A Palmprint Verification System

[redacted]

Brief Description

Introduction

Palmprint verification is a type of biometric authentication with a wide range of application scenarios. In fact, it is very robust to kinds of variations (e.g., Illumination and time) and is more secure than face. However, its potentiality for commerce has not been fully investigated. With more advanced camera and sensors, modern mobile phones can obtain high-quality pamlprint images. And with more efficient and accurate algorithms and increasing computing power on mobile devices, palmprint verification is a very promising authentication technology.

We implemented a palmprint verification system based on mobile phones. The system totally runs on a mobile phone. All data are stored on the local device, and the whole computation process is completed locally.

We referred to the paper published by Prof. Lin Zhang et al. Towards Palmprint Verification On Smartphones and used the MPD dataset to train our models.

Modules

The main process of the verification is:

- Use YOLO-v4-tiny to detect key points (Double Finger Gap and Palm Center)
- Construct a local coordinate system and extract the ROI
- Resize the ROI to 224 x 224 and feed the resized ROI to the MobileFaceNet to extract a 512-dimensional feature vector
- Compute the dot product of the features. If the dot product (which represents similarity) is greater than the specified threshold, the verification is successful.

There are 4 corresponding modules in the application:

- YOLO-v4-tiny object detector

  The detector detects key points (Double Finger Gaps and Palm Center) in an image or video. The trained model is converted by Tencent's NCNN framework for deployment on Android devices.

- Image Processing Module

  After detecting the key points on the palm, a local coordinate system is constructed. A specific area is extracted as the ROI. The ROI is then rotated and resized to 224 x 224.

- Feature Extraction Module

  A MobileFaceNet is trained for feature extraction. The model takes a 224 x 224 ROI as input and outputs a 512 dimensional feature vector.

- Local Database Module

When users register their palmprints, the extracted features are stored in a local database. When users verify their palmprints, the system compare their palmprints with the data stored in the local database. If the similarity is greater than a specified threshold, the verification is successful.

Project Structure

The project consists of 2 parts:

- Android Application: implementation of the Android application
- Models: models definition, training and validation programs

```
├─android
│  └─main
│      ├─assets
│      ├─cpp
│      │  └─ncnnvulkan
│      │      ├─arm64-v8a
│      │      ├─armeabi-v7a
│      │      ├─include
│      │      │  ├─glslang
│      │      │  │  ├─Include
│      │      │  │  ├─MachineIndependent
│      │      │  │  │  └─preprocessor
│      │      │  │  ├─Public
│      │      │  │  └─SPIRV
│      │      │  ├─ncnn
│      │      │  └─SPIRV
│      │      ├─x86
│      │      └─x86_64
│      ├─java
│      │  └─top
│      │      └─sun1999
│      └─res
│          ├─filepaths
│          ├─layout
│          ├─mipmap-xxxhdpi
│          ├─values
│          └─xml
└─model
    ├─dataset
    ├─result
    └─saved_models
        └─MobileFaceNet_v2_20220614_231536
```

Implemented Requirements

Palmprint registration

If the user is using the application for the first time, the system will prompt the user to register his/her palmprint first. The system stores the data of the user's palmprint for future verification.

The user can either select an image or take an image of his/her palm. If a palm is successfully detected in the image, the feature of the palm will be stored on the phone. If no palm is detected, the system will prompt a message and require the user to select or take an image again.

code：

```java
@Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        Bitmap image = byteToBitmap(getPhotoByUrl(currentPhotoPath));

        long startTime = System.currentTimeMillis();
        Box[] result = YOLOv4.detect(image, threshold, nms_threshold);
        Bitmap tmpimage = Util.extractROI(image, result);
        if (tmpimage == null) {
            tvInfo.setText("Yolo 识别失败");
            resultImageView.setImageBitmap(image);
            return;
        }
        resultImageView.setImageBitmap(tmpimage);

        // preparing input tensor
        final Tensor inputTensor = TensorImageUtils.bitmapToFloat32Tensor(tmpimage,
                TensorImageUtils.TORCHVISION_NORM_MEAN_RGB, TensorImageUtils.TORCHVISION_NORM_STD_RGB, MemoryFormat.CHANNELS_LAST);

        // running the model
        final Tensor outputTensor = module.forward(IValue.from(inputTensor)).toTensor();


        long endTime;
        myVec = outputTensor.getDataAsFloatArray();

        double sum = 0;
        for (float tmp : myVec) {
            sum += tmp * tmp;
        }
        sum = Math.sqrt(sum);
        for (int i = 0; i < myVec.length; i++) {
            myVecDouble[i] = myVec[i] / sum;
        }
        endTime = System.currentTimeMillis();
        Log.e("myVec", myVec.toString());

        inputName();

        tvInfo.setText(String.format(Locale.CHINESE,
                "ImgSize: %dx%d\nUseTime: %d ms\n特征向量已保存",
                image.getHeight(), image.getWidth(), endTime - startTime));
    }
```

Palmprint verification

The user can tap on the button "Verify palmprint" and place his/her hand in the camera. The verification result will be shown on the screen.

code:

```java
         @Override
        public void analyze(ImageProxy image, final int rotationDegrees) {
            final Bitmap bitmapsrc = imageToBitmap(image);  // 格式转换
            Thread detectThread = new Thread(() -> {
                Matrix matrix = new Matrix();
                matrix.postRotate(rotationDegrees);
                width = bitmapsrc.getWidth();
                height = bitmapsrc.getHeight();
                Bitmap bitmap = Bitmap.createBitmap(bitmapsrc, 0, 0, width,
height, matrix, false);

                startTime = System.currentTimeMillis();
                Box[] result = YOLOv4.detect(bitmap, threshold, nms_threshold);
                endTime = System.currentTimeMillis();

                final Bitmap mutableBitmap = bitmap.copy(Bitmap.Config.ARGB_8888,
true);
                float strokeWidth = 4 * (float) mutableBitmap.getWidth() / 800;
                float textSize = 30 * (float) mutableBitmap.getWidth() / 800;

                Canvas canvas = new Canvas(mutableBitmap);
                boxPaint.setAlpha(255);
                boxPaint.setTypeface(Typeface.SANS_SERIF);
                boxPaint.setStyle(Paint.Style.STROKE);
                boxPaint.setStrokeWidth(strokeWidth);
                boxPaint.setTextSize(textSize);
                for (Box box : result) {
                    boxPaint.setColor(box.getColor());
                    boxPaint.setStyle(Paint.Style.FILL);
                    String score = Integer.toString((int) (box.getScore() *
100));
                    canvas.drawText(box.getLabel() + " [" + score + "%]",
                            box.x0 - strokeWidth, box.y0 - strokeWidth
                            , boxPaint);
                    boxPaint.setStyle(Paint.Style.STROKE);
                    canvas.drawRect(box.getRect(), boxPaint);
                }

                String res;
                Bitmap roi = Util.extractROI(mutableBitmap, result, true);

                if (Util.names.size() == 0) {
                    res = "请录入掌纹";
                } else {
                    if (roi == null) {
                        res = "请将摄像头对准手掌";
                    } else {
                        final Tensor inputTensor = TensorImageUtils.bitmapToFloat32Tensor(roi,
                                TensorImageUtils.TORCHVISION_NORM_MEAN_RGB, TensorImageUtils.TORCHVISION_NORM_STD_RGB, MemoryFormat.CHANNELS_LAST);

                        // running the model
                        final Tensor outputTensor = module.forward(IValue.from(inputTensor)).toTensor();

                        float[] vec = outputTensor.getDataAsFloatArray();
                        float sum = 0;
                        for (float tmp : vec) {
                            sum += tmp * tmp;
                        }
                        sum = (float) Math.sqrt(sum);

                        double dot = 0;
                        double vect[] = Util.vecs.get(palmIndex);
                        for (int i = 0; i < 512; i++) {
                            dot += vec[i] / sum * vect[i];
                        }

                        endTime = System.currentTimeMillis();
                        String res1 = dot > 0.4 ? "掌纹匹配成功" : "掌纹匹配失败";
                        res = String.format(Locale.CHINESE,
                                "相似度: %.3f %s\nImgSize: %dx%d\nUseTime: %d ms\n%d",
                                dot, res1, mutableBitmap.getHeight(), mutableBitmap.getWidth(), endTime - startTime,palmIndex);

                    }

                }
                runOnUiThread(() -> {
                    resultImageView.setImageBitmap(mutableBitmap);
                    tvInfo.setText(res);
                });
```

```
            }, "detect");
            detectThread.start();
        }
```

Visualization of the Verification Process

When the user verify his/her palmprint, the key points detected in the image are shown with its confidence. The local coordinate system is also shown in the image. After extracting the ROI, the system shows the ROI on the screen. This helps users understand the process of palmprint verification.

code:

```
public static Bitmap extractROI(Bitmap img, Box[] yoloRes) {
        Integer H = img.getHeight(), W = img.getWidth();
        float x1 = -1;
        float y1 = 0;
        float x2 = 0;
        float y2 = 0;
        float x3 = 0;
        float y3 = 0;

        if (yoloRes.length != 3) {
            return null;
        }
        for (Box box : yoloRes) {
            if (box.getLabel().equals("double gap")) {
                if (x1 == -1) {
                    x1 = (box.x0 + box.x1) / 2;
                    y1 = (box.y1 + box.y0) / 2;
                } else {
                    x2 = (box.x0 + box.x1) / 2;
                    y2 = (box.y1 + box.y0) / 2;
                }
            } else {
                x3 = (box.x0 + box.x1) / 2;
                y3 = (box.y1 + box.y0) / 2;
            }
        }
        if (x2 == 0 || x3 == 0) {
            return null;
        }

        float x0 = (x1 + x2) / 2;
        float y0 = (y1 + y2) / 2;

        double unitLen = Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));

        float k1 = (y1 - y2) / (x1 - x2); // line AB
```

```java
        float b1 = y1 - k1 * x1;

        float k2 = (-1) / k1;
        float b2 = y3 - k2 * x3;

        float tmpX = (b2 - b1) / (k1 - k2);  // AB中点
        float tmpY = k1 * tmpX + b1;

        float vec0 = x3 - tmpX;
        float vec1 = y3 - tmpY;
        double sidLen = Math.sqrt(vec0 * vec0 + vec1 * vec1); // C到AB中点距离
        vec0 = (float) (vec0 / sidLen);      //OC单位向量
        vec1 = (float) (vec1 / sidLen);

        double angle;
        if (vec1 < 0 && vec0 > 0) angle = Math.PI / 2 - Math.acos(vec0);
        else if (vec1 < 0 && vec0 < 0) angle = Math.acos(-vec0) - Math.PI / 2;
        else if (vec1 >= 0 && vec0 > 0) angle = Math.acos(vec0) - Math.PI / 2;
        else angle = Math.PI / 2 - Math.acos(-vec0);

        Matrix matrix = new Matrix();
        matrix.postRotate((float) (-angle / Math.PI * 180));
        img = Bitmap.createBitmap(img, 0, 0, img.getWidth(), img.getHeight(),
matrix, true);

        List<Double> xy0 = onePoint(x0 - W / 2, y0 - H / 2, angle);
        x0 = (float) (xy0.get(0) + img.getWidth() / 2);
        y0 = (float) (xy0.get(1) + img.getHeight() / 2);

        Matrix matrix1 = new Matrix();
        matrix1.postScale((float) (224 / (unitLen * 5 / 2)), (float) (224 /
(unitLen * 5 / 2)));
        img = Bitmap.createBitmap(img,
                (int) Math.round(x0 - unitLen * 5 / 4),
                (int) Math.round(y0 + unitLen / 4),
                (int) Math.round(unitLen * 5 / 2),
                (int) Math.round(unitLen * 5 / 2),
                matrix1, true);
        Log.e("size", String.valueOf(img.getHeight()));
        Log.e("width", String.valueOf(img.getWidth()));
        return img;
    }

    public static Bitmap extractROI(Bitmap img, Box[] yoloRes, Boolean draw) {
        Integer H = img.getHeight(), W = img.getWidth();
        float x1 = -1;
        float y1 = 0;
        float x2 = 0;
        float y2 = 0;
        float x3 = 0;
        float y3 = 0;

        if (yoloRes.length != 3) {
            return null;
        }
        for (Box box : yoloRes) {
            if (box.getLabel().equals("double gap")) {
                if (x1 == -1) {
                    x1 = (box.x0 + box.x1) / 2;
                    y1 = (box.y1 + box.y0) / 2;
                } else {
                    x2 = (box.x0 + box.x1) / 2;
                    y2 = (box.y1 + box.y0) / 2;
                }
            } else {
                x3 = (box.x0 + box.x1) / 2;
                y3 = (box.y1 + box.y0) / 2;
            }
        }
        if (x2 == 0 || x3 == 0) {
            return null;
        }

        float x0 = (x1 + x2) / 2;
        float y0 = (y1 + y2) / 2;


        double unitLen = Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));

        float k1 = (y1 - y2) / (x1 - x2); // line AB
        float b1 = y1 - k1 * x1;

        float k2 = (-1) / k1;
        float b2 = y3 - k2 * x3;

        float tmpX = (b2 - b1) / (k1 - k2);  // AB中点
        float tmpY = k1 * tmpX + b1;

        float vec0 = x3 - tmpX;
        float vec1 = y3 - tmpY;
        double sidLen = Math.sqrt(vec0 * vec0 + vec1 * vec1); // C到AB中点距离
        vec0 = (float) (vec0 / sidLen);     //OC单位向量
        vec1 = (float) (vec1 / sidLen);

        double angle;
        if (vec1 < 0 && vec0 > 0) angle = Math.PI / 2 - Math.acos(vec0);
        else if (vec1 < 0 && vec0 < 0) angle = Math.acos(-vec0) - Math.PI / 2;
        else if (vec1 >= 0 && vec0 > 0) angle = Math.acos(vec0) - Math.PI / 2;
        else angle = Math.PI / 2 - Math.acos(-vec0);

        Matrix matrix = new Matrix();
        matrix.postRotate((float) (-angle / Math.PI * 180));
        Bitmap newimg = Bitmap.createBitmap(img, 0, 0, img.getWidth(),
img.getHeight(), matrix, true);
```

```


        Canvas canvas = new Canvas(img);
        Paint boxPaint = new Paint();
        float strokeWidth = 4 * (float) img.getWidth() / 800;
        float textSize = 30 * (float) img.getWidth() / 800;
        boxPaint.setColor(Color.argb(255, 255, 165, 0));
        boxPaint.setAlpha(255);
        boxPaint.setTypeface(Typeface.SANS_SERIF);
        boxPaint.setStyle(Paint.Style.STROKE);
        boxPaint.setStrokeWidth(strokeWidth);
        boxPaint.setTextSize(textSize);
        canvas.drawLine(x1, y1, x2, y2, boxPaint);
        canvas.drawLine(x0, y0, x3, y3, boxPaint);

        List<Double> xy0 = onePoint(x0 - W / 2, y0 - H / 2, angle);
        x0 = (float) (xy0.get(0) + newimg.getWidth() / 2);
        y0 = (float) (xy0.get(1) + newimg.getHeight() / 2);

        Matrix matrix1 = new Matrix();
        matrix1.postScale((float) (224 / (unitLen * 5 / 2)), (float) (224 /
(unitLen * 5 / 2)));
        try {
            newimg = Bitmap.createBitmap(newimg,
                    (int) Math.round(x0 - unitLen * 5 / 4),
                    (int) Math.round(y0 + unitLen / 4),
                    (int) Math.round(unitLen * 5 / 2),
                    (int) Math.round(unitLen * 5 / 2),
                    matrix1, true);
        } catch (Exception e) {
            return null;
        }

        return newimg;
    }

```

Palmprint recognition

In addition to verification of a single user, our system also supports multi-user palmprint verification. When a user registers his/her palmprint, he/she is required to fill in his/her name. When verifying, the system compares the palmprint with all the palmprint data stored in the database and find the user's identity. This can be applied to identify recognition in payment scenarios (similar to AliPay's face recognition payment in supermarket).

code:

```
@Override
```

```java
        public void analyze(ImageProxy image, final int rotationDegrees) {
            final Bitmap bitmapsrc = imageToBitmap(image);  // 格式转换
            Thread detectThread = new Thread(() -> {
                Matrix matrix = new Matrix();
                matrix.postRotate(rotationDegrees);
                width = bitmapsrc.getWidth();
                height = bitmapsrc.getHeight();
                Bitmap bitmap = Bitmap.createBitmap(bitmapsrc, 0, 0, width, height, matrix, false);

                startTime = System.currentTimeMillis();
                Box[] result = YOLOv4.detect(bitmap, threshold, nms_threshold);
                endTime = System.currentTimeMillis();

                final Bitmap mutableBitmap = bitmap.copy(Bitmap.Config.ARGB_8888, true);
                float strokeWidth = 4 * (float) mutableBitmap.getWidth() / 800;
                float textSize = 30 * (float) mutableBitmap.getWidth() / 800;

                Canvas canvas = new Canvas(mutableBitmap);
                boxPaint.setAlpha(255);
                boxPaint.setTypeface(Typeface.SANS_SERIF);
                boxPaint.setStyle(Paint.Style.STROKE);
                boxPaint.setStrokeWidth(strokeWidth);
                boxPaint.setTextSize(textSize);
                for (Box box : result) {
                    boxPaint.setColor(box.getColor());
                    boxPaint.setStyle(Paint.Style.FILL);
                    String score = Integer.toString((int) (box.getScore() * 100));
                    canvas.drawText(box.getLabel() + " [" + score + "%]",
                            box.x0 - strokeWidth, box.y0 - strokeWidth
                            , boxPaint);
                    boxPaint.setStyle(Paint.Style.STROKE);
                    canvas.drawRect(box.getRect(), boxPaint);
                }

                String res;
                Bitmap roi = Util.extractROI(mutableBitmap, result, true);

                if (Util.names.size() == 0) {
                    res = "请录入掌纹";
                } else {
                    if (roi == null) {
                        res = "请将摄像头对准手掌";
                    } else {
                        final Tensor inputTensor = TensorImageUtils.bitmapToFloat32Tensor(roi,
                                TensorImageUtils.TORCHVISION_NORM_MEAN_RGB, TensorImageUtils.TORCHVISION_NORM_STD_RGB, MemoryFormat.CHANNELS_LAST);

                        // running the model
                        final Tensor outputTensor = module.forward(IValue.from(inputTensor)).toTensor();

                        float[] vec = outputTensor.getDataAsFloatArray();
                        float sum = 0;
                        for (float tmp : vec) {
                            sum += tmp * tmp;
                        }
                        sum = (float) Math.sqrt(sum);

                        List<Double> dot = new ArrayList<>();
                        double tmpDot = 0f;
                        for (double[] vect : Util.vecs) {
                            for (int i = 0; i < vect.length; i++) {
                                tmpDot += vec[i] / sum * vect[i];
                            }
                            dot.add(tmpDot);
                            tmpDot = 0f;
                        }
                        double maxDot = -1;
                        for (int i = 0; i < dot.size(); i++) {
                            if (dot.get(i) > maxDot) {
                                maxDot = dot.get(i);
                            }
                        }
                        String name = Util.names.get(dot.indexOf(maxDot));

                        endTime = System.currentTimeMillis();
//                        String res1 = dot > 0.432 ? "掌纹匹配成功" : "掌纹匹配失败";
                        res = String.format(Locale.CHINESE,
                                "%s，你被我认出来了! \n相似度: %.3f\nImgSize: %dx%d\nUseTime: %d ms",
                                name, maxDot, mutableBitmap.getHeight(), mutableBitmap.getWidth(), endTime - startTime);

                    }

                }
                runOnUiThread(() -> {
                    resultImageView.setImageBitmap(mutableBitmap);
                    tvInfo.setText(res);
                });
            }, "detect");
            detectThread.start();
        }
```

Accuracy

The MPD ROI dataset contains 16,000 images from 400 hands of 200 persons. We used images from 160 subjects for training and images from the remaining 40 subjects for testing. We performed **10-fold cross validation** on test set (use 9 folds to obtain the best threshold and test on the remaining 1 fold), the average accuracy is **98.86%**.

For multi-person recognition, we performed Top-1 test. For each palm, randomly select an image as the registration image, take the rest of the images as input, and compare the selected image with the registration images one by one to find the palm from which the input image is taken. **The accuracy of top-1 test is 97.70%**.

For palmprint recognition, we determine the threshold of dot product based on the False Acceptance Rate (FAR). The table shows the thresholds and corresponding True Positive Rates (TPR) when FAR takes different values. **For the sake of security**, we chose the threshold corresponding to **FAR=1e-4** as the final threshold to build a smartphone palmprint verification application, i.e. threshold = 0.432. The corresponding TPR is 91.4%.

FAR	Threshold	TPR
1e-1	0.1726	0.998
1e-2	0.2979	0.986
1e-3	0.4222	0.921
1e-4	**0.4320**	**0.914**

Advantages and disadvantages

Advantages

1. High security

   Palmprint contains more information and is more specific than face (especially when wearing a mask). Therefore, palmprint verification is more secure than face verification. Since the algorithm is efficient, the whole computation precess is completed on local device. No network transmission is required, which ensures security and privacy.

2. Robust to variations

   Palmprint verification is very robust to kinds of variations. The algorithm can work normally in different Illumination conditions, with sightly different posture and changes over time.

3. Wide application scenarios

   Palmprint verification is a very promising authentication technology and has a wide range of application scenarios. It can be used to unlock device, pay, check-in and in other scenarios requiring identity authentication. And it is not affected by wearing masks during the pandemic.

Disadvantages

1. The recognition speed is slow

   Currently it takes more than a hundred milliseconds to identify an image, so only 5-10 frames per second can be identified. This speed is not up to the standard of practical application.

2. The recognition accuracy is low

   Because the training of neural network needs a lot of data, but the palmprint data set is small at present, so the training effect of network has not reached the ideal state.

Possible Improvement

1. Speed up identification

   Currently it takes more than a hundred milliseconds to identify an image, so only 5-10 frames per second can be identified. In the later stage, network structure and computing performance should be optimized to achieve a speed of more than 15 frames per second.

2. Improve recognition accuracy

   At present, there are few palmprint data sets, so the effect of network training has not reached the ideal state. More palmprint data could be collected in the future to train the network so that the recognition accuracy is usable.