Use of Auxiliary Classifier Generative Adversarial Network in Touchstroke Authentication

Debzani Deb
Department of Computer Science
Winston-Salem State University
Winston-Salem, NC, USA
debd@wssu.edu

Mina M. Guirguis
Department of Computer Science
Winston-Salem State University
Winston-Salem, NC, USA
guirguismm@wssu.edu

2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA) | 978-1-7281-8470-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICMLA51294.2020.00049

Abstract— **With the growing popularity of smartphones, continuous and implicit authentication of such devices via behavioral biometrics such as touch dynamics becomes an attractive option, especially when the physical biometrics are challenging to utilize, or their frequent and continuous usage annoys the user. However, touch dynamics is vulnerable to potential security attacks such as shoulder surfing, camera attack, and smudge attack. As a result, it is challenging to rule out genuine imposters while only relying on models that learn from real touchstrokes. In this paper, a touchstroke authentication model based on Auxiliary Classifier Generative Adversarial Network (AC-GAN) is presented. Given a small subset of a legitimate user's touchstrokes data during training, the presented AC-GAN model learns to generate a vast amount of synthetic touchstrokes that closely approximate the real touchstrokes, simulating imposter behavior, and then uses both generated and real touchstrokes in discriminating real user from the imposters. The presented network is trained on the Touchanalytics dataset and the discriminability is evaluated with popular performance metrics and loss functions. The evaluation results suggest that it is possible to achieve comparable authentication accuracies with Equal Error Rate ranging from 2% to 11% even when the generative model is challenged with a vast number of synthetic data that effectively simulates an imposter behavior. The use of AC-GAN also diversifies generated samples and stabilizes training.**

Keywords—***Touch dynamics, Behavioral Biometrics, Continuous authentication, GAN, AC-GAN***

I. Introduction

Over the last few years, the world has witnessed the explosive growth of consumers who are increasingly using their smartphones for anytime-anywhere computing and the enhancements of their daily lives. During the Covid-19 era, smartphones are being regarded as lifelines and became an absolute crucial for distance learning and working. Since these devices store a mounting quantity of user's private and sensitive information, securing these devices from adversary attacks continues to be a significant concern for both manufacturers and users. Physical biometrics (face, fingerprints, iris, etc.) has often been promoted as the most secure means for log-in authentication for smartphones. However, there is a need for additional security measures after the initial log-in, known as continuous and implicit user authentication [1]. In such authentication, the system keeps monitoring the user in a continuous manner throughout their interactions with the device, and the process is implicit such as all authentication is carried out in the background without interrupting the user or requiring any active user cooperation. Strong physical biometrics are not appropriate for such implicit authentication as they require either full or partial cooperation from the users at regular intervals, which results in annoying the user.

Recent research has shown promising results in using behavioral biometrics [2] to verify users implicitly and continuously on smartphones. Today's smartphones are equipped with a plethora of sensors and accessories and could be used to extract user behavioral attributes such as touch dynamics, keystroke dynamics, and gait recognition. This paper focuses on touch dynamics [3,4], which captures the way a user touches a touchscreen device and its usage on continuous and implicit user authentication.

In touch dynamics continuous authentication, the system continuously monitors the raw touch data and extracts touchstroke features. These include the area of the screen covered by the touch stroke, touch pressure, speed, velocity, and acceleration of the x, y-positions on the screen [3]. After observing the user behavior for a while, the system learns her touch dynamics by performing statistical analysis or using machine learning. Then, at a later time, after the initial log-in by using a password/pin or physical biometric, the system continuously compares current user behavior with the learned user model to make an authentication decision. The training phase in such authentication is different from typical classification as the only training data available is merely the smartphone owner's data. It is highly unlikely that many users will share a smartphone, and therefore the classifier can only assume the availability of the owner's data that belongs to a single class instance. The challenge is to train a classifier with two different predictions, such as owner and attacker, where the attacker instance does not belong to prior-learned class [5]. Most of the prior works [3,6,7] on smartphone touchstroke authentication addressed this challenge by simulating one or more random users as attackers and the authentication problem is naturally fitted as a binary-class classification problem, where the model is trained using a particular user's touchstone data as the owner's and the others' as attacker's.

While the abovementioned strategy performs well in preventing random attackers (someone who does not know the owner and can be simulated by a random user), however, it is most likely not to prevent attacks from a genuine imposter (someone who knows the owner and deliberately trying to imitate her behavior). Therefore, authentication becomes vulnerable to smudge attacks and shoulder surfing. Recent

978-1-7281-8470-8/20/$31.00 ©2020 IEEE
DOI 10.1109/ICMLA51294.2020.00049

studies reveal that the maleficent actors were trained to deceive the biometric authentication system by mimicking their targets through media recordings or other means in which they are allowed to observe and practice their target's behavior [8]. This poses a challenge, such as how to differentiate the owner from imposters to defend the attack knowing that an imposter already knew the target behavior.

To address this challenge, a more robust behavioral biometric authentication based on generative adversarial network (GAN) [9] is presented, which learns how to mimic any data distribution while requiring fewer input data. GANs are a particular type of deep neural network model where two networks such as Generator and Discriminator are trained simultaneously in zero-sum game theory, with the former focused on data generation and the later centered on discrimination. While GANs and other generative models have been applied to an array of computer vision [9, 10, 11, 12, 13] and natural language processing [14,15,16] problems, biometric authentication has yet to receive thorough exploration. Given a small subset of a legitimate user's touchstroke data during training, the GAN model can learn to generate a vast amount of synthetic touchstroke data that closely approximates the real data and then uses both generated and real data in discriminating real user from the imposter. However, GANs are typically challenging to train and suffer from model collapse [17,18] problem in which the Generator starts generating samples that have little variety.

This paper proposes a touchstroke authentication model based on an extension of generic GANs, Auxiliary Classifier Generative Adversarial Network (AC-GAN) [19], an additional task-specific auxiliary classifier that optimizes by back-propagating classification loss through the Discriminator and the Generator. The auxiliary classifier has the effect of stabilizing the training process, and prior researches [20,21] on image and text data has shown excellent performances while alleviating the mode collapse problem. In the proposed AC-GAN authentication model, the Generator synthesizes touchstroke data conditioned on a class label. The Discriminator classifies between real and generated touchstrokes and assigns them a class label such as owner vs. imposter. Our goal is to develop a more robust biometric authentication system with higher accuracy and security.

The rest of this paper is organized as follows. Section 2 reviews some core concepts of GAN and AC-GAN. Then in Section 3, the architecture of the proposed authentication system is detailed. Section 4 focuses on the steps that are followed to train and test the AC-GAN network. Section 5 discusses the experimental setup, data set, and evaluation results. Section 6 concludes the paper.

II. Background

This section reviews some core concepts of GANs and the additional improvements that AC-GAN offers. GAN models assume the availability of real data **x** drawn from a distribution p_r, and exploit a generative model to generate synthetic data that closely resembles **x**. A generative model *G* takes as input a random noise **z** and generates a sample *G(**z**)*, such as the output can be regarded as a sample drawn from a distribution: $G(\mathbf{z}) \sim p_g$. The objective for *G* is to approximate p_r using p_g.

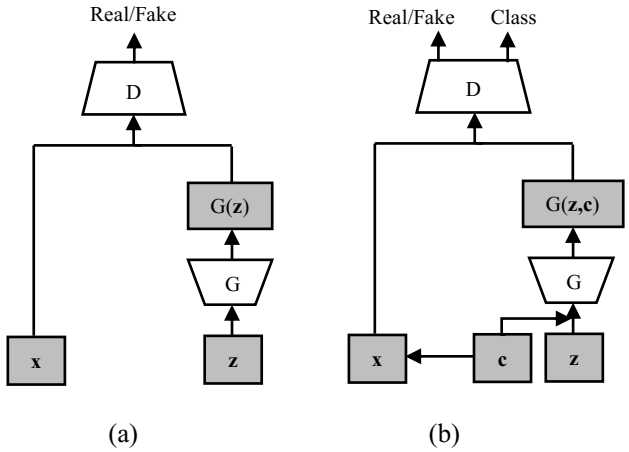

Fig. 1. The general structure of (a) GAN and (b) AC-GAN, where *x* denotes the real touchstroke, *c* the class label, *z* the noise, *G* the Generator, and *D* the Discriminator.

GAN consists of two separate neural networks: a Generator *G* that takes a random noise vector **z**, and outputs synthetic data *G(**z**)*; a Discriminator *D* that takes an input **x** or *G(**z**)* and output a probability *D(**x**)* or *D(G(**z**))* to indicate whether the input is generated (fake) or from the real data distribution (Fig. 1(a)). Both of the Generator and Discriminator in GAN models can be arbitrary neural networks. The Generator *G* and Discriminator *D* in GAN models are trained by forming a two-player min-max game where *G* tries to generate realistic data to fool the Discriminator while *D* tries to distinguish between real and synthetic data [9]. The Discriminator is trained to maximize the log-likelihood it assigns the input to its correct source (real/fake) as in (1) [9].

$$L = E[\log P(S = real\ |X_{real})] + E[\log P(S = fake\ |X_{fake})] \quad (1)$$

While GANs are able to generate synthetic data with higher accuracies, they suffer from problems like training instability, nonconvergence, and mode collapse. Multiple improvements have been suggested to fix these problems, including using deep convolutional layers for the networks, varying architectures, and modified objective functions for D and G.

Conditional GAN (CGAN) [22] is one such improvement where the GAN network is augmented using side information to add more structure to the network and stabilize training. The original setup of a GAN has no control dependent on random noise. However, if auxiliary information is provided during the generation, the GAN can be driven to output data with desired properties. In CGAN, both Generator and Discriminator are supplied with class labels **c** in order to produce class conditional samples. The input/noise and **c** are combined in a joint hidden representation and fed as an additional input layer in both networks. The training of the GAN model is changed so that the Generator is provided both with random noise and a conditional input **c**, and attempts to generate synthetic data based on that condition. The

Discriminator is provided with both real and generated data as input and must classify whether the input is real or fake as before.

Extending these ideas, Odena *et al*. [19] proposed the Auxiliary Classifier Generative Adversarial Network (AC-GAN). In that model, the Generator synthesizes data conditioned on a class label, and the Discriminator not only classifies between real and generated input data, but also assigns them a class label. In addition to Generator and Discriminator models, AC-GAN is equipped with an additional task-specific auxiliary classifier with the purpose of reconstructing the class labels (Fig. 1(b)). In the AC-GAN, every generated sample has a corresponding class label **c** in addition to the noise **z**. *G* uses both inputs to generate synthetic touchstrokes *G(z,c)*. The Discriminator facilitated by the auxiliary classifier can be provided with either generated or real touchstrokes as input and outputs both a probability distribution over sources (real/fake) and a probability distribution over the class labels. The objective function has two parts: the log-likelihood of the correct source, L_S (2), and the log-likelihood of the correct class, L_C (3) . D is trained to maximize $L_S + L_C$ while G is trained to maximize $L_C - L_S$. The resulting Generator learns a latent space representation independent of the class label, unlike the conditional GAN.

$$L_S = E[\log P(S = real\ |X_{real}) + E[\log P(S = fake|X_{fake})] \quad (2)$$

$$L_c = E[\log P(C = c\ |X_{real})] + E[\log P(C = c\ |X_{fake})] \quad (3)$$

III. Architectural Framework

The presented touch biometric authentication architecture contains two process modules, one executes in the target (smartphone) device, and the other executes in the server-side (Fig. 2). As the proposed authentication system needs to act instantly with higher accuracies, the mobile side component is designed to be a lightweight process running without being computation or resource-heavy. Training a GAN model is computationally demanding, and therefore the proposed server-side component deals with this heavyweight process. First, the smartphone owner interacts with the touchscreen device and the smartphone sensors and accessories capture raw touch biometrics such as x- and y-coordinates of the finger, its pressure on the screen, the area of the screen covered by the finger, the finger orientation with respect to the screen and the screen orientation. During the training period, these raw data are continuously communicated with the server where touch stroke features are extracted. The features extracted are then used as the input of the AC-GAN model, where the model is trained based on the owner's touch biometrics and generated synthetic data using TensorFlow and python libraries. Once trained, a compact version of the Discriminator model is created using TensorFlow Lite [23], which is designed to execute models efficiently on mobile and other embedded devices with limited compute and memory resources. The compact model is then deployed in the target device and becomes ready to make predictions. While deployed, user's interactions are continuously monitored, touch biometrics features are extracted, and the deployed model relentlessly looks out for imposters.

IV. Training and Testing of AC-GAN Network

This section details the steps that are followed in order to train and test the AC-GAN model. Firstly, a single raw touch stroke is analyzed, and a feature vector with 31 dimensions is formed according to the procedure depicted in [3]. As part of the proposed AC-GAN authentication system, three models, such as the Generator, the Discriminator, and the composite models, are developed. The input noise for the Generator model is created by randomly generating a point **z** (100 dimensions) in the latent space and then using an embedding layer with *glorot_normal* as kernel initializer in order to assign a random class label **c** (randomly selected integers in

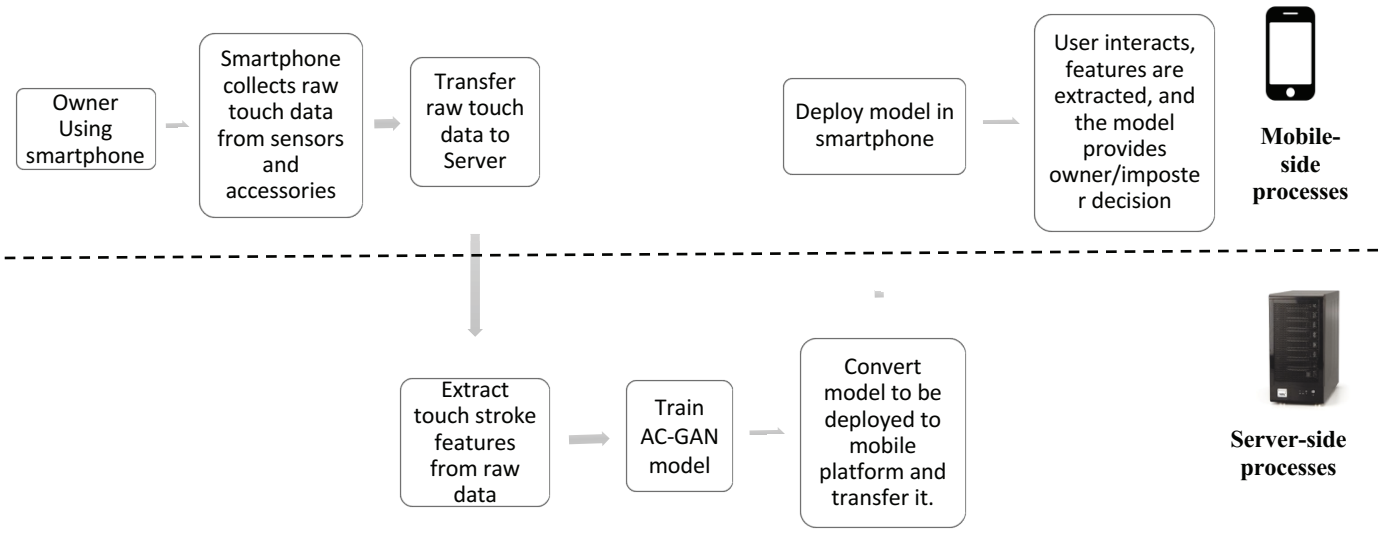

Fig 2. Architectural Framework for the touch biometrics authentication system

[0,1] inclusively) to **z**. The Hadamard product between latent point **z** and class conditional embedding **c** is then provided as input to the Generator**.** The Generator is designed as a deep neural network (DNN) containing four dense layers with *LeakyReLU* activation and *BatchNormalization* as specified in Table I and generates a class conditional synthetic feature vector of size 31 with *tanh* activation and *glorot_normal* as kernel initializer.

The Discriminator model is provided with either a real feature vector or a synthetic one (generated by the Generator) as input and then predict whether the input touchstroke is real or synthetic, and the auxiliary classifier predicts the class label of the touchstroke vector. In this study, both the Discriminator and the auxiliary classifier are implemented as a single DNN with two outputs. Table I shows the DNN structure and parameters. The first output of the Discriminator is a single probability via the sigmoid activation function that indicates the "*realness*" of the input touchstroke and is optimized using *binary cross entropy* like a normal GAN Discriminator model. The second output is a probability of the touchstroke belonging to either the *owner* (indicated by label "1") or *imposter* (indicated by label "0") class via the *softmax* activation function and is optimized using *categorical cross entropy*. The model is fitted with *Adam* version of stochastic gradient descent with *learning rate* being 0.0002 and *momentum* being 0.5.

The AC-GAN composite model is created by packing the Generator model on top of the Discriminator model. The Discriminator model within the composite model takes the synthetic touchstrokes generated by the Generator model as input and predicts both the realness of the generated output and the class label. As the purpose of the composite model is to enhance the quality of the generated touchstrokes by tricking the Discriminator, during training, the Discriminator's weights are not updated, only the Generator's weights are updated. This has the effect of updating the Generator toward getting better at generating real samples on the next phase of training. In the composite model, the Discriminator model is therefore set as non-trainable to prevent it from being updated when the composite model is updated.

Once all three models are built, the AC-GAN network is ready for training. Fig. 3 contains the pseudo-codes of AC-GAN authentication that illustrates how the network works in the training stage. Once trained, the compact Discriminator model is deployed in the mobile platform and make authentication decisions as outlined in Fig. 4.

TABLE I. STRUCTURES AND PARAMETERS USED IN GENERATOR AND DISCRIMINATOR NETWORKS

Generator		**Discriminator**	
Layer	**output_shape**	**Layer**	**output_shape**
Input	(None, 100)	Input	(None, 31)
Dense	(None, 256)	Dense	(None, 1024)
LeakyReLU	(None, 256)	LeakyReLU	(None, 1024)
BatchNormalization	(None, 256)	Dense	(None, 512)
Dense	(None, 512)	LeakyReLU	(None, 512)
LeakyReLU	(None, 512)	Dropout	(None, 512)
BatchNormalization	(None, 512)	Dense	(None, 256)
Dense	(None, 1024)	LeakyReLU	(None, 256)
LeakyReLU	(None, 1024)	Dropout	(None, 256)
BatchNormalization	(None, 1024)	Dense	(None, 1)
Dense	(None, 31)	Dense	(None, 2)

AC-GAN Training
Inputs: Training instances X (31 dimensions feature vector) with class labels Y
Inputs: Batch Size (B_s = 32)
Output: Trained AC-GAN model along with training losses
1. Initialize Discriminator D and Generator G
2. For each training step
   a. Get a sample batch X_B and Y_B of size B_s from X and Y.
   b. Create a new batch of noise z and random sample labels c and use them as input to the Generator to generate a batch of class conditioned synthetic touchstrokes X_B'
      X_B' = Generator.predict (z, c)
   c. $x = X_B + X_B'$ the training dataset x therefore contains 2 * B_s elements.
   d. Populate array y (real/fake label) with B_s 1s and B_s 0s to indicate that the first half of the training set is real touchstrokes, and the second half is synthetic.
   e. Populate array *aux_y* (class labels) with Y_B (class labels of real touchstrokes) plus c (generated class labels for synthetic touchstrokes).
   f. Train the Discriminator with (x, [y, *aux_y*]) as input. During training, the Discriminator gradients are updated to maximize $L_S + L_C$ as in (2) and (3).
      Discriminator.train_on_batch(x, [y, aux_y])
   g. Create 2 * B_s noise z′ and random labels c' and train the Generator with them via the combined model. The Discriminator is set as non-trainable in order to prevent weights updating. The goal here is to train the Generator to trick the Discriminator, therefore all labels in y' are set to 1, or to real, although they are synthetic samples.
      combined.train_on_batch([z′, c′], [y′, c′]
   h. Continue steps *a* to *g* for all remaining batches in the training data set.
   i. Calculate the average Discriminator training loss over all batches for both the y (real/fake label) and *aux_y* (class label) outputs.
   j. Calculate the average Generator training loss over all batches through the combined model.

Fig. 3. Training stages of AC-GAN

AC-GAN Testing
Inputs: Testing instances X_t (31 dimensions feature vector) with class labels Y_t
Output: Testing Losses, Prediction scores: ROC, EER
1. Follow steps a – e of the training algorithm in Fig. 3 to create x_t,y_t, and *aux_*y_t from the testing data set (X_t, Y_t) and generated synthetic touchstrokes.
2. Evaluate the Discriminator on (x_t, [y_t, *aux_*y_t]).
3. Calculate the Discriminator testing loss for both the y_t (real/fake) and *aux_*y_t (class label) outputs.
4. Calculate the Generator testing loss.
5. Calculate ROC and EER.

Fig. 4. Testing stages of AC-GAN

V. EXPERIMENTS AND EVALUATAION

A. Dataset

This study adopted Touchanalytics [3] dataset for experiments. There are 21,158 touch strokes in total, belongs to 41 subjects, which are collected from four different Android phones. For each stroke, 31 functional features can be derived

[3]. Since each feature does not fall in the same range, they are standardized to be in the range [-1,1]. The authentication scenario considers two classes only, such as owner and imposter; however, the Touchanalytics dataset is a collection of touchstrokes data that belongs to 41 users. For a legitimate user, a user-specific dataset is created by extracting all of her touch data from the main dataset. In order to keep classes balanced, as many samples from the negative class (other users) are obtained as there are samples of the legitimate user and these samples are added to make a complete user-specific dataset for such user. This user-specific dataset is divided into training (80%) and testing (20%) sets, and is utilized during training and testing, as outlined in Fig 3 and Fig. 4. Note that these are not the sole data used during training and testing, an equal amount of synthetic data generated by the AC-GAN model are also merged along with the real train and test datasets and the merged ones are utilized in learning and making predictions. The model is fit for 25 training epochs and a mini-batch size of 32 samples is used.

B. Evaluation Metrics

To evaluate the performance of the trained Discriminator as a classifier, various standard evaluation metrics such as Precision, Recall, F1, ROC, and Equal Error Rate (EER) score were utilized. In this study, Precision is the ratio of correctly predicted 'owner' observations to the total predicted 'owner' observations. Recall is the ratio of correctly predicted 'owner' observations to all observations in the actual 'owner' class. In other words, Precision and Recall are all interested in predicting the true answer of the positive label. The EER is the error rate where the False Acceptance Rate (FAR) and False Rejection Rate (FRR) coincide. Lower EER values signify better accuracies. F1 score takes both Recall and Precision into account, and therefore it provides a useful indicator.

However, there are some desired properties that an efficient GAN network should fulfill such as the ability to distinguish generated samples from real ones, the ability to generate diverse samples (sensitivity to overfitting, mode collapse, etc.), the ability to generate samples that closely approximate the real data, model having a low sample and computational complexity, etc. Quantitively evaluating these properties is challenging, and most of the currently available measures are specifically suitable for computer vision or text analysis tasks. Inception score (IS) and Fr´echet Inception distance (FID) [24] are two such measures correlated with the visual quality of generated images and are not suitable for non-image datasets. This study investigated generation loss (ability to discriminate between real/fake samples) and classification loss (ability to classify owner/imposter correctly) to measure some of the desired GAN properties quantitively.

C. Results

For experimental evaluation, ten subjects (first Column in Table. II) are randomly selected from Touchanalytics dataset. As explained in section V.A, each subject's touch strokes (second column in Table II) are extracted as legitimate data, and a same number of other users' touchstrokes are added as fraud data to make the class balanced. During training and testing, an equal number of synthetic data of random classes are generated to make the total data set size as depicted in the third column of Table II. The training and testing are performed for each authentic user individually, and the performances for each user

TABLE II. AUTHENTICATION PERFORMANCES FOR TEN RANDOMLY SELECTED SUBJECTS FROM TOUCHANALYTICS DATASET

Subject ID	Subject instances	Dataset size	Pre.	Rec.	F1	ROC	EER
2	**1230**	4920	**0.98**	**0.98**	**0.98**	**0.976**	**0.024**
35	1063	4252	0.94	0.96	0.95	0.949	0.063
23	969	3876	0.96	**0.98**	0.97	0.969	0.039
3	759	3036	0.93	0.91	0.92	0.913	0.092
27	609	2436	0.95	**0.88**	**0.91**	0.917	0.110
11	445	1780	0.95	0.95	0.95	0.952	0.049
16	382	1528	**0.90**	0.89	0.89	**0.888**	**0.111**
12	342	1368	0.92	0.96	0.94	0.938	0.076
4	241	964	0.96	0.95	0.95	0.953	0.053
30	**225**	900	0.93	0.91	0.92	0.922	0.087
Mean			**0.94**	**0.94**	**0.94**	**0.94**	**0.07**
Median			**0.95**	**0.95**	**0.95**	**0.94**	**0.07**

in terms of Precision, Recall, F1, ROC, and EER are shown in Table II. The maximum and minimum performances achieved for each metric across all subjects are highlighted in the table, along with the mean and median of all metrics. The Precision scores are relatively high, with mean: 0.94 and median: 0.95. The Recall and F1 values are similarly high, with the exception of subject # 27. These results indicate that it is possible to achieve acceptable authentication accuracies even when the machine learning model is challenged with a large number of synthetic data that effectively simulates an imposter behavior. The EER performance ranges from 2% to 11% with a median of 7%, which is comparable with performances achieved by other touchstroke authentication systems [3,4] that are not challenged by generated data. The results in Table II further reveal that the proposed system comparatively performs better when there are more data available to learn and to generate from.

As outlined in section V.B, we also studied generation loss L_S in (2) and classification loss L_C in (3) to understand the network's ultimate efficacy. Fig. 5 shows the Discriminator's generation (real/fake) loss during training and testing with respect to subject # 2's touchstroke data. It is evident that after the ~15th or so epoch, the Discriminator becomes quite capable of distinguishing real data from synthetic data. Some shaky behavior is noticed from the test loss, but it is quite expected. From this result, it can be concluded that the generated samples are a realistic approximation of the distribution of natural touchstrokes; otherwise, Discriminator would not be able to classify them as real/fake so effectively. Fig. 6, on the other hand, shows the Discriminator's classification (owner/imposter) loss during training and testing with respect to subject # 2's touchstroke data. The plot in Fig. 6 shows that at about 20th epoch, the test loss reaches its minimum with some expected wobble afterward. This result suggests that the generated touchstrokes are similar to real ones as the classification network, which learns features for discriminating touchstrokes generated for different classes, can correctly classify them. These plots do not show any symptom for mode collapse, which may indicate that the generated samples are diverse.

VI. CONCLUSION

This paper presents a touchstroke authentication model based on AC-GAN. Given a small subset of a legitimate user's touchstroke data during training, the presented AC-GAN model learns to generate a vast amount of synthetic touchstrokes that closely approximate the real touchstrokes, simulating imposter

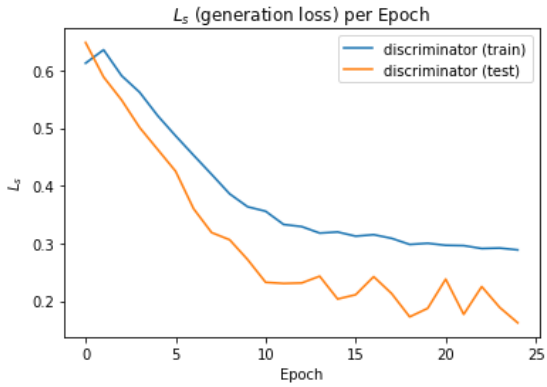

Fig. 5. Train and Test Generation loss (L_S) for the Discriminator (Subject #2)

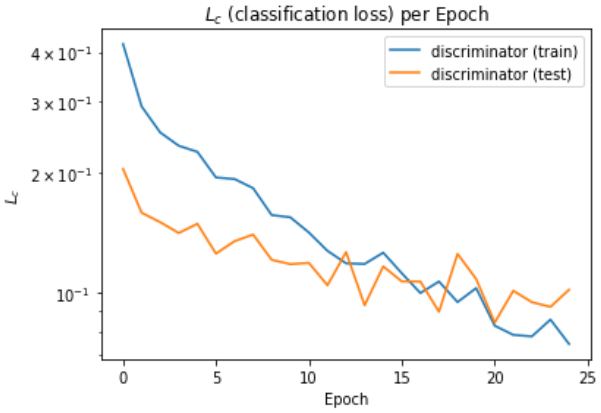

Fig. 6. Train and Test Classification loss (L_C) for the Discriminator (Sub. #2)

behavior, and then uses both generated and real touchstrokes in discriminating real user from the imposters. The presented authentication relies on an architecture where the computationally demanding AC-GAN training takes place on the server-side, and the lightweight mobile side performs the authentication. The presented network is trained on Touchanalytics dataset, and the discriminability is evaluated with popular performance metrics and loss functions. The evaluation results suggest that it is possible to achieve comparable authentication accuracies (EER ranging from 2% to 11%) even when the generative model is challenged with a vast number of synthetic data that effectively simulates an imposter behavior. Use of AC-GAN also diversify generated samples and stabilizes training. The future works will focus on fine tuning the model in order to achieve better accuracies and investigating the impact of posture variation on the presented authentication.

ACKNOWLEDGMENT

We would like to acknowledge the support provided by NSF award # 1900087.

REFERENCES

[1] V. M. Patel, R. Chellappa, D. Chandra, B. Barbello, “Continuous user authentication on mobile devices: Recent progress and remaining challenges”. IEEE Signal Processing Magazine, vol. 33, issue. 4, pp. 49–61, 2016.

[2] A. Mahfouz, T. M. Mahmoud, and A. S. Eldin, “A survey on behavioral biometric authentication on smartphones,” Journal of Information Security and Applications, vol. 37, pp. 28–37, 2017.

[3] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication", IEEE Transactions on Information Forensics and Security, vol. 8, no. 1, pp. 136-148, 2013.

[4] Z. Sitova, J. Sedenka, Q. Yang et al., “HMOG: new behavioral biometric features for continuous authentication of smartphone users,” IEEE Transactions on Information Forensics and Security, vol. 11, no. 5, pp. 877–892, 2016.

[5] R. Domingues, M. Filippone, P. Michiardi and J. Zouaoui, "A comparative evaluation of outlier detection algorithms: Experiments and analyses", Pattern Recognition, vol. 74, pp. 406-421, 2017.

[6] A. Roy, T. Halevi, and N. Memon. “An hmm-based behavior modeling approach for continuous mobile authentication”, In 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3789–3793. IEEE, 2014.

[7] A. Serwadda, V. Phoha, and Z. Wang, “Which verifiers work?: A benchmark evaluation of touch-based authentication algorithms,” in Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on, pp. 1–8, 2013.

[8] G. Ye, Z. Tang, X. Chen, K. Kim, B. Taylor, and Z. Wang, “Cracking android pattern lock in five attempts”, The Network and Distributed System Security Symposium, 2017.

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Advances in neural information processing systems, pp. 2672– 2680, 2014.

[10] Y. Jin, J. Zhang, M. Li, Y. Tian, H. Zhu, and Z. Fang, “Towards the automatic anime characters creation with generative adversarial networks,” https://arxiv.org/abs/1708.05509

[11] R. A. Yeh, C. Chen, T. Lim, M. Hasegawa-Johnson, and M. N. Do, “Semantic image inpainting with perceptual and contextual losses,” CoRR, vol. abs/1607.07539, 2016. [Online]. Available: http://arxiv.org/abs/1607.07539

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. “Image-to-image translation with conditional adversarial networks”. arXiv:1611.07004, 2016.

[13] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas. “Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks”. arXiv:1612.03242, 2016.

[14] S. Rajeswar, S. Subramanian, F. Dutil, C. J. Pal, and A. C. Courville, “Adversarial generation of natural language,” CoRR, vol. abs/1705.10929, 2017. Available: http: //arxiv.org/abs/1705.10929

[15] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, “Long text generation via adversarial training with leaked information,” CoRR, vol. abs/1709.08624, 2017. Available: http://arxiv.org/abs/1709.08624

[16] J. Li, W. Monroe, T. Shi, A. Ritter, and D. Jurafsky, “Adversaria learning for neural dialogue generation,” arXiv:1701.06547, 2017

[17] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, “On convergence and stability of GANs,” https://arxiv.org/abs/1705.07215

[18] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training GANs,” in Advances in Neural Information Processing Systems, pp. 2234–2242, 2016.

[19] A. Odena, C. Olah, and J. Shlens, “Conditional image synthesis with auxiliary classifier GANs,” arXiv:1610.09585, 2016.

[20] A. Mino and G. Spanakis, "Logan: Generating logos with a generative adversarial neural network conditioned on color", 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)., pp. 965-970, 2018.

[21] A. Dash, et al. "TAC-GAN-Text Conditioned Auxiliary Classifier Generative Adversarial Network." arXiv:1703.06412 (2017).

[22] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” arXiv preprint arXiv:1411.1784, 2014.

[23] https://www.tensorflow.org/lite

[24] M. J. Chong, D. Forsyth, “Effectively unbiased fid and inception score and where to find them”. arXiv:1911.07023 (2019)