

Ministry of Higher Education  
and Scientific Research

\*\*\* \* \*\*\*

University of Carthage

\*\*\* \* \*\*\*

**National Institute of Applied  
Science and Technology**



## **Mandatory Summer Internship Report**

**Field: Software Engineering**  
**Level: 4<sup>th</sup> grade**

**Subject:**

# **Creating an offline signature verification system using Convolutional Neural Networks**

Prepared By: **Amal Mtibaa**

**Company:**



**University year: 2018/2019**

Ministry of Higher Education  
and Scientific Research

\*\*\* \* \*\*\*

University of Carthage

\*\*\* \* \*\*\*

**National Institute of Applied  
Science and Technology**



## **Mandatory Summer Internship Report**

**Field: Software Engineering**

**Level: 4<sup>th</sup> grade**

**Subject:**

# **Creating an offline signature verification system using Convolutional Neural Networks**

Prepared By: **Amal Mtibaa**

Company:

**Wevioo**

<i><b>Company Responsible</b></i> <b>Mr. Khaled Ben Driss</b>	<i><b>Opinion of the Trainees Committee</b></i>

**University year: 2018/2019**

# Acknowledgments

Before any development on this professional experience, it appears important to begin this report with acknowledgments.

First, I want to warmly thank **Mr. Khaled BEN DRISS**, executive director at WEVIOO, for his welcome, his trust and his constant sharing of his expertise.

I also want to thank **Mme. Salma LANDOULSI**, my supervisor, for guiding the team in achieving the goal as well as her encouragement to maintain the progress in track.

I would like to express my deepest gratitude to **Mr. Mahmoud BEN AMOR** for giving necessary advice and guidance and arranging all facilities so my internship takes place in the best possible conditions. I choose this moment to acknowledge his contribution gratefully.

Finally, I want to thank the whole WEVIOO team for their welcome, their team work spirit and the amazing ambience especially my two good friends and colleagues: **Yasmine** and **Wassim**.

# Table of content

Acknowledgments .....	ii
List of Figures .....	iv
List of Tables.....	v
<b>1 Introduction .....</b>	<b>1</b>
<b>2 Host Company presentation.....</b>	<b>1</b>
2.1 Company creation .....	1
2.2 Field of activity .....	2
<b>3 Project presentation.....</b>	<b>2</b>
3.1 Project objective .....	2
3.2 Functional and non-functional requirements specification.....	2
3.2.1 Actors and functional requirements specification.....	2
3.2.2 Non-functional requirements specification.....	2
3.3 Project global architecture.....	3
<b>4 Internship environment .....</b>	<b>3</b>
4.1 Work team .....	3
4.2 Tools and Technologies .....	3
<b>5 Internship Journal.....</b>	<b>4</b>
5.1 Creating the signature verification system.....	4
5.2 Creating the demo Web application.....	4
5.3 Creating VAE model for skilled forgeries.....	5
<b>6 Work description.....</b>	<b>5</b>
6.1 Basics explanation .....	5
6.2 Offline Signature verification system .....	6
6.2.1 Data Collection.....	7
6.2.2 Pre-processing.....	7
6.2.3 Model creation .....	7
6.2.3.1 CNN architecture.....	7
6.2.3.2 Siamese network.....	8
6.2.3.3 Model definition and training.....	8
6.2.3.4 Result interpretation .....	9
6.2.4 Evaluation.....	10
6.2.5 Deployment.....	11
6.3 Creating the demo Web application.....	11
6.4 Creating the VAE model.....	13
<b>7 Acquired skills consolidation.....</b>	<b>15</b>
<b>8 Conclusion .....</b>	<b>15</b>
References & Bibliography .....	16

# List of Figures

<b>Figure 1.</b> Logo Wevioo .....	1
<b>Figure 2.</b> Global architecture of the check verification system .....	3
<b>Figure 3.</b> Gantt chart “Creating the signature verification system” .....	4
<b>Figure 4.</b> Gantt chart “Creating the web application” .....	5
<b>Figure 5.</b> Gantt chart “Creating VAE model” .....	5
<b>Figure 6.</b> Steps for creating the proposed signature verification system.....	6
<b>Figure 7.</b> Pre-processing pipeline for a signature.....	7
<b>Figure 8.</b> VGG19 Architecture .....	7
<b>Figure 9.</b> Siamese network Architecture.....	8
<b>Figure 10.</b> Example of an input for training the Siamese network .....	8
<b>Figure 11.</b> Trpilet cost function.....	9
<b>Figure 12.</b> Example of the testing algorithm outputs on two examples.....	9
<b>Figure 13.</b> Accepted and not accepted signatures according to distances and cosine of similarity.....	10
<b>Figure 14.</b> ROC curve of the developed model .....	11
<b>Figure 15.</b> Technical architecture for the developed web application .....	12
<b>Figure 16.</b> Example of the Front end for CIN information extraction.....	12
<b>Figure 17.</b> Example of the Front end for the developed check verification system.....	12
<b>Figure 18.</b> System diagram of the VAE architecture used.....	13
<b>Figure 19.</b> Example of data augmentation on a single signature .....	13
<b>Figure 20.</b> ROC plots for easy (left) and hard signatures (right) .....	14
<b>Figure 21.</b> T-SNE representation of the latent vector of a real signature and its forgery .....	14

## List of Tables

<b>Table 1.</b> List of static and dynamic features of a signature .....	5
<b>Table 2.</b> Testing results on different datasets (rounded).....	10
<b>Table 3.</b> False positive and false negative decisions.....	11
<b>Table 4.</b> Table of acquired skills consolidation.....	15

# 1 Introduction

No wonder that businesses recognize signatures as the primary way of authenticating transactions. People sign checks, authorize documents and contracts, validate credit card transactions and verify activities through signatures. As the number of signed documents has increased tremendously, so has the growth of signature fraud.

According to recent studies, check fraud costs banks about \$900M per year with 22% of all fraudulent checks attributed to signature fraud. With more than 70 million checks written each year in Tunisia, visually comparing signatures with manual effort on the hundreds of checks processed daily proves impractical.

That's why, the innovation department of Wevoo has thought of creating a robot that verify the validity of checks in order to automate the checks verification procedure.

It is in this context that the company has offered me to be part of this project as my 4rth year summer internship from the National institute of applied science and technologies (INSAT).

In the present report, I will describe the company as well as the project. Then I will specify the used technologies, the internship journal and the missions accorded to me. Finally I will end up with presenting the results and a conclusion of this work.

## 2 Host Company presentation

### 2.1 Company creation

Wevoo is an international consulting and digital services group. When it was founded in 1998, the group was known as OXIA. In 2016, the group decided to change its name to WEVIOO in order to unite its consulting services and digital services of innovation around a global and unique identity.



**Figure 1.** Logo Wevoo

## 2.2 Field of activity

Wevioo supports its customers in their digital transformation projects. Customer-oriented, the group brings its expertise and know-how in 3 areas:

- ✓ Consulting
- ✓ The Digital
- ✓ IOT (Internet of Things)

As a committed partner Wevioo provides its customers with digital innovation solutions that are perfectly suited to their agility, performance and international development challenges.

## 3 Project presentation

### 3.1 Project objective

This project aim to automate the check verification procedure by comparing the extracted signature of a given check to its client reference signature. In order to increase performance, the system will give the possibility to add the extracted signature to the client signatures database, so that the system will compare a given signature with more than one reference signature.

### 3.2 Functional and non-functional requirements specification

#### 3.2.1 Actors and functional requirements specification

Since checks verification is very critical, and in order to reduce the possibility of accepting a non-valid signature, the system can, in few cases, not judge and require the decision of a supervisor. This supervisor is the only actor interacting with the system. The robot automatically execute the check verification scripts with every new input. It will store the ones that the system couldn't decide on. So that, the supervisor can in any time consult those checks and judge manually. The robot will then use these decisions and increase his performances. It is important to specify that the company consultants wanted the decision to be based on a similarity measures, not an output of classification.

#### 3.2.2 Non-functional requirements specification

Our solution need to respect the following exigencies:

- ✓ Reliability: The system should maintain his performance in time.
- ✓ Auditability: The system should offer logs in order to easily find out the cause of an error.
- ✓ Scalability: The system should handle a wide variety of sizes.



### 3.3 Project global architecture

The following figure (Figure 2) captures the main processes that the robot execute with every new input: scanned check.

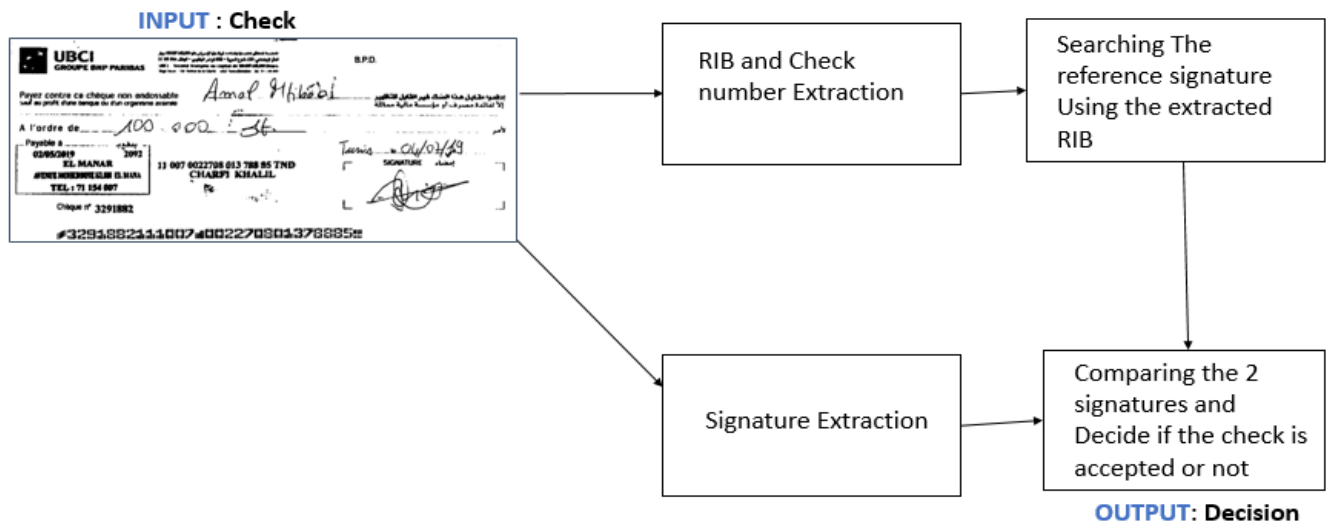


Figure 2. Global architecture of the check verification system

## 4 Internship environment

### 4.1 Work team

For this project, I worked with two interns with the supervision of Mme. Salma LANDOULSI and Mr. Mahmoud BEN AMOR. Few meetings with the company consultants were arranged in order to present our ideas and work progress.

### 4.2 Tools and Technologies

For the signature verification system, I created a deep learning model using:

- ✓ **Python:** an interpreted, high-level, general-purpose programming language. It offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows developers to write reliable systems. Developers get to put all their effort into solving a machine learning problem instead of focusing on the technical nuances of the language.



- ✓ **Keras:** an open-source neural-network library written in Python. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.



For the Web application, I used:

- ✓ **Angular 8:** a TypeScript-based open-source web application framework. It's the most preferred framework for creating interactive components for a website.
- ✓ **Flask:** is a micro web framework written in Python. Flask API is a drop-in replacement for Flask that provides an implementation of browsable APIs. It is an easy way to create RESTful web service for the developed python scripts.



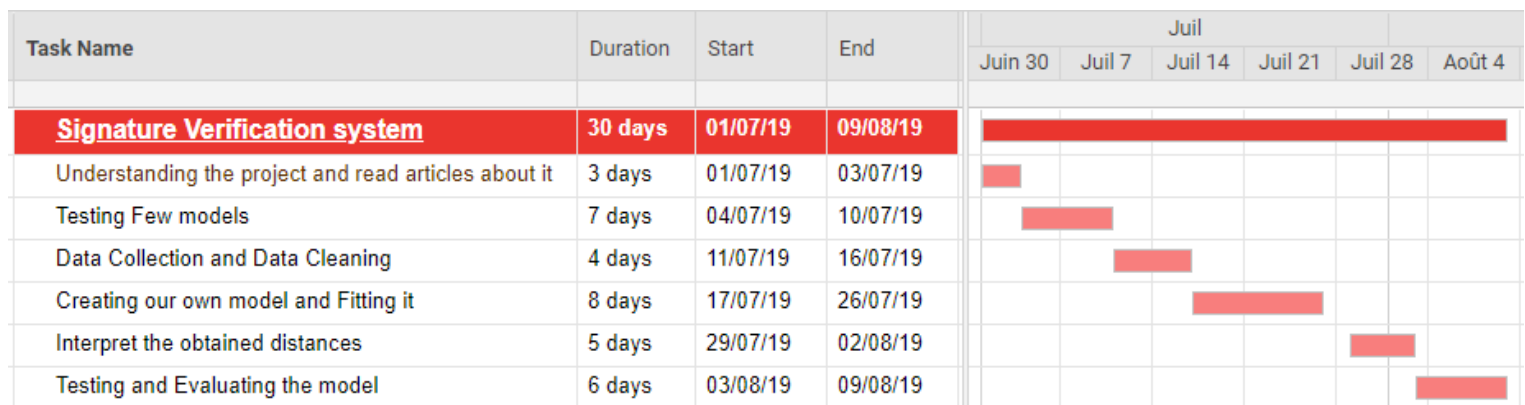
## 5 Internship Journal

During my internship, I worked on 3 main missions which are:

- ✓ Creating the signature verification system.
- ✓ Creating a Web Application for demo.
- ✓ Creating a VAE model for skilled forgeries.

### 5.1 Creating the signature verification system






My main mission was to create a deep learning model that compute the percentage of similarity between 2 given signatures and decide whether these signatures are corresponding to the same client or not. The following figure (Figure 3) shows the different steps of creating this system.



**Figure 3.** Gantt chart “Creating the signature verification system”

### 5.2 Creating the demo Web application






For demo purposes, and in order to convince the potential clients, I was asked to create a web application that display the offered functionalities of the check verification project as well as another project that extract information from a Tunisian national identity card (CIN). Figure 4 display here the different steps of this mission.

Task Name	Duration	Start	End	Août	
				Août 11	Août 18
<b>Creating the Web Application</b>	6 days	12/08/19	20/08/19		
Develop Front End for CIN Extraction	2 days	12/08/19	13/08/19		
Develop Front End for check verification	2 days	14/08/19	15/08/19		
Develop the FLASK API	1 day	16/08/19	16/08/19		
Establish the link between the different parts	1 day	20/08/19	20/08/19		

**Figure 4.** Gantt chart “Creating the web application”

### 5.3 Creating VAE model for skilled forgeries.

After creating our signature verification system, Mme. Salma Landoulsi proposed to test the VAE (Variational auto encoding) approach in order to detect skilled forgeries. The next figure (Figure 5) describes the main phases of this mission.

Task Name	Duration	Start	End	Août	
				Août 18	Août 25
<b>Creating VAE model</b>	8 days	21/08/19	30/08/19		
Read articles about VAE	3 days	21/08/19	23/08/19		
Create the vanilla VAE model	3 days	26/08/19	28/08/19		
Fitting the model	1 day	29/08/19	29/08/19		
Testing and evaluating the model	1 day	30/08/19	30/08/19		

**Figure 5.** Gantt chart “Creating VAE model”

## 6 Work description

### 6.1 Basics explanation

The physical act of signing a signature requires coordinating the brain, eyes, arms, fingers, muscles and nerves. Considering all factors in play, it’s no wonder that people don’t sign their name exactly the same every time. Personality, emotional state, health, conditions under which the individual signs, space available for the signature and many other factors all influence signature-to-signature deviations. Signature verification systems, however, are based on signature features to distinguish frauds from genuine. The following table (Table 1) present a list of these features.

Static features	Dynamic features
<ul style="list-style-type: none"> <li>- Shaky handwriting</li> <li>- Letter Propositions</li> <li>- Signature shape/dimension</li> <li>- Smoothness of curves</li> </ul>	<ul style="list-style-type: none"> <li>- Pen lifts</li> <li>- Speed</li> <li>- Pen pressure</li> <li>- Acceleration pattern</li> </ul>

**Table 1.** List of static and dynamic features of a signature

There are two types of signature verification systems in the market:

- **Offline signature verification:** Deployed where there is no scope for monitoring real time signature activity of a person. In other words, it uses only a static two dimensional image for verification. The features mainly used here, are static in nature: image texture, geometry and topology (shape, size aspect ratio etc.), stroke positions, hand writing similarity etc.
- **Online signature verification:** Capture of crucial dynamic features, such as speed, acceleration and pressure... This type of system is more accurate as even for the copy machine or an expert since it is virtually impossible to mimic unique behavior patterns and characteristics of the original signer.

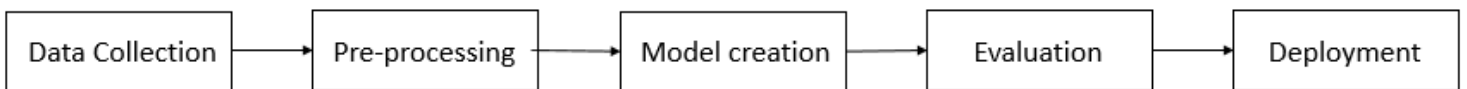
The range of signature forgeries falls into the following three categories:

- **Random forgery:** has little or no similarity to the genuine signatures. This type of forgery is created when the forger has no access to the authentic signature.
- **Unskilled (Trace-over) Forgery:** The signature is traced over appearing as a faint indentation on the sheet of paper underneath.
- **Skilled forgery:** Produced by a perpetrator that has access to one or more samples of the authentic signature and can imitate it after much practice. Skilled forgery is the most difficult of all forgeries to authenticate.

In Tunisian banks, when a client want to have a check book, he must sign a paper to save his signature. This signature is called the “reference signature”. Since the used signatures databases are two dimensional images, I developed an offline verification solution. And, since we have only one reference signature as a start, my solution can detect only the first 2 types of forgeries. For the skilled forgeries, I tested another approach that will be explained later.

## 6.2 Offline Signature verification system

The proposed signature verification system involves Convolutional Neural Networks (CNN) for extracting features and Siamese network for computing similarity. The following figure (Figure 6) present the steps for creating this solution.



**Figure 6.** Steps for creating the proposed signature verification system

### 6.2.1 Data Collection

Here I used the database offline SigComp2009 in [1] consisting of data from 79 individuals: 12 genuine signatures for each individual. I had to do some manual data cleaning to make sure that every individual is referenced by similar signatures that can be accepted.

### 6.2.2 Pre-processing

As all the SigComp2009 offline signatures are of different size, they must first and foremost be pre-processed. Each signature was gone through a pre-processing pipeline: it goes through gray scale transformation, then noise removal and black and white transformation. After that, it goes through padding before resizing to a NumPy array of size 224x224 which is the input size of the network. This pipeline is shown in Figure 7 as follows.

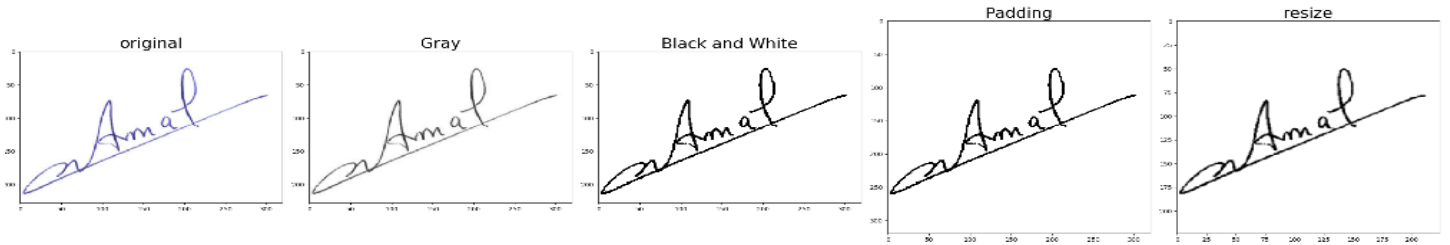


Figure 7. Pre-processing pipeline for a signature

### 6.2.3 Model creation

#### 6.2.3.1 CNN architecture

CNN consists of multiple layers, performing operations such as convolutions, max-pooling and dot products (fully connected layers). Convolutional layers and fully connected layers have learnable parameters that are optimized during training. The kind of architecture of CNN determines how many layers it has, what the function of each layer is and how the layers are connected. For my main training tasks, I used the VGG19 CNN architecture. This network contains a total of 16 layers with learnable parameters as shown in Figure8. I used the predefined Keras VGG19 model to transform a 224x224 signature into 512-dimentional vector.

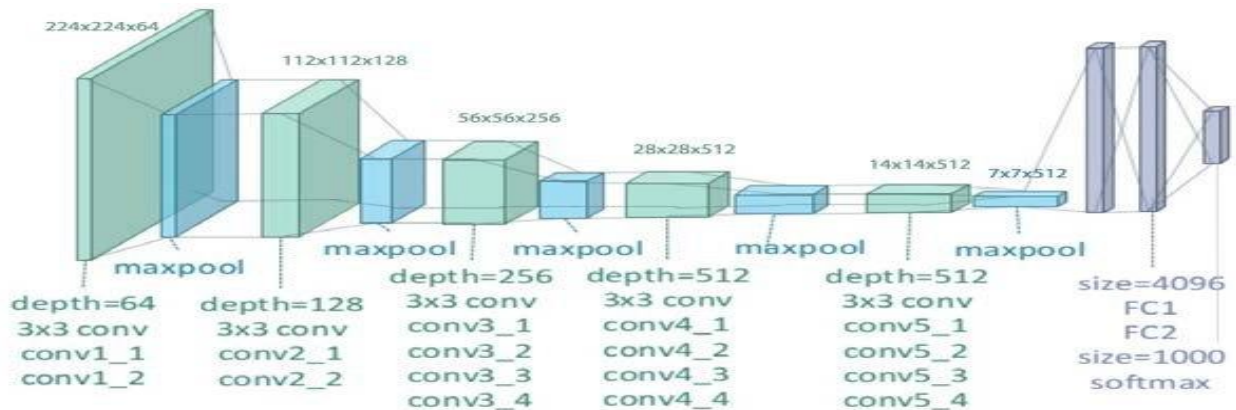


Figure 8. VGG19 Architecture

### 6.2.3.2 Siamese network

Siamese neural network is an artificial neural network that use the same weights while working in tandem on two different input vectors to compute comparable output vectors. The Siamese network has two input fields to compare two patterns and one output whose state value corresponds to the similarity between the two patterns as shown in Figure 9.

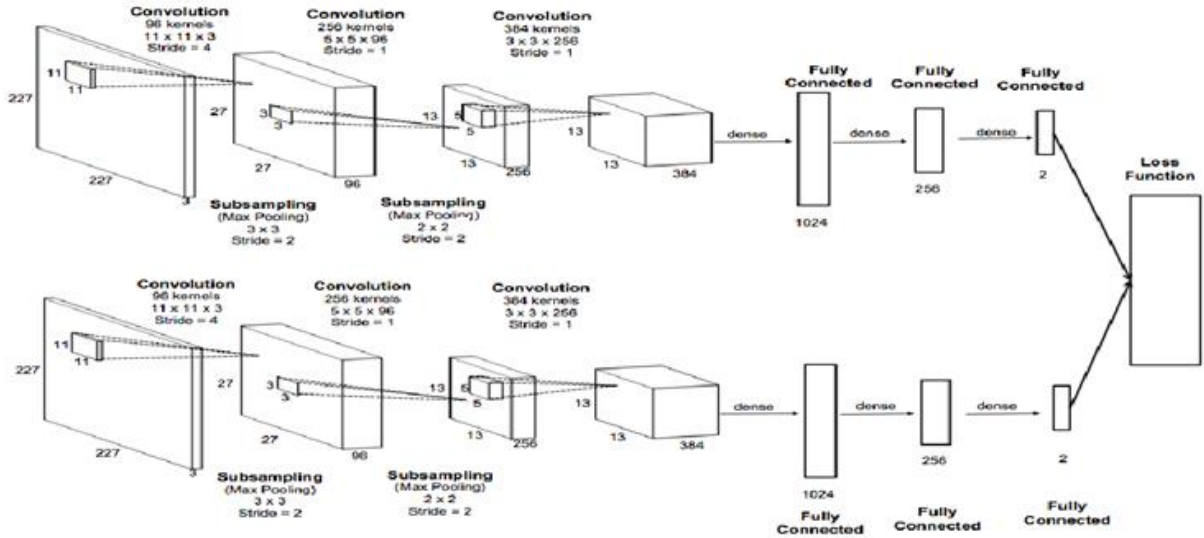


Figure 9. Siamese network Architecture

### 6.2.3.3 Model definition and training

For an image  $x$ , we denote its encoding  $f(x)$ , where  $f$  is the function computed by the VGG19 neural network. Learning in Siamese networks should be done with a loss function. The function that I used is called triplet loss.

Training will use triplets of images (A, P, N):

- A is an "Anchor" image which is a signature of a person.
- P is a "Positive" image which is a signature of the same person as the Anchor image.
- N is a "Negative" image which is a signature of a different person than the Anchor image.

The following figure: Figure 10 is an example of one input to train the Siamese network.

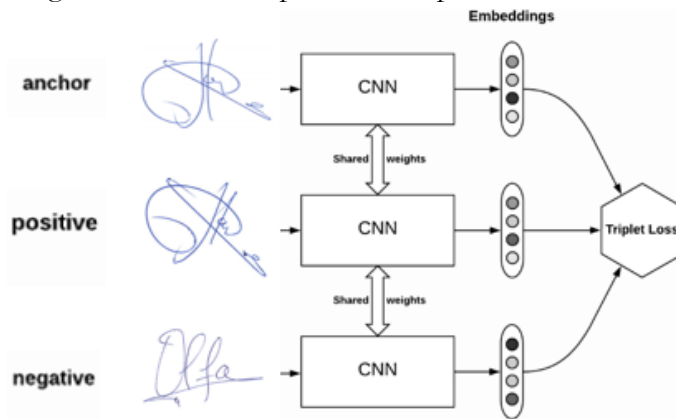


Figure 10. Example of an input for training the Siamese network

For learning by triplet loss, a baseline vector (A) is compared against a positive vector (P) and a negative vector (N). The negative vector will force learning in the network, while the positive vector will act like a regularizer. The triplet loss function tries to "push" the encodings of two images of the same person (Anchor and Positive) closer together, while "pulling" the encodings of two images of different persons (Anchor, Negative) further apart. Thus, I would like to minimize the following "triplet cost" in Figure 11.  $\alpha$  is called the margin. It is a hyperparameter that we picked manually and  $[x]_+$  is a notation of  $\max(0, x)$ . Here we used  $\alpha=0.2$

$$\mathcal{J} = \sum_{i=1}^m \left[ \underbrace{\|f(A^{(i)}) - f(P^{(i)})\|_2^2}_{(1)} - \underbrace{\|f(A^{(i)}) - f(N^{(i)})\|_2^2}_{(2)} + \alpha \right]_+$$

**Figure 11.** Trpilet cost function

After defining the model and preparing the corresponding dataset, the model was trained with 10 000 triplets: 10 epochs and 18 as batch size. I saved the weights of the trained model and I created an algorithm that take any 2 signatures, pre-process them, call the model to compute their encoded vectors and then return the Euclidean distance between them as well as the cosine of the angle between these 2 vectors. Figure 12 shows the output of this algorithm on 2 examples.

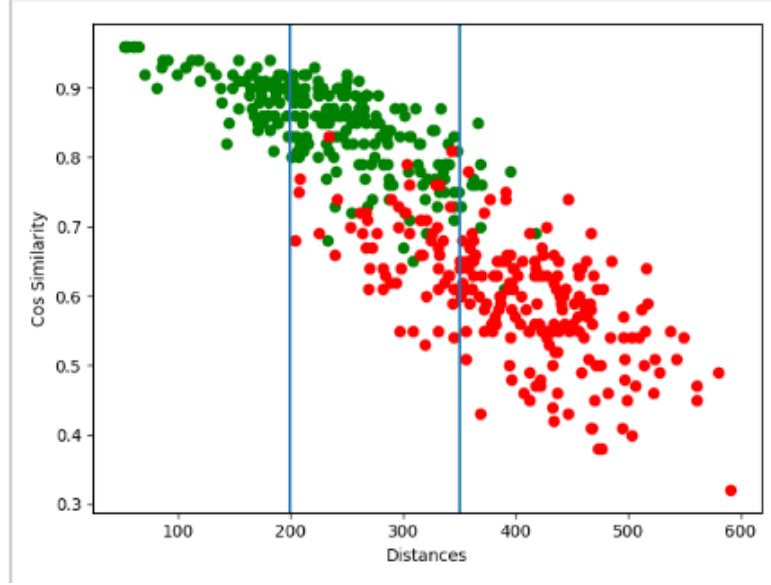


**Figure 12.** Example of the testing algorithm outputs on two examples

#### 6.2.3.4 Result interpretation

Now, I have to interpret the distance that the model gives as an output to judge whether two given signatures are similar or not. To do so, I created a testing dataset that have for each line two signatures from the training dataset and a label (1 for similar and 0 for not). I called the model and stored its outputs, so I can plot the labels according to the distance and the cosine as shown in Figure 13. The green points are the accepted signatures and the red ones aren't. I noticed that the signatures that have distances less than 200 are always accepted and the ones that have distances more than 350 are not. I based my decision mostly on the distance because that's what the model was trained on. However, I noticed that the cosine between these vectors could give an idea if they are similar or not. So, for the interval between 200 and 350, my algorithm consider the ones that have cosine more than 0.78 accepted and the ones that have less than 0.75 aren't. The interval between 0.75 and 0.78 is where the system cannot judge.





**Figure 13.** Accepted and not accepted signatures according to distances and cosine of similarity

#### 6.2.4 Evaluation

In order to evaluate the developed system and his capability of predicting whether two signatures are similar or not, I collected signatures from different databases. I even created my own by asking employees in Wevoo to sign in papers. Using these databases, I choose random samples to create a testing dataset that have for each line two signatures and a label (1 for similar, 0 for not). As shown in Table 2, the success range represent the percentage of well predicted signatures, the fail range represent the ones that the system failed at predicting, and the “can’t judge range” represent the percentage of signatures that the system couldn’t decide on.

The “Can’t judge range” was added in order to minimize the fail range. It also reduced the success range, but here, I focused more on the fail range due the criticality of the system. The accuracy here is the sum of the success and “can’t judge” range.

	SigComp2009 (468)	SigComp2011 (Dutch)(80)	Kaggle dataset (425)	Tunisian dataset (136)
Success Range	89 %	86 %	84 %	86 %
Fail Range	2.7 %	7.5 %	5 %	5 %
Can’t judge Range	8.1 %	6.25 %	9 %	8 %
Accuracy	97 %	92.25 %	94 %	95 %

**Table 2.** Testing results on different datasets (rounded)

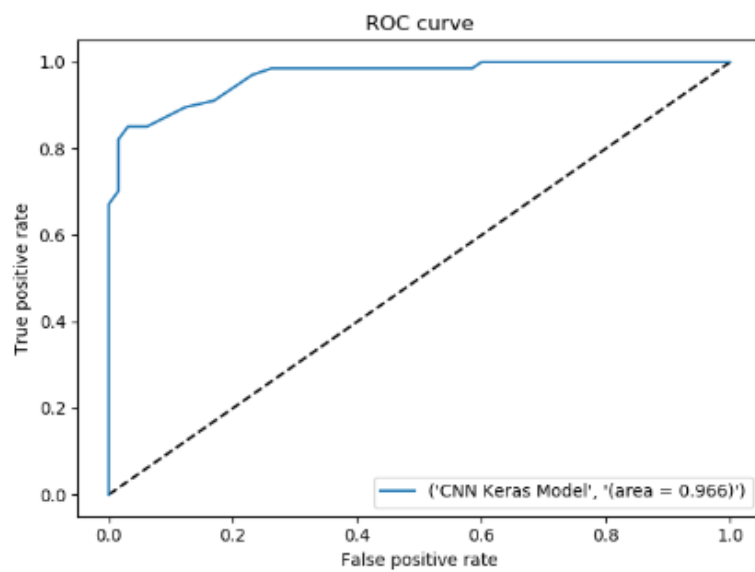
In our case, accepting a non-valid signature (false positive) is more critical than refusing a valid one (False negative). That’s why, while choosing the thresholds, I tried to minimize the false positive decisions. Table 3 shows the number of false positive and false negative decisions for each dataset.



	SigComp2009 (468)	Kaggle dataset (425)	Tunisian dataset (136)
Total fail	13	26	7
False positive	2	11	0
False negative	11	15	7

**Table 3.** False positive and false negative decisions

I also choose random lines where the system has made a decision (accepted or not) to plot the following Receiver Operator Characteristic (ROC) curve as shown in Figure 14. ROC curve is used to show the diagnostic ability of binary classifiers. It's true that our system isn't a binary classifier but, I used it as a way to compare this model with others. I also calculated the Area under curve (AUC) which is 0.966 which can be considered as a general measure of predictive accuracy.



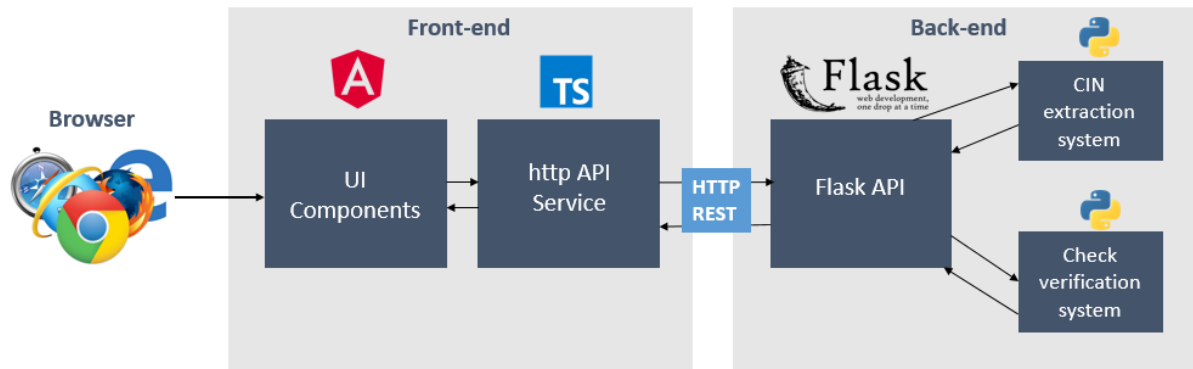
**Figure 14.** ROC curve of the developed model

### 6.2.5 Deployment

For this part, I organized the different scripts and prepared the prediction function so it can be used by the check verification system. The following section will describe furthermore how the developed signature verification system will be used.

## 6.3 Creating the demo Web application

Since this project is part of a very big automating system, other projects were involved. For demo purposes, I was asked to combine the developed check verification system with a CIN information extractor project that was developed by another team in a Web application. I was in charge of developing the web service that call the different scripts including my signature verification system and creating the front end. Figure 15 represent the technical architecture of this application.



**Figure 15.** Technical architecture for the developed web application

For the CIN information extractor, the user should submit a pdf File that contain the scanned CIN: a page for each side. Then, he will have, as shown in Figure 16, the extracted information with the possibility to download them in an Excel file, as well as the extracted CIN images.

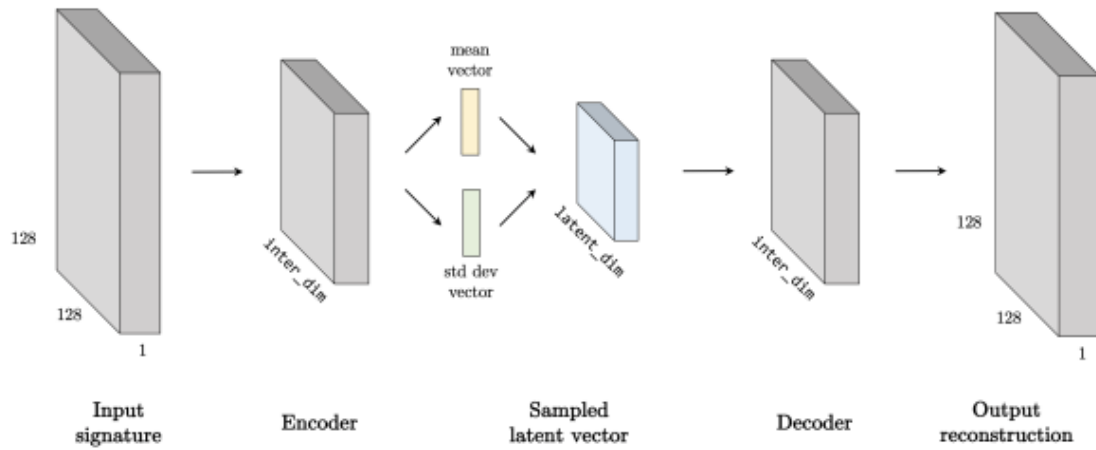
**Figure 16.** Example of the Front end for CIN information extraction

For the check verification system, the user should submit a check image of a stored client. The system execute the processes of Figure 2 and send, as show in Figure 17, the client information as well as the decision and the similarity percentage which is calculated as cosine of similarity x 100.

**Figure 17.** Example of the Front end for the developed check verification system

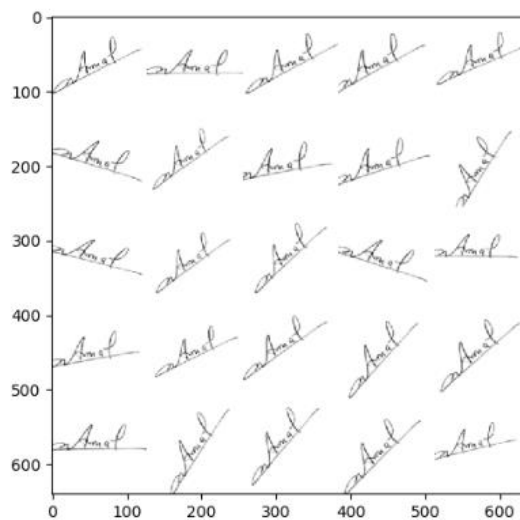
## 6.4 Creating the VAE model

In order to detect skilled forgeries, we need to study the main features of a person signature. To do so, we need a lot of samples of the same signature and a lot of skilled forgeries. My mission was here to explore another approach for signature verification using Variational Auto-Encoding (VAE) by experimenting the project referenced in [2]. VAE allows the creation of a latent distribution space to extract features of a signature. It consists of three main things which are: encoder, decoder and loss function. Figure 18 represent here the used VAE architecture.



**Figure 18.** System diagram of the VAE architecture used

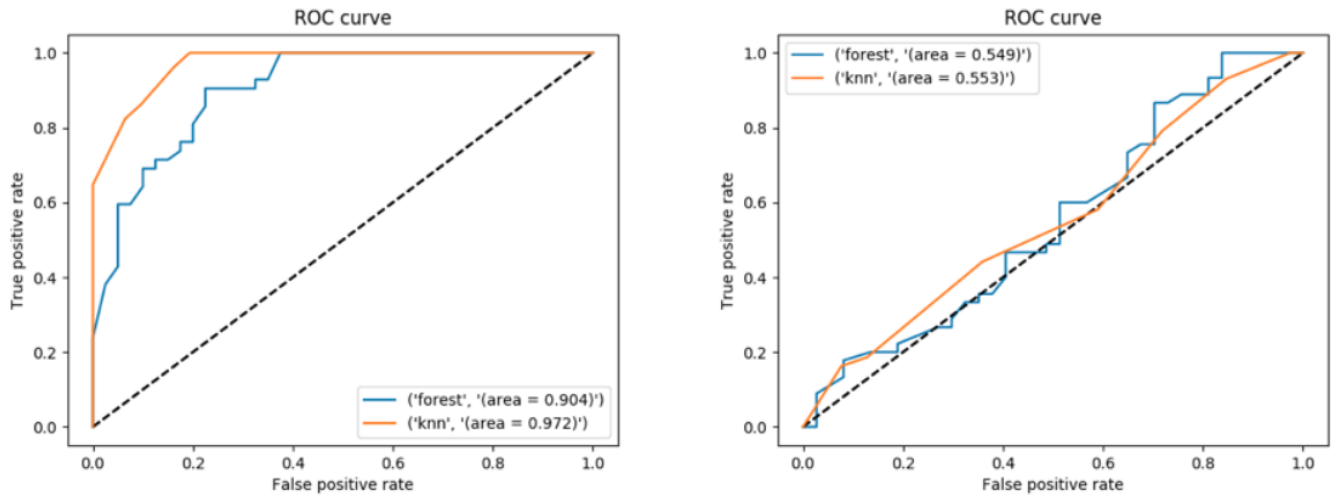
The used training dataset is the ICDAR competition dataset [3] which is a small dataset that won't be enough for training. To attempt to remedy the small data problem, I used data augmentation, so single signatures will be better generated by the Variational auto encoders (VAEs). Data augmentation was done by combining five different transformations: rotation angle (up to 20 degrees), channel shift (changes gray scale intensity), width shift, height shift and zoom range. Figure 19 present an example of generated signatures using the Keras ImageDataGenerator library.



**Figure 19.** Example of data augmentation on a single signature

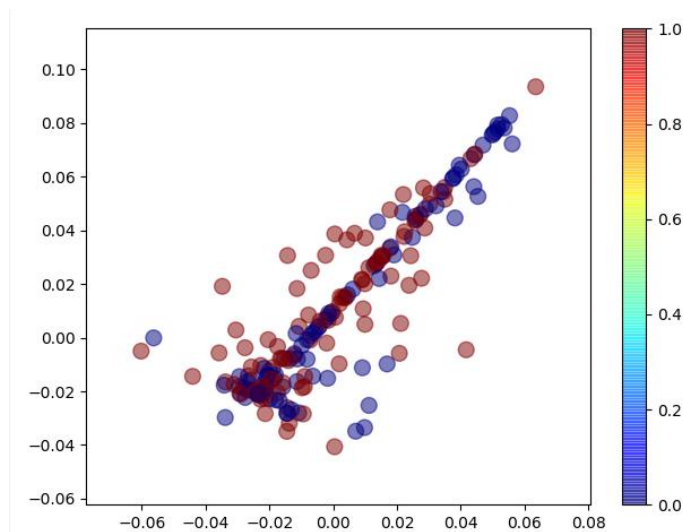
With this approach, each signature has her own model that I train by providing genuine signatures only, such that the VAEs fail to reconstruct forgeries but successfully reconstruct genuine signatures. Given 128x128 signature, the model will give as a result the latent representation of that signature.

Using the created VAE models, I can train KNN and Random Forest classifiers to detect forgeries from the latent space representations and from reconstruction loss. I trained only few signatures due the lack of time and materials, I tested the classification output and plotted the ROC curve as seen in Figure 20. Certain complex signatures are much harder to convincingly forge thus are easy to classify (left side). Similarly other signatures are much easier to forge and thus much more difficult to classify (right side).



**Figure 20.** ROC plots for easy (left) and hard signatures (right)

Figure 21 represent here a t-SNE representation of the latent vector of a random real signature and its skilled forgery. Indeed, this looks challenging for a classifier to distinguish between a real signature and its skilled forgery.



**Figure 21.** T-SNE representation of the latent vector of a real signature and its forgery

Overall, the performances of this approach isn't as good as it should be in our context. There are other approaches that seems to be more efficient. However, the small data problem persist in any approach. That's why it's important to work on collecting suitable data. Even for the CNN developed signature verification system, I believe that if we fit the model with Tunisian signatures, it will give better performances in the check verification procedure.

## 7 Acquired skills consolidation

The following table (Table 4) present the main subjects studied during my education at INSAT that helped in this internship.

Subject name	Acquired skills
Data Mining	Data Analyze, classification, getting familiar with python libraries.
Mathematics for Machine learning	Basic understanding of statistics for machine learning.
Software architecture	Building a web applications respecting best practices.

**Table 4.** Table of acquired skills consolidation

## 8 Conclusion

During my summer internship, I had the chance to be part of an innovative project that aim to automate check verification in Tunisia. My main mission was to create an offline signature verification system using convolutional neural networks. It was my first experience with deep learning, but having the opportunity to brainstorm with such talented people helped me identifying and solving problems.

In addition to the technical side, this project was an opportunity to understand the work in a professional hierarchy within a large company and to face the inherent difficulties such as planning and distributing time and effort.

On the whole, this internship was a useful experience. I have gained new knowledge, skills and met many new people. The most important things after two months as an intern in WEVIOO were the personal reflections I had on myself. It was a very open minded experience that made me definitely different and even more enthusiast about the word of data science.

## References

[1] V.L. Blankers, C.E. van den Heuvel, K. Franke, L. Vuurpijl (et al), "The ICDAR 2009 Signature Verification Competition", in Proceedings of the 2009 10th International Conference on Document Analysis and Recognition, pp. 1403-1407, 2009.

[2] PsiPhiTheta, « Catch-me-if-you-can » A ML project exploring alternative approaches to signature verification, Git repository: <https://github.com/PsiPhiTheta/catch-me-if-you-can>

[3] Marcus Liwicki, Muhammad Imran Malik, C Elisa Van Den Heuvel, Xiaohong Chen, Charles Berger, Reinoud Stoel, Michael Blumenstein, and Bryan Fount. Signature verification competition for online and offline skilled forgeries (sigcomp2011). In *2011 International Conference on Document Analysis and Recognition*, pages 1480–1484. IEEE, 2011.

## Bibliography

- 1- Dey, Sourish. "Signature Fraud Detection- An Advanced Analytics Approach." *Linkedin.Com*, 13 Mar. 2017, <https://www.linkedin.com/pulse/signature-fraud-detection-advanced-analytics-application-sourish-dey/>. Accessed 28 Aug. 2019.
- 2- Prabhu. "Understanding of Convolutional Neural Network (CNN) — Deep Learning." *Medium*, Medium, 4 Mar. 2018, [medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148](https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148).
- 3- "Keras Documentation." *Keras.Io*, <https://keras.io/>
- 4- "Siamese Network - Special Applications: Face Recognition & Neural Style Transfer | Coursera." *Coursera*, 2017, <https://www.coursera.org/lecture/convolutional-neural-networks/siamese-network-bjhmj>. Accessed 28 Aug. 2019.
- 5- Olivier Moindrot. "Triplet Loss and Online Triplet Mining in TensorFlow." *Olivier Moindrot Blog*, 19 Mar. 2018, <https://omoindrot.github.io/triplet-loss? Fbclid =IwAR3 GmA2yhg1x K BXtjDTRPFWkw9beYBOAAZNUSScBrrsc9mYUBohuq5JDdqU>.
- 6- jadevaibhav. "Jadevaibhav/Signature-Verification-Using-Deep-Learning." *GitHub*, 7 July 2018, [github.com/jadevaibhav/Signature-verification-using-deep-learning](https://github.com/jadevaibhav/Signature-verification-using-deep-learning). Accessed 28 Aug. 2019.
- 7- Vivek Vyas. "Variational Auto Encoders." *Medium*, Towards Data Science, 16 Aug. 2017, <https://towardsdatascience.com/variational-auto-encoders-fc701b9fc569>. Accessed 28 Aug. 2019.
- 8- "Angular." *Angular.Io*, 2019, [angular.io/start](https://angular.io/start). Accessed 29 Aug. 2019.