Proof Of Liveness

1/ Plan

-x- 2 march → 13 march (1.5 week) : Research and Setup
Upload first project summary that would work locally

-x- 14 march → 20 march (1 week) : Preps for prototype build (Research on tech tools Python libraries, Cairo, useful firmwares…)
Upload PrepDoc with project summary that is decentralized

-21 march → 3 april (2 weeks) : Scope Statement (expert meetings and full bibliographie)
Upload Scope Statement

-4 april → 10 april (1 week) : Conception document and small builds, tests, preps
Upload Doc de Conc

-11 april → 9 may (5 weeks) : Local Prototype
1 week image dataset creation
2 weeks platform build
1 week user data collection
1 week classifier training
Upload V1

-9 may → 23 may (2 weeks) : CAIRO BUILD
Deploy V1 on STARKNET

####### HALF TIME ####### (1 week lifeboat)

- 1 june → Local V2 (1 month)
- 1 july → RAPPORT + Donjon feedback + updates (1 month)

Project done by 1st August → 1 month for new Project

2/ Ideas

- Machine Learning algorithm that verifies that the user of the contract is a human.
- Neural network that distinguishes human faces from “other” pictures.
- Human mouse trajectory recognition.
- Completely decentralized algorithm (Open source) *not rely, or as less as possible on using Security by obscurity.*
- Not worrying about the robustness of the algorithm, the goal is to discourage a certain threshold of bots. Simply adding an extra task already removes (personal estimation) 50% of bots. Machine learning algorithms repulce up to 80%, 90% of most advanced bots. There must be a real motivation and dedication to make a powerful intelligent bot, but then, the danger of

employing real human farms exists also. Then the effort needed to develop a 100% bot repellent algorithm, seems unnecessary. Complementary document verification is necessary for highly critical systems.
- Preserving privacy as much as possible.
- Integration de starkware en live.
- Proof of livenes: Is 100% robustness even necessary? After a difficulty level, it is easier to hire real human farms.
- The goal should be to make tests that are difficult for AIs and easy for humans. Maybe make a Cognitive test that is not too hard but the assignment is a picture of a handwritten sentence. That almost unnoticeable step in complexity for the human is a requirement for the bot to have a pattern recognition engine. Combine a few of these tasks and mix them up, and we should get a test that is hard to pass for most of the bots and easy for humans.
- Setting a pin that has to be entered manually with on screen buttons, analyzing typing.
- The facial recognition method can be combined with DLT (Distributed Ledger Technology) for full decentralized KYC.
- ML algorithms that are based on image analysis tend to transform color images to gray levels. Maybe we can exploit that.
- Sliders that don't go all the way but need to be stopped at a certain percentage. Human approximation error may be the judge of liveness. Distance error from the mean human data can be one feature. Simpler classifier but can be used on more platforms. Problem of not available data. And use of GAN probably required, or good human model function.
- Table with 4x4 noisy pictures of different objects. The task is to select for example 1 red car, 2 numbers 3. The task can be a picture of a handwritten phrase. Randomize the pictures and the tasks. Make the process as difficult as possible for AI's while maintaining the process simple enough for humans not to frustrate them.
- Images for the database may come from google street view captures. Anyway making a new database makes the problem a little harder for classifiers as they have to have a good generalization ability. It is highly probable that the classifier may have used the same publicly available data for training.
-Rather than focusing the project on the performance and precision of our classifier, focus it on the weaknesses of the attacker's classifier. We have to explore the weak points of classifiers rather than relying on enhancing the strong parts. Make them miss.
- Make estimates of the percentages of bots that could be stopped.
- The Security test won't be theoretically unbreakable but will be in calculatory terms so difficult that the training of such algorithms would be so expensive it wouldn't be worth it. The bigger problem would be maintaining the test easy for humans.
- Importance of the project is more profound than it seems. Battle against centralisation and security based on Trust. We need decentralization, privacy, based on transparency and verifications. Example of google cloud for training and AI. How can we trust the execution. That problem is at the heart of Starknet.
- Use Keras or other toy datasets to experiment with noise.

3/ Useful links:

-**JavaScript CAPTCHA Interface:** https://dev.to/meatboy/series/6335
Captcha Sniper, tool to solve CAPTCHA's. This tutorial uses JavaScript, TypeScript, Node and React stacks, API's like Canvas and Web Crypto. On the frontend side, React for rendering CAPTCHA widget and displaying an image from the backend side. React will render the canvas object from Web API and handle user input.

-How to bypass captcha slider: https://filipvitas.medium.com/how-to-bypass-slider-captcha-with-js-and-puppeteer-cd5e28105e3c
Slider is a fast and easy way to eliminate the majority of bots because most of them don't execute Javascript. This method is user friendly and natural for phone users. Puppeteer is a tool for bypassing sliders. War between websites and bots is never over.

- **DLT example** https://github.com/Matus23/KYC-Ethereum-smart-contracts

- **Mouse tracking example** : https://github.com/2justinmorgan/thesis

-Python Face recognition
https://realpython.com/face-detection-in-python-using-a-webcam/
Using OpenCv and FaceCascades, an open source library with Computer Vision algorithms. Using webcams would be useful only as a way to capture the face. A GAN trained classifier would then need to face the picture and tell if it's real or fake.

Vitalik's introduction to ZK proofs:
https://medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558cea649

Flattening (Transforming code into simple operations); Then conversion to rank-1 constraint system (R1CS is essentially a sequence of groups of vectors) and then converting to QAP form (Based on Lagrange interpolation). Instead of checking the constraints in the R1CS individually, we can now check all of the constraints at the same time by doing the dot product check on the polynomials.

4/ Project Summary

A/ Local Version

- The program will consist of an interface which will ask the user to click a box A then box B. The NN will determine whether the mouse trajectory was human or AI drawn. A time span of 7 seconds will be available to execute the drawing.
- Data gathering: To train our network we need big amounts of data (The more the better). We need human drawings which we will ask for in the database available via link in the article. At least we would need around 30 x 50 = 1500 human mouse trajectories. The AI trajectories will be synthesized functionally.
- Feature Extraction : 36 features using the Sigma LogNormal model.
- Training with Random Forest classifier.
- Test with 30% of the database.
- Decentralization and application development

B/ Decentralized Version

- The program is a test of liveness that will ask the user of the smart contract to solve certain tasks. They will consist of selecting the right pictures from a few proposed ones. For example, from a table of 16 pictures select 2 numbers “3” and 1 cat. The user will have 10s to choose. *(Problem of how to store the labels, in general how to make an open source test because we have to store the questions and answers. Maybe it won’t matter because the decider will not decide to pass the test or not depending on only the quality of the answer but on the humanness.).* Object recognition is still considered a very hard problem for ML algorithms.

1) Construct the database. Make a database of 500 pictures (The probability of repetition of a picture in a new test is 3.2%). Find pictures of objects and make them as complicated for computer reading as possible. Color, rotation, angles, noise play may be a good idea. (The environment is already very uncomfortable for bots.)
2) Make a local prototype environment for data gathering capturing features that will help the decider if the response is human or not (number of good answers, time of answer…). Depending on the similarity of data, the number of data will be defined later. Expect a bare minimum of 30 persones x 50 tests = 1500 data samples.
3) Build a machine learning classifier of Humans/Bots based on the extracted features in the data samples. A GAN will be needed to simulate 1500 responses of a bot that “knows the answers” is a genius of object recognition, or partially sees, or gives random responses.
4) Test the program locally 20 new persons and 20 GAN responses will have to be classified with a rate higher than 80%.
5) Cairo build of the app
6) DApp lunch!
7) Donjon finds whole in the app architecture:)

8) Updates and fixes

5/ Meetings

Nicolas Perrin-Gilbert : perrin@isir.upmc.fr ISIR Researcher (Friday 04/03/22 at 17h):

- Proof of livenes by Machine Learning, (Without document verification), that is completely robust, **Doesn't exist** to date. Big companies protect themselves by using abnormally huge amounts of data to train their Machine Learning algorithms and that way ensuring that very few teams in the world have that much data to train on. For example Facebook is able to do training on user mouse trajectory on Millions of subjects per day. That way having access to the top quality of data which cant be attained in normal conditions.
- **Warning**: Keep in mind that for successful training, a very large data set is required. Not only large but representative of the population.
- Proof of livenes is generally not based on machine learning, because it's not completely robust. A more **dynamic** solution is required: Changing the nature of the test before there is a breach. The Goal is to make the task of breaching the test by a bot as hard as possible before it has the time to train himself to resolve the test. Nevertheless We have to keep in mind that the test is never 100% impenetrable.
- **Particular problem** of supervised learning: Algorithms must classify a face and "something else".Training by giving to the "else" category random pictures is dangerous. ML algorithms have pretty unpredictable responses to unseen data. To Check: GAN (Adversarial NN). Generator and discriminant methods.
- **Warring**: Security by obscurity, big companies hide in top secrecy their trained neural network.
- **Decentralization**, and open sourcenes gives the project an interesting aspect as we must avoid using hiding anything.

Mael Franceschetti - Phd student at LIP6 - Team SMA (Friday 18/03/22 / ZOOM)

- Using a trained NN based test is problematic if the code needs to be open source. The trained network is used after as the classifier but without updating the network. So if the weights of the network are visible a person might use it to exploit it and train their network to trick it.
- The test needs to rely on problems that are in general very hard for AI algorithms to resolve. Object detection on noisy images is one example.
- Using LSTM makes it harder to use the trained classifier.
- Haar Cascade also.
- Using Evolutionary Strategies instead of Gradient Descent based algorithms is a better option, even if slower they are better at exploring and evading local optimums.
- The database should be constructed independently, takes time but is more adapted and practical to use. A lot of people should get tested for the database in order to have good results if there are not a lot of features or the data are not very different from person to person.

- Project options: Develop a way to make noisy images and trick the most advanced classifiers. Develop a test with noisy images that classifies humans from bots depending on the response time, errors made, etc… Use of GAN needed.
- Switching the focus of the project from the performance of the classifier to the hardness of the test naturally for an AI.

Felix Poirier - Equipe Data Ledger (24/03/22)

- **Warning**: Starknet and CAIRO are very new environments. Be wary on the time and complexity of programming, there are very few libraries available. The project needs to be done incrementally. Finish a small project but until the end, and then upgrade it little by little.
- There is no need to make the picture database available, maybe only make 10% available. The rest should be on a server and that doesn't make the app less decentralized. Solves maybe the safety issue.
- Does the discriminant need to be open source? If yes, it can still be trained locally. And run it as a classifier only on the smart contract, which can be a completely separate one. That can be updatable also.

Jerome Caporossi - Ledger Innovation Team (28/03/22)

Isabelle Blosch - Computer Vision Departement / LIP6 (04/04/22 16h)

6/ Bibliographie

[1] - Nicolas Perrin-Gilbert, course on supervised learning

Example of architecture for the MNIST number classification problem: Pixels as input neurons, and 10 output neurons. The softmax function then transforms the output values in a probability distribution. Entrainement: Iteratively, we pick random batches from the train data set. The minimisation of the negative log likelihood is a cross entropy minimization between the real probability distribution and the model. Gradient descent to do the minimisation. Backpropagation is a particular way of doing that for neural networks. This simple architecture can attain 90% of good classification on the MNIST database.
Convolutions: Adding filter layers to the input images.
Max-Pooling:A problem with the output feature maps is that they are sensitive to the location of the features in the input. One approach to address this sensitivity is to down sample the feature maps. Operation that calculates the maximum value in each patch of each feature map. Highlights the most present feature in the patch, not the average presence of the feature in the case of average pooling.
Dropout: Technique used to prevent overfitting. Dropout works by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0 at each update of the training phase.

GANs: Given a training set, this technique learns to generate new data with the same statistics as the training set. The generator is not trained to minimize the distance to a specific image, but rather to fool the discriminator.

[2] - Gunjan, V. K., Senatore, S., Kumar, A., Gao, X. Z., & Merugu, S. (Éds.). (2020). Advances in Cybernetics, Cognition, and Machine Learning for Communication Technologies. Lecture Notes in Electrical Engineering. https://doi.org/10.1007/978-981-15-3125-5

Artificial Neural Network and Partial Pattern Recognition to Detect Malware : Signature bases and behavior based Malware detection, heuristic based (works on unknown malware as well). Methodology used: Opcode extraction, Feature selection, Feature vector construction, Classifier architecture, Training, Testing. An unknown malware detection system that is automated. OpCode extraction was done with 3-gram model. ANN classification of malware families almost with accuracy of 100%.

Classification of Remote Sensing Images Based on K-Means Clustering and Artificial Bee Colony Optimization: An automatic method for the hyper spectral image classification and it classifies the images into multiple land cover objects.
Preprocessing: Improve the image data that eliminates the unwanted distortion and makes the images suitable for further processing. The first step is image reshaping and the second step is conversion of high resolution input satellite image into grayscale (Gray scale conversion makes the segmentation and classification easier.)
Segmentation: Dividing an image into smaller objects (simplify the representation of an image). Here K-Means Clustering Algorithm is an unsupervised approach which is used on unlabeled data: Determine the similar groups in the data and numbers of groups are represented with the variable K.
Classification done with *Swarm* Intelligence: Particle Swarm Optimization Algorithm (This algorithm is based on the bird flocking), Artificial Bee Colony Optimization (inspired by the nature of honey bees for solving unconstrained functions).

Smart KYC Using Blockchain and IPFS: The proposed system will replicate the functionality of the legacy KYC system. By using the immutable property of Distributed Ledger Technology (DLT) and InterPlanetary File System (IPFS), a tamper-proof system can be formed. We need to shift gears and abandon the old methodology of doing KYC with and document verification.
DLT: recording of transaction of assets in detailed format stored in a digital system simultaneously at multiple places. Used to store both static and dynamic data such as registry and transactions.
IPFS: A p2p protocol designed to act as a ubiquitous file system for all computer systems and is open source. Every node possesses a collection of hashed files. Any client who wants to retrieve any of these files is presented with a simple abstraction layer where only a hash of the file needs to be called to get the file. IPFS then digs across the nodes and supplies the client with the called file.

The system proposed in this paper incorporates all the functionality provided by a standard KYC system. The user provides a username for the creation of the wallet which will be used for storing all data relevant to him. Document submission method and third party (approving authority) verification. Only the hash is stored on the blockchain.

A Comparative Case Study on Machine Learning Based Multi-biometric Systems:
A Multibiometric system which can be formed by clubbing two or more traits has shown the better result in developing a more secure and reliable authentication system. Generally, it takes any one of the biometric identities such as face, finger, iris, retina, signature, gait, hand geometry or palm prints in single/unimodal. Biometric recognition system which first enroll the users in advance and testify them during security checks. Several *limitations* such as presence of noise in data collection at sensors, difficulties in grabbing the identity from some users (non universality), problem of individuality (similar faces of twins). Suffer to fulfill the requirement in commercial and security environments. Need to mix biometric modalities, makes systems also more robust. Also helps with common problems of sensor collecting data: Noise (reasons are defective conditions such as bad illumination, dirt on lenses…). One new problem arises: possibility of a tie: use of NNs. Table summarizing divergent architectures and their use. For face recognition Radial Basis Function utilized with one hidden layer.

[3] - Meng Joo Er, Shiqian Wu, Juwei Lu, & Hock Lye Toh. (2002). Face recognition with radial basis function (RBF) neural networks. *IEEE Transactions on Neural Networks*, *13*(3), 697-710. https://doi.org/10.1109/tnn.2002.1000134

Machine recognition of human faces is a widely researched area because of various needs from the need to statically match a face with biometric documents to real-time matching of surveillance video images. Considered still a difficult problem, the trickiest being real time identification. Reasons: Face images are highly variable and sources of variability are sometimes variable to individuals (Pose, hair, make up..) and even sometimes dependent on the environment, lighting.. Combined ,they can be greater than the variations due to changes of face identity.
Challenges: 1\ What features can be used to represent a face under environmental changes? And 2\ How to classify a new face image based on the chosen representation?
Statistical vs NN approach. NNs have many advantages such as learning ability and good generalization. Most widely used for facial recognition are **multilayered networks** coupled with **backpropagation**. Disadvantages: Computationally heavy, no optimality guarantee (and in my opinion: explicability).
Features of RBF networks: universal approximators, good approximation property, learning speed is fast because of locally tuned neurons, compact topology.
Moody and Darken: “RBF neural networks are best suited for learning to approximate continuous or piecewise continuous, real-valued mapping where the input dimension is sufficiently small.” When implemented, following **characteristics** are held:

- High dimensions: 128 x 128 pictures means 16 384 features.
- Small sample set: Only 1-10 images per person.

Those are the key differences with classical pattern recognition like character recognition. Leads to the following **problems**:

- Overfitting problem. Often if the dimension is comparable to the training set size, poor generalization.
- Overtraining. High dimensions means slow convergence.
- Small-sample effect. For an application with a large number of features, the training sample size needs to be large. Even Exponentially higher.
- Singular problem: ?

Methods to circumvent those problems:

1) Number of input variables is reduced through feature selection with **PCA**. Then generate a set of most discriminant features using **FLD**. Same features are compacted as close as possible.
2) New clustering algo so that homogeneous data could be clustered.
3) 2 Criteria for width estimation of RBF units controlling the generalization of the classifier.
4) New hybrid learning algo is presented to train de RBF NN so that the dimensions of the search space are reduced in the gradient paradigm.

Architecture:

Geometrically, the key idea of an RBF neural network is to partition the input space into a number of subspaces which are in the form of hyperspheres. Accordingly, clustering algorithms (k-means clustering…) are often used in RBF NN. However those are unsupervised algorithms as no category information about patterns is used. Therefore, the use of class membership implies that we should propose a supervised procedure to cluster the training patterns and determine the initial Gaussian widths.

Extracting Face Feature:

PCA**:** Dimension Reduction. PCA can be thought of as fitting a p-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a principal component. If some axis of the ellipsoid is small, then the variance along that axis is also small. projecting each data point onto only the first few principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible.

FLD: The goal is to project the data to a new space. Then, once projected, they try to classify the data points by finding a linear separation. The idea proposed by Fisher is to maximize a function that will give a large separation between the projected class means while also giving a small variance within each class, thereby minimizing the class overlap. In other words, FLD selects a projection that maximizes the class separation: A large variance among the dataset classes / A small variance within each of the dataset classes.

Initialization of RBF NN:

Structure Determination: Search of hidden layer.

Estimation of Widths: select the widths in such a way that they would minimize overlapping of nearest neighbors of different classes to preserve local properties, as well as maximize the generalization ability of the network.

Hybrid Algo:

The adjustment of RBF unit parameters is a nonlinear process while the identification of weight is a linear one. Though we can apply the gradient paradigm to find the entire set of optimal parameters, the paradigm is generally slow and likely to become trapped in local minima. Here, a hybrid learning algorithm, which combines the gradient paradigm and the linear least square paradigm to adjust the parameters, is presented. Two passes : forward and backward.

Results:
Experiments done on the ORL database (400 images of 40 individuals, and for some it was taken at a different time: causes nice variability), 200 were set in the training set and 200 in the test set. Clustering is done before learning and the separation of data becomes better and better as the dimension decreases. Error of classification after learning: If the information is sufficient (feature dimension is larger than 20), the results are stable. Otherwise, the error rate will increase drastically. On the other hand, it does not mean that more information (dimension is larger than 30) will result in higher performance. The reason may be that high dimensions will lead to complexity in structure and increase difficulty in learning as the addition of some unimportant information may become noise and degrade the performance.

Discussion:
General and efficient approach for designing an RBF neural classifier to cope with high-dimensional problems in face recognition. Error decreases in function of the number of dimensions with PCA but doesn't decrease monotonically with PCA + FLD, even increasing at some point. The choice of data is crucial for good generalization. A dataset has been constructed with 600 images of 30 individuals, in different poses and variability in facial expression. The results are 100% rate classification. Concerning the initialization, Theoretically, the final results should be the same regardless of initial parameters if the learning algorithm is good enough for optimization. Difficulties for neural networks to train in face recognition is the small sample data. This severe limitation always results in poor generalization. One method to help with generalization is adding noise. In this experiment it didn't give any better results with small noise and bigger noise deteriorated the results. High dimension may be one of the reasons that lead to poor generalization.

Conclusion:
When the number of dimensions is high (close to the input size) which is common for face recognition problems, the system easily overfits and generalizes poorly. This RBF approach tackles the problem. Simulations show excellent results in terms of precision and efficiency.

[4] - Lior Goldberg Shahar Papini Michael Riabzev. (2021) Cairo – a Turing-complete STARK-friendly CPU architecture.

Proof systems allow one party to prove to another party that a certain statement is true. Often statements will be represented in terms of polynomial equations. This makes the process of representing a statement that one wishes to prove or verify complicated. Cairo allows writing a program that describes that statement, instead of writing a set of polynomial equations.

The naive approach of statement proof would be for a user to send the input of the computation to the other user for him to redo the computation and verify the output. Issues: lack of Privacy, And Efficiency. ZK solves Privacy and succinct verification, exponentially more efficient then re execution.
Cairo: Efficient (the corresponding AIR (algebraic intermediate representation) will be as efficient as possible), Practical (supports conditional branches, memory, function calls, and recursion), Proofs for Cairo programs are generated frequently and verified by an on-chain contract, more restricted memory model (the values of all the memory cells are chosen by the prover and do not change when the code is executed), Builtins (The Cairo architecture supports the implementation of predefined operations directly), efficient public memory, proving programs where only the hash (rather than the code) is known to the verifier and proving multiple different programs in one proof to reduce the amortized verification costs.
CPU AIR --> CAIRO
The Cairo framework is designed to convince a verifier that a piece of code ran successfully with some given output.
Von Neumann architecture: moving from a computation described as a computer program to a computation described as an AIR.
A program, called “the bootloader” computes and outputs the hash of the bytecode of
another program and then starts executing it as above. Improves privacy and scalability as the verifier only needs the hash of a program.
When designing ordinary instruction sets that will be executed in a physical chip built of transistors, it should minimize the latency of the execution and the number of required transistors. The metric for an efficient AIR is different: we have to minimize the number of variables in a system of polynomial equations (nb of trace cells used).
No general purpose registers, and all the operands of an instruction are memory cells.
A Cairo Program has input, data not shared with the verifier. The output which is generated during the execution of the program is shared with the verifier (The prover sends the start and end addresses of the segment in the public memory).
Cairo doesn’t have a special instruction to invoke a builtin. Instead, one should simply read or write values in the memory cells affected by the builtin. It just means that the same memory is shared between the CPU and the builtins. Adding builtins implies adding constraints, which increases the verification time.
The deterministic machine is used by the prover and non deterministic by the verifier.

[5] - Alejandro Acien , Aythami Morales , Julian Fierrez, Ruben Vera-Rodriguez. (2021). BeCAPTCHA-Mouse: Synthetic Mouse Trajectories and Improved Bot Detection. https://arxiv.org/pdf/2005.00890.pdf

How to distinguish between human users and artificial intelligence during computer interactions is not a trivial task. This challenge was discussed by Alan Turing in 1950. Bots are expected to be responsible for more than 40% of the web trac with more than 43% of all login attempts in the next few years. Also, bots are used to influence and divide society on social networks. Bots are becoming more and more sophisticated, being able to mimic human online behaviors. Also,

algorithms to distinguish them are also getting very complex. 2 common methods: Active Detection (online tasks that are dicult for software bots to solve while being easy for legitimate human users to complete) and Passive Detection (transparent and analyze the users behavior). The Turing Test was designed as a task in which machines had to prove they were human, meanwhile in current CAPTCHA systems humans have to prove they are not machines: The responsibility of proving is on the wrong side.
Human - System interaction depends on the human, sensor, task. fusion of 2 biometric modalities (mouse dynamics and keystrokes) has been shown to outperform significantly each individual modality.
BeCAPTCHA based on two points: Mouse dynamics to extract neuromotor features, Generating synthetic mouse trajectory to improve the learning of bot detectors.

Mouse Trajectory Analysis:

Fine neuromotor movements are difficult to emulate for bots. Sigma Lognormal model to model the trajectories. The model states that the velocity profile of the human hand movements can be decomposed into primitive strokes. 6 parameters and 6 features for each plus N = 37 feature set size.

Trajectory Synthesis:

A mouse trajectory is defined by two main characteristics: the shape and the velocity profile. Two methods for generating such trajectories:

- Function-based Trajectories: First define initial point and ending point, second choose one of the 3 velocity profiles: Constant, logarithmic, Gaussian. Third, generate a sequence X between initial and final points according to the velocity profile and x to the shape profile (linear, quadratic, and exponential). Another important factor to consider is the number of stops, which can be modeled by calculating the mean and std deviation, of the number of points in human examples. Synthesis using Gaussian distribution.
- GAN based trajectories: The aim of the Generator is to fool the Discriminator by generating fake trajectories. The input of the Generator consists of a seed vector of R random numbers, The output of the Generator are two coordinate vectors x, y. The input of the Discriminator consists of a batch including two types of trajectories: Bot(Generated by the generator) and Human(real mouse trajectories DB). During the training phase, the GAN architecture will improve the ability of the Generator to fool the Discriminator. The process of learning is guided by the real mouse trajectory DB. Once the Generator is trained, we can use it to synthesize mouse trajectories very similar to the human ones. Then the Discriminator is used as a classification layer to distinguish between bot and real mouse trajectories.

Experiment:

5k of human trajectories, 5k gan and 5k function, all have a variety of lengths, directions, and velocities. 70% / 30% for train / test. Short trajectories have lover scores because less information. Second, logarithmic trajectory shapes achieve the worst classification performance, as we expected, because the shape of logarithmic functions better the human trajectories shapes. The velocity profile is the most significant parameter. The GAN Generator results in lower classification errors compared with the function-based method. This is surprising after visualizing the high similarity between human and GAN-generated trajectories. The synthetic samples generated by the GAN Generator seem very realistic to the human eye, the RF

classifiers were capable of detecting synthetic samples with high accuracy. These high classification rates suggest that GAN generators introduce patterns that allow its detection.
Conclusion:
These generators are very useful both training stronger bot detectors and evaluating them in comprehensive and worst case scenarios. We have been able to discriminate between humans and bots with up to 98,7% of accuracy, even with bots of high realism, and only one mouse trajectory as input. Random Forests (RF) have demonstrated to perform the best in all scenarios evaluated followed by an LSTM network.

[6] Matus Drgon, Lamprini Georgiou, and Aggelos Kiayias. (2020). Robust KYC via Distributed Ledger Technology. https://www.researchgate.net/profile/Matus-Drgon-2/publication/346485525_Robust_KYC_via_Distributed_Ledger_Technology/links/5fc4abb8a6fdcc6cc684f44f/Robust-KYC-via-Distributed-Ledger-Technology.pdf

[7] Justin Morgan. (2021). Clustering Web Users By Mouse Movement to Detect Bots and Botnet Attacks. https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=3792&context=theses

[8] Tiago S. Nazar´e, Gabriel B. Paranhos da Costa, Welinton A. Contato, and Moacir Ponti. (2017). Deep Convolutional Neural Networks and Noisy Images. https://sites.icmc.usp.br/moacir/papers/Nazare_CIARP2017_DNN-Noise.pdf
Code Available
Deep Convolutional neural networks have really good results in image classification problems. However, little research has been done to understand the impact of image quality on the classification results.
Models are often trained on noisy/restored training set versions and
tested on images with the same noise configuration. Usually they perform worse than
a model trained and tested on the original data.
Here deep NN are targeted which are capable of learning even from low quality data, then
If adding noise, restoration to the training model can help build more resilient models.
Restoring of models is done using the Non-Local Means algorithm or by median filtering 3x3.
Gaussian noise applied on images seems less impactful for human eyes than Salt and pepper noise. Even when denoising techniques were applied, the accuracy still decreased. Denoising creates blurry images, removing relevant information.

[9] Gabriel B. Paranhos da Costa, Welinton A. Contato, Tiago S. Nazare, Joao E. S. Batista Neto, Moacir Ponti. (2016). An empirical study on the effects of different types of noise in image classification tasks.
https://arxiv.org/pdf/1609.02781.pdf