**PRACTICAL: 14**

**AIM:**

Create an application that uses end-to-end process of training a machine learning model that can recognize handwritten digit images with TensorFlow and deploy it to an Android app.

**THEORY:**

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

TensorFlow Lite is a set of tools to help developers run TensorFlow models on mobile, embedded, and IoT devices. It enables on-device machine learning inference with low latency and a small binary size. TensorFlow Lite consists of two main components: The TensorFlow Lite interpreter, which runs specially optimized models on many different hardware types, including mobile phones, embedded Linux devices, and microcontrollers. The TensorFlow Lite converter, which converts TensorFlow models into an efficient form for use by the interpreter, and can introduce optimizations to improve binary size and performance.

TensorFlow Lite is designed to make it easy to perform machine learning on devices, "at the edge" of the network, instead of sending data back and forth from a server. For developers, performing machine learning on-device can help improve:

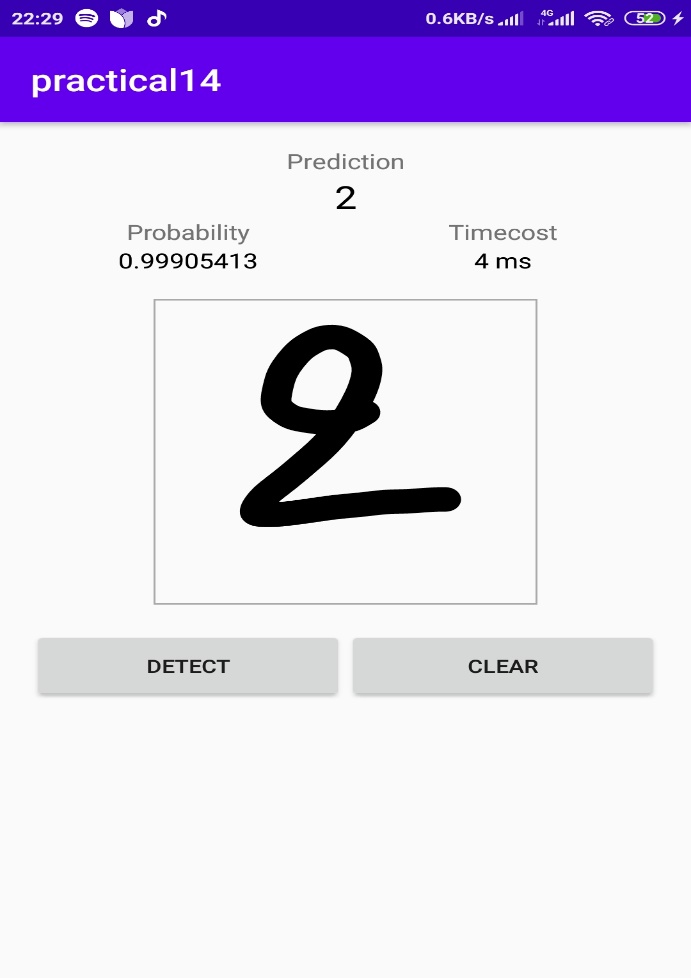
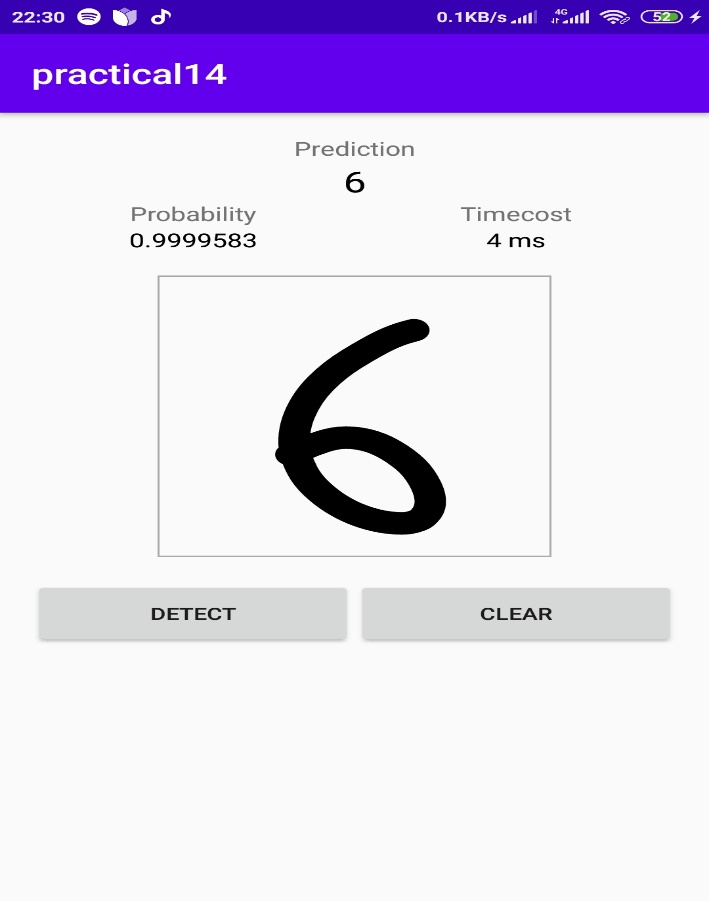
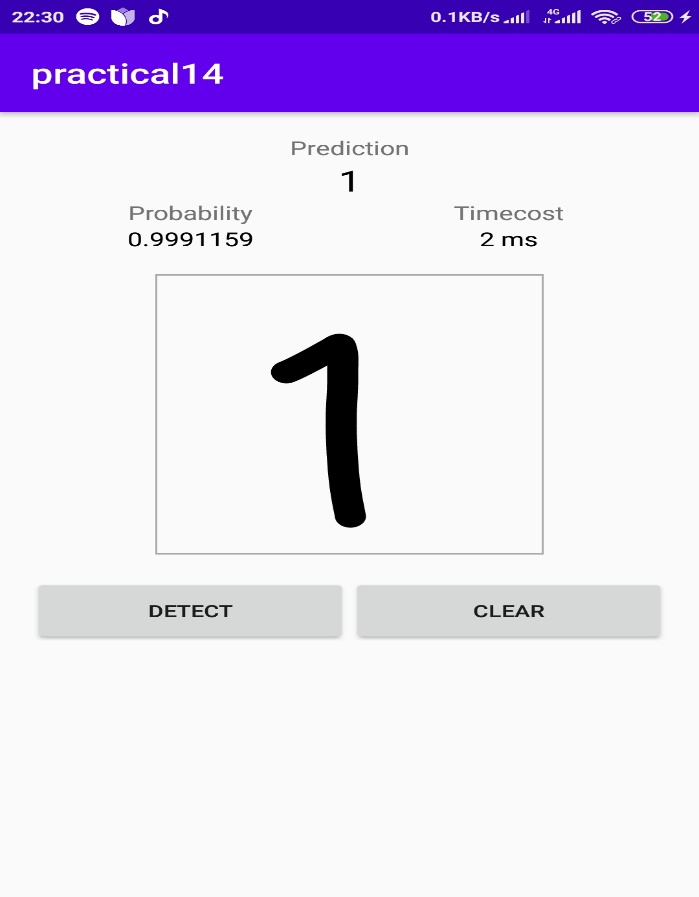
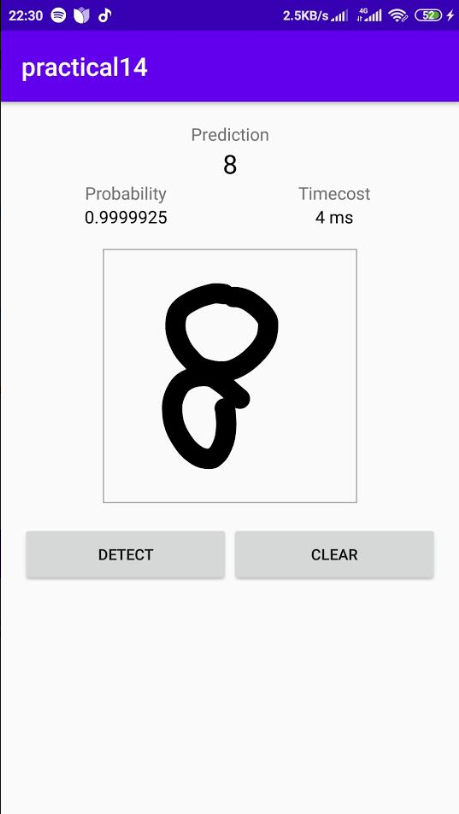
* Latency*:* there's no round-trip to a server
* Privacy*:* no data needs to leave the device
* Connectivity*:* an Internet connection isn't required
* Powerconsumption*:* network connections are power hungry

TensorFlow Lite works with a huge range of devices, from tiny microcontrollers to powerful mobile phones.

**CODE:**

|  |
| --- |
| **MainActvity.java**  package com.example.prac\_14\_wcmc\_17it015;  import android.graphics.Bitmap;  import android.os.Bundle;  import android.util.Log;  import android.widget.TextView;  import android.widget.Toast;  import com.nex3z.fingerpaintview.FingerPaintView;  import java.io.IOException;  import androidx.appcompat.app.AppCompatActivity;  import butterknife.BindView;  import butterknife.ButterKnife;  import butterknife.OnClick;  public class MainActivity extends AppCompatActivity {  private static final String LOG\_TAG = "lele";  @BindView(R.id.fpv\_paint) FingerPaintView mFpvPaint;  @BindView(R.id.tv\_prediction) TextView mTvPrediction;  @BindView(R.id.tv\_probability) TextView mTvProbability;  @BindView(R.id.tv\_timecost) TextView mTvTimeCost;  private Classifier mClassifier;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  ButterKnife.bind(this);  init();  }  @OnClick(R.id.btn\_detect)  void onDetectClick() {  if (mClassifier == null) {  Log.e(LOG\_TAG, "onDetectClick(): Classifier is not initialized");  return;  } else if (mFpvPaint.isEmpty()) {  Toast.makeText(this, R.string.please\_write\_a\_digit, Toast.LENGTH\_SHORT).show();  return;  }  Bitmap image = mFpvPaint.exportToBitmap(  Classifier.IMG\_WIDTH, Classifier.IMG\_HEIGHT);  Result result = mClassifier.classify(image);  renderResult(result);  }  @OnClick(R.id.btn\_clear)  void onClearClick() {  mFpvPaint.clear();  mTvPrediction.setText(R.string.empty);  mTvProbability.setText(R.string.empty);  mTvTimeCost.setText(R.string.empty);  }  private void init() {  try {  mClassifier = new Classifier(MainActivity.this);  } catch (IOException e) {  Toast.makeText(this, e.getMessage().toString(), Toast.LENGTH\_LONG).show();  Log.e(LOG\_TAG, "init(): Failed to create Classifier", e);  }  }  private void renderResult(Result result) {  mTvPrediction.setText(String.valueOf(result.getNumber()));  mTvProbability.setText(String.valueOf(result.getProbability()));  mTvTimeCost.setText(String.format(getString(R.string.timecost\_value),  result.getTimeCost()));  }  }  **classifier.java**  package com.example.prac\_14\_wcmc\_17it015;  import android.app.Activity;  import android.content.res.AssetFileDescriptor;  import android.graphics.Bitmap;  import android.os.SystemClock;  import android.util.Log;  import org.tensorflow.lite.Interpreter;  import java.io.FileInputStream;  import java.io.IOException;  import java.nio.ByteBuffer;  import java.nio.ByteOrder;  import java.nio.MappedByteBuffer;  import java.nio.channels.FileChannel;  import java.util.Arrays;  public class Classifier {  private static final String LOG\_TAG = Classifier.class.getSimpleName();  private static final String MODEL\_NAME = "mnist.tflite";  private static final int BATCH\_SIZE = 1;  public static final int IMG\_HEIGHT = 28;  public static final int IMG\_WIDTH = 28;  private static final int NUM\_CHANNEL = 1;  private static final int NUM\_CLASSES = 10;  private final Interpreter.Options options = new Interpreter.Options();  private final Interpreter mInterpreter;  private final ByteBuffer mImageData;  private final int[] mImagePixels = new int[IMG\_HEIGHT \* IMG\_WIDTH];  private final float[][] mResult = new float[1][NUM\_CLASSES];  public Classifier(MainActivity activity) throws IOException {  mInterpreter = new Interpreter(loadModelFile(activity), options);  mImageData = ByteBuffer.allocateDirect(  4 \* BATCH\_SIZE \* IMG\_HEIGHT \* IMG\_WIDTH \* NUM\_CHANNEL);  mImageData.order(ByteOrder.nativeOrder());  }  public Result classify(Bitmap bitmap) {  convertBitmapToByteBuffer(bitmap);  long startTime = SystemClock.uptimeMillis();  mInterpreter.run(mImageData, mResult);  long endTime = SystemClock.uptimeMillis();  long timeCost = endTime - startTime;  Log.v(LOG\_TAG, "classify(): result = " + Arrays.toString(mResult[0])  + ", timeCost = " + timeCost);  return new Result(mResult[0], timeCost);  }  private MappedByteBuffer loadModelFile(Activity activity) throws IOException {  AssetFileDescriptor fileDescriptor = activity.getAssets().openFd(MODEL\_NAME);  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);  }  private void convertBitmapToByteBuffer(Bitmap bitmap) {  if (mImageData == null) {  return;  }  mImageData.rewind();  bitmap.getPixels(mImagePixels, 0, bitmap.getWidth(), 0, 0,  bitmap.getWidth(), bitmap.getHeight());  int pixel = 0;  for (int i = 0; i < IMG\_WIDTH; ++i) {  for (int j = 0; j < IMG\_HEIGHT; ++j) {  int value = mImagePixels[pixel++];  mImageData.putFloat(convertPixel(value));  }  }  }  private static float convertPixel(int color) {  return (255 - (((color >> 16) & 0xFF) \* 0.299f  + ((color >> 8) & 0xFF) \* 0.587f  + (color & 0xFF) \* 0.114f)) / 255.0f;  }  }  **result.java**  package com.example.prac\_14\_wcmc\_17it015;  public class Result {  private final int mNumber;  private final float mProbability;  private final long mTimeCost;  public Result(float[] probs, long timeCost) {  mNumber = argmax(probs);  mProbability = probs[mNumber];  mTimeCost = timeCost;  }  public int getNumber() {  return mNumber;  }  public float getProbability() {  return mProbability;  }  public long getTimeCost() {  return mTimeCost;  }  private static int argmax(float[] probs) {  int maxIdx = -1;  float maxProb = 0.0f;  for (int i = 0; i < probs.length; i++) {  if (probs[i] > maxProb) {  maxProb = probs[i];  maxIdx = i;  }  }  return maxIdx;  }  }  **App.gradle**  apply plugin: 'com.android.application'  android {  compileSdkVersion 28  defaultConfig {  applicationId "com.example.prac\_14\_wcmc\_17it015"  minSdkVersion 15  targetSdkVersion 28  versionCode 1  versionName "1.0.0"  testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  }  buildTypes {  release {  minifyEnabled false  proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  }  }  aaptOptions {  noCompress "tflite"  noCompress "lite"  }  compileOptions {  sourceCompatibility JavaVersion.VERSION\_1\_8  targetCompatibility JavaVersion.VERSION\_1\_8  }  }  dependencies {  implementation fileTree(dir: 'libs', include: ['\*.jar'])  implementation 'com.android.support:appcompat-v7:28.0.0'  implementation 'com.android.support:support-v4:28.0.0'    implementation 'com.nex3z:finger-paint-view:0.1.0'  implementation 'org.tensorflow:tensorflow-lite:1.13.1'  implementation 'com.jakewharton:butterknife:10.0.0'  annotationProcessor 'com.jakewharton:butterknife-compiler:10.0.0'  testImplementation 'junit:junit:4.12'  androidTestImplementation 'com.android.support.test:runner:1.0.2'  androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'  }  **activity\_main.xml**  <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical"  android:padding="16dp"  tools:context=".MainActivity">  <TableLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content">  <TextView  style="@style/ResultText"  android:text="@string/prediction"/>  <TextView  android:id="@+id/tv\_prediction"  style="@style/ResultText"  android:textSize="24sp"  android:textColor="@android:color/black"  android:text="@string/empty"  tools:text="1"/>  <TableRow>  <TextView  style="@style/ResultText"  android:text="@string/probability"/>  <TextView  style="@style/ResultText"  android:text="@string/timecost"/>  </TableRow>  <TableRow>  <TextView  android:id="@+id/tv\_probability"  style="@style/ResultText"  android:textColor="@android:color/black"  android:text="@string/empty"  tools:text="0.9"/>  <TextView  android:id="@+id/tv\_timecost"  style="@style/ResultText"  android:textColor="@android:color/black"  android:text="@string/empty"  tools:text="10ms"/>  </TableRow>  </TableLayout>  <com.nex3z.fingerpaintview.FingerPaintView  android:id="@+id/fpv\_paint"  android:layout\_width="200dp"  android:layout\_height="200dp"  android:layout\_gravity="center"  android:layout\_marginTop="16dp"  android:foreground="@drawable/shape\_rect\_border"/>  <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="16dp"  android:orientation="horizontal">  <Button  android:id="@+id/btn\_detect"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_weight="1"  android:text="@string/detect"/>  <Button  android:id="@+id/btn\_clear"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_weight="1"  android:text="@string/clear"/>  </LinearLayout>  </LinearLayout>  **AndroidManifest.xml**  <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android"  package="com.example.prac\_13\_wcmc\_17it015">  <application  android:allowBackup="true"  android:icon="@mipmap/ic\_launcher"  android:label="@string/app\_name"  android:roundIcon="@mipmap/ic\_launcher\_round"  android:supportsRtl="true"  android:theme="@style/AppTheme">  <activity android:name=".MainActivity">  <intent-filter>  <action android:name="android.intent.action.MAIN" />  <category android:name="android.intent.category.LAUNCHER" />  </intent-filter>  </activity>  </application>  </manifest> |

**OUTPUT:**

****

**LATEST APPLICATIONS:**

Netflix, Tinder, Snapchat.

**LEARNING OUTCOME:**

I learnt how to work with machine learning algorithms in android studio as well as I also learnt how to combine android studio and trainng an algorithm and implement in real application.