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] 6초
```
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] 0초
```
dataset_path = "/content/drive/MyDrive/dataset"
```

[3] 0초
```
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] 0초
```
images, labels = load_data(dataset_path)
```

[5] 0초
```
# 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] 0초
```
# 데이터 분할
x_train, x_test, y_train, y_test = train_test_split(augmented_images, augmented_labels, test_size=0.2, random_state=42)
```

[7] 0초
```
model = models.Sequential([
    layers.Conv2D(32, (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(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(3, activation='softmax')  # 클래스 수만큼의 출력
])
```

[8] 0초
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        320

 max_pooling2d (MaxPooling2  (None, 15, 15, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

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

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 64)                65600

 dense_1 (Dense)             (None, 3)                 195

=================================================================
Total params: 121539 (474.76 KB)
Trainable params: 121539 (474.76 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[9] 0초
```
# 모델 컴파일
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

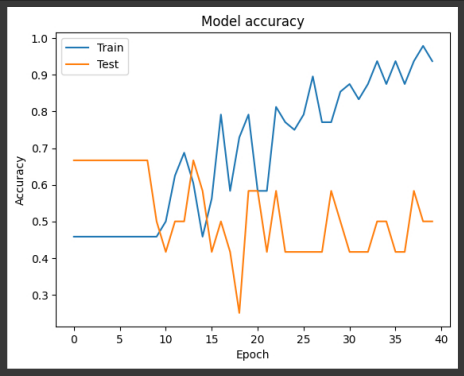
14초
```
# 모델 학습
history = model.fit(x_train, y_train, epochs=40, batch_size=32, validation_data=(x_test, y_test))

print(history.history)

import matplotlib.pyplot as plt
```

```
Epoch 1/40
2/2 [==============================] - 6s 954ms/step - loss: 1.0928 - accuracy: 0.4583 - val_loss: 0.9535 - val_accuracy: 0.6667
Epoch 2/40
2/2 [==============================] - 0s 86ms/step - loss: 1.0698 - accuracy: 0.4583 - val_loss: 0.9160 - val_accuracy: 0.6667
Epoch 3/40
2/2 [==============================] - 0s 99ms/step - loss: 1.0409 - accuracy: 0.4583 - val_loss: 0.9403 - val_accuracy: 0.6667
Epoch 4/40
2/2 [==============================] - 0s 103ms/step - loss: 1.0449 - accuracy: 0.4583 - val_loss: 0.9193 - val_accuracy: 0.6667
Epoch 5/40
2/2 [==============================] - 0s 111ms/step - loss: 1.0267 - accuracy: 0.4583 - val_loss: 0.9105 - val_accuracy: 0.6667
Epoch 6/40
2/2 [==============================] - 0s 105ms/step - loss: 1.0210 - accuracy: 0.4583 - val_loss: 0.9210 - val_accuracy: 0.6667
Epoch 7/40
2/2 [==============================] - 0s 126ms/step - loss: 1.0150 - accuracy: 0.4583 - val_loss: 0.9309 - val_accuracy: 0.6667
```

```
Epoch 8/40
2/2 [==============================] - 0s 94ms/step - loss: 1.0059 - accuracy: 0.4583 - val_loss: 0.9203 - val_accuracy: 0.6667
Epoch 9/40
2/2 [==============================] - 0s 109ms/step - loss: 0.9976 - accuracy: 0.4583 - val_loss: 0.8817 - val_accuracy: 0.6667
Epoch 10/40
2/2 [==============================] - 0s 137ms/step - loss: 0.9717 - accuracy: 0.4583 - val_loss: 0.9059 - val_accuracy: 0.5000
Epoch 11/40
2/2 [==============================] - 0s 186ms/step - loss: 0.9566 - accuracy: 0.5000 - val_loss: 0.9236 - val_accuracy: 0.4167
Epoch 12/40
2/2 [==============================] - 0s 236ms/step - loss: 0.9427 - accuracy: 0.6250 - val_loss: 0.9421 - val_accuracy: 0.5000
Epoch 13/40
2/2 [==============================] - 0s 230ms/step - loss: 0.9412 - accuracy: 0.6875 - val_loss: 0.9002 - val_accuracy: 0.5000
Epoch 14/40
2/2 [==============================] - 0s 126ms/step - loss: 0.9046 - accuracy: 0.6042 - val_loss: 0.8036 - val_accuracy: 0.6667
Epoch 15/40
2/2 [==============================] - 0s 88ms/step - loss: 0.9044 - accuracy: 0.4583 - val_loss: 0.8710 - val_accuracy: 0.5833
Epoch 16/40
2/2 [==============================] - 0s 95ms/step - loss: 0.8722 - accuracy: 0.5625 - val_loss: 0.9212 - val_accuracy: 0.4167
Epoch 17/40
2/2 [==============================] - 0s 164ms/step - loss: 0.8265 - accuracy: 0.7917 - val_loss: 0.7776 - val_accuracy: 0.5000
Epoch 18/40
2/2 [==============================] - 0s 84ms/step - loss: 0.8883 - accuracy: 0.5833 - val_loss: 0.8018 - val_accuracy: 0.4167
Epoch 19/40
2/2 [==============================] - 0s 117ms/step - loss: 0.8140 - accuracy: 0.7292 - val_loss: 1.0260 - val_accuracy: 0.2500
Epoch 20/40
2/2 [==============================] - 0s 96ms/step - loss: 0.8120 - accuracy: 0.7917 - val_loss: 0.8695 - val_accuracy: 0.5833
Epoch 21/40
2/2 [==============================] - 0s 122ms/step - loss: 0.7709 - accuracy: 0.5833 - val_loss: 0.8262 - val_accuracy: 0.5833
Epoch 22/40
2/2 [==============================] - 0s 194ms/step - loss: 0.7614 - accuracy: 0.5833 - val_loss: 0.9579 - val_accuracy: 0.4167
Epoch 23/40
2/2 [==============================] - 0s 184ms/step - loss: 0.7372 - accuracy: 0.8125 - val_loss: 0.9499 - val_accuracy: 0.5833
Epoch 24/40
2/2 [==============================] - 0s 103ms/step - loss: 0.6987 - accuracy: 0.7708 - val_loss: 0.7902 - val_accuracy: 0.4167
Epoch 25/40
2/2 [==============================] - 0s 91ms/step - loss: 0.6915 - accuracy: 0.7500 - val_loss: 0.8373 - val_accuracy: 0.4167
Epoch 26/40
2/2 [==============================] - 0s 120ms/step - loss: 0.6097 - accuracy: 0.7917 - val_loss: 1.0187 - val_accuracy: 0.4167
Epoch 27/40
2/2 [==============================] - 0s 106ms/step - loss: 0.6286 - accuracy: 0.8958 - val_loss: 0.9452 - val_accuracy: 0.4167
Epoch 28/40
2/2 [==============================] - 0s 106ms/step - loss: 0.5725 - accuracy: 0.7708 - val_loss: 0.8684 - val_accuracy: 0.4167
Epoch 29/40
```

1초

```
[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()
```



0초

```
[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(
```


✓ 0초 오후 11:57에 완료됨