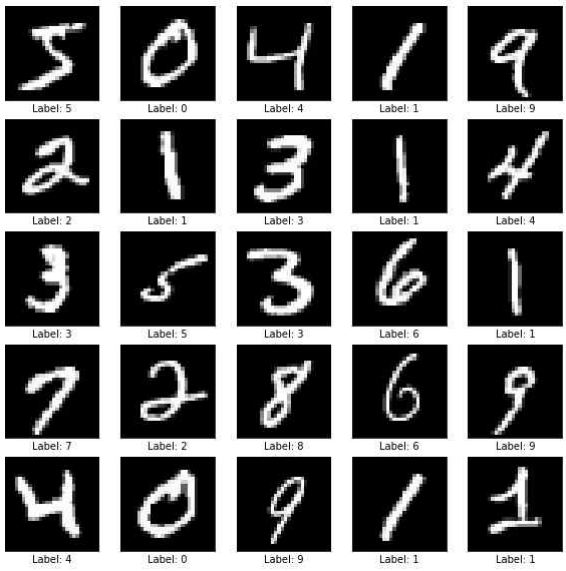
열 및 통계물리 2 개인 프로젝트:
MNIST 데이터베이스를 이용하여 숫자를 인식하는 어플리케이션 개발

물리학과 201500946 민선기

1. 개발 목적

인공신경망을 통해 학습하는 것을 실제로 구현해보기 위하여 가장 간단한 어플리케이션을 구현하였다. 수업에서 배운 FCN과 CNN과정을 통하여 MNIST 데이터베이스에 대한 학습을 실시하였고, 이를 tensorflow lite로 변환하여 안드로이드 기반 장비에 사용할 수 있도록 하였다.

2. 개발 과정



[그림 1] 학습시킬 MNIST 데이터베이스의 샘플

위 그림 1과 같은 데이터베이스를 학습시키기 위하여 CNN 과정을 거치게 되었다.

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Reshape(target_shape=(28, 28, 1)),
    keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation=tf.nn.relu),
    keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation=tf.nn.relu),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Dropout(0.25),
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dropout(0.5),

    keras.layers.Dense(10)
])

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 110s 58ms/step - loss: 0.1775 - accuracy: 0.9480
Epoch 2/5
1875/1875 [==============================] - 109s 58ms/step - loss: 0.0746 - accuracy: 0.9769
Epoch 3/5
1875/1875 [==============================] - 109s 58ms/step - loss: 0.0582 - accuracy: 0.9824
Epoch 4/5
1875/1875 [==============================] - 109s 58ms/step - loss: 0.0485 - accuracy: 0.9849
Epoch 5/5
1875/1875 [==============================] - 110s 59ms/step - loss: 0.0399 - accuracy: 0.9874
<tensorflow.python.keras.callbacks.History at 0x7fa45ab12a90>
```

[그림 2] 학습하게 될 CNN 과정

이 CNN과정에서 sequential을 이용하였고, activation함수 relu로 두 번의 convolution, 한번의 Maxpooling을 하여 layer를 쌓았다.

학습할 때 optimizer는 adam을 사용하였고, 5번의 epoch를 통해 학습하였다.

```
# Convert Keras model to TF Lite format.
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the TF Lite model as file
f = open('mnist.tflite', "wb")
f.write(tflite_model)
f.close()
```

```
INFO:tensorflow:Assets written to: /tmp/tmpj3zqurcb/assets
```

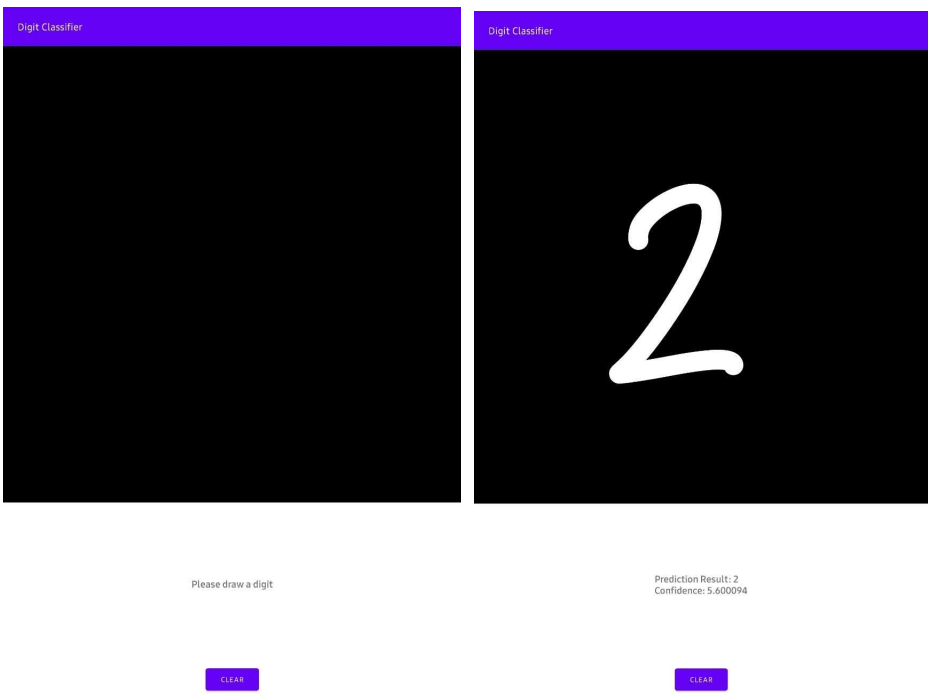
```
# Download the digit classification model if you're using Colab,
# or print the model's local path if you're not using Colab.
try:
  from google.colab import files
  files.download('mnist.tflite')
except ImportError:
  import os
  print('TF Lite model:', os.path.join(os.getcwd(), 'mnist.tflite'))
```

[그림 3] 안드로이드 기반으로 사용하기 위하여 TF lite 형식으로 변환

그림 3의 코드를 통해 안드로이드 기반으로 사용하기 위하여 TF lite 형식으로 변환하였고, 이를 android studio에서 사용하였다.

안드로이드 어플리케이션에서 그림을 통해 학습된 숫자를 받기 위하여 DRAW code를 사용하였으며, 이를 dependency에 추가하였다.

최종적으로 구현하게 된 어플리케이션의 모습은 다음과 같다.



3. 결론 및 고찰

CNN을 이용해 MNIST 데이터베이스를 학습하는 것을 완료하였고, 이를 실제로 구동하여 잘 맞는지 확인 또한 할 수 있었다. 하지만, MNIST 데이터베이스가 서구권에서 만들어져, 4와 6, 0 이 세 개의 숫자는 잘 판별이 되지 않는데, 다른 데이터베이스를 만들어서 학습을 한다면 이와같은 문제는 해결할 수 있을 것이라고 생각한다.

4. 참고문헌

https://youtu.be/vL0J0Fyfa4k
https://github.com/tensorflow/examples/tree/master/lite/examples/digit_classifier
https://github.com/divyanshub024/AndroidDraw