Weekly Report (16/04-22/04)

ZHOU, Zhenyu

April 22, 2022

1 Back Propagation of Image Blocks Translation & Rotation

From the previous formula, it can be seen that $\frac{\partial IBTRTL}{\partial F(I^a)}$ can not be differentiable because the feature map is divided to image blocks result in different coefficient with different image blocks. But from below formula, the $\frac{\partial IBTRTL}{\partial F(I^a)B_{mn}}$ can be differentiable. The B_{mn} is the image block, while m represents the image block sequence number along the x-axis direction, n is the image block sequence number along the y-axis direction. Then the pixels on the image blocks can be back propagation along neural network.

$$IBTRTL = \frac{1}{I}\sum_{i=1}^{I}[L(F(I_i^a), F(I_i^p)) - L(F(I_i^a), F(I_i^n)) + m]_+ \tag{1}$$

As for the $L()$ function, it is used to get the sum of minimal MSE value of each image block that will be shifted and rotated in a range.

$$L(F_1, F_2) = \sum_{m=1}^{M}\sum_{n=1}^{N} \min_{-Ws\le w\le Ws, -Hs\le h\le Hs, -\Theta_r\le\theta\le\Theta_r} D_{w,h,\theta}(F_1B_{mn}, F_2B_{mn}) \tag{2}$$

And the $D()$ is to calculate the MSE of two input feature maps, just as show as below.

$$D_{w,h,\theta}(F_1B_{mn}, F_2B_{mn}) = \frac{1}{|C_{w,h,\theta}|}\sum_{(x,y)\in C_{w,h,\theta}} (F_1B_{mn}^{(w,h,\theta)}[x,y] - F_2B_{mn}[x,y])^2 \tag{3}$$

In terms of $C_{w,h,\theta}$, it represents the common region between two feature maps after shifting along x-axis with w, shifting along y-axis with h, and rotating with a. In this kind of situation, we assume that we shift and rotate anchor w_{ap}, h_{ap}, a_{ap} with positive sample, and shift and rotate anchor w_{an}, h_{an}, a_{an} with negative sample can get minimal MSE. Then the partial differentiation of the loss function with respect to each variable can be written as follows:

$$\frac{\partial Loss}{\partial F_i^p} = \begin{cases} 0, if(x,y) \notin C_{w_{ap},h_{ap},a_{ap}} \ or \ Loss = 0 \\ \frac{-2(F_i^a[[x_{w_{ap}}, y_{h_{ap}}] * M(a_{ap})] - F_i^p[x,y])}{N|C_{w_{ap},h_{ap},a_{ap}}|}, otherwise \end{cases} \tag{4}$$

The $M(a_{ap})$ is the rotation matrix:

$$\begin{bmatrix} cos(rot(a_{ap})) & -sin(rot(a_{ap})) \\ sin(rot(a_{ap})) & cos(rot(a_{ap})) \end{bmatrix}$$

$$\frac{\partial Loss}{\partial F_i^n} = \begin{cases} 0, if(x,y) \notin C_{w_{an},h_{an},a_{an}} \ or \ Loss = 0 \\ \frac{-2(F_i^a[[x_{w_{an}}, y_{h_{an}}] * M(a_{an})] - F_i^n[x,y])}{N|C_{w_{an},h_{an},a_{an}}|}, otherwise \end{cases} \tag{5}$$

The $M(a_{an})$ is the rotation matrix:

$$\begin{bmatrix} cos(rot(a_{an})) & -sin(rot(a_{an})) \\ sin(rot(a_{an})) & cos(rot(a_{an})) \end{bmatrix}$$

$$\frac{\partial Loss}{\partial F_i^a[x,y]} = -\frac{\partial Loss}{\partial F_i^p[[x - w_{ap}, y - h_{ap}] * M(-a_{ap})]} + \frac{\partial Loss}{\partial F_i^n[[x - w_{an}, y - h_{an}] * M(-a_{an})]} \tag{6}$$

2 Experiments and Results

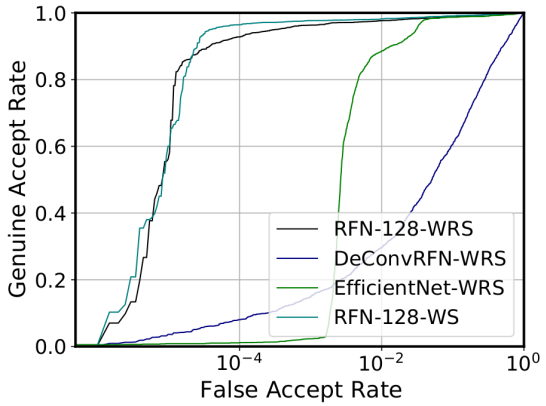
Firstly, all models are pre-trained on the Finger Knuckle V1 Database, and then fine-tuning on the rest database. On the Finger Knuckle V1 Database, it contains 512 subjects, and each of subjects has 5 finger knuckle samples. In this kind of situation, I use three samples of these five samples to pre-train my models.

I also continue to train the DeConvRFN (change the RFN-128 convolution layer with deformable convolution) and EfficientNet, the loss

converged to the local minimal point. As for the EfficientNet, I change the original EfficientNet-B0 to fit my application. It has 9 stages in totally, the 9th stage is classification task with FC layer, so I replace it with convolution layer for output feature maps. Meanwhile, I delete the stage7 and stage8, and change stage3 and stage4 with stride 1.

2.1 Within-Database Experiments

2.1.1 Index Finger Knuckle of Hand Dorsal Image Database

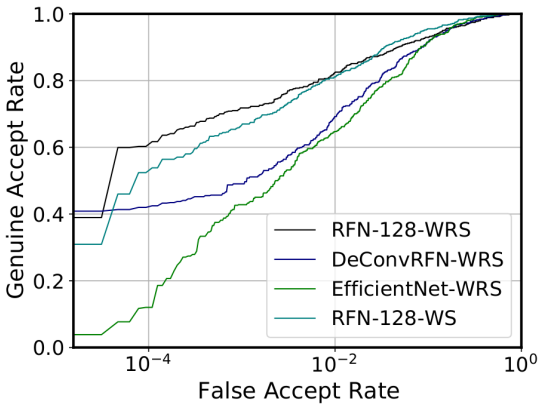


Model	Genuine	Imposter	EER
RFN-WRS	2848	2024928	0.02001
DeConvRFN-WRS	2848	2024928	0.23350
EfficientNet-WRS	2848	2024928	0.03386
RFN-WS	2848	2024928	0.01650

As for the experiment, the dataset totally contains 712 subjects, and I use the segmented Index finger knuckle as my dataset. And I fine-tuned my model on the first sample of each subject, and then use the rest four sample as the testing dataset. At the testing process, it has $712 * 4 = 2848$ genuine matching scores, and has $712 * 711 * 4 = 2024928$ imposter matching scores. The performance of RFN-128-WRS and RFN-128-WS is similar, but the RFN-128-WS is slightly better than RFN-128-WRS depend on the EER value.

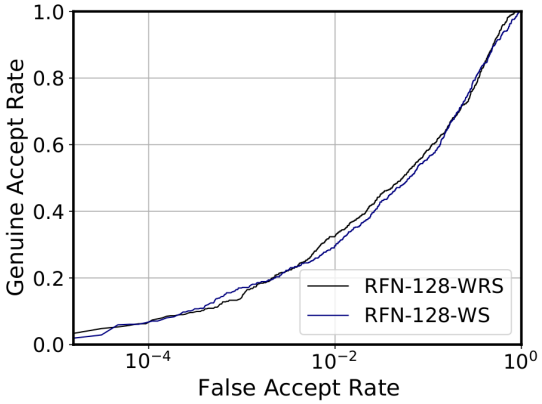
2.1.2 Finger Knuckle V3 Database with Deformation

The Finger Knuckle V3 Database have 1-104 subjects that have two session samples, and the rest subjects of first session 105-221 just is one session samples. So as the first experiment, I firstly fine-tuned my model on the one-session 1-104 dataset, and test on the another session 1-104 subjects. So it will have $104*6 = 624$ genuine matching scores, and have $104 * 103 * 6 = 64272$ imposter matching scores. From the below figure, if the false accept rate is below 10^{-2}, the RFN-128-WRS is better than the RFN-128-WS.



Model	Genuine	Imposter	EER
RFN-WRS	624	64272	0.08014
DeConvRFN-WRS	624	64272	0.09296
EfficientNet-WRS	624	64272	0.09377
RFN-WS	624	64272	0.06732

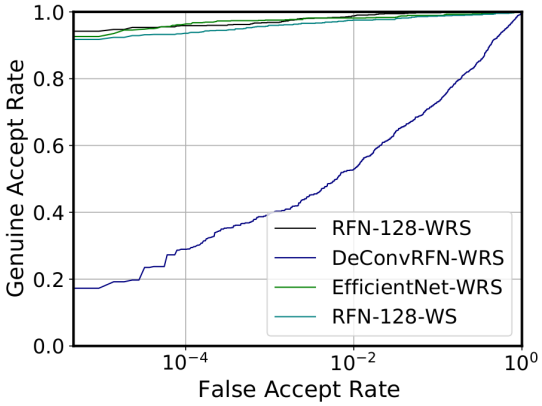
Because If I have a two session protocol, I should fine-tune my model on the 105-221 subjects, and use the 1-104 subjects to use two session protocol to evaluate my model performance. The result is shown as below, the matching performance is very low.



Model	Genuine	Imposter	EER
RFN-WS	624	64272	0.25802
RFN-WRS	624	64272	0.26924

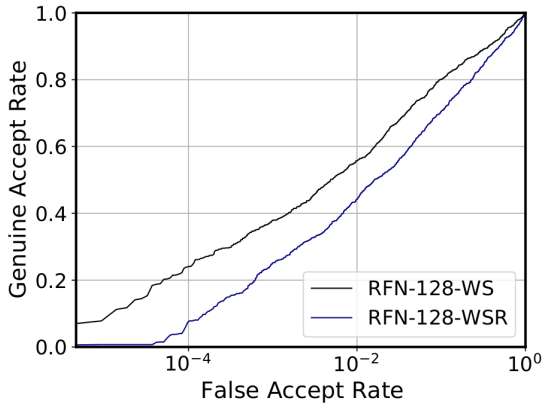
2.1.3 2D Samples of 3D Finger Knuckle Database

First experiment on the database is to use the one session 190 subjects image to fine-tune models and then to test on the another session 190 subjects. It has $190 * 6$ genuine matching scores and $190 * 189 * 6$ imposter matching scores. From the result, we can see that these RFN-128-WRS, RFN-128-WS, EfficientNet can get very high matching accuracy. Meanwhile, the RFN-WRS has the minimal EER value among these models.



Model	Genuine	Imposter	EER
RFN-WRS	1140	215460	0.01053
DeConvRFN-WRS	1140	215460	0.19299
EfficientNet-WRS	1140	215460	0.01754
RFN-WS	1140	215460	0.02281

And then use the two session protocol. I use the rest samples of session1, and it has 191-228 subjects. In this kind of situation, the training dataset is too small. The two session protocol will test on the 190 subjects, these subjects can offer two session samples. Due to the training set is too small, so the matching performance is not



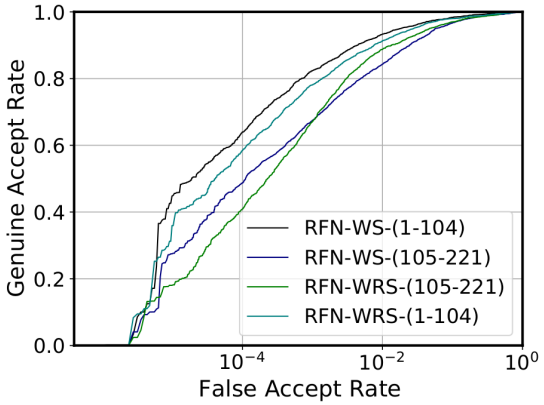
Model	Genuine	Imposter	EER
RFN-WS	1140	215460	0.15978
RFN-WRS	1140	215460	0.21053

2.2 Cross-Database Experiments

I firstly pre-trained my models on the Finger Knuckle V1 Database, and then fine-tuned models on the Finger Knuckle V3 Database (with deformable). I use these kind training method, and use these models to test performance on the Index Finger Knuckle of Hand Dorsal Image and Tsinghua Finger Knuckle Database as a cross database experiment. The label in the finger curve, the content in parentheses

indicates the training samples. Such as RFN-WS(1-104), it uses 1-104 subjects of Finger Knuckle V3 Database to train models.

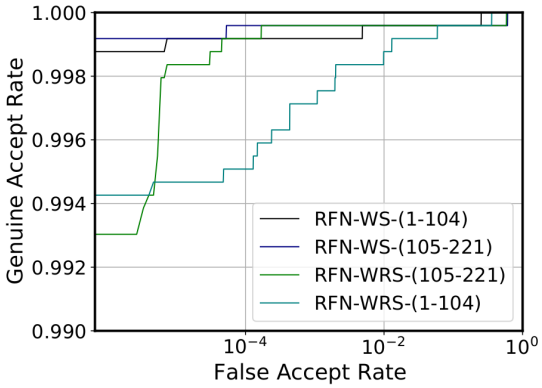
2.2.1 Index Finger Knuckle of Hand Dorsal Image



Model	Genuine	Imposter	EER
RFN-WS-(1-104)	3560	2531160	0.03508
RFN-WS-(105-221)	3560	2531160	0.05421
RFN-WRS-(105-221)	3560	2531160	0.04888
RFN-WRS-(1-104)	3560	2531160	0.03876

The database totally has 712 subjects, and each subject has 5 samples. Therefore, it will have 712∗5 genuine matching scores and 712∗711∗5 imposter matching scores. From the cure, the performance of RFN-WS and RFN-WRS is similar, and the RFN-WS is slightly better than RFN-WRS while using the same training samples.

2.2.2 Tsinghua Finger Knuckle Database



Model	Genuine	Imposter	EER
RFN-WS-(1-104)	2440	1485960	0.00082
RFN-WS-(105-221)	2440	1485960	0.00041
RFN-WRS-(105-221)	2440	1485960	0.00041
RFN-WRS-(1-104)	2440	1485960	0.00200

The database has 610 subjects, and each subject can offer 4 samples. Then as the cross database experiment, it will have $610 * 4$ genuine matching scores and $610 * 609 * 4$ imposter matching scores. In this database, all models can get very high matching performance from the table and figure.

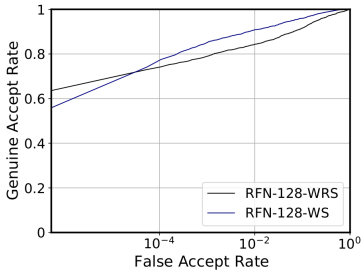
2.3 Challenging Protocol Experiments

I have a challenging protocol experiment on the 2d image of 3d finger knuckle database. The experiment protocol is similar with the "Experiments Results using Challenging Protocol".

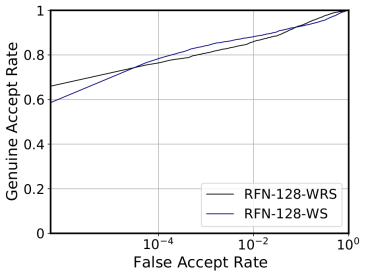
2.3.1 One-session

I pre-trained my model on Finger Knuckle V1 Database. As for the one-session experiment, I was firstly fine-tuning my model on the second session forefinger dataset which has 190 subjects, and then use the

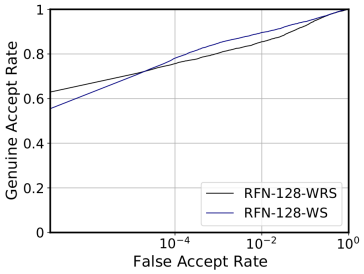
all-to-all protocol to evaluate matching accuracy on the first session of forefinger and middle finger, then combine both of them.



(a) Index Finger



(b) Middle Finger

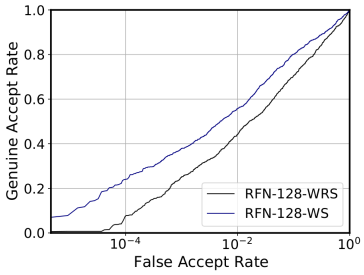


(c) Combine

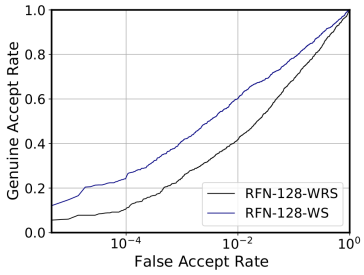
Finger	Model	Genuine	Imposter	EER
Combination	RFN-WRS	13680	7469280	0.08100
	RFN-WS	13680	7469280	0.06403
Forefinger	RFN-WRS	6840	1863226	0.08861
	RFN-WS	6840	1863226	0.05351
Middle	RFN-WRS	6840	1863226	0.07664
	RFN-WS	6840	1863226	0.07558

2.3.2 Two-session

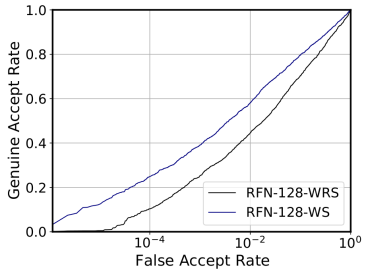
I also pre-trained model on Finger Knuckle V1 Database, and then trained model on the 191-228 subjects of first session of forefinger dataset. Using the rest 1-190 subjects samples as the testing dataset, I use the two session protocol to evaluate matching performance.



(a) Index Finger



(b) Middle Finger



(c) Combine

Finger	Model	Genuine	Imposter	EER
Combination	RFN-WRS	2280	864120	0.21138
	RFN-WS	2280	864120	0.16184
Forefinger	RFN-WRS	1140	215460	0.21053
	RFN-WS	1140	215460	0.15978
Middle	RFN-WRS	1140	215460	0.21603
	RFN-WS	1140	215460	0.16561

From the above result, the matching accuracy is very low result from the training dataset is too small. The training dataset just contains 191-228 subjects, 38 subjects. Meanwhile, the performance of forefinger, middle or combination is similar.

3 Key Points of Blood Vessel of Finger Nail

3.1 Coordinate Regression

Coordinate Regression directly uses the coordinates of the key points as the target that the final network needs to return to. In this case, the

direct position information of each coordinate point can be directly obtained.

3.2 Heatmap

The heat map here is a map, each channel in the picture represents the key points of a category, and there are several channels if there are key points of several categories. The key points position on a channel map is a two-dimensional Gaussian distribution cantered on it, and the pixel value of the remaining positions is 0. The result of network prediction is also a heat map. The most direct way to extract the coordinates is to extract the point in a channel whose pixel response is greater than a certain threshold and has the largest response. The coordinates of this point are the coordinates of the key points of the category.

3.3 Heatmap & Offsets

Different from pure Heatmap, Google's Heatmap refers to that the probability value of all points within a certain range from the target key points is 1. Outside Heatmap, use Offsets, that is, the offset to indicate the relationship between the pixel position within a certain range from the target key points and the target key points.

3.4 Next Week Plans

Because our dataset is labelled by coordinates, for a quick test, we can firstly use the coordinate regression method. I have listed a several models can regress the key points coordinate:

1. MTCNN (Multi-task Cascaded Convolutional Networks)
https://github.com/ipazc/mtcnn
2. DeepPose (Human Pose Estimation via Deep Neural Networks)

The heatmap network directly returns the probability of each type of key points. To a certain extent, each point provides supervision information. The network can converge quickly. At the same time, predicting the position of each pixel can improve the positioning accuracy of key points. In addition, practice has proved that Heatmap

is indeed much better than Coordinate. But the label should also be a heatmap, so we should change the label type.

1. HRNet
https://github.com/leoxiaobin/deep-high-resolution-net.pytorch
2. OpenPose, SimpleBaselines, and HigerHRNet etc.