Dog"])
disp.plot()
plt.show()
```

✓ 3.6s

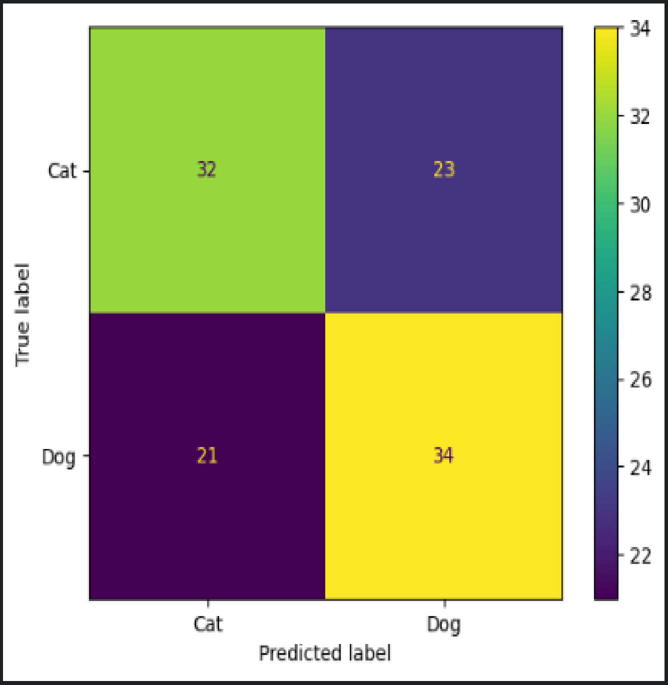
```
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 384ms/step
```



```
preds = model.predict(test_data)
y_pred = (preds > 0.5).astype(int)

cm = confusion_matrix(test_data.classes, y_pred)
disp = ConfusionMatrixDisplay(cm, display_labels=["Cat", "Dog"])
disp.plot()
plt.show()
```

✓ 3.5s

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 394ms/step
```


True label
Cat
Dog
Cat
Dog
Predicted label