

Tinker board

IoT를 위한 최선의 엣지 단말기



MAKER SPACE
G·CAMP

Contents

- Tensorflow Lite 설명
- Andorid Tensorflow Lite를 위한 환경 설정
- Tensorflow Lite 모델 생성 후 Tinker Edge R 실행



Andorid Tensorflow Lite

- 현재 안드로이드에서는 Tensorflow Lite만 공식적으로 지원
- Tensorflow Lite는 Tensorflow Model을 모바일 환경에서 구동하도록 해줌
- Model 학습을 모바일에서 직접 하는 것은 아니며 학습된 Model을 모바일에 올려서 Inference 할 수 있도록 지원
- 현재는 공식적으로 MobileNet 모델로 객체 감지, 필기 숫자 인식, 이미지 분류 전이 학습, 핫워드 감지, 동작 인식, 모바일 객체 인식, audio_Classified 등의 여러 모델을 지원하며, 지속적으로 포팅 중

Andorid Tensorflow Lite

- 작업 순서

PC에서 Model 학습 -> 학습한 Model을 Tensorflow Lite Transfer ->

안드로이드 프로젝트 생성 -> assets 폴더를 만들고 Model 삽입 ->

Tensorflow Lite 모듈을 사용할 수 있도록 gradle 내용 추가 ->

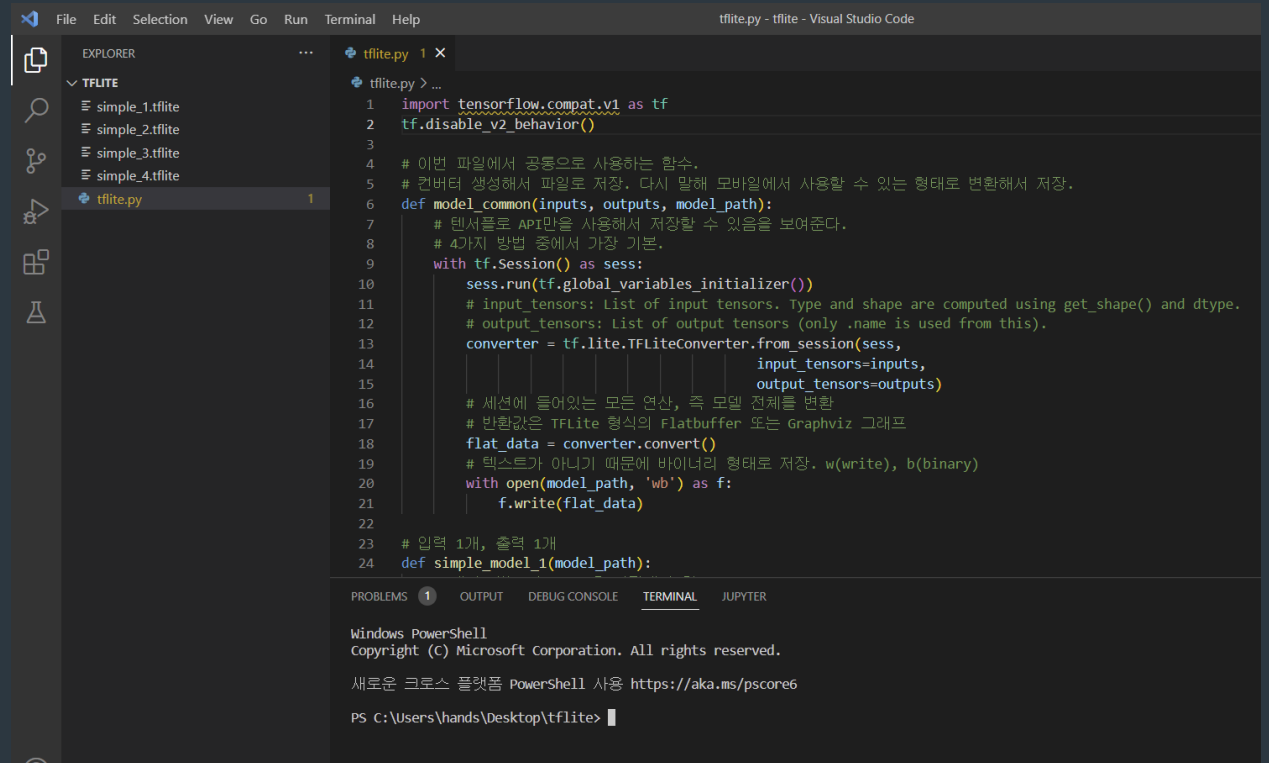
gradle을 수정해서 Model 압축 막고 동기화 ->

tflite Model을 로딩하고 run 함수를 호출해서 결과를 가져오고 표시

Andorid Tensorflow Lite

X86에서 Python 설치 후 tflite 생성

- Python 설치
- Visual studio code 설치
- Tflite 폴더 생성
- tflite.py 파일 생성
- Vscode 실행 후 Tflite import
- Pip install tensorflow로 설치



The screenshot shows the Visual Studio Code interface with a file explorer on the left and a code editor on the right. The file explorer shows a folder named 'TFLITE' containing files 'simple_1.tflite', 'simple_2.tflite', 'simple_3.tflite', 'simple_4.tflite', and 'tflite.py'. The code editor displays the contents of 'tflite.py', which is a Python script for converting a TensorFlow model to TFLite format. The script includes comments in Korean and Python code for importing TensorFlow, disabling v2 behavior, and using the TFLiteConverter to convert a model from a session to a flat buffer format. The terminal at the bottom shows the PowerShell prompt and the command 'PS C:\Users\hands\Desktop\tflite>'.

```
1 import tensorflow.compat.v1 as tf
2 tf.disable_v2_behavior()
3
4 # 이번 파일에서 공통으로 사용하는 함수.
5 # 컨버터 생성해서 파일로 저장. 다시 말해 모바일에서 사용할 수 있는 형태로 변환해서 저장.
6 def model_common(inputs, outputs, model_path):
7     # 텐서플로 API만을 사용해서 저장할 수 있음을 보여준다.
8     # 4가지 방법 중에서 가장 기본.
9     with tf.Session() as sess:
10         sess.run(tf.global_variables_initializer())
11         # input_tensors: List of input tensors. Type and shape are computed using get_shape() and dtype.
12         # output_tensors: List of output tensors (only .name is used from this).
13         converter = tf.lite.TFLiteConverter.from_session(sess,
14                                                         input_tensors=inputs,
15                                                         output_tensors=outputs)
16         # 세션에 들어있는 모든 연산, 즉 모델 전체를 변환
17         # 반환값은 TFLite 형식의 Flatbuffer 또는 Graphviz 그래프
18         flat_data = converter.convert()
19         # 텍스트가 아니기 때문에 바이너리 형태로 저장. w(write), b(binary)
20         with open(model_path, 'wb') as f:
21             f.write(flat_data)
22
23 # 입력 1개, 출력 1개
24 def simple_model_1(model_path):
```

Android TensorFlow Lite

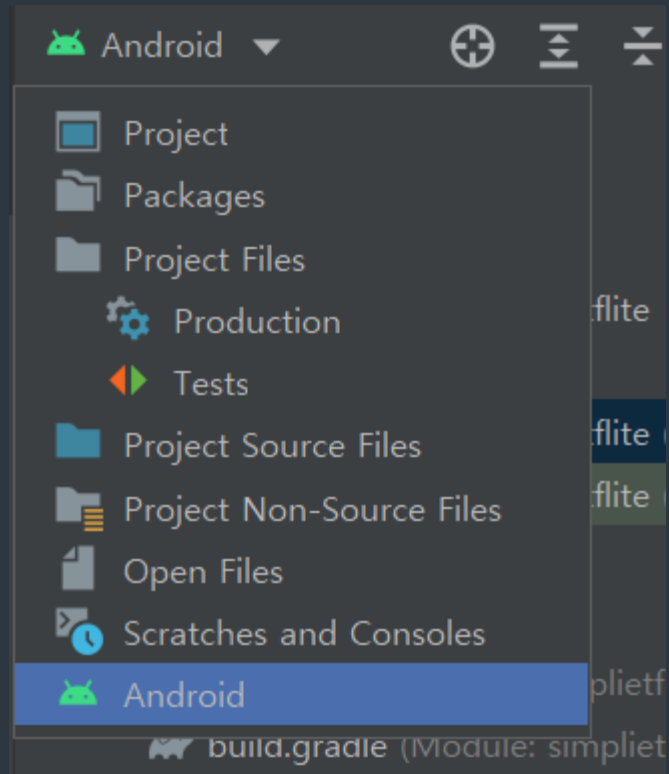
tfLite.py

```
tflite.py > ...  
1  import tensorflow.compat.v1 as tf  
2  tf.disable_v2_behavior()  
3  
4  # 이번 파일에서 공통으로 사용하는 함수.  
5  # 컨버터 생성해서 파일로 저장. 다시 말해 모바일  
6  def model_common(inputs, outputs, model_path
```

- 현재 Tensorflow는 2버전 바이너리만 제공되기에 특정 명령어를 처음에 선언하여 Tensorflow 1 명령어를 사용 가능케 해야 함

Andorid Tensorflow Lite

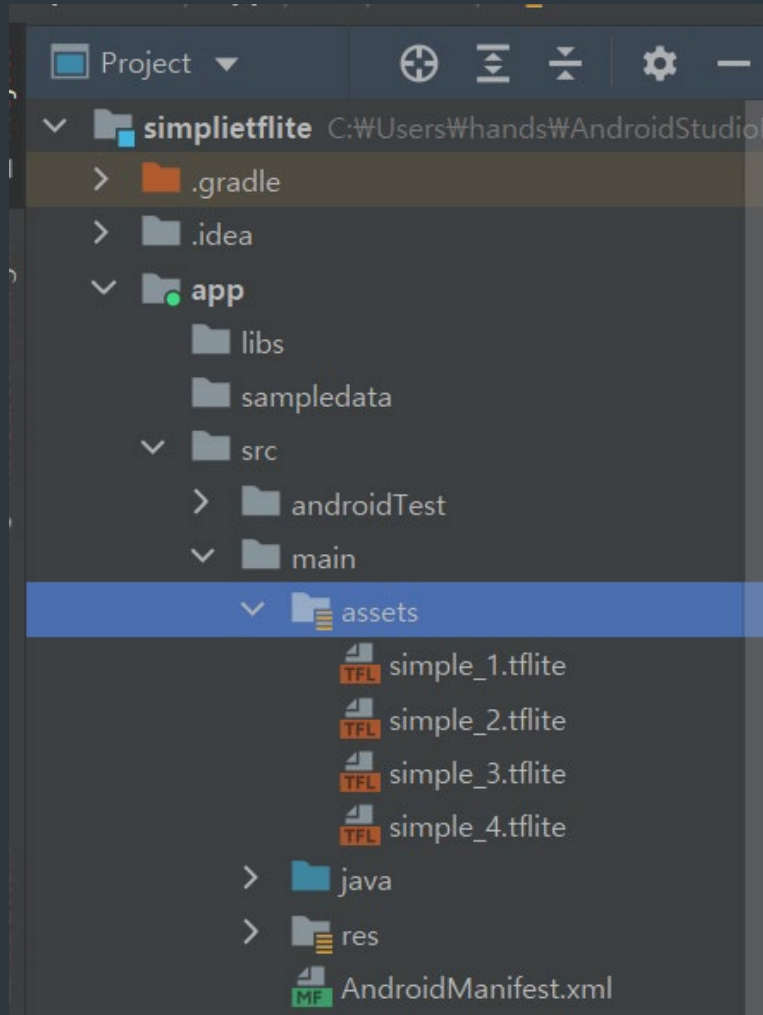
simple.tflite



- Android Tree를 Project Tree로 변경

Andorid Tensorflow Lite

simple.tflite



- Main 폴더에 assets 폴더 생성 후
simple.tflite 파일 입력

Android Tensorflow Lite

tf lite.py

```
# 입력 1개, 출력 1개
def simple_model_1(model_path):
    # 에러. 반드시 shape을 지정해야 함.
    # x = tf.placeholder(tf.int32)

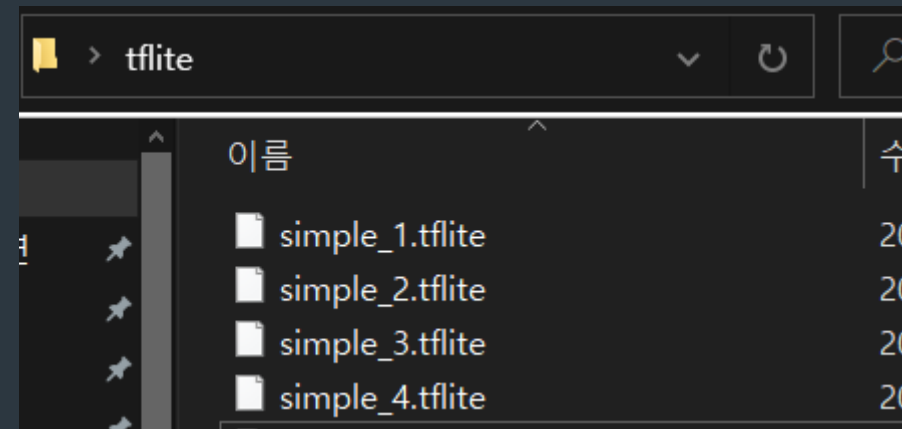
    # 안드로이드에서 전달한 입력과 출력 변수가 플레이스 홀더와 연동
    x = tf.placeholder(tf.int32, shape=[1])
    out = x * 5

    model_common([x], [out], model_path)

    # 에러. 반드시 [] 형태로 전달해야 함.
    # model_common(x, out, model_path)
```

```
64     model_common([x1, x2], [out_1, out_2], model_path)
65
66     simple_model_1('simple_1.tflite')
67     simple_model_2('simple_2.tflite')
68     simple_model_3('simple_3.tflite')
69     simple_model_4('simple_4.tflite')
70
```

- 가장 간단한 tf model을 정의후 tflite model 파일 생성



Android Tensorflow Lite

<Gradle>

```
dependencies {  
    implementation fileTree(dir: 'libs', include: ['*.jar'])  
    implementation 'androidx.appcompat:appcompat:1.5.0'  
    implementation 'com.google.android.material:material:1.6.1'  
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'  
    testImplementation 'junit:junit:4.13.2'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'  
    implementation 'org.tensorflow:tensorflow-lite:+'  
}
```

tflite 모듈을 사용할 수 있도록
gradle 파일에 내용을 추가

Gradle 파일의 dependencies
영역 끝 부분에 tflite 모듈을
화면과 같이 추가

Android Tensorflow Lite

<Gradle>

```
android {
    compileSdk 32

    defaultConfig {
        applicationId "com.example.simplietflite"
        minSdk 21
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }

    aaptOptions {
        noCompress "tflite"
    }
}
```

gradle 파일을 수정해서 모델
파일이 압축되지 않도록 하기

android 영역을 찾아서 마지막에
aaptOptions를 추가

메모리를 절약하기 위해 리소스를
압축하는데 그럴 경우 모델을
올바로 읽어들이지 수가 없음

완료 후 Gradle 동기화 진행

Andorid Tensorflow Lite

activity_main.xml



4개의 텐서플로우 라이트 모델을
선택할 수 있는 버튼 및 모델
결과값을 호출하는
activity_main.xml 제작

Andorid Tensorflow Lite

Mainactivity

```
import android.view.View;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import org.tensorflow.lite.Interpreter; // 핵심 모듈
|
import java.io.FileInputStream;
import java.io.IOException;
import java.nio.MappedByteBuffer;
import java.nio.channels.FileChannel;

public class MainActivity extends AppCompatActivity {
    @Override
```

Java 파일 입출력 함수 및
텐서플로우 라이트 인터프리터
함수 импорт

Android TensorFlow Lite

Mainactivity

```
findViewById(R.id.button_1).setOnClickListener(new View.OnClickListener() {
    // 리스너의 기능 중에서 클릭(single touch) 사용
    @Override
    public void onClick(View v) {
        // input : 텐서플로 모델의 placeholder에 전달할 데이터(3)
        // output: 텐서플로 모델로부터 결과를 넘겨받을 배열. 덮어쓰기 때문
        int[] input = new int[]{3};
        int[] output = new int[]{0}; // 15 = 3 * 5, out = x * 5

        // 1번 모델을 해석할 인터프리터 생성
        Interpreter tfLite = getTfLiteInterpreter("simple_1.tflite");

        // 모델 구동.
        // 정확하게는 from_session 함수의 output_tensors 매개변수에 전달된
        tfLite.run(input, output);

        // 출력을 배열에 저장하기 때문에 0번째 요소를 가져와서 문자열로 변환
        tv_output.setText(String.valueOf(output[0]));
    }
});
```

각 버튼마다 텐서플로우 라이트
모델을 해석해 결과값을 표현해
UI에 표시하는 코드 추가

Andorid Tensorflow Lite

Mainactivity

```
// 모델을 읽어오는 함수로, 텐서플로 라이트 홈페이지에 있다.  
// MappedByteBuffer 바이트 버퍼를 Interpreter 객체에 전달하면 모델 해석을 할 수 있다.  
private MappedByteBuffer loadModelFile(Activity activity, String modelPath) throws IOException {  
    AssetFileDescriptor fileDescriptor = activity.getAssets().openFd(modelPath);  
    FileInputStream inputStream = new FileInputStream(fileDescriptor.getFileDescriptor());  
    FileChannel fileChannel = inputStream.getChannel();  
    long startOffset = fileDescriptor.getStartOffset();  
    long declaredLength = fileDescriptor.getDeclaredLength();  
    return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset, declaredLength);  
}
```

Java 코드로 메모리 버퍼에 올라간 모델을 읽어오는 함수를 추가함으로 Android가 텐서플로우
라이트 모델을 인터프리터 객체에 전달해 모델 해석을 가능케 하는 코드

Andorid Tensorflow Lite

실행 화면

[GitHub 링크](#)



기존에 생성한 tflite 모델 파일들이
버튼에 따라 각각 실행됨을 확인