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

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

```
import os
import cv2
import numpy as np
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
from sklearn.model_selection import train_test_split


# Defining the directory and categories
dataset_dir = "/content/drive/MyDrive/sih_road_dataset"
categories = ["good", "poor", "satisfactory", "very_poor"]

# Loading images and labels
images = []
labels = []
for idx, category in enumerate(categories):
    path = os.path.join(dataset_dir, category)
    for img_name in os.listdir(path):
        img_path = os.path.join(path, img_name)
        img = cv2.imread(img_path)
        if img is not None:
            img_resized = cv2.resize(img, (224, 224))
            images.append(img_resized)
            labels.append(idx)

# Converting to numpy arrays, normalize, and one-hot encode labels
images = np.array(images) / 255.0
labels = to_categorical(np.array(labels), num_classes=4)

# Splitting into training and testing datasets
x_train, x_test, y_train, y_test = train_test_split(images, labels, test_size=0.2)



# CNN model
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(4, activation='softmax')
])

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

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
# Using ImageDataGenerator for data augmentation
datagen = ImageDataGenerator(
    shear_range=0.1,
    zoom_range=0.1,
    width_shift_range=0.2,
    height_shift_range=0.2
)

datagen.fit(x_train)

# Training the model
model.fit(datagen.flow(x_train, y_train, batch_size=32), epochs=10, validation_data=(x_test, y_test))
```

```
Epoch 1/10
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` cl
  self._warn_if_super_not_called()
52/52 ━━━━━━━━━━━━━━━━━━━━ 181s 3s/step - accuracy: 0.5620 - loss: 3.8259 - val_accuracy: 0.8265 - val_loss: 0.4604
Epoch 2/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 174s 3s/step - accuracy: 0.8480 - loss: 0.4174 - val_accuracy: 0.9108 - val_loss: 0.2280
Epoch 3/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 206s 3s/step - accuracy: 0.8844 - loss: 0.3206 - val_accuracy: 0.9253 - val_loss: 0.2106
```

```
Epoch 4/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 193s 4s/step - accuracy: 0.8864 - loss: 0.3198 - val_accuracy: 0.9253 - val_loss: 0.2265
Epoch 5/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 187s 3s/step - accuracy: 0.8908 - loss: 0.2927 - val_accuracy: 0.9349 - val_loss: 0.1953
Epoch 6/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 177s 3s/step - accuracy: 0.9010 - loss: 0.2446 - val_accuracy: 0.9157 - val_loss: 0.2376
Epoch 7/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 174s 3s/step - accuracy: 0.9104 - loss: 0.2481 - val_accuracy: 0.9133 - val_loss: 0.2588
Epoch 8/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 174s 3s/step - accuracy: 0.9154 - loss: 0.2532 - val_accuracy: 0.8723 - val_loss: 0.4007
Epoch 9/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 178s 3s/step - accuracy: 0.9112 - loss: 0.2337 - val_accuracy: 0.8819 - val_loss: 0.3380
Epoch 10/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 174s 3s/step - accuracy: 0.8830 - loss: 0.2929 - val_accuracy: 0.9181 - val_loss: 0.2390
<keras.src.callbacks.history.History at 0x7d741c54a830>
```

```
# Prediction function
def predict_image(image_path):
    img = cv2.imread(image_path)
    img_resized = cv2.resize(img, (224, 224)) / 255.0
    img_expanded = np.expand_dims(img_resized, axis=0)
    prediction = model.predict(img_expanded)
    print(f"Predicted category: {categories[np.argmax(prediction)]}")
```

```
predict_image("/content/drive/MyDrive/sih_road_dataset/very_poor/verypoor_008.jpg")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step
Predicted category: very_poor
```

```
predict_image("/content/drive/MyDrive/sih_road_dataset/satisfactory/satis_060.jpg")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
Predicted category: poor
```

```
predict_image("/content/drive/MyDrive/sih_road_dataset/very_poor/verypoor_047.jpg")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
Predicted category: very_poor
```

```
predict_image("/content/drive/MyDrive/sih_road_dataset/very_poor/verypoor_039.jpg")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
Predicted category: poor
```

```
predict_image("/content/drive/MyDrive/sih_road_dataset/satisfactory/satis_007.jpg")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
Predicted category: satisfactory
```

```
predict_image("/content/drive/MyDrive/sih_road_dataset/good/good_029.JPG")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step
Predicted category: good
```

```
predict_image("/content/drive/MyDrive/sih_road_dataset/good/good_044.JPG")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
Predicted category: good
```

```
predict_image("/content/road_dmg.jpg")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
Predicted category: satisfactory
```