Fingerprint Classification Using Convolutional Neural Networks and Ridge Orientation Images

John M. Shrein
The University of Memphis
Memphis, TN 38152, USA
jmshrein@memphis.edu

***Abstract*—Deep learning is currently popular in the field of computer vision and pattern recognition. Deep learning models such as Convolutional Neural Networks (CNNs) have been shown to be extremely effective at solving computer vision problems with state-of-the-art results. This work uses the CNN model to achieve a high degree of accuracy on the problem of fingerprint image classification. It will also be shown that effective image preprocessing can greatly reduce the dimensionality of the problem, allowing fast training times without compromising classification accuracy, even in networks of moderate depth. The proposed approach has achieved 95.9% classification accuracy on the NIST-DB4 dataset with zero sample rejection.**

Keywords—Convolutional Neural Network, CNN, Machine Learning, Fingerprint Classification, Biometrics

I. Introduction

Fingerprint recognition is a common real-world task in applications such as law enforcement, forensics, and biometric identification. The IAFIS (Integrated Automated Fingerprint Identification System) database, maintained by the FBI, stores over 100 million fingerprints. After receiving a fingerprint identification request, it takes roughly 30 minutes for the FBI computers to return a response [1]. While fingerprint classification by itself is not a method of identifying an individual, accurate classification methods may speed up the process of identification by reducing the overall search space, thus reducing both wait time and computational overhead. In fact, the FBI itself states in its NBIS fingerprint image software guide that "Identifying a fingerprint's class effectively reduces the number of candidate searches required to determine if a fingerprint matches a print on file. These […] strategies are critical for the FBI to manage the searching of its fingerprint repository [2]."

The usefulness of fingerprint classification in modern applications was discovered and utilized by William Herschel in 1858, who realized that fingerprints were both unique and permanent throughout an individual's lifetime. In 1897, Sir Edward Richard Henry and two Bengali police officers devised a taxonomy for fingerprint classification that is still used today. The Henry classification scheme was extremely useful when all classification and identification were done by hand, but the speed of automated algorithms allows fingerprints to be classified and identified much faster, albeit with a less complex classification scheme. Most modern fingerprint classification algorithms are derived from Henry's original classification system but are limited to five classes (whorl, left loop, right loop, arch, and tented arch), shown in figure 1.

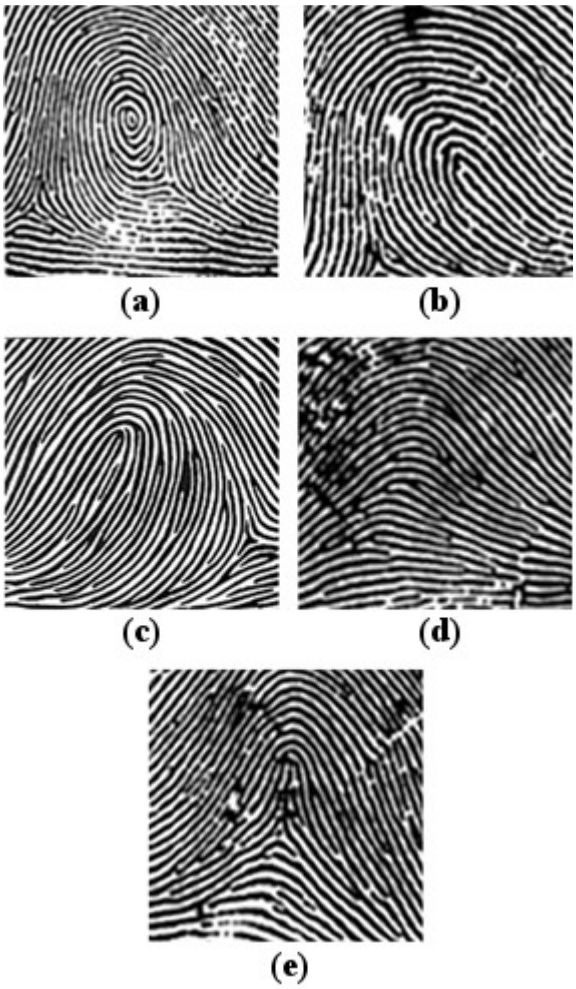

Fig. 1. Fingerprint classes: (a) Whorl, (b) right loop, (c) left loop, (d) arch, (e) tented arch

These types of fingerprints are not distributed uniformly in nature. Table I shows the natural distribution of these five classification patterns [4]. It also shows that most human fingerprints fit into the whorl and loop class patterns, which

978-1-5386-2726-6/17/$31.00 ©2017 IEEE

make up about 93.4% of all fingerprints. Because the arch and tented arch classes make up only 6.6% of the fingerprint patterns found naturally, and because these two types of prints are very similar, researchers often combine these into a single class for the task of classification.

TABLE I. DISTRIBUTION OF FINGERPRINT PATTERNS FOUND IN NATURE [4]

Whorl	27.9%
Left Loop	33.8%
Right Loop	31.7%
Arch	3.7%
Tented Arch	2.9%

Without the use of any classification scheme, the worst-case search space size for identification contains N fingerprint images that must be considered. By utilizing an accurate classification method, the worst-case search space size drops to $\sim N/3$. However, many AFIS systems use all ten fingerprints to fully identify an individual, giving a much larger search space of $(N)^{10}$ with no classification and $(N/3)^{10}$ with classification, resulting in a considerable reduction in time and computation.

II. RELATED WORK

In [5], a comparison of classification approaches was conducted. Many feature extraction methods were reviewed and re-implemented with machine learning-based classifiers including decision trees, support vector machines using polynomial, radial basis functions, and Pearson VII Function-Based Universal kernels, K-nearest neighbor for k = 5 and k = 10 and using Euclidean and Heterogeneous Value Difference Metric (HVDM), and fixed rule based methods. The majority of these methods use NIST special database 4 [3], though several use NIST special database 9 [11] and NIST special database 14 [12]. The authors of [5] created three additional fingerprint databases that are synthetically generated from the SFinGe [13] tool for more thorough testing. These datasets were not used for testing in this work due to lack of availability and prohibitive cost of the required software.

The authors of [5] also create a series of ensemble classifiers constructed from the various models from previous works. The ensemble classifiers take either majority decision or consensus to decide the class of each input image. This strategy achieves 98.69% accuracy on NIST-DB4-F and 98.45% accuracy on NIST-DB4-S. However, with 30.24% and 34.97% rejection respectively on these datasets, about one third of the samples seen are discarded, making this strategy too fragile to be used in a practical setting. With rejection at 5.45%, 89.26% accuracy is achieved on NIST-DB4-S and with rejection at 5.52%, 90.45% accuracy is achieved on NIST-DB4-F. To put the rejection parameter in perspective, recall that the natural distribution of arch and tented arch prints combined makes up 6.6% of all prints. Therefore, a 10% rejection rate could conceptually eliminate these two classes and small portion of another class and a 34.97% rejection rate eliminates more prints than are in the largest class.

In [10], a convolutional neural network approach to fingerprint classification was implemented and tested. Their baseline model was able to achieve an accuracy of 90.73 on NIST-DB4_F and 88.91 on NIST_DB4_S using zero rejection and non-ambiguous data (samples with two class labels). One of the claims by the authors is that smaller, shallower nets such as the LeNet model [8] lack the expressiveness needed to adequately solve the fingerprint classification problem. In the work presented below, it will be shown that shallower nets can in fact be trained to solve this problem with a high degree of accuracy. A distinction must be made that the authors of [10] use 227x227 pixel images. The preprocessing steps, if any, used by the authors are not discussed, but it is mentioned that the images were 'fitted' to 227x227x pixels, perhaps by down-sampling, cropping, or both. However, in this work, normalized orientation angle images with 32x32 pixels are used as input to the neural network, allowing for a smaller network depth and vastly fewer parameters that must be learned. In [10], it may be the case that much less upfront time for preprocessing is needed, but the time to train the network is almost certainly longer (with other factors such as hardware and software packages being the same).

In [4], a multilayer perceptron neural network model is used to classify fingerprints. Considerable preprocessing is done to binarize the input images using the "ridge-valley" algorithm described in [20]. A FFT filter is applied to the binarized image to improve quality. An extension of the "ridge-valley" binarization algorithm is used to find ridge directions in the filtered fingerprint image. These ridge directions make up the features that are used for learning. This results in 1680 directional features which are then reduced to 64 or 96 using the Karhunen-Loeve (K-L) Transform. The model in [4] achieves 90.2% accuracy with 10% rejects. Several strategies, such as a priori distribution of fingerprint classes and reduction of the NIST-DB4 dataset to match the natural distribution, are used to increase this probability to 95.4% with 10% rejects.

In [32], Michelsanti, et al. use transfer learning to repurpose a pre-trained deep convolutional neural network for the task of fingerprint classification. Transfer learning seems to present a viable method of training larger networks on datasets with a limited number of images as fewer images, such as those in NIST-DB4, can be used to fine-tune the network. They mention that their method avoids timely preprocessing, though the input size to the network is 224x224 and it is stated that the images are cropped to isolate the center region, though it is not mentioned if the cropped region is centered on the region of interest or if all images are equally cropped. Two networks were trained, VGG-S and VGG-F, presumably corresponding to the F and S partitions of NIST-DB4, though which network was trained and tested on each set was not mentioned. Even though pre-trained networks were used, fine-tuning took around 9 hours for VGG-S and 30 hours for VGG-F. The authors achieve 94.4% accuracy and 95.05% accuracy. Whereas in this work, only the non-ambiguously labeled data was used for training and testing, the authors of [32] train their networks using only one

label but test using either label (i.e., the classification is considered correct if the network chooses either label). Lastly, only the 4-class classification problem is considered and no results are published for the 5-class problem. In the method proposed below, an accuracy of 95.0% and 95.9% was achieved on the 4-class classification problem.

III. Data Preprocessing

A. Database Selection

To choose an adequate database to use for training, several qualities (number of samples, existence of class labels, transformation consistency, luminosity consistency, region of interest isolation, etc.,) were used to evaluate and choose from the many public databases that are available. Ultimately, NIST-DB4 was chosen because the image quality is acceptable, there are class labels for each fingerprint, there are 4000 samples, it is freely available, and many existing approaches also use it. The primary fault in NIST-DB4 is that 17.5% of the labels are erroneous or ambiguous. Additionally, there are only 2000 unique prints, each of which were sampled twice, which is not ideal because 4000 samples is a very limited number of samples with which to train and test a convolutional neural network. Typically, many more images are preferred for adequate training. Furthermore, researchers tend to remove the ambiguously labeled prints, reducing the number of unique prints to 1650 (3300 total). Lastly, the prints from NIST-DB4 are rolled ink imprints that were later scanned. Modern fingerprint scanning devices vary greatly in quality, and though image quality may not be perfect, ink smudging is no longer a problem.

Many of the existing approaches to fingerprint classification use the orientation field or singular points as features for classification. This work uses orientation angles as the primary feature. However, instead of using orientation angles to generate an orientation field, the angles themselves are normalized and stored as MxM images, referred to in this work as "orientation angle images", where M represents the width or height of the image (assumed here to be square for simplicity) divided by the local window size N (typically 8 or 16). The term "orientation image" typically refers to images in which NxN pixel blocks, or windows in which the local orientation is calculated, are preserved and in which the orientation angle is represented as an oriented line in the range $[-\pi/2, \pi/2]$. This distinction is necessary because the CNN operates on normalized pixel representations of orientation angles instead of a set of vectorized orientations, such as those used in PCASYS and other methods. Figures 2 and 3 illustrate the difference between these two types of images.

By pre-calculating the local orientations, the image size, and therefore the dimensionality of the training data, is significantly reduced. In the case of NIST-DB4, the reduction is from 262144 (512 x 512) dimensional vectors to 4096 (64 x 64) or even 1024 (32 x 32) dimensional vectors. This greatly enhances the feasibility of training, especially with smaller datasets and fewer samples. The number of model parameters (weights) is also significantly reduced, requiring fewer computational resources and less time to train the network. Even when using 32 x 32 pixel images, the number of parameters that must be trained may be several hundred thousand. With 512 x 512 pixel images, there can easily be millions of parameters, even for shallower networks.

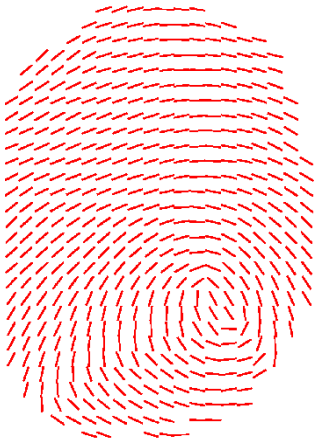

Fig. 2. Fingerprint orientation field

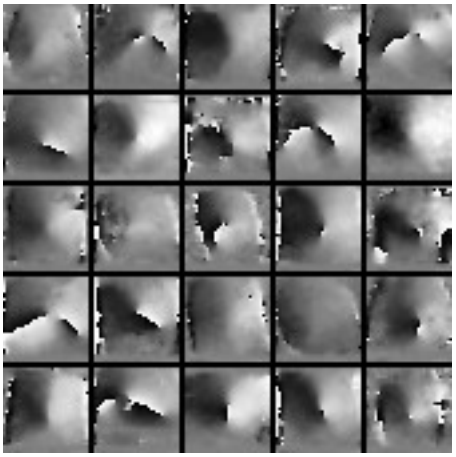

Fig. 3. 5x5 grid of orientation angle images extracted using Rao's method [11] representing 25 images of varying classes

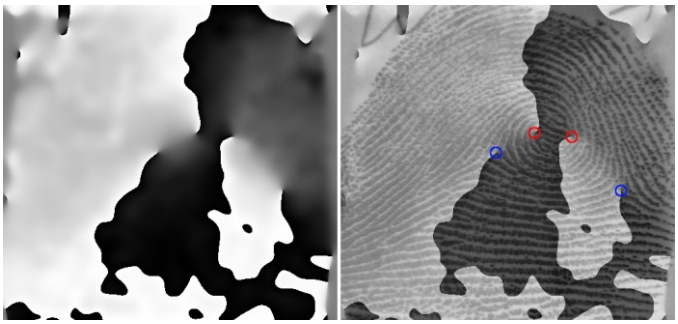

Fig. 4. (left) Orientation angle image using Thai's method [29]. (right) Correspondence of large changes in orientation angle and singular points.

Singular points, which have been used as the sole feature for classification in some works [22, 23, 24, 25, 26, 27], are a derivative feature that are generally calculated from the orientation angles, commonly by using the Poincarè method [28]. Therefore, such singular points are "encoded" within the orientation angle images and manifest visually as extreme changes in angle, which is exactly what the Poincarè index

measures. From orientation angle images, morphological processing techniques alone can be used to locate singular points in fingerprint images (though distinguishing between types of singular points may still require analytical methods). Figure 4 shows a visual correlation between singular points and friction ridge orientation.

One of the strengths of convolutional neural nets is their ability to learn which features are useful for classification without explicit feature engineering. In this work, even though the orientation angles are pre-calculated and fed to the network as images and no singular points are explicitly calculated, some of the convolutional filters must invariably model singular points.

B. Fingerprint orientation angle extraction

Several methods of orientation angle extraction exist. In [11], Rao describes one classical approach based on image gradient estimation. In [29], Thai also uses a gradient-based approach that generally results in smoother images than Rao's. However, during testing, the CNN tended to achieve higher performance with images obtained using Rao's approach. Figures 3 and 4 show examples of orientation angle images computed using Rao's method [11] and Thai's method [29], respectively. As described below and shown in figure 5, there are five primary steps involved in Rao's algorithm.

These steps are described in more detail below. First, the raw image is smoothed using a Gaussian filter given by:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2+y^2}{2\sigma^2}} \quad (1)$$

Second, the gradient of the image is estimated using horizontal and vertical Sobel filters given by:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (2)$$

These filters are convolved with the smoothed 2D image. From the resulting 2D gradient image, the orientations at each point can be computed using the formula:

$$\tan^{-1}\left[\frac{S_y}{S_x + \varepsilon}\right], \text{where } \varepsilon \text{ is a small constant.} \quad (3)$$

After computing the orientations for each point, the local dominant orientation can be calculated by averaging the orientation over an $N \times N$ block. The Gaussian filter is then applied once again to smooth the final image.

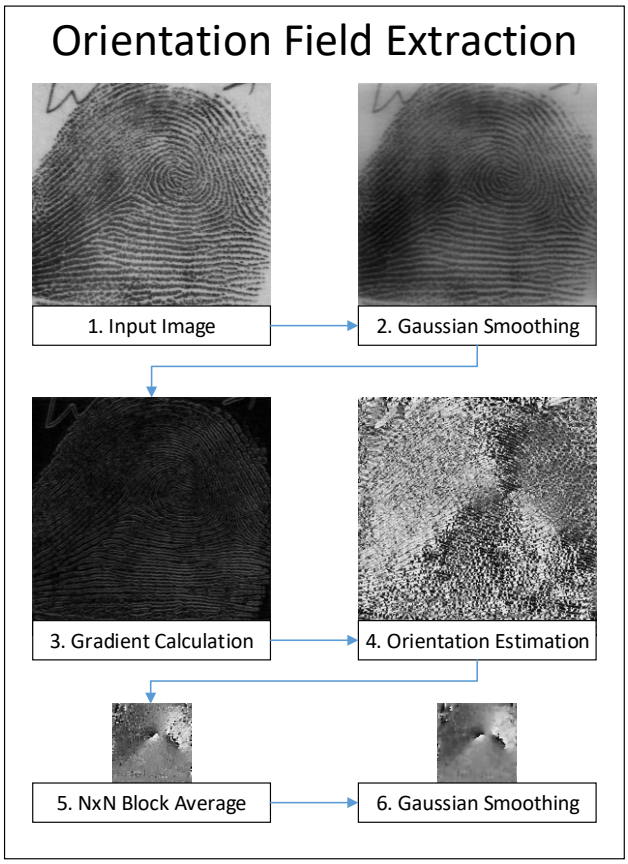

Fig. 5. Visualization of orientation angle extraction using Rao's approach [11]

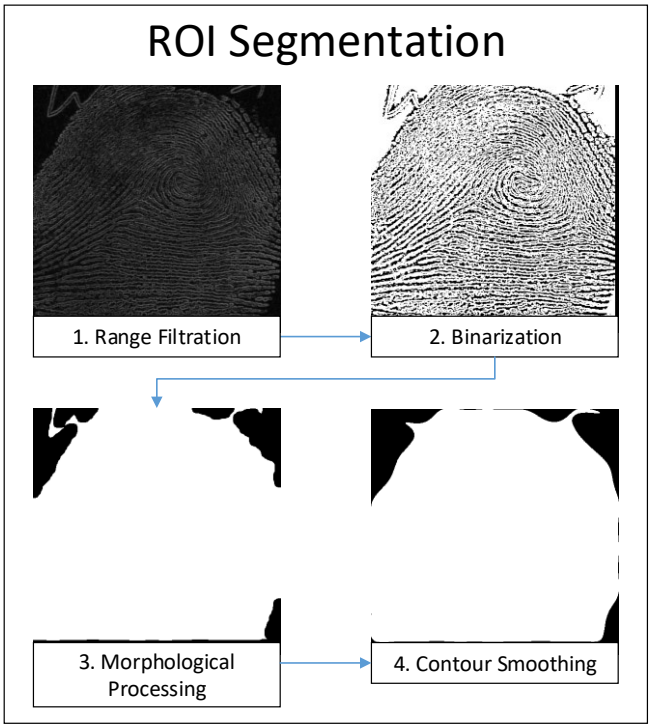

Fig. 6. ROI Segmentation of fingerprint images using techniques described in [30]

Optionally, region of interest (ROI) segmentation may be used to remove the background portions of the image. The steps involved are shown in figure 6, however discussion of this process is limited because network training and testing was found to be equally or more effective without using any segmentation.

C. *Data augmentation and normalization*

To facilitate network training, the mean value of each pixel over the entire population was calculated and subtracted from that pixel (feature-wise centering) and each pixel was divided by the standard deviation of the corresponding pixels over the entire population (feature-wise normalization).

Data augmentation was used to generate additional images from the existing set of training samples. Data augmentation is a powerful strategy that makes the best possible use of each training sample by performing minute transformations to the image, effectively creating new training images. The augmentation parameters used in this case include random rotations in the range [-15 degrees, 15 degrees], vertical and horizontal translations expressed as a fraction of the width and height of the image in the range [-0.2, 0.2], shearing expressed in radians in the range [-0.2, 0.2], and zooming (enlargement or reduction) in the range of [.8, 1.2]. The image was not flipped vertically or horizontally as this may unintentionally confuse the model as to which class the image belongs to (i.e., a horizontal flip of a left loop becomes a right loop). 1000 augmented images were created for each training sample in real-time during training.

IV. CNN Model Architecture

Convolutional Neural Networks (CNN) have proven to be very effective at image classification tasks including classic problems such as MNIST handwritten digit recognition, CIFAR-10, and CIFAR-100, as well as offering state-of-the-art solutions to real world problems such as facial recognition [15], pose estimation [16], motion synthesis [17], colorization of gray-scale images [18], and many others. The basic CNN architecture upon which this work is modeled is shown in figure 7. Discussion of the operational characteristics of CNNs will be limited in this work as they are well described in many other publications [6,7,8].

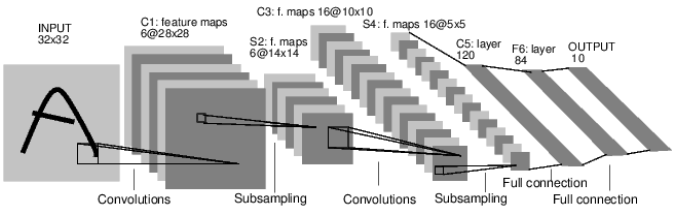

Fig. 7. Architecture of Lenet-5, an archetypal CNN model [8]

Several software packages are available that provide a framework for implementing neural networks. The work presented here uses Keras, which is a Python library that provides easy to use abstractions to powerful deep learning libraries such as Theano and Tensorflow (used here). The CNN used in this work uses the construction shown in figure 8. The types of layers used are described as follows:

- Convolutional Layer: Each convolutional layer consists of a bank of filters (weights) that are convolved with the output of the previous layer, or the input image if it is the first layer, to produce some response.
- Max-Pooling Layer: The max-pooling layer performs sub-sampling on the outputs generated by the previous convolutional layer(s) by selecting the maximum value in an MxM window. In this work, all max-pooling layers are 2x2 with a stride of 2, thus reducing the size of the output by 2.
- Fully Connected Layer: The fully connected layer connects a set of neurons to each of the neurons of the previous layer. The model shown above has 128 3x3 feature maps in the last max-pooling layer that ultimately form 8192 neurons connected to the 32 neurons in the proceeding fully connected layer for a total of 262176 weights (32 bias weights).

The Keras API provides several types of additional "layers" that simplify certain aspects of implementation. In the model shown in figure 8, the flatten layer simply vectorizes and connects all of the neurons from the outputs of the previous layer to all of the neurons in the fully connected layer. The dropout layer specifies that some fraction of inputs to the next layer should be turned off during the current training step, thus nullifying their contribution to the next layer. Ideally, this helps to prevent overfitting of the model and attempts to ensure that there are multiple independent representations of the input being learned.

All of the convolutional layers and the first fully connected layer use the rectified linear activation function (ReLu):

$$f(x) = \begin{cases} 0 \; if \; x \leq 0 \\ x \; if \; x > 0 \end{cases} \tag{4}$$

ReLu activation has been shown to reduce the need to perform unsupervised pre-training in deep neural networks. [19] and is the most widely used activation function used in modern deep architectures [21].

The softmax function, given as:

$$P(y = j) = \frac{e^{z_j}}{\sum_k e^{z_k}}, for \; j = 1, \ldots, K \tag{5}$$

scales the CNN outputs such that they are all in the range [0,1] and sum to 1, allowing them to be interpreted as a categorical probability distribution over K classes. Thus, the highest value can be taken as the model's prediction that a given data sample is a member of that class.

Categorical cross-entropy, given as:

$$H(P, Q) = -\sum_i p_i \log(q_i) - (1 - p_i) \log(1 - q_i) \tag{6}$$

is used, where p_i is the true distribution of the data and q_i is the distribution predicted by the model.

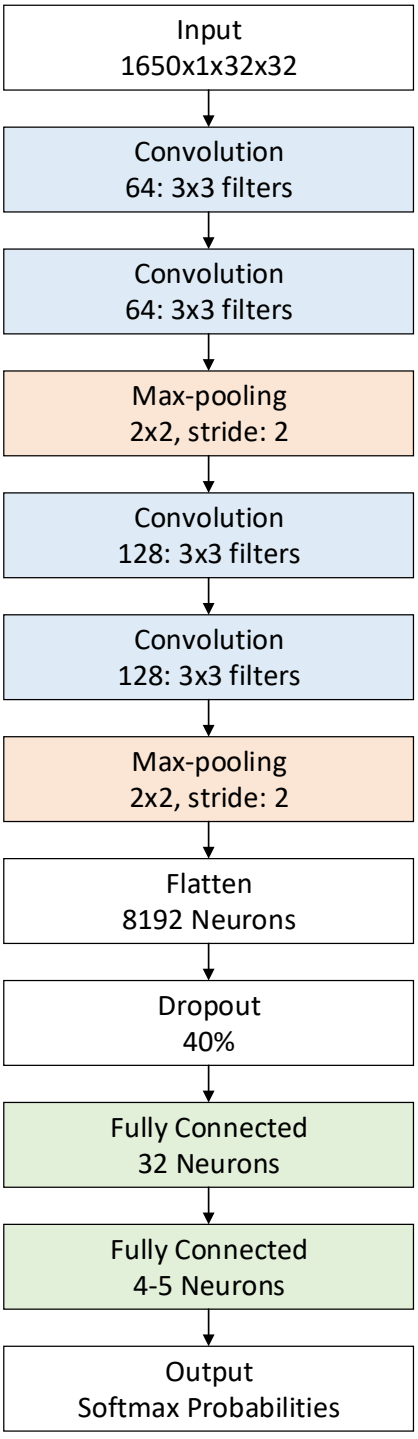

Fig. 8. CNN model architecture used in this work

All experiments were conducted on a single laptop with an Intel Core i7-6700HQ CPU, 16GB memory, and an Nvidia GTX 960m with 4GB GDDR5 memory. Typically, network training took between 17 and 20 seconds per epoch for up to 50 epochs, totaling between 850 and 1000 seconds.

V. Experimental Results

The model described above was tested using four configurations in which either the F (first rolling) set or S (second rolling) set of fingerprints from NIST-DB4 was used for training and the opposing set was used for testing and either the 5-class labels or 4-class labels were used. Any samples containing two labels were removed, leaving 1650 samples in both the F and S sets. Results are as follows:

TABLE II. Confusion matrix for CNN training on set F and testing on set S with 5 classes

			Predicted		
True	Whorl	Left	Right	Arch	Tent
Whorl	391	1	4	0	0
Left	8	361	2	4	3
Right	5	2	356	5	5
Arch	1	11	6	358	4
Tent	0	14	7	18	84
Accuracy			93.9%		

TABLE III. Confusion matrix for CNN training on set F and testing on set S with 4 classes

			Predicted	
True	Whorl	Left	Right	Arch/Tent
Whorl	287	5	2	2
Left	3	363	2	10
Right	4	1	354	14
Arch/Tent	3	26	10	464
Accuracy			95.0%	

TABLE IV. Confusion matrix for CNN training on set S and testing on set F with 5 classes

			Predicted		
True	Whorl	Left	Right	Arch	Tent
Whorl	385	4	5	2	0
Left	3	354	1	8	12
Right	0	0	363	4	6
Arch	0	2	7	371	0
Tent	0	3	15	10	95
Accuracy			95.0%		

TABLE V. Confusion matrix for CNN training on set S and testing on set F with 4 classes

			Predicted	
True	Whorl	Left	Right	Arch/Tent
Whorl	389	3	4	0
Left	4	357	0	17
Right	1	0	365	7
Arch/Tent	0	14	17	472
Accuracy			95.9%	

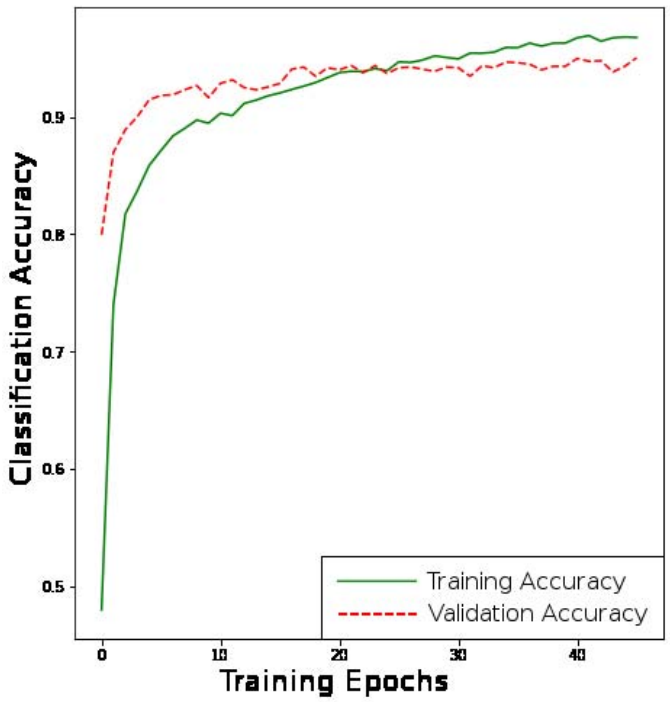

Fig. 9. Training and Validation Accuracy for NIST-DB4 trained on S set and tested on F set with 4 classes (95.9% acc.)

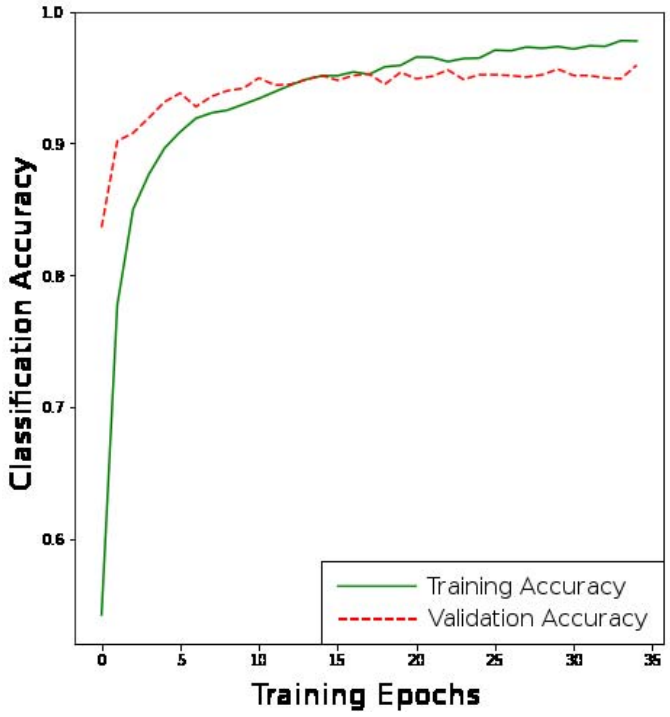

Fig. 10. Training and Validation Accuracy for NIST-DB4 trained on S set and tested on F set with 5 classes (95.0% acc.)

Figures 9 and 10 show training and validation accuracy for NIST-DB4 trained on the S set and tested on the F set. Notice that the model converges relatively quickly (~15 epochs) and overfitting is relatively low. Overfitting is more pronounced without data augmentation.

Table 6 shows the proposed approach compared to the top ensemble method described in [5] and the CNN model described in [17]. The accuracy percentage is shown with rejection rate in parentheses. Clearly, a 30.24% and 34.97% rejection rate makes usage in practical situations unrealistic.

TABLE VI. COMPARISON OF ACCURACY WITH OTHER APPROACHES (REJECTION RATE SHOWN IN PARENTHESES)

	NIST-DB4 F	NIST-DB4 S
Wilson, et al. [4]	-	90.80 (10.00)
Ensemble Method [5]	98.69 (30.24)	98.45 (34.97)
Peralta, et al. [17]	90.73	88.91
Proposed Approach	95.03	93.93

VI. CONCLUSION AND FUTURE WORK

In this work, a new approach to fingerprint classification using convolutional neural networks was created and tested. It has been shown that a relatively shallow network can be effectively trained to recognize fingerprint classes with a high degree of accuracy. Furthermore, with a small amount of preprocessing, the dimensionality of the network inputs can be significantly reduced by a factor of 256 (in the case of NIST-DB4), thus reducing network training time.

In future works, additional steps may be taken to augment the classification process. In [4], Wilson, Candela, and Watson consider the a priori distribution of fingerprints found in nature and removed images from the NIST-DB4 dataset to reflect this natural distribution. Many methods, including the FBI NBIS software using the PCASYS algorithm [31], allow for some sample rejection. Additionally, even though the model produces highly accurate results with zero rejection, allowing for a small percentage of rejected images should increase accuracy to nearly 100% on the tested datasets.

The authors of [5] created three additional databases, HQNoPert (High Quality with No Perturbations), Default (middle quality and some slight perturbations), and VQandPert (Varying Quality with moderate Perturbation), using the SFinGe tool for use in testing classification algorithms. Each database contains 10,000 fingerprint images with 288x384 pixels following a natural distribution. Though these prints are synthetically generated, they are highly similar to real fingerprints. They also contain only true class labels, unlike the ambiguous labeling of NIST4, allowing for more confidence in training and testing on the entire dataset instead of only a subset. Due to the cost of the SFinGe software, testing on these databases was not possible for this work, however future work using this model should include testing on these three databases.

REFERENCES

[1] FBI. (2014). Integrated Automated Fingerprint Identification System. Retrieved from https://www.fbi.gov/about-us/cjis/fingerprints_biometrics/iafis

[2] Craig I. Watson, et al. (2007). User's Guide to NIST Biometric Image Software (NBIS). National Institute of Standards and Technology. Retrieved from

http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=51097

[3] Craig I. Watson. (1992). NIST Special Database 4: 8-bit Gray Scale Images of Fingerprint Image Groups. National Institute of Standards and Technology

[4] Charles L. Wilson, Gerald T. Candela, and Craig I. Watson. (1994). Neural network fingerprint classification. Journal of Artificial Neural Networks 1.2, pp.203-228.

[5] Galar, M. et al. (2015). A survey of fingerprint classification Part II: Experimental analysis and ensemble proposal. Knowledge-Based Systems 81

[6] Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems 25, pp. 1097-1105

[7] Y. LeCun, et al. (1990). Handwritten digit recognition with a back-propagation network. Advances in Neural Information Processing Systems, 2,396- 404, Morgan Kaufman.

[8] Y. LeCun, et al. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov 1998.

[9] M. F. Fahmy, and M. A. Thabet, "A fingerprint segmentation technique based on Morphological processing," ISSPIT 2013.

[10] Daniel Peralta, et al. (2017). Robust classification of different fingerprint copies with deep neural networks for database penetration rate reduction. ArXiv:1703.07270v1 [cs.CV]

[11] Craig I. Watson. (1993). NIST Special Database 9: 8-bit Gray Scale Images of Fingerprint Images of Mated Fingerprint Card Pairs. National Institute of Standards and Technology

[12] Craig I. Watson. (1994). NIST Special Database 14: Mated Fingerprint Card Pairs 2. National Institute of Standards and Technology

[13] R. Cappelli. (2004). SFinGe: An Approach to Synthetic Fingerprint Generation. International Workshop on Biometric Technologies (BT2004).

[14] Francois Chollet, et al. (2015). Keras: The Python Deep Learning Library. https://github.com/fchollet/keras

[15] Yaniv Taigman, et al. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification. Conference on Computer Vision and Pattern Recognition (CVPR).

[16] Guilhem Cheron, Ivan Laptev, Cordelia Schmid. (2015). P-CNN: Pose-Based CNN Features for Action Recognition. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 3218-3226

[17] Daniel Holden, Jun Saito, Taku Komura. (2016). A deep learning framework for character motion synthesis and editing. ACM Transactions on Graphics Volume 35, Issue 4, Article 138.

[18] Satoshi Iizuka, Edgar Simo-Serra, Hiroshi Ishikawa. (2016). Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. ACM Transactions of Graphics Volume 35, Issue 4, Article 110.

[19] Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. (2011). Deep sparse rectifier neural networks. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics.

[20] R. M. Stock and C. W. Swonger. (1969). Development and evaluation of a reader of fingerprint minutiae. Technical Report CAL No. M-2478-X-1:13-17

[21] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. (2015). "Deep learning." Nature 521.7553, pp. 436-444.

[22] [22] K. Karu, A.K. Jain. (1996). Fingerprint classification, Pattern Recognition, Volume 29, Issue 3 pp. 389-404.

[23] Q. Zhang, H. Yan. (2004) Fingerprint classification based on extraction and analysis of singularities and pseudo ridges, Pattern Recognition, Volume 37, Issue 11, pp. 2233–2243.

[24] H.O. Nyongesa, et al. (2004). Fast robust fingerprint feature extraction and classification, Journal of Intelligent & Robotic Systems, Volume 40, Issue 1, pp. 103–112.

[25] L. Wang, M. Dai. (2007). Application of a new type of singular points in fingerprint classification, Pattern Recognition Letters, Volume 28, Issue 13, pp. 1640–1650.

[26] J. Li, W.-Y. Yau, H. Wang. (2008). Combining singular points and orientation image information for fingerprint classification, Pattern Recognition, Volume 41, Issue 1, pp. 353-366.

[27] M. Liu. (2010). Fingerprint classification based on adaboost learning from singularity features, Pattern Recognition, Volume 43, Issue 3, pp. 1062–1070.

[28] M. Kawagoe, A. Tojo. (1984). Fingerprint pattern classification, Pattern Recognition, Volume 17, Issue 3, pp. 295–303.

[29] R. Thai. (2003). Fingerprint Image Enhancement and Minutiae Extraction. The University of Western Australia. Retrieved from http://www.peterkovesi.com/studentprojects/raymondthai/RaymondThai.pdf

[30] M. F. Fahmy, and M. A. Thabet. (2013). A fingerprint segmentation technique based on Morphological processing. ISSPIT 2013.

[31] G.T. Candela, P.J. Grother, C.I. Watson, R.A. Wilkinson, C.L. Wilson, PCASYS – a pattern-level classification automation system for fingerprints, Technical Report, UNIST Interagency/Internal Report (NISTIR) - 5647, 1995.

[32] D. Michelsanti, et al. (2017). Fast Fingerprint Classification with Deep Neural Network. In VISAPP - International Conference on Computer Vision Theory and Applications