BLOCKCHAIN AND CRYPTOCURRENCIES

BLOCKCHAIN FOR SNEAKERS AUTHENTICATION AND GENERATION OF RELATED NFTS WITH GAN

June 2022

Francesco Vannoni
francesco.vannoni2@studio.unibo.it

Mirko Del Moro
mirko.delmoro@studio.unibo.it

Contents

Abstract

The sneakers market is one of the most important trends nowadays. Besides the usual market, the reselling one is born: companies such as Nike or Adidas put on the market a limited number of sneakers per model, and people try to buy them not only for themselves but often to resell them to a higher price. This new market has brought about counterfeiting problems. To overcome this problem we propose an authentication system based on blockchain. Each sneaker is associated with a QR Code which is linked to the immutable information contained in the blockchain. In addition to that we create an associated NFT to add uniqueness to the product. This NFT is generated with a style transfer that tries to combine the sneakers' image and an artwork generated by a Generative Adversarial Network.

1 Introduction

Our task is to create a system based on blockchain to authenticate sneakers. The task can be split into two subproblems: the development of a blockchain containing the necessary information to authenticate the products and the creation, by means of a GAN, of an NFT associated with the product itself.
The utility of a system based on blockchain is that data are immutable and can't be modified, while the NFT is created to add uniqueness to the product itself.
A summary of our system can be seen in Figure 1.

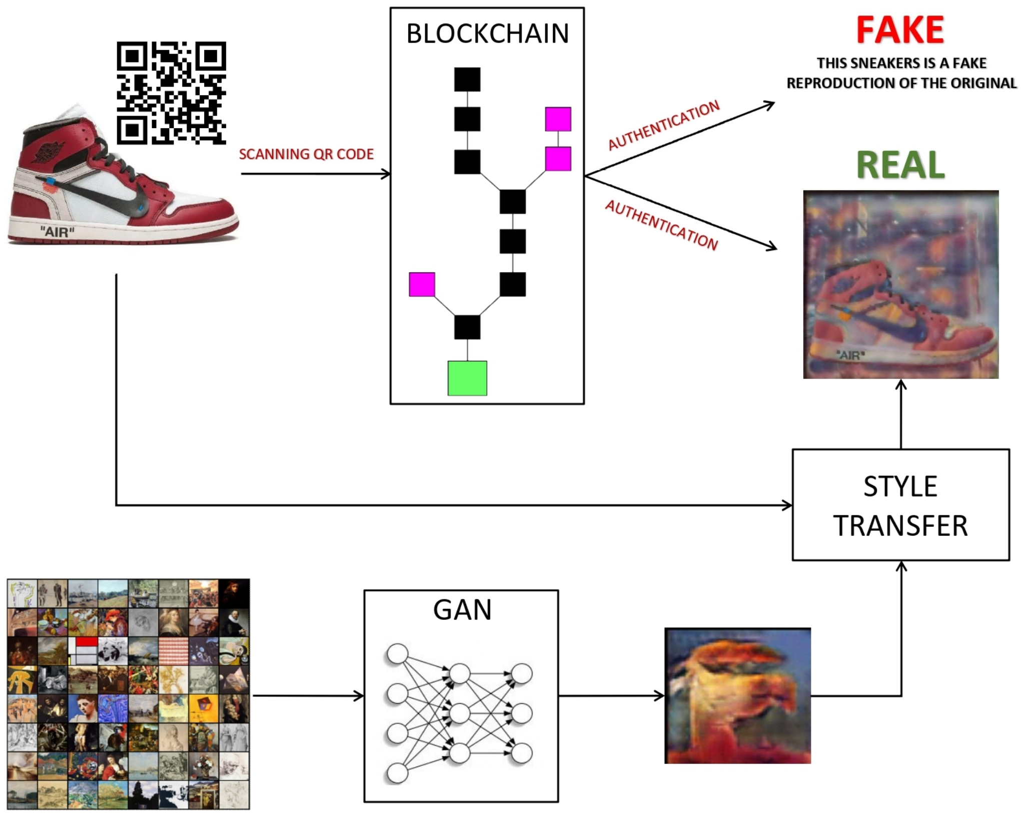


Figure 1: Summary of the system

We create a QR Code for each sneaker containing the identifier of the transactions stored in the blockchain. By scanning the QR Code we check if the identifier exists in a transaction of the blockchain. If it doesn't exist the user receives a message telling him that the product is fake. On the contrary the user receives a confirmation of the authenticity of its product and the correspondent NFT is shown to him when the transaction exists. The NFT image is generated through automatic tools. The sneakers' image and a style image are passed as input to a style transfer which combines them and generates a new image. Also, the style image is automatically produced by a Generative Adversarial Network which is trained on a dataset of artworks. In order to deal with the smart contract and the mining process for the NFT, we have worked on the Ropsten Test Network.

2 Blockchain for Sneakers Authentication

Blockchain is one of the most trendy technologies of the last year [1]. A blockchain is a growing list of records, called blocks, that are securely linked together using cryptography. An important blockchain feature is the immutability of data. Once that data has been validated and accepted it is stored in a block and cannot be modified anymore.
We have tried to exploit this feature to create a system able to authenticate products, in particular sneakers.

2.1 Blockchain Features

The blockchain we have developed is a very simple one. In this section we are going to list the structural choices we have made:

- **Blocks Hash**: each block id has been computed through a hash function. In particular, we have used the "SHA-256" hash function. Each block contains information regarding the hash of the previous block.
- **Transactions Hash**: for simplicity reasons, we haven't compute the hash of each transaction.
- **Difficulty**: as far as the chain becomes longer we have incremented a difficulty factor which is used to make it more difficult to mine a block. In particular, the difficulty factor represents the number of zeros that the block has to start with.
- **Proof of Work**: we have simulated the proof of work validation strategy. The functions try different values of nonce to get a hash that satisfies the difficulty criteria.

2.2 Transactions Structure and Data Retrieval

The data we need to authenticate has been stored in transactions. The transactions we have developed are very simple but represent the central part of the whole system. Each transaction contains two pieces of data:

1. **ID**: the transaction id
2. **NFT Link**: the link to the NFT corresponding to the sneaker

In order to verify if a sneaker is authentic, we provide the client with a QR Code. By scanning the QR Code the user is redirected to a webpage. From the webpage a request is automatically sent to the blockchain. In particular, the request contains the id coded by the QR Code. The id is extracted and is used by the system to check if it corresponds to a real transaction in the blockchain. If the transaction is found, the system will show the user the correspondent NFT. Otherwise, the user will see a message telling him that the product is fake.

3 NFT

For each sneakers, we have associated an NFT [2], which is automatically generated. We have created a style image by training a Generative Adversarial Network on an artwork dataset and we have implemented a style transfer that combines the style image with the sneakers' picture and generates the NFT image. By using the Ropsten test network we have deployed the NFT smart contract and minted the NFT on the Ethereum blockchain in order to make it impossible to edit, modify, or delete.

3.1 GAN Generation

The approach we have followed to generate the style image is based on the Generative Adversarial Network [3]. GANs are deep generative models composed of two networks, a generator, and a discriminator, which are one against the other.
Generative modeling is an unsupervised learning task in machine learning that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset.
One neural network, called the generator, generates new data instances, while the other, the discriminator, evaluates them for authenticity. The discriminator decides whether each instance of data that it reviews belongs to the actual training dataset or not.

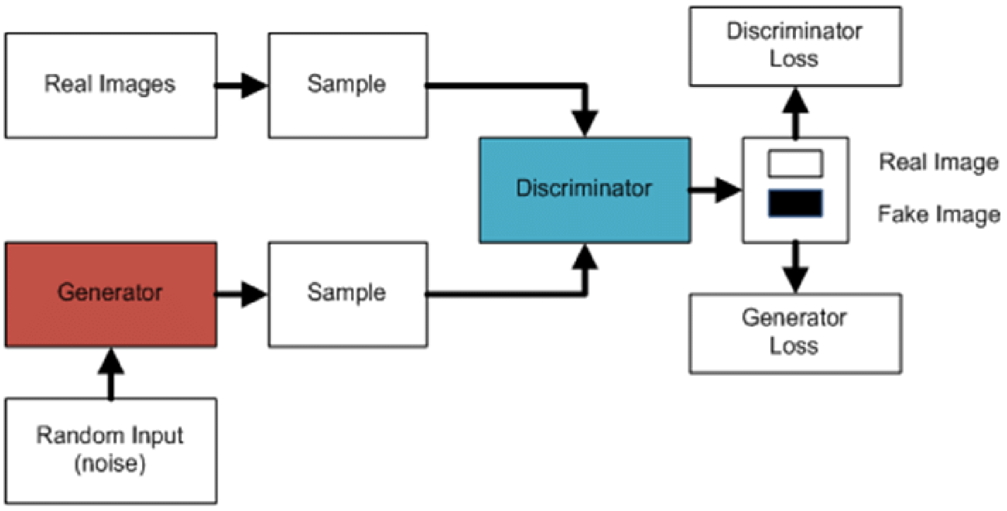


Figure 2: Gan Architecture

3.1.1 Dataset

We want to automatically generate images in the art domain. We have used a dataset publicly available on Kaggle named "Best Artworks of All Time". It contains a collection of artworks of the most influential artists of all time. The data are divided into folders, one for each artist, but for our task, the artist categories are not needed so we have used a unique folder with all artworks without the author's name.
We have worked with 8683 artworks of 50 different artists.

3.1.2 Generator

The input to the generator is typically a vector or a matrix of random numbers which is used as a seed for generating an image. The generator will convert a latent tensor into an image tensor. To achieve this, layers of transposed convolutions are used.
After each deconvolutional layer we have used Batch Normalization and ReLU as activation, except for the last layer where you have used Tanh as the activation function.

3.1.3 Discriminator

The discriminator Network discriminates the manipulated pictures, takes images as input, and tries to classify them as “real” or “generated”. We have used a convolutional neural network (CNN) which outputs a single number for every image. Further, we have used a stride of 2 to progressively reduce the size of the output feature map.
After each convolutional layer we have used Batch Normalization and LeakyReLU as activation. Lastly, we have flattened the last layer and passed it through a Sigmoid.

3.1.4 Training and Results

To work with the images of the artworks we have resized them to 64x64 and we have center cropped them to ensure that they are all in the same shape and size. We also have created a squared grid containing 64 different pictures and for the training, we have considered each grid as a single image. For what regards the generator network, we have used the discriminator as a part of loss function. We have generated a batch of images using the generator and we have passed it into the discriminator. We have calculated the loss by setting the target labels to 1 (real), in order to "fool" the discriminator and we have used the loss to perform gradient descent i.e. change the weights of the generator, so it gets better at generating real-like images to “fool” the discriminator.
Since the discriminator is a binary classification model, we have used the binary cross-entropy loss function to quantify how well it can differentiate between real and generated images.
We have defined a fit function to train the discriminator and generator in tandem for each batch of training data.
We have trained for 150 epochs with a batch size of 128 and we have used Adam optimizer for both generator and discriminator. During training, the generator progressively becomes better at creating images that look real, while the discriminator becomes better at telling them apart.

The process reaches equilibrium when the discriminator can no longer distinguish real images from fakes.
The images begin as random noise, and increasingly resemble new artworks over time. Figure 3 shows an example of an output of the GAN. We get a grid containing 64 generated artworks that don't exist in nature.

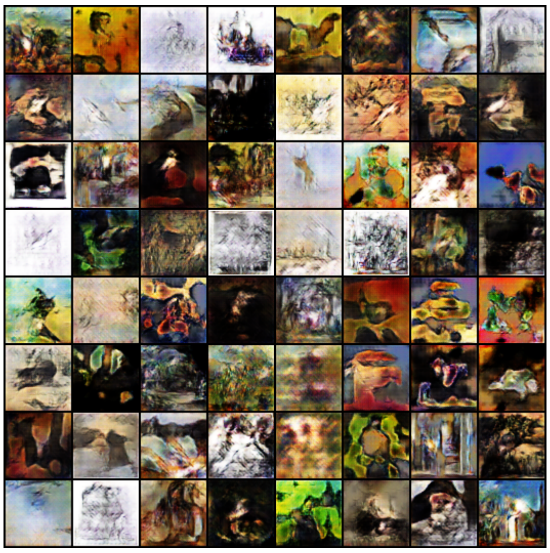


Figure 3: Example of generated images through GAN

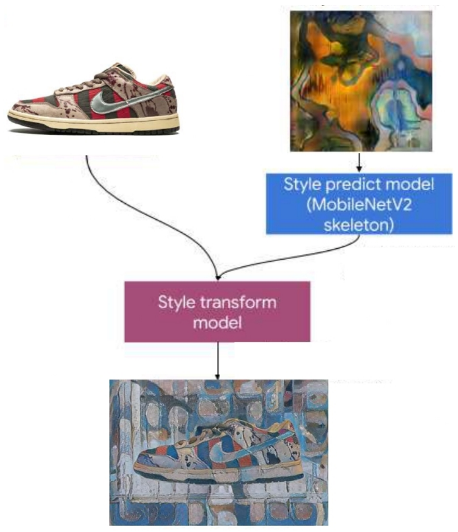
3.2 Style Transfer

Style transfer is an optimization technique used to create a new image by blending two images. The two images in input are a content image and a style reference one, and the output image looks like the content image, but "painted" in the style of the reference image [4]. We have taken the sneakers' image as content image and as style image we have picked one of the 64 pictures output by the GAN randomly.

This Style Transfer model consists of two submodels:

1. **Style prediciton model**: a MobilenetV2-based neural network that takes an input style image to a 100-dimension style bottleneck vector.

2. **Style transform model**: a neural network that applies a style bottleneck vector to a content image and creates a stylized image.



3.3 Smart Contract and Minting

Since uploading images to the blockchain is expensive, we have uploaded the link of the images to the blockchain and we have stored the image on an Interplanetary File System (IPFS). The process we will show in this section has been done following the instructions of [5].
As a first step, we have exploited "Pinata" to upload for free images using IPFS. Once the image is created and uploaded with Pinata we have to associate the metadata to create the NFT. The metadata contains four fields:

- image: link to the image
- token_id: id of the token, corresponding to the id of the transaction
- name: NFT name
- sneakers_type: model of the sneakers

We have generated a specific .json file for each image and we have uploaded it to Pinata.

3.3.1 Environment Set-up

In order to make Ethereum development easy, we have used the blockchain developer Alchemy. We have worked on the Ropsten test network since it is free to use.
As a digital wallet, to send and receive transactions, we have used Metamask. Working on the Ropsten test network we have used a faucet to get fake free Ethereum.
After developing the metadata, the blockchain, and the digital wallet, we have worked on smart contracts. In order to do it, we have used the ERC-721 Standard. We have exploited the development environment Hardhat to build smart contracts locally before deploying them to the live chain. Hardhat is a software used to compile, deploy, test, and debug your Ethereum software. It helps developers manage and automate the recurring tasks that are inherent to the process of building smart contract.

3.3.2 Smart Contract deployment and compilation

The smart contract is written in Solidity by exploiting the openzeppelin library. A counter has been implemented to keep track of the total number of NFTs minted and assign the unique ID on our new NFT. Two additional variables have been used as inputs: "address recipient" (the address of where the NFT should be sent to) and "string memory tokenURI" (the URI, Uniform Resource Identifier, of the NFT metadata).
We have saved on an environment file the Alchemy API key and the Metamask private key to be able to interlink between the two. After that, we have implemented the smart contract and we have compiled it. Once compiled we have deployed the smart contract by using a script in javascript. If no error is found, the script returns the address corresponding to the contract deployed.

3.3.3 Minting

Minting refers to the process of turning a digital file into an NFT on the Ethereum blockchain. This NFT has been stored on the decentralized database making it impossible

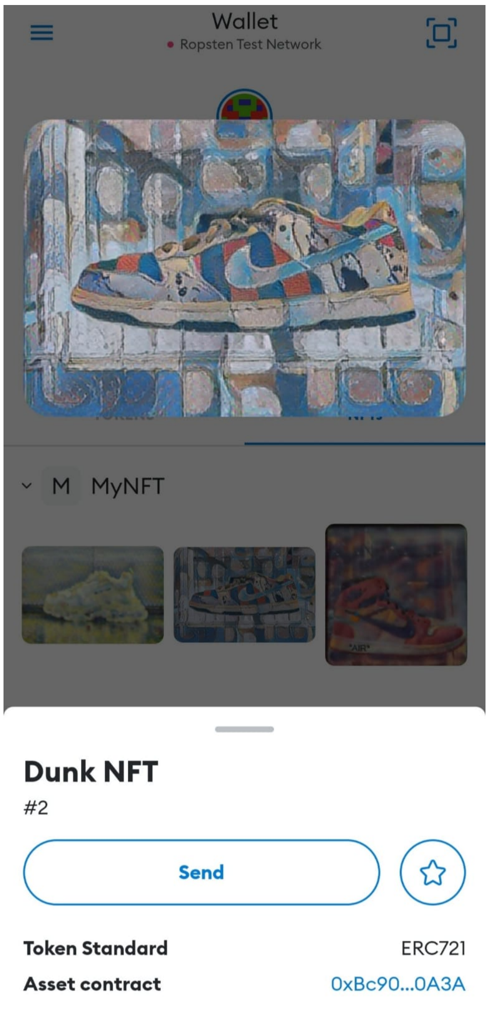
to edit, modify, or delete.
In order to mint, we have to specify some variables:

- API URL: the URL from the alchemy app
- Private Key: private Metamask key
- Public Key: public Metamask key
- Contract Address: address of the contract deployed
- Transaction Variables: necessary variables to mint (from, to, gas, nonce...)

The last step before minting is to sign the transaction. Once the transactions have been signed we can mint.

3.3.4 Viewing minted NFT and Transactions

We can view the results of the process in the Metamask app (Figure 4).



Dunk NFT
#2

Token Standard	ERC721
Asset contract	0xBc90...0A3A

Figure 4: NFT on Metamask Wallet

In particular, we can see the details about the smart contract implemented as the Token Standard (ERC-721) and the contract address (corresponding to the address of the contract deployed)

4 Conclusions

In this section, we are going to discuss general conclusion on the project and possible developments. The system we have generated is able to exploit the blockchain to authenticate a product and is correctly linked to the corresponding NFT (if it exists). The whole system can be framed as a 2-step authentication. In order to get the final NFT, the data has to be validated by two blockchains.
The entire process has been developed on a test network (Ropsten) by using fake Ethereum, but it is easily applicable also in a real scenario.
The system could be improved by enhancing the security of the blockchain and making it more complex.
We could also improve the GAN system by enhancing the resolution of the image generated. To do it, we should train it with batches of a less number of images than 64. We haven't been able to do it due to the low amount of resources available.

5 References

[1] Y. Zhang, *Blockchain*, pp. 115–118. Cham: Springer International Publishing, 2020.

[2] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (NFT): overview, evaluation, opportunities and challenges," *CoRR*, vol. abs/2105.07447, 2021.

[3] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[4] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, "Neural style transfer: A review," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 11, pp. 3365–3385, 2020.

[5] "How to deploy nft smart contracts." `/https://betterprogramming.pub/how-to-deploy-nft-smart-contracts-9271ce5e91c0`.