```
"""
This project is focused on building a deep learning model to detect sickle cells from blood smear images.
The main idea was to take medical image data and use CNNs to classify whether sickle cells are present or not.
I worked with a dataset from Uganda that includes both labeled positive images and clear negative images.
This is part of my learning journey in machine learning and applying it to real-world health problems.
Everything is done in Google Colab using TensorFlow and Keras.
"""
```

```
import os
import zipfile
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!cp -r "/content/drive/MyDrive/Colab Notebooks/sickle_cell_dataset" ./sickle_cell_dataset
```

Double-click (or enter) to edit

```
import shutil
from sklearn.model_selection import train_test_split
import glob

# Paths
positive = glob.glob('/content/sickle_cell_dataset/Labelled/*.jpg')
negative = glob.glob('/content/sickle_cell_dataset/Clear/*.jpg')

# Split
pos_train, pos_val = train_test_split(positive, test_size=0.2, random_state=42)
neg_train, neg_val = train_test_split(negative, test_size=0.2, random_state=42)

# Create directories
os.makedirs('dataset/train/positive', exist_ok=True)
os.makedirs('dataset/train/negative', exist_ok=True)
os.makedirs('dataset/val/positive', exist_ok=True)
os.makedirs('dataset/val/negative', exist_ok=True)

# Move files
for f in pos_train:
    shutil.copy(f, 'dataset/train/positive')
for f in pos_val:
    shutil.copy(f, 'dataset/val/positive')
```

```
for f in neg_train:
    shutil.copy(f, 'dataset/train/negative')
for f in neg_val:
    shutil.copy(f, 'dataset/val/negative')
```

```
train_gen = ImageDataGenerator(rescale=1./255)
val_gen = ImageDataGenerator(rescale=1./255)

train_data = train_gen.flow_from_directory('dataset/train',
                                           target_size=(150, 150),
                                           batch_size=32,
                                           class_mode='binary')

val_data = val_gen.flow_from_directory('dataset/val',
                                       target_size=(150, 150),
                                       batch_size=32,
                                       class_mode='binary')
```

```
Found 454 images belonging to 2 classes.
Found 115 images belonging to 2 classes.
```

```
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(),
    Dropout(0.5),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')
])

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

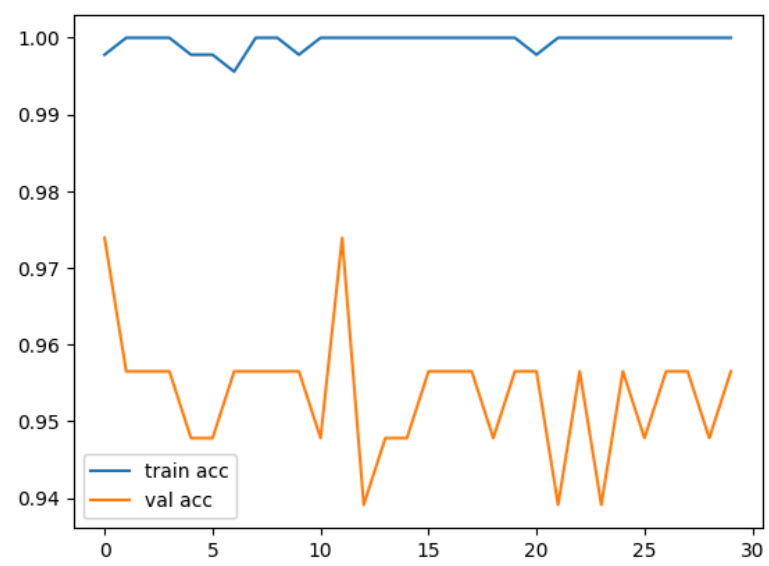
```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
history = model.fit(train_data, validation_data=val_data, epochs=30)
```

```
Epoch 1/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 229ms/step - accuracy: 0.9962 - loss: 0.0067 - val_accuracy: 0.9739 - val_loss: 0.1207
Epoch 2/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 217ms/step - accuracy: 1.0000 - loss: 0.0028 - val_accuracy: 0.9565 - val_loss: 0.1280
Epoch 3/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 5s 207ms/step - accuracy: 1.0000 - loss: 0.0029 - val_accuracy: 0.9565 - val_loss: 0.1129
Epoch 4/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 200ms/step - accuracy: 1.0000 - loss: 0.0027 - val_accuracy: 0.9565 - val_loss: 0.1575
Epoch 5/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 201ms/step - accuracy: 0.9972 - loss: 0.0086 - val_accuracy: 0.9478 - val_loss: 0.1495
Epoch 6/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 4s 286ms/step - accuracy: 0.9996 - loss: 0.0110 - val_accuracy: 0.9478 - val_loss: 0.1190
```

```
Epoch 7/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 5s 242ms/step - accuracy: 0.9966 - loss: 0.0148 - val_accuracy: 0.9565 - val_loss: 0.1040
Epoch 8/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 201ms/step - accuracy: 1.0000 - loss: 0.0071 - val_accuracy: 0.9565 - val_loss: 0.1303
Epoch 9/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 234ms/step - accuracy: 1.0000 - loss: 0.0049 - val_accuracy: 0.9565 - val_loss: 0.0999
Epoch 10/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 5s 194ms/step - accuracy: 0.9992 - loss: 0.0031 - val_accuracy: 0.9565 - val_loss: 0.1230
Epoch 11/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 205ms/step - accuracy: 1.0000 - loss: 0.0033 - val_accuracy: 0.9478 - val_loss: 0.2177
Epoch 12/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 199ms/step - accuracy: 1.0000 - loss: 0.0035 - val_accuracy: 0.9739 - val_loss: 0.0771
Epoch 13/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 4s 275ms/step - accuracy: 1.0000 - loss: 0.0024 - val_accuracy: 0.9391 - val_loss: 0.2611
Epoch 14/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 217ms/step - accuracy: 1.0000 - loss: 7.2987e-04 - val_accuracy: 0.9478 - val_loss: 0.1775
Epoch 15/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 206ms/step - accuracy: 1.0000 - loss: 0.0034 - val_accuracy: 0.9478 - val_loss: 0.1975
Epoch 16/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 198ms/step - accuracy: 1.0000 - loss: 0.0012 - val_accuracy: 0.9565 - val_loss: 0.1938
Epoch 17/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 4s 269ms/step - accuracy: 1.0000 - loss: 0.0015 - val_accuracy: 0.9565 - val_loss: 0.1604
Epoch 18/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 205ms/step - accuracy: 1.0000 - loss: 0.0011 - val_accuracy: 0.9565 - val_loss: 0.2182
Epoch 19/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 204ms/step - accuracy: 1.0000 - loss: 0.0025 - val_accuracy: 0.9478 - val_loss: 0.2110
Epoch 20/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 6s 249ms/step - accuracy: 1.0000 - loss: 0.0020 - val_accuracy: 0.9565 - val_loss: 0.1899
Epoch 21/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 5s 212ms/step - accuracy: 0.9980 - loss: 0.0027 - val_accuracy: 0.9565 - val_loss: 0.1418
Epoch 22/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 203ms/step - accuracy: 1.0000 - loss: 0.0018 - val_accuracy: 0.9391 - val_loss: 0.3240
Epoch 23/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 206ms/step - accuracy: 1.0000 - loss: 0.0026 - val_accuracy: 0.9565 - val_loss: 0.1208
Epoch 24/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 4s 258ms/step - accuracy: 1.0000 - loss: 0.0017 - val_accuracy: 0.9391 - val_loss: 0.3388
Epoch 25/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 198ms/step - accuracy: 1.0000 - loss: 9.2682e-04 - val_accuracy: 0.9565 - val_loss: 0.1907
Epoch 26/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 203ms/step - accuracy: 1.0000 - loss: 7.3622e-04 - val_accuracy: 0.9478 - val_loss: 0.2292
Epoch 27/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 3s 203ms/step - accuracy: 1.0000 - loss: 6.2807e-04 - val_accuracy: 0.9565 - val_loss: 0.1904
Epoch 28/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 4s 264ms/step - accuracy: 1.0000 - loss: 4.8047e-04 - val_accuracy: 0.9565 - val_loss: 0.1821
Epoch 29/30
```

```
plt.plot(history.history['accuracy'], label='train acc')
plt.plot(history.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
```




```
final_train_acc = history.history['accuracy'][-1]
final_val_acc = history.history['val_accuracy'][-1]

print(f"Final Training Accuracy: {final_train_acc:.2f}")
print(f"Final Validation Accuracy: {final_val_acc:.2f}")
```

```
Final Training Accuracy: 1.00
Final Validation Accuracy: 0.96
```

```
 """
 Sickle Cell Classification with CNN

In this project, I built a deep learning model to classify sickle cell blood images using a CNN.
The dataset I used includes labeled positive images, unlabeled positive images, and clear negative images
collected from the Teso region in Uganda. I excluded the unclear images to focus on higher quality samples.
The goal of this project was to detect sickle cells from microscopic images using image classification.

I trained the model using a convolutional neural network (CNN) with several convolutional and max pooling layers,
followed by dense layers for classification. I used ImageDataGenerator to rescale and feed the image data into the model.
I trained the model over 30 epochs and observed that as the number of epochs increased,
the training accuracy improved and eventually reached 1.00. However, the validation accuracy started to decrease
after a certain point, which is a sign of overfitting.

The final training accuracy was 1.00, and the final validation accuracy was 0.96.
Based on this, I think the model performs well considering the size of the dataset.
Since the dataset is not that large, this kind of pattern is expected, and I believe this model still gives solid performance.
If the dataset was larger or more balanced, the generalization could be even better.
I also plan to experiment more with data augmentation, dropout, and early stopping to see if I can reduce overfitting
and improve validation accuracy further.

This project shows a real-world application of machine learning in medical diagnostics,
```

```
and I’m planning to keep building on it.
"""
```

```
'\nSickle Cell Classification with CNN\n\nIn this project, I built a deep learning model to classify sickle cell blood images using a CNN.\nThe dataset I used includes labele
d positive images, unlabeled positive images, and clear negative images\ncollected from the Teso region in Uganda. I excluded the unclear images to focus on higher quality sa
mples.\nThe goal of this project was to detect sickle cells from microscopic images using image classification.\n\nI trained the model using a convolutional neural network (C
NN) with several convolutional and max pooling layers,\nfollowed by dense layers for classification. I used ImageDataGenerator to rescale and feed the image data into the mod
el.\nI trained the model over 30 epochs and observed that as the number of epochs increased,\nthe training accuracy improved and eventually reached 1.00. However, the validat
ion accuracy started to decrease\nafter a certain point, which is a sign of overfitting.\n\nThe final training accuracy was 1.00, a'
```

```
Start coding or generate with AI.
```