Keras와 Webcam을 이용한 mnist 실습

The Scipy Ecosytem

김 대 환

2022.

텐서플로우 vs. Keras

- **텐서플로우 vs. Keras**
  - 텐서플로우는 구글에서 개발하고, 공개한 머신러닝 프레임워크
  - 케라스는 텐서플로우 위에서 동작하는 프레임워크
- **텐서플로우와 케라스의 장단점**
  - 텐서플로우는 머신러닝을 처음 접하는 사용자가 사용하기 어려운 부분이 있다
  - 텐서플로우는 스레딩이나 큐(Queue) 등의 메커니즘을 세밀하게 사용할 수 있고, 내부 구조를 확인할 수 있는 디버거 등을 활용할 수 있다
  - 신경망을 유심히 관찰하고 연구 개발해야 하는 경우라면 텐서플로우 사용을 권장
  - 케라스는 사용자 친화적으로 개발되었다
  - 간단한 신경망의 경우 몇 줄의 코딩만으로 구현 가능
  - 비교적 단순한 신경망을 구성하거나 기존의 갖추어진 기능만을 사용할 경우 케라스를 권장
    - ✓ 텐서플로우 1.2 버전부터 케리스와 통합 (tf.keras)
    - ✓ 주요 틀은 tf.keras로 구현하고, 내용은 순수한 텐서플로우로 채워 넣는 방법이 가장 좋은 옵션이 될 수 있다

TF(Keras)와 OpenCV를 이용한 웹캠 사용

- **Keras를 이용한 mnist 데이터 세트 학습**

```
import tensorflow as tf

# 데이터 세트 다운로드와 머신러닝 네트워크 준비
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

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

model.fit(x_train, y_train, epochs=10)
model.evaluate(x_test, y_test)

model.save_weights('./mnist_keras_weight/mnist_checkpo
int')
```

- Mnist 데이터 세트을 입력으로 신경망 학습을 진행
- 6만개의 손글씨 숫자 이미지를 10회 반복 학습

TF(Keras)와 OpenCV를 이용한 웹캠 사용

- **Keras를 이용한 mnist 데이터 세트 학습**
  - 학습이 진행되는 동안 에러는 줄고, 정확도는 향상된다
  - 학습 후, 업데이트 된 가중치를 이용하여 손글씨 이미지의 숫자를 인식한다
  - 업데이트된 가중치는 따로 파일로 저장하여 다음에 손글씨를 예측할 때 사용한다

TF(Keras)와 OpenCV를 이용한 웹캠 사용

- 학습된 신경망을 이용하여 손글씨 숫자 인식을 테스트

```
import cv2
import tensorflow as tf
import numpy as np
import math

def process(img_input):

    gray = cv2.cvtColor(img_input, cv2.COLOR_BGR2GRAY)
    gray = cv2.resize(gray, (28, 28), interpolation=cv2.INTER_AREA)

    (thresh, img_binary) = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV ¦ cv2.THRESH_OTSU)

    h,w = img_binary.shape

    ratio = 100/h
    new_h = 100
    new_w = w * ratio
```

TF(Keras)와 OpenCV를 이용한 웹캠 사용

- 학습된 신경망을 이용하여 손글씨 숫자 인식을 테스트

```
    img_empty = np.zeros((110,110), dtype=img_binary.dtype)
    img_binary = cv2.resize(img_binary, (int(new_w), int(new_h)), interpolation=cv2.INTER_AREA)
    img_empty[:img_binary.shape[0], :img_binary.shape[1]] = img_binary

    img_binary = img_empty

    cnts = cv2.findContours(img_binary.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # 컨투어의 무게중심 좌표를 구한다.
    # 컨투어(contour)란 동일한 색 또는 동일한 픽셀값(강도,intensity)을 가진 영역의 경계선에 대한 정보
다. 물체의 윤곽선, 외형을 파악하는데 사용된다.
    M = cv2.moments(cnts[0][0])
    center_x = (M["m10"] / M["m00"])
    center_y = (M["m01"] / M["m00"])

    # 무게 중심이 이미지 중심으로 오도록 이동시킨다.
    height,width = img_binary.shape[:2]
    shiftx = width/2-center_x
    shifty = height/2-center_y
```

TF(Keras)와 OpenCV를 이용한 웹캠 사용

- 학습된 신경망을 이용하여 손글씨 숫자 인식을 테스트

```
    Translation_Matrix = np.float32([[1, 0, shiftx],[0, 1, shifty]])
    img_binary = cv2.warpAffine(img_binary, Translation_Matrix, (width,height))

    img_binary = cv2.resize(img_binary, (28, 28), interpolation=cv2.INTER_AREA)

    flatten = img_binary.flatten() / 255.0

    return flatten

# 저장된 학습 모델을 불러와 사용할 때도 학습 시 사용한 네트워크 구조를 가져야 한다.
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

# 학습된 모델 로드
model.load_weights('./mnist_keras_weight/mnist_checkpoint')
```

TF(Keras)와 OpenCV를 이용한 웹캠 사용

- 학습된 신경망을 이용하여 손글씨 숫자 인식을 테스트

```
cap = cv2.VideoCapture(1)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
print('original size: %d, %d' % (width, height))

# 카메라 해상도 변경이 필요할 때
dispW=640
dispH=480
cap.set(cv2.CAP_PROP_FRAME_WIDTH, dispW)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, dispH)

while(True):
    ret, img_color = cap.read()
    if ret == False:
        break;

    img_input = img_color.copy()
```

TF(Keras)와 OpenCV를 이용한 웹캠 사용

- 학습된 신경망을 이용하여 손글씨 숫자 인식을 테스트

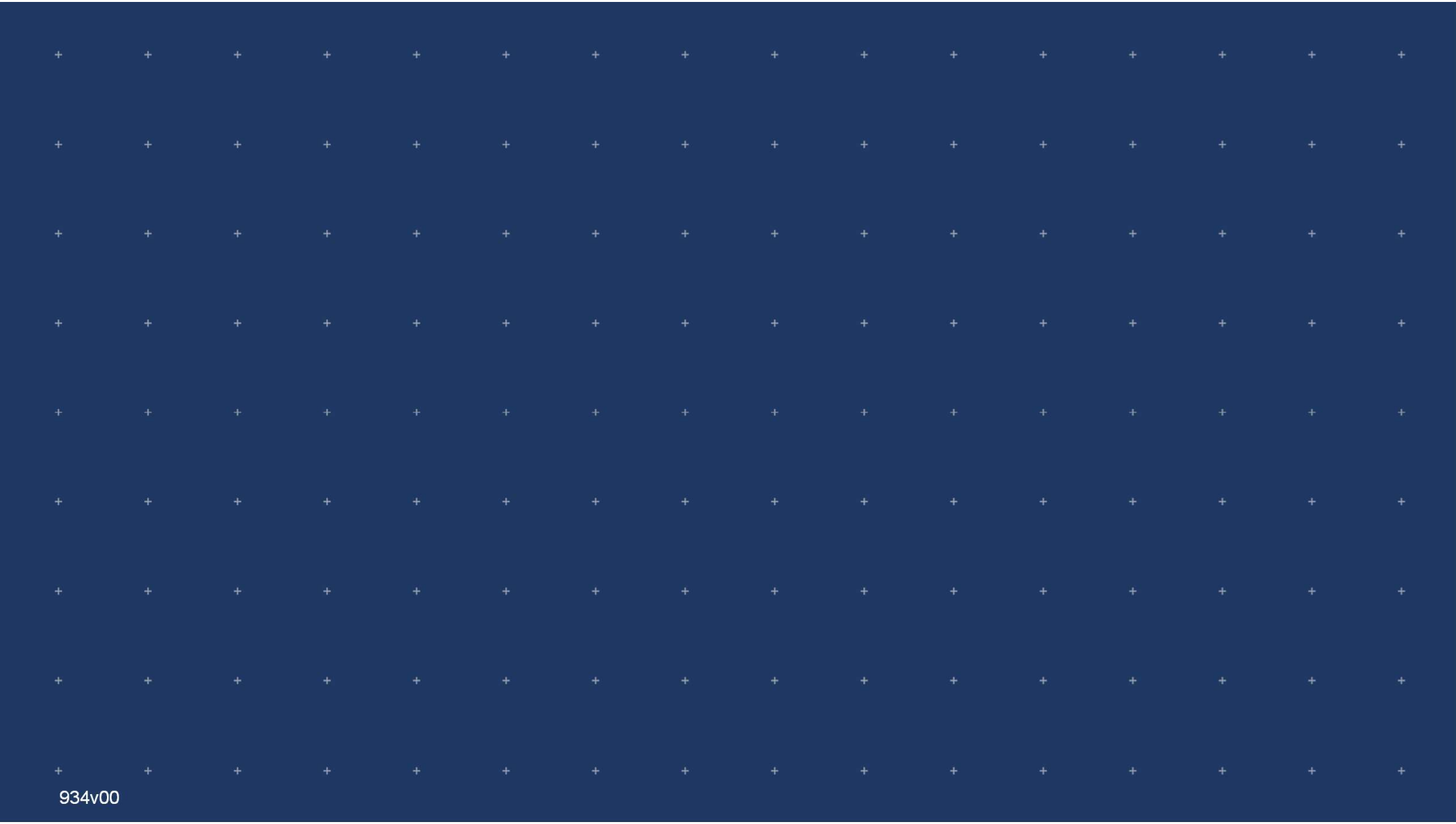
```
# 카메라 해상도 변경이 필요할 때
    cv2.rectangle(img_color, (250, 150),  (dispW-250, dispH-150), (0, 0, 255), 3)
    cv2.imshow('bgr', img_color)
    img_roi = img_input[150:dispW-150, 250:dispH-250]

    key = cv2.waitKey(1)

    if key == 27:
        break
    elif key == 32:
        flatten = process(img_roi)

        predictions = model.predict(flatten[np.newaxis,:])

        with tf.compat.v1.Session() as sess:
            print(tf.argmax(predictions, 1).eval())

        cv2.imshow('img_roi', img_roi)
        cv2.waitKey(0)
```

TF(Keras)와 OpenCV를 이용한 웹캠 사용

- **학습된 신경망을 이용하여 손글씨 숫자 인식을 테스트**

```
cap.release()
cv2.destroyAllWindows()
```

  - 테스트는 미리 손글씨로 적어 둔 숫자를 웹캠에 비추어 인식을 시작한다.
  - 숫자를 빨간색 사각형 안에 넣고 스페이스바를 누르면 인식이 시작된다
  - 숫자 이미지와 함께 터미널에 결과를 표시한다
  - 계속 다른 숫자들을 테스트 해본다


934v00