sleep_detector.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유 상원

+ 코드 + 텍스트

RAM 디스크

파일

..
.config
.ipynb_checkpoints
drive
.Trash-0
.file-revisions-by-id
.shortcut-targets-by-id
MyDrive
Colab Notebooks
dataset
sleep_left
sleep_right
wake
sample_data
image_classification_model.h5

디스크 81.67 GB 사용 가능

[1]
```
import os
import cv2
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

[2]
```
dataset_path = "/content/drive/MyDrive/dataset"
```

[3]
```
def load_data(dataset_path):
    images = []
    labels = []
    classes = os.listdir(dataset_path)
    for i, class_name in enumerate(classes):
        class_path = os.path.join(dataset_path, class_name)
        for image_name in os.listdir(class_path):
            image_path = os.path.join(class_path, image_name)
            image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
            image = cv2.resize(image, (32, 32))
            images.append(image)
            labels.append(i)
    images = np.array(images, dtype=np.float32) / 255.0
    labels = np.array(labels)
    return images, labels
```

[4]
```
images, labels = load_data(dataset_path)
```

[5]
```
# 데이터 증강
datagen = ImageDataGenerator(
    rotation_range=1,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True
)

augmented_images = []
augmented_labels = []

for i in range(images.shape[0]):
    image = images[i]
    image = np.expand_dims(image, axis=-1)
    image = np.expand_dims(image, axis=0)
    label = labels[i]
    for _ in range(10):  # 각 이미지당 10장의 추가 생성
        for x_augmented, y_augmented in datagen.flow(image, [label], batch_size=1):
            augmented_images.append(np.squeeze(x_augmented))
            augmented_labels.append(y_augmented[0])
            break

augmented_images = np.array(augmented_images)
augmented_labels = np.array(augmented_labels)
```

[6]
```
# 데이터 분할
x_train, x_test, y_train, y_test = train_test_split(augmented_images, augmented_labels, test_size=0.2, random_state=42)
```

```
model = models.Sequential([
    layers.Conv2D(64, (3, 3), activation='relu', input_shape=(32, 32, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(3, activation='softmax')  # 클래스 수만큼의 출력
])
```

[8]
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 64)        640

 max_pooling2d (MaxPooling2  (None, 15, 15, 64)        0
 D)

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        36928

 max_pooling2d_1 (MaxPoolin  (None, 6, 6, 64)          0
 g2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 128)         73856

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 128)               262272

 dense_1 (Dense)             (None, 3)                 387

=================================================================
Total params: 374083 (1.43 MB)
Trainable params: 374083 (1.43 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

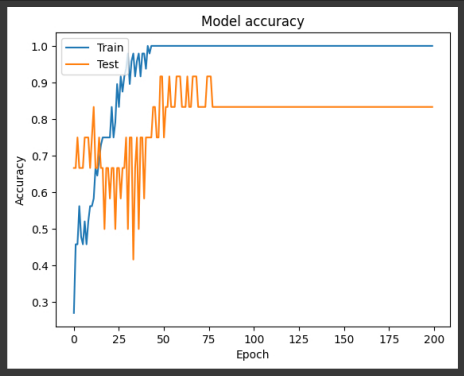
[9]
```
# 모델 컴파일
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

[10]
```
# 모델 학습
history = model.fit(x_train, y_train, epochs=200, batch_size=32, validation_data=(x_test, y_test))

print(history.history)

import matplotlib.pyplot as plt
```

```
Epoch 173/200
2/2 [==============================] - 0s 84ms/step - loss: 4.9172e-04 - accuracy: 1.0000 - val_loss: 1.0213 - val_accuracy: 0.8333
Epoch 174/200
2/2 [==============================] - 0s 69ms/step - loss: 4.8615e-04 - accuracy: 1.0000 - val_loss: 1.0237 - val_accuracy: 0.8333
Epoch 175/200
2/2 [==============================] - 0s 82ms/step - loss: 4.8034e-04 - accuracy: 1.0000 - val_loss: 1.0211 - val_accuracy: 0.8333
Epoch 176/200
2/2 [==============================] - 0s 67ms/step - loss: 4.7317e-04 - accuracy: 1.0000 - val_loss: 1.0179 - val_accuracy: 0.8333
Epoch 177/200
2/2 [==============================] - 0s 70ms/step - loss: 4.6764e-04 - accuracy: 1.0000 - val_loss: 1.0147 - val_accuracy: 0.8333
Epoch 178/200
2/2 [==============================] - 0s 83ms/step - loss: 4.6238e-04 - accuracy: 1.0000 - val_loss: 1.0123 - val_accuracy: 0.8333
Epoch 179/200
2/2 [==============================] - 0s 71ms/step - loss: 4.5792e-04 - accuracy: 1.0000 - val_loss: 1.0127 - val_accuracy: 0.8333
Epoch 180/200
```

```
2/2 [==============================] - 0s 70ms/step - loss: 4.5180e-04 - accuracy: 1.0000 - val_loss: 1.0177 - val_accuracy: 0.8333
Epoch 181/200
2/2 [==============================] - 0s 70ms/step - loss: 4.4392e-04 - accuracy: 1.0000 - val_loss: 1.0274 - val_accuracy: 0.8333
Epoch 182/200
2/2 [==============================] - 0s 85ms/step - loss: 4.5226e-04 - accuracy: 1.0000 - val_loss: 1.0367 - val_accuracy: 0.8333
Epoch 183/200
2/2 [==============================] - 0s 81ms/step - loss: 4.4439e-04 - accuracy: 1.0000 - val_loss: 1.0325 - val_accuracy: 0.8333
Epoch 184/200
2/2 [==============================] - 0s 70ms/step - loss: 4.3415e-04 - accuracy: 1.0000 - val_loss: 1.0287 - val_accuracy: 0.8333
Epoch 185/200
2/2 [==============================] - 0s 84ms/step - loss: 4.2782e-04 - accuracy: 1.0000 - val_loss: 1.0221 - val_accuracy: 0.8333
Epoch 186/200
2/2 [==============================] - 0s 69ms/step - loss: 4.2019e-04 - accuracy: 1.0000 - val_loss: 1.0129 - val_accuracy: 0.8333
Epoch 187/200
2/2 [==============================] - 0s 71ms/step - loss: 4.1722e-04 - accuracy: 1.0000 - val_loss: 1.0029 - val_accuracy: 0.8333
Epoch 188/200
2/2 [==============================] - 0s 69ms/step - loss: 4.1259e-04 - accuracy: 1.0000 - val_loss: 1.0012 - val_accuracy: 0.8333
Epoch 189/200
2/2 [==============================] - 0s 71ms/step - loss: 4.1016e-04 - accuracy: 1.0000 - val_loss: 1.0031 - val_accuracy: 0.8333
Epoch 190/200
2/2 [==============================] - 0s 72ms/step - loss: 4.0866e-04 - accuracy: 1.0000 - val_loss: 1.0082 - val_accuracy: 0.8333
Epoch 191/200
2/2 [==============================] - 0s 70ms/step - loss: 4.0376e-04 - accuracy: 1.0000 - val_loss: 1.0206 - val_accuracy: 0.8333
Epoch 192/200
2/2 [==============================] - 0s 70ms/step - loss: 3.9491e-04 - accuracy: 1.0000 - val_loss: 1.0296 - val_accuracy: 0.8333
Epoch 193/200
2/2 [==============================] - 0s 77ms/step - loss: 3.8802e-04 - accuracy: 1.0000 - val_loss: 1.0345 - val_accuracy: 0.8333
Epoch 194/200
2/2 [==============================] - 0s 69ms/step - loss: 3.8294e-04 - accuracy: 1.0000 - val_loss: 1.0372 - val_accuracy: 0.8333
Epoch 195/200
2/2 [==============================] - 0s 72ms/step - loss: 3.8004e-04 - accuracy: 1.0000 - val_loss: 1.0396 - val_accuracy: 0.8333
Epoch 196/200
2/2 [==============================] - 0s 72ms/step - loss: 3.7535e-04 - accuracy: 1.0000 - val_loss: 1.0417 - val_accuracy: 0.8333
Epoch 197/200
2/2 [==============================] - 0s 88ms/step - loss: 3.7302e-04 - accuracy: 1.0000 - val_loss: 1.0421 - val_accuracy: 0.8333
Epoch 198/200
2/2 [==============================] - 0s 88ms/step - loss: 3.6726e-04 - accuracy: 1.0000 - val_loss: 1.0385 - val_accuracy: 0.8333
Epoch 199/200
2/2 [==============================] - 0s 84ms/step - loss: 3.6271e-04 - accuracy: 1.0000 - val_loss: 1.0352 - val_accuracy: 0.8333
Epoch 200/200
2/2 [==============================] - 0s 77ms/step - loss: 3.5972e-04 - accuracy: 1.0000 - val_loss: 1.0335 - val_accuracy: 0.8333
{'loss': [1.0949788093566895, 1.0430947542190552, 1.0328944492149353, 1.0266998124887793, 1.0090082883834839, 1.0019887685775757, 0.9765724539756775, 0.9762846827507019, 0.9306554794311523, 0.9114353060722351, 0.881
```

[11]
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



[12]
```
# 모델 저장
model.save("image_classification_model.h5")
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead
  saving_api.save_model(
```