

Alexandre Berard, Julien Liottard, Marie Vialle

Client : Jean-Pierre Ceysson, CARI Electronic  
Tuteur : André Lagrèze, IUT de Valence

Projet de deuxième année de DUT Informatique

IUT de Valence

# Rapport de projet - IA Embarquée

mars 2020 (version 2)



# Sommaire

<b>1</b>	<b>Présentation du projet et de ses objectifs</b>	<b>7</b>
1.1	Contexte de présentation de la mission . . . . .	7
1.1.1	L'entreprise CARI Electronic . . . . .	7
1.1.1.1	Situation géographique . . . . .	7
1.1.1.2	Domaine d'activité . . . . .	7
1.1.1.3	Services proposés . . . . .	7
1.1.2	Présentation du projet . . . . .	7
1.1.2.1	Situation du projet dans le système existant . . . . .	7
1.1.2.2	Problématique de l'entreprise . . . . .	7
1.2	Recueil des besoins . . . . .	8
1.2.1	Les besoins de l'entreprise . . . . .	8
1.2.1.1	Objectif principal du projet . . . . .	8
1.2.1.2	Objectif(s) secondaire(s) . . . . .	8
1.2.2	Besoins techniques . . . . .	8
1.2.2.1	Besoin de rapidité . . . . .	8
1.2.2.2	Besoin de précision . . . . .	8
1.3	L'élaboration du cahier des charges . . . . .	8
1.3.1	Analyse conceptuelle . . . . .	8
1.3.1.1	Modélisation sous forme d'un diagramme de cas d'utilisation . . . . .	8
1.3.1.2	Prise en compte de l'aspect embarqué du système . . . . .	9
1.3.2	Propositions de l'entreprise . . . . .	9
1.3.2.1	Utiliser les outils et matériels conseillés . . . . .	9
1.3.2.2	Abstraire la récupération des données par la carte . . . . .	10
1.3.2.3	Maximiser la vitesse de calcul . . . . .	10
1.3.3	Modifications apportées au cours du projet . . . . .	10
1.3.3.1	Adaptation du développement dans l'optique de preuve de faisabilité . . . . .	10
1.3.3.2	Normalisation du format des données pour le réseau de neurones . . . . .	10
<b>2</b>	<b>Mise en oeuvre du projet</b>	<b>11</b>
2.1	Les différents axes et répartition du travail . . . . .	11
2.1.1	Les différentes phases de travail . . . . .	11
2.1.1.1	Conception . . . . .	11
2.1.1.2	Développement . . . . .	11
2.1.1.3	Gestion de projet . . . . .	11
2.1.2	L'utilisation de la méthode agile . . . . .	11
2.1.2.1	Présentation de la méthode scrum . . . . .	11
2.1.2.2	Définition des différents sprints . . . . .	12
2.1.2.3	Contenu des sprints . . . . .	12

2.1.3	La répartition du travail . . . . .	12
2.1.3.1	Attribution des rôles dans la phase de développement . . . . .	12
2.1.3.2	Place de la gestion de projet . . . . .	12
2.2	Le cadre technique du projet . . . . .	12
2.2.1	Cadre STM32 . . . . .	12
2.2.1.1	Répercussion de cet IDE sur le projet . . . . .	12
2.2.1.2	Intégration du module IA . . . . .	12
2.2.1.3	Paramétrage de l'application et de la carte . . . . .	12
2.2.2	Utilisation de TensorFlow . . . . .	12
2.2.2.1	Présentation de TensorFlow . . . . .	12
2.2.2.2	Installation . . . . .	12
2.2.2.3	Prise en main . . . . .	13
2.2.2.4	Utilisation dans le projet . . . . .	18
2.2.3	Tests réalisés . . . . .	18
2.2.3.1	Allumer une LED en fonction de l'état du réseau de neurones . .	18
2.2.3.2	Différencier plusieurs types de cartes électroniques par l'IA . . .	18
2.2.3.3	Tests du temps de réaction de l'IA sur carte . . . . .	18
2.3	Les défis et solutions aux problèmes rencontrés . . . . .	18
2.3.1	Les défis de la gestion de projet . . . . .	18
2.3.1.1	Mise en place d'un planning pour la conception . . . . .	18
2.3.1.2	Gestion des réunions avec le client et le tuteur . . . . .	18
2.3.2	Les défis techniques initiaux . . . . .	18
2.3.2.1	Aptitude de l'IA à analyser des courbes de tension . . . . .	18
2.3.2.2	Obtenir des quantités de banques d'apprentissage et de tests . .	18
2.3.3	Les difficultés rencontrés au cours du projet - Livre blanc ? . . . . .	19
2.3.3.1	Le paramétrage du réseau de neurones . . . . .	19
2.3.3.2	La compatibilité de TensorFlow et le module IA de STM32CubeMx	19
2.3.3.3	Les difficultés liées à l'utilisation de STM32CubeMx . . . . .	20
2.3.3.4	La prise en compte du réglage de l'oscilloscope . . . . .	20
<b>3</b>	<b>Bilan et perspectives</b>	<b>20</b>
3.1	L'état final du projet . . . . .	20
3.1.1	Description des fonctionnalités du produit final . . . . .	20
3.1.1.1	Facultés de l'IA pour la différenciation . . . . .	20
3.1.1.2	Formatage des données sur carte . . . . .	20
3.1.1.3	Résultats de l'analyse des données de la carte . . . . .	20
3.1.2	Description de tâches complexes . . . . .	20
3.1.2.1	Apprentissage d'anomalies . . . . .	20
3.1.2.2	Réduction du temps de calcul . . . . .	20
3.1.3	Description des tâches à réaliser . . . . .	20
3.1.3.1	Récupération des données en temps réel . . . . .	20

3.1.3.2	Précision du réseau de neurones à améliorer . . . . .	21
3.2	Les apports du projet . . . . .	21
3.2.1	Apports humains et professionnels . . . . .	21
3.2.1.1	Apports professionnels . . . . .	21
3.2.1.2	Apports humains . . . . .	21
3.2.2	Apports techniques . . . . .	21
3.2.2.1	Nouvelle méthode (Agile) . . . . .	21
3.2.2.2	Se familiariser avec STM32CubeMx et la librairie TensorFlow . .	21
3.2.2.3	Notions d'électronique et de programmation embarquée . . . . .	21
3.3	Perspectives de l'évolution du projet . . . . .	21
3.3.1	Amélioration de l'IA . . . . .	21
3.3.1.1	Détection d'anomalies et de leur type . . . . .	21
3.3.1.2	Réduction du temps de calcul . . . . .	21
3.3.1.3	Aspect "temps réel" . . . . .	21
3.3.2	Portabilité de l'application . . . . .	21
3.3.2.1	Rendre possible l'import de banques personnalisées . . . . .	21
3.3.2.2	Rendre possible l'import direct sur carte . . . . .	21
<b>4</b>	<b>Annexes (début sur chaque page impaire)</b>	<b>22</b>
4.1	annexes . . . . .	22
4.2	table des figures . . . . .	23
4.3	Accronymes . . . . .	23
4.4	définitions . . . . .	23
4.5	sources et bibliographies . . . . .	23

## Introduction

Afin de répondre à la demande de tester l'implémentation d'un réseau de neurones artificiels sur une carte et de faire un compte rendu de sa rapidité, l'entreprise CARI Electronic, a proposé un projet aux étudiants du département informatique de l'IUT<sup>1</sup> de Valence. Nous étions intéressés par cette opportunité de découvrir l'intelligence artificielle, un domaine que nous ne côtoyons pas durant notre formation.

Ce projet est proposé par M. Jean-Pierre Ceysson à des étudiants de seconde année du département informatique de l'IUT de Valence et le tuteur est M. André Lagrèze. La mission consiste à développer un inspecteur d'erreurs lors de la mise sous tension d'un appareil. Plus spécifiquement, le projet consiste à créer un réseau de neurones artificiels<sup>2</sup> puis à l'implémenter sur une carte STM32<sup>3</sup>, afin de pouvoir rendre compte de l'état de l'appareil lors du démarrage via une courbe de points. L'utilisateur sera informé de l'état grâce à des LED<sup>4</sup> présentes sur la carte. Ce projet doit répondre à différentes contraintes qui ont guidé sa conception, telles que la contrainte de temps réel ou la contrainte de précision.

Afin de répondre au mieux aux besoins dont nécessitait ce projet, nous avons dégagé la problématique suivante : comment implémenter un réseau de neurones sur un système embarqué pour qu'il détecte des anomalies à travers la mise en tension d'un appareil tout en respectant une contrainte de rapidité et de précision ?

Concernant les rôles des membres de l'équipe de projet, Marie Vialle était la responsable de la documentation pour normaliser les documents rendus, ainsi que la chef de projet, en charge de superviser et faire le lien entre le tuteur, le client et l'équipe. Alexandre Berard était le responsable planning et devait s'assurer que celui-ci soit respecté. Enfin, Julien Liottard était en charge du suivi des coûts, il devait donc faire en sorte que le projet soit rentable.

Dans un premier temps, nous aborderons une présentation complète du projet et de ses objectifs, puis nous verrons comment nous avons mis en oeuvre sa conception. Pour finir, nous ferons un bilan et exposerons les perspectives possibles.

---

1. Tous les mots marqué d'un astérisque sont définis en pied de page.

Institut Universitaire Technologique

2. Système informatique s'inspirant du fonctionnement du cerveau humain pour apprendre.

3. La famille STM32 est une série de microcontrôleurs 32-bits en circuits intégrés réalisés par STMicroelectronics.

4. Light-Emitting Diode

## Remerciements

C'est à M. André Lagrèze, notre tuteur de projet, que nous adressons nos plus grands remerciements pour le temps qu'il nous a consacré, et les suggestions dont il nous a fait part lorsque nous rencontrions une difficulté.

Nous tenons également à témoigner notre gratitude à Jean-Pierre Ceysson, qui a accepté de mettre à notre disposition une carte STM32 de test pour notre développement.

Nous n'oublions pas non plus les services que nous a rendu M. Sébastien Jean, du corps enseignant de l'IUT de Valence, pour l'échantillonnage des données ainsi que le livre<sup>5</sup> qui nous aura servi à nous renseigner sur les réseaux de neurones artificiels.

De plus, nous souhaitons remercier M. Guillaume De Souza, notre professeur de communication qui nous aura guidé pour la rédaction des documents techniques tout au long du projet.

Un grand merci enfin à XXX et ZZZ, qui ont gentiment accepté de relire ce rapport.

---

5. TensorFlow pour le deep learning, de la régression linéaire à l'apprentissage par renforcement - Barath Ramsundar et Reza Bosagh Zadeh

# 1 Présentation du projet et de ses objectifs

## 1.1 Contexte de présentation de la mission

### 1.1.1 L'entreprise CARI Electronic

**1.1.1.1 Situation géographique** L'entreprise CARI Electronic se situe entre Romans-sur-Isère et Valence, au sein du pôle d'activité Rovaltain.

#### 1.1.1.2 Domaine d'activité

La firme opère dans de nombreux secteurs d'activités. En effet, en aéronautique elle a déjà réalisé des cartes pour une manette d'avion. En Internet des objets, elle a fait part à un projet avec des systèmes de surveillances de consommation de l'énergie de logements européens. Dans le domaine de l'énergie et dans le développement durable, l'entreprise a déjà produit des cartes de détection de radioactivité et de contrôle vidéo. Dans le médical, CARI Electronic a fabriqué des cartes contrôle-commande pour aider les systèmes paramédicaux. Dans l'industrie, elle a supporté un industriel pour un dispositif d'éclairage. Enfin dans l'audition, l'entreprise a développé des systèmes audio complets.

#### 1.1.1.3 Services proposés

La société est spécialisée dans la fabrication de matériel électronique mélangeant PCB<sup>6</sup> et matériaux complémentaires pour former le hardware. Plus particulièrement, elle peut s'occuper de l'étude et la conception des maquettes, les tests des prototypes, la fabrication et le câblage des cartes jusqu'aux tests fonctionnels.

## 1.1.2 Présentation du projet

### 1.1.2.1 Situation du projet dans le système existant

L'entreprise CARI Electronic souhaite s'étendre dans le domaine de l'intelligence artificielle, il n'existe donc pas d'ancien projet sur lequel nous pourrions nous appuyer. En effet, l'entreprise aimerait constater de la faisabilité de l'utilisation d'une IA dans un circuit imprimé en guise de premiers pas vers le deep learning<sup>7</sup>. Notre rôle serait alors de faire des tests sur l'efficacité d'un réseau de neurones artificiel d'une certaine taille implémenté sur une carte afin de rendre compte du temps par rapport à la contrainte de temps réel, et surtout de sa faisabilité.

#### 1.1.2.2 Problématique de l'entreprise

Comme dit précédemment, l'entreprise souhaiterait exploiter nos données et nos recherches pour savoir si les projets qu'elle imagine en IA seraient réalisables. La contrainte de rapidité et de précision est alors extrêmement importante. En effet, le temps d'analyse idéal serait de 1 à 2

---

6. Printed Circuit Board ou carte à circuit imprimé

7. Apprentissage profond en français, est une méthode d'apprentissage automatisée qui met notamment en jeu des couches abstraites qui communiquent entre-elles sur un schéma entrées-sorties



millisecondes.

Une autre problématique serait l'utilisation de TensorFlow et son implémentation sur une carte STM32. Là encore, ne sachant pas si cela était faisable, nous avons à notre charge les recherches et les tests. Le client nous a alors orienté, en nous fournissant une carte STM32 Nucleo 144, vers l'environnement STM32CubeMX, et son extension X-Cube-IA.

## **1.2 Recueil des besoins**

### **1.2.1 Les besoins de l'entreprise**

#### **1.2.1.1 Objectif principal du projet**

L'objectif principal de ce projet est la création d'un réseau de neurones artificiels, son implémentation sur une carte qui doit, elle, être fiable à un objet mis sous tension. Cette même intelligence artificielle devra être capable d'analyser des courbes de mise sous tension, afin de déterminer en temps réel, s'il y a une défaillance, grâce à l'allumage de LED (par exemple).

#### **1.2.1.2 Objectif(s) secondaire(s)**

Un des objectifs secondaires du projet serait la précision de l'intelligence artificielle. En effet, le temps réel prime sur la précision, mais cela reste un objectif important pour reconnaître les différentes courbes qui lui sont présentées.

### **1.2.2 Besoins techniques**

#### **1.2.2.1 Besoin de rapidité**

Tout l'intérêt du projet repose dans le fait que le réseau de neurones doit être rapide. En effet, il doit pouvoir reconnaître des erreurs dans les éléments qu'il analyse le plus rapidement possible, de préférence de l'ordre de quelques millisecondes. En effet, même 3 secondes seraient considérées comme longues dans notre cas.

#### **1.2.2.2 Besoin de précision**

Le second principal enjeu de ce projet est la contrainte de précision. En effet, le réseau de neurones artificiel doit être précis afin de détecter les plus petites erreurs, ainsi que pour les différencier entre elles. Dans l'idéal chaque erreur doit être signalée d'une manière différente.

## **1.3 L'élaboration du cahier des charges**

### **1.3.1 Analyse conceptuelle**

#### **1.3.1.1 Modélisation sous forme d'un diagramme de cas d'utilisation**

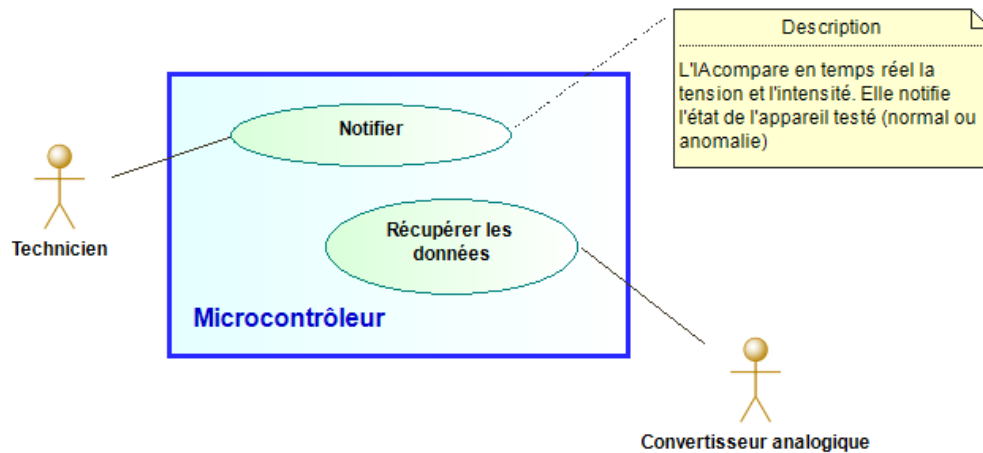


Figure 1 : Diagramme de cas d'utilisation

La carte reçoit un résultat d'analyse de l'intelligence artificielle, et la carte doit notifier l'utilisateur de l'état de ce même résultat, en utilisant par exemple des LED.

### 1.3.1.2 Prise en compte de l'aspect embarqué du système

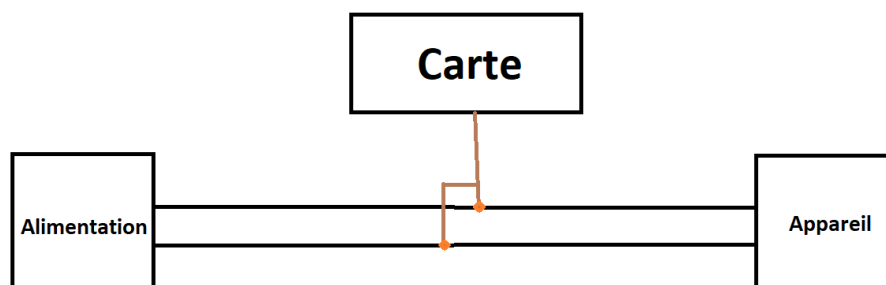


Figure 2 : Cadre technique

En ce qui concerne le cadre technique de notre produit final, c'est une carte branchée entre l'alimentation d'un appareil et l'appareil en question.

L'aspect embarqué concerne donc le réseau de neurones qui est implémenté sur la carte. Tout le projet repose dessus.

## 1.3.2 Propositions de l'entreprise

### 1.3.2.1 Utiliser les outils et matériels conseillés

Afin de mener à bien le projet, notre client nous a proposé d'utiliser l'outil open source d'apprentissage automatique développé par Google, TensorFlow<sup>8</sup>. De plus, nous avons récupéré une carte STM32F767ZI sur laquelle le client peut travailler. Ce point est important car nous avons donc utilisé l'environnement que fournit ST, soit STM32CubeMX<sup>9</sup> qui nous a permis de

8. <https://www.tensorflow.org/>

9. <https://www.st.com/en/development-tools/stm32cubemx.html>

générer du code en fonction des paramètres que nous avons choisi dans le logiciel, nous avons aussi utilisé l'IDE STM32CubeIDE<sup>10</sup>. Pour finir, nous avons intégré le module X-Cube-MX<sup>11</sup>.

### **1.3.2.2 Abstraire la récupération des données par la carte**

???

### **1.3.2.3 Maximiser la vitesse de calcul**

???

## **1.3.3 Modifications apportées au cours du projet**

### **1.3.3.1 Adaptation du développement dans l'optique de preuve de faisabilité**

???

#### **1.3.3.2 Normalisation du format des données pour le réseau de neurones**

Après concertation avec le client, nous avons revu le formatage des données à apprendre avec le réseau de neurones. Premièrement, le client a proposé qu'au lieu de faire un rapport entre le minimum et le maximum de la courbe (pour ramener les données dans un intervalle  $[0, 1]$ ), il serait plus sage de fixer un maximum commun à chaque courbes à analyser, ce qui fera moins de calculs et qui améliore aussi la performance du réseau de neurones. Nous avons choisi de maximiser la courbe à 20 Volts (20,48 pour être précis), ce qui laisse un panel assez large de cartes à faire apprendre. Il faudra aussi penser à revoir la précision des points donnés au réseau de neurones. Le fait est que, pour la réception des données, il faudra installer un CT sensor (definition : Outil permettant la mesure des courants alternatifs, utilisés principalement dans les alimentations) sur la carte Nucleo. La précision de cette acquisition est de l'ordre de 1024 points, et étant donné que les données acquises sur l'oscilloscope sont plus précises, il nous faut revoir le format donné. Au final, le calcul est le suivant, pour chaque données :

$$E(x*(p/m))/p$$

(à revoir). où E est une fonction qui arrondit les valeurs, p étant la précision, m le maximum global, et x chaque point de la courbe. Sur carte, le seul calcul à faire sera de diviser chaque valeurs par 1024, de sorte à obtenir un nombre flottant dans l'intervalle  $[0,1]$  nécessaire pour le passage dans le réseau de neurones, ces données seront sûrement compatible avec ce qui aura été appris.

---

10. <https://www.st.com/en/development-tools/stm32cubeide.html>

11. <https://www.st.com/en/embedded-software/x-cube-ai.html>

## **2 Mise en oeuvre du projet**

### **2.1 Les différents axes et répartition du travail**

#### **2.1.1 Les différentes phases de travail**

##### **2.1.1.1 Conception**

Lors des quatre premiers mois du projet, nous nous sommes penchés sur la conception du projet. Durant cette phase, nous avons fait une analyse approfondie du projet via des diagrammes UML<sup>12</sup>, en étudiant les divers objectifs et charges afin d'aboutir à un cahier des charges détaillés.

##### **2.1.1.2 Développement**

La seconde phase, de trois mois, a été dédiée au développement du projet. Durant ce temps, nous avons réalisé notre réseau de neurone et nous l'avons implémenté sur la carte, en fonction du cahier des charges que nous avons réalisé pendant la phase précédente.

##### **2.1.1.3 Gestion de projet**

En parallèle des deux phases précédemment citées, nous avons géré notre planning grâce à un diagramme de Gantt<sup>13</sup> et nous suivons des coûts concernant le projet en fonction de nos heures de travail. Cette gestion de projet nous a permis d'estimer le temps de réalisation de chaque tâche ainsi que d'anticiper les livrables que nous devons rendre.

Plus spécifiquement, le suivi des coûts nous a permis de suivre le potentiel coût de revient du projet et son évolution au cours des semaines. Il nous a facilité l'estimation d'un prix de vente virtuel à proposer à un client et ainsi, le possible bénéfice qu'aurait engendré le projet dans un contexte concret. On peut commenter que l'évolution du coût lors de la phase de conception et de développement est restée stable, même s'il a finalement diminué par rapport au prix initial. Pour ce qui concerne les chiffres, le coût de la conception a été évalué à 13 986€ pour un bénéfice final de 9 700€ et le coût du développement a été chiffré à 14 430€ pour un bénéfice estimé de 9716,8€.

#### **2.1.2 L'utilisation de la méthode agile**

##### **2.1.2.1 Présentation de la méthode scrum**

Cette méthode s'appuie sur le découpage d'un projet en différentes phases nommées "sprints". Ces sprints peuvent durer entre quelques heures et un mois, et est lui-même découpé en plusieurs stories. Ces stories sont, découpées en tâches, afin de bien guider le projet.

À la fin de chaque sprint, un rapport est produit afin de rendre compte de l'avancée et des points à améliorer. Enfin, chaque nouveau sprint commence par une rétrospective, qui permet d'analyser le déroulement du sprint achevé et des pratiques à améliorer.

---

12. Unified Modeling Language

13. Outil utilisé en ordonnancement et en gestion de projet et permettant de visualiser dans le temps les diverses tâches composant un projet

### **2.1.2.2 Définition des différents sprints**

Le développement a été découpé en plusieurs sprints d'une durée moyenne de 3 semaines <sup>14</sup>, à la fin desquels nous produisons un rapport détaillé au client et au tuteur, que nous leur envoyons par mail. Cela nous a permis d'avoir de nombreux retours concernant les points à améliorer.

### **2.1.2.3 Contenu des sprints**

## **2.1.3 La répartition du travail**

### **2.1.3.1 Attribution des rôles dans la phase de développement**

En conséquence de la dualité matériel d'informatique embarquée et développement d'une IA, les tâches ont pu être séparées aux membres du groupe pour une parallélisation du travail.

Ainsi après la phase de conception du projet, c'est Marie Vialle qui a pris en main le logiciel STM32Cube et a intégré le logiciel dans la carte nucleo fournie. L'implémentation de l'intelligence artificielle et le formatage des données ont été réalisés par Alexandre Bérard. Pour entraîner l'IA, Julien Liottard a réalisé un échantillonnage d'extraits de montée de tension d'appareils.

### **2.1.3.2 Place de la gestion de projet**

## **2.2 Le cadre technique du projet**

### **2.2.1 Cadre STM32**

#### **2.2.1.1 Répercussion de cet IDE sur le projet**

#### **2.2.1.2 Intégration du module IA**

#### **2.2.1.3 Paramétrage de l'application et de la carte**

### **2.2.2 Utilisation de TensorFlow**

#### **2.2.2.1 Présentation de TensorFlow**

#### **2.2.2.2 Installation**

Afin d'installer TensorFlow, nous avons suivis des instructions que nous avons trouvées sur le site de tensorflow <sup>15</sup>, que nous allons redétailler ici-bas. Pour plus de simplicité, nous avons utilisé le terminal Ubuntu (à définir) sur Windows.

Tout d'abord, nous avons vérifié que Python et son environnement étaient bien configurés via les commandes :

---

14. cf annexe n° ?

15. <https://www.tensorflow.org/install/pip?lang=python3>

```
python3 --version
pip3 --version
virtualenv --version
```

Si les packages ne sont pas installés, nous utilisons :

```
sudo apt update
sudo apt install python3-dev python3-pip
sudo pip3 install -U virtualenv
```

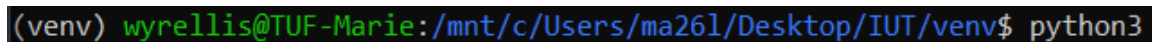
Nous avons ensuite créé un environnement virtuel :

```
virtualenv --system-site-packages -p python3 ./venv
```

Afin d'activer l'environnement, nous utilisons la commande :

```
source ./venv/bin/activate
```

Une fois que virtualenv est actif, le prompt du shell possède le préfixe (venv)



```
(venv) wyrellis@TUF-Marie:/mnt/c/Users/ma261/Desktop/IUT/venv$ python3
```

Figure ? : Visuel du préfixe *venv* sur le prompt du shell

On continue en améliorant pip pour qu'il ait toutes les fonctionnalités à jour :

```
pip install --upgrade pip
pip list
```

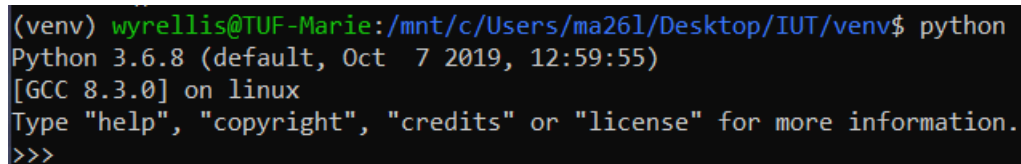
Pour finir, afin de quitter l'environnement virtuel, nous utilisons :

```
deactivate
```

Puis pour utiliser TensorFlow, nous pouvons lancer Python via

```
python3
```

qui nous permet d'ouvrir le shell (à définir) Python.



```
(venv) wyrellis@TUF-Marie:/mnt/c/Users/ma261/Desktop/IUT/venv$ python
Python 3.6.8 (default, Oct 7 2019, 12:59:55)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 2 : Visuel du shell python

Pour activer Tensorflow (dans le shell de Python) :

```
import tensorflow
```

### 2.2.2.3 Prise en main

Tout d'abord, nous avons utilisé la documentation tensorflow<sup>16</sup> pour nous permettre de

---

16. <https://www.tensorflow.org/tutorials/quickstart/beginner>

comprendre comment est construit un réseau de neurones simple, qui peut reconnaître des images, apprendre, et donner le taux de précision associé à ses estimations. D'après les codes que nous avons pu voir, nous en avons déduits certaines informations :

```
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test, verbose=2)
```

Ici, nous en avons déduit que "model" correspond à notre réseau de neurones, sur lequel nous pouvons appliquer deux méthodes : `.fit` et `.evaluate`. `.fit` permet de faire apprendre le réseau de neurones avec un dataset<sup>17</sup> (ici mnist via le lien de la partie ) contenant les éléments à analyser (images ou vecteurs) et leur label correspondant.

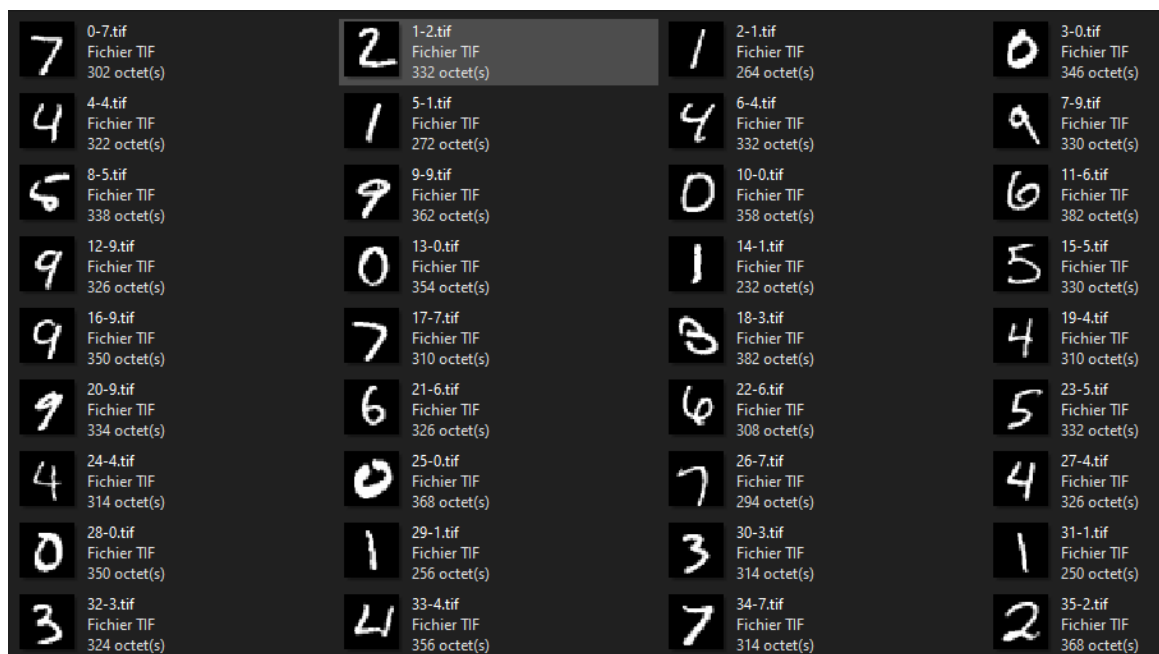


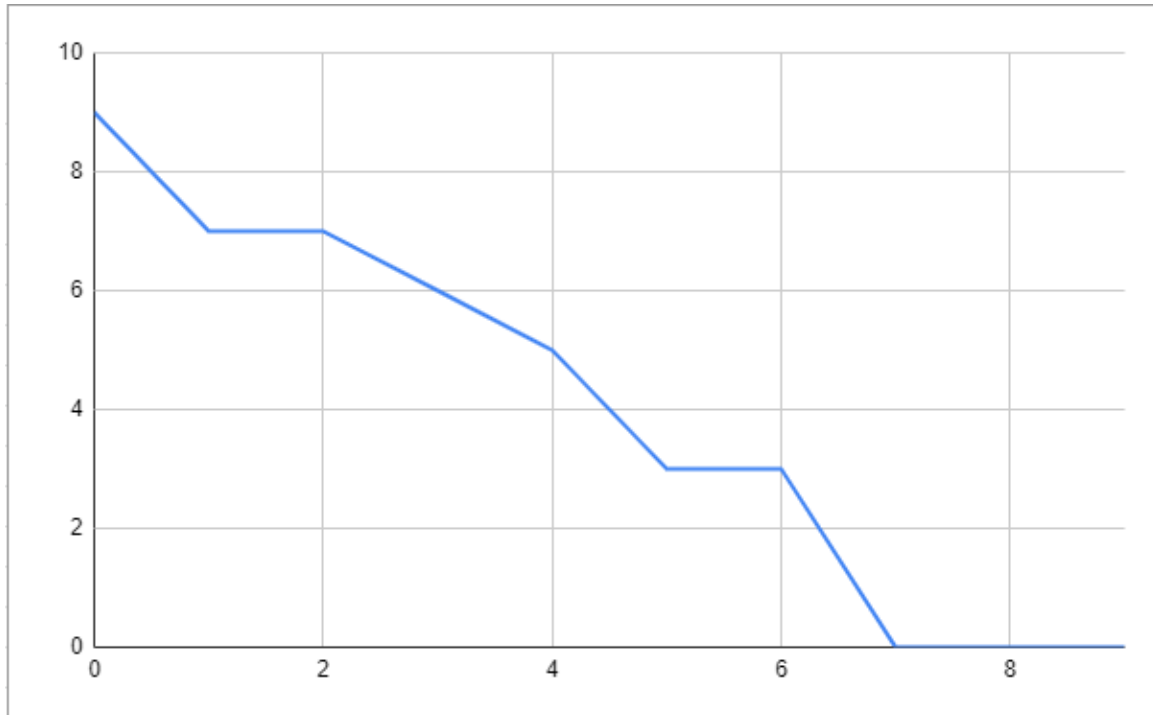
Figure ? : Visuel des images et de leurs labels

`.evaluate` permet lui de tester le réseau de neurones. Il utilise un dataset comme le `.fit` mais pour donner une estimation de la précision du réseau de neurones qui a précédemment été calculé (grâce à l'apprentissage). Il est donc important de mettre des valeurs différentes pour le dataset de test et le dataset d'apprentissage.

Après avoir globalement compris ce que faisait TensorFlow, il faut comprendre comment utiliser la librairie dans sa globalité afin de créer un code pouvant faire apprendre des courbes à notre réseau de neurones. Tout d'abord, nous avons créé un script Python qui permet de générer des courbes simples de 10 points.

---

17. ensemble de données regroupées en masse, ici on utilise on regroupe 60000 images avec leurs labels



*Figure ? : Visuel d'une courbe*

Nous en avons réalisé deux sortes : les courbes croissantes et décroissantes. Pour entraîner le réseau, nous avons fait quelques variations ce qui rend chaque courbe différente.

Nous avons décidé d'enregistrer ces graphiques dans des fichiers CSV (à définir), pour nous habituer à le faire lorsqu'il s'agira de récupérer des données sur l'oscilloscope, où ici nous n'aurons d'autre choix que cette extension. Pour ce faire, nous utilisons les fonctionnalités de Python qui permettent d'écrire et lire dans ces fichiers. La création de fichiers CSV se fait dans le fichier `tenpointsvectorgenerator.py`<sup>18</sup>, et la lecture de ceux-ci s'effectue dans `tenpointsvectorreader.py`<sup>19</sup>.

Nous devons associer à chaque courbe un label le décrivant, ici, dans cet exemple, nous distinguons une courbe croissance (ASC) d'une courbe décroissante (DESC), ce label sera inclus dans le fichier CSV. Avec ceci, nous pouvons utiliser un réseau de neurones permettant de distinguer une courbe qui croît d'une qui décroît.

Grâce aux ressources et exemples en ligne<sup>20</sup>, nous avons pu à minima comprendre quel est le format de données que peut prendre les réseaux de neurones de TensorFlow. TensorFlow peut apprendre les structures de données de NumPy<sup>21</sup>, nous devons donc convertir nos graphiques et labels en structure NumPy compréhensible par TensorFlow, ce dernier pouvant prendre en entrée une liste de nombre flottants (à définir) entre 0 et 1 et un label pour l'apprentissage du

18. <https://github.com/wyrellis/IA-Embarquee/blob/master/pyfiles/vectorLearning/tenpointsvectorgenerator.py>

19. <https://github.com/wyrellis/IA-Embarquee/blob/master/pyfiles/vectorLearning/tenpointsvectorreader.py>

20. [https://www.tensorflow.org/tutorials/load\\_data/numpy](https://www.tensorflow.org/tutorials/load_data/numpy)

21. Librairie de structures mathématiques



réseau de neurones. Pour ce faire, les valeurs du graphique doivent être formatées (ce sont des nombres entiers sur un axe de nombre entiers de 0 à 10), on procède comme suit :

- On soustrait toutes les valeurs du graphique par son minimum, cela aura pour effet de mettre la valeur minimale du graphique à 0.
- On divise toutes ces valeurs par le nouveau maximum, ce qui positionnera toutes les valeurs du graphique dans l'intervalle  $[0,1]$ .

Cette procédure est effectuée dans la fonction `format_vector` du fichier `tenpointsvectorIA.py`<sup>22</sup>. Le label doit également être formaté, puisqu'on ne peut pas donner de variable de type chaîne de caractères à notre réseau, mais que des types entiers. On choisit de mettre les labels dans un tableau<sup>23</sup> et de les associer à l'index correspondant dans le tableau. Ce formatage est fait dans la fonction `format_label` du même fichier (avec le label passé en paramètre). Une fois nos données formatées et mises dans des tableaux NumPy, nous pouvons directement passer à la gestion de notre réseau de neurones.

La configuration suivante est tirée de l'exemple donné sur les ressources en ligne sur TensorFlow<sup>24</sup>, cette configuration semble bien fonctionner avec les données tirées de structures de données simple NumPy. On modifie la configuration pour qu'elle puisse s'appliquer à notre exemple, on pose une première couche de 16 neurones (qui permettent d'effectuer la prise de décision), puis une seconde de 2 neurones, qui correspondra à la sortie. La compilation du modèle est également nécessaire à son fonctionnement.

Une fois configuré, le réseau est prêt à l'apprentissage. Le principe d'apprentissage repose sur le fait d'apprendre au réseau de neurones à associer un type de courbe à un nom de type de courbe (aussi appelé label). Pour cela, on génère un nombre assez conséquent de graphiques (ici 1000) qui lui seront donnés. Le réseau de neurones va modifier chacun de ses neurones en fonction du graphique et du label associé, et ce, autant de fois qu'il y a de graphiques à apprendre. Le réseau prendra en compte une liste NumPy de graphiques et une liste NumPy de labels (formatés comme décrit précédemment, et les deux listes doivent avoir une taille équivalente, puisque TensorFlow associe les deux éléments par leur index dans les listes).

On peut utiliser la fonction TensorFlow `.fit()` sur le modèle pour cela. Cette dernière prend aussi en paramètre le nombre d'itérations de l'apprentissage, cela à pour effet de 'rabâcher' les données et d'améliorer les résultats du réseau. TensorFlow appelle ces itérations les 'Epochs'.

---

22. <https://github.com/wyrellis/IA-Embarquee/blob/master/pyfiles/vectorLearning/tenpointsvectorIA.py>

23. `labels = ['ASC','DESC']`

24. [https://www.tensorflow.org/tutorials/load\\_data/numpy](https://www.tensorflow.org/tutorials/load_data/numpy)

```

Train on 2000 samples
Epoch 1/13
2000/2000 [=====] - 1s 628us/sample - loss: 0.5128 - sparse_categorical_accuracy: 0.6875
Epoch 2/13
2000/2000 [=====] - 0s 74us/sample - loss: 0.3309 - sparse_categorical_accuracy: 1.0000
Epoch 3/13
2000/2000 [=====] - 0s 66us/sample - loss: 0.1639 - sparse_categorical_accuracy: 1.0000
Epoch 4/13
2000/2000 [=====] - 0s 71us/sample - loss: 0.0596 - sparse_categorical_accuracy: 1.0000
Epoch 5/13
2000/2000 [=====] - 0s 72us/sample - loss: 0.0188 - sparse_categorical_accuracy: 1.0000
Epoch 6/13
2000/2000 [=====] - 0s 70us/sample - loss: 0.0051 - sparse_categorical_accuracy: 1.0000
Epoch 7/13
2000/2000 [=====] - 0s 74us/sample - loss: 0.0013 - sparse_categorical_accuracy: 1.0000
Epoch 8/13
2000/2000 [=====] - 0s 71us/sample - loss: 2.7600e-04 - sparse_categorical_accuracy: 1.0000
Epoch 9/13
2000/2000 [=====] - 0s 71us/sample - loss: 6.3600e-05 - sparse_categorical_accuracy: 1.0000
Epoch 10/13
2000/2000 [=====] - 0s 72us/sample - loss: 1.7754e-05 - sparse_categorical_accuracy: 1.0000
Epoch 11/13
2000/2000 [=====] - 0s 72us/sample - loss: 5.1236e-06 - sparse_categorical_accuracy: 1.0000
Epoch 12/13
2000/2000 [=====] - 0s 72us/sample - loss: 1.5814e-06 - sparse_categorical_accuracy: 1.0000
Epoch 13/13
2000/2000 [=====] - 0s 74us/sample - loss: 5.6529e-07 - sparse_categorical_accuracy: 1.0000
500/1 - 0s - loss: 3.1614e-07 - sparse_categorical_accuracy: 1.0000
Test with one vector of the set:
[[1. 0.75 0.75 0.625 0.5 0.25 0.25 0. 0. 0. ]]
[[5.8892527e-07 9.900096e-01]]
DESC

```

Figure ? : Visuel de la phase d'apprentissage répartie sur plusieurs 'Epochs'

En dernier lieu, on procède à un test, on fournit au réseau de neurones d'autres graphiques et labels, mais c'est à son tour de nous dire à quel label ce graphique appartient. Un pourcentage de réussite sera donné à la fin. (image montrant le pourcentage de réussite). On a aussi décidé de faire quelques tests individuels, où l'on peut voir l'estimation que le réseau porte sur les labels ASC et DESC.

```

Vector n°1:
Graphique (sous forme de liste)
[[0. 0.11111111 0.22222222 0.33333333 0.44444444 0.55555556
 0.55555556 0.77777778 1. 1. ]]
Résultat du réseau [ASC,DESC]
[[9.8186111e-01 4.4962962e-06]]
Label choisi (par défaut, celui qui le meilleur résultat)
ASC
Vector n°2:
Graphique (sous forme de liste)
[[0. 0.11111111 0.22222222 0.33333333 0.44444444 0.66666667
 0.77777778 0.77777778 0.88888889 1. ]]
Résultat du réseau [ASC,DESC]
[[9.8582226e-01 3.3522290e-06]]
Label choisi (par défaut, celui qui le meilleur résultat)
ASC
Vector n°3:
Graphique (sous forme de liste)
[[1. 0.88888889 0.77777778 0.77777778 0.55555556 0.44444444
 0.33333333 0.22222222 0.11111111 0. ]]
Résultat du réseau [ASC,DESC]
[[2.1314747e-07 9.4558895e-01]]
Label choisi (par défaut, celui qui le meilleur résultat)
DESC

```

Figure ? : Test du réseau de neurones

Le réseau de neurones est ici fonctionnel, nous avons dû trouver le nombre optimal de graphiques

à apprendre et d'itérations d'apprentissage pour faire fonctionner au mieux notre réseau de neurones.

#### **2.2.2.4 Utilisation dans le projet**

### **2.2.3 Tests réalisés**

#### **2.2.3.1 Allumer une LED en fonction de l'état du réseau de neurones**

#### **2.2.3.2 Différencier plusieurs types de cartes électroniques par l'IA**

#### **2.2.3.3 Tests du temps de réaction de l'IA sur carte**

## **2.3 Les défis et solutions aux problèmes rencontrés**

### **2.3.1 Les défis de la gestion de projet**

**2.3.1.1 Mise en place d'un planning pour la conception** Avant d'attaquer le développement du projet, nous avons préalablement établi un planning qui nous a permis d'anticiper les diverses tâches telles que la rédaction de livrables ou le déroulement des sprints. Comme mentionné dans la partie 2.1.1.3 Gestion de projet (lien ?), nous avons utilisé le modèle Gantt <sup>25</sup>. Le projet se déroulant tout au long du semestre 4 du DUT, nous avons réparti les tâches sur les 10 semaines de ce semestre (sans compter les vacances et le moment du stage). En parallèle des cycles de développement (1 et 2), il nous faudra nous renseigner sur les outils de développement de la carte, et sur la carte en elle-même, pour pouvoir savoir comment procéder pour les cycles 3 et 4. Nous décidons d'attribuer cette tâche à Marie Vialle. Avant de démarrer le cycle 2, il faudra aussi se renseigner rapidement sur l'utilisation d'un oscilloscope, pour générer des courbes de tension utile à l'intelligence artificielle. Cette tâche, de courte durée (1 jour), aura été attribuée à Alexandre Berard et Julien Liottard, pour pouvoir générer le plus de courbes possible en étant à deux sur cette tâche. A la fin du cycle 4, le produit fini, nous nous consacrerons à la préparation de l'oral de présentation de ce projet.

#### **2.3.1.2 Gestion des réunions avec le client et le tuteur**

### **2.3.2 Les défis techniques initiaux**

#### **2.3.2.1 Aptitude de l'IA à analyser des courbes de tension**

**2.3.2.2 Obtenir des quantités de banques d'apprentissage et de tests** Après s'être renseignés sur l'utilisation d'un oscilloscope, nous avons pu générer des courbes de démarrage de diverses cartes. Pour configurer le réseau de neurones, il est utile de créer une grande banque

---

<sup>25</sup>. Le diagramme de projet est disponible ici : <https://github.com/wyrellis/IA-Embarquee/tree/master/Planning%20S4> (lien URL)

de données, contenant un nombre pondéré de types de cartes différents. Les modèles de carte utilisées étaient un Arduino Duemilanove<sup>26</sup>, une STM32 MB997B et une Microship PICDEM 2 PLUS DEMO BOARD<sup>27</sup>. Les deux premières étaient alimentées par un générateur avec 5 volts et la dernière avec 9 volts. A l'aide de fils et de l'oscilloscope, nous avons récupéré la tension en fonction du temps et avons enregistré la courbe sous forme de fichier .CSV.

Pour obtenir assez de données, nous avons répété plus d'une centaine de fois le démarrage de la carte pour chaque modèle.

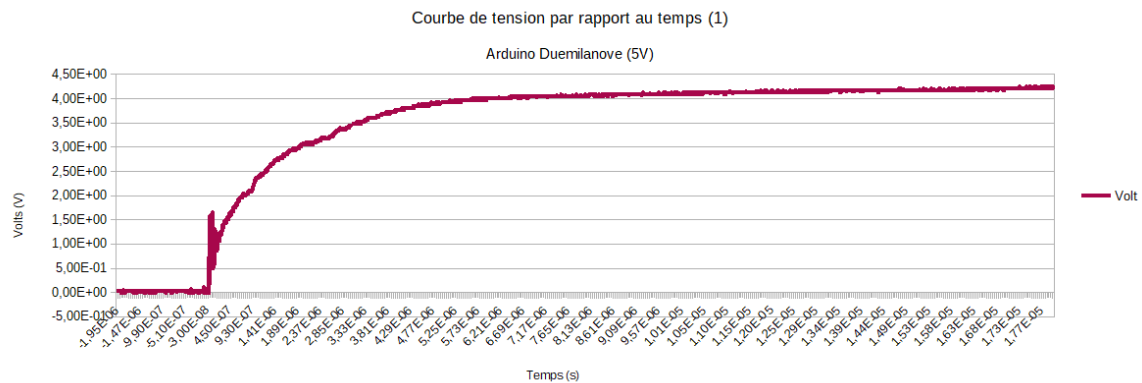


Figure ?? : Graphique acquis grâce à l'oscilloscope

### 2.3.3 Les difficultés rencontrés au cours du projet - Livre blanc ?

**2.3.3.1 Le paramétrage du réseau de neurones** Dans un soucis de précision, nous avons amélioré la forme de notre réseau de neurones. Il existe globalement trois manières d'arranger notre réseau de neurones ; *Small CNN*, *Medium CNN* et *Large CNN* Selon (lien : <https://blog.engineering.publicissapient.fr/2017/04/11/tensorflow-deep-learning-episode-3-modifiez-votre-reseau-de-neurones-en-toute-simplicité/>) l'article TensorFlow Deep Learning – Episode 3 du blog Publicissapient, on obtient respectivement 1,39%, 1,03% et 0,82% selon l'arrangement. Malgré le fait qu'il est intéressant de gagner en précision, il ne faut pas oublier le fait d'effectuer des calculs rapidement. A titre d'exemple, en *Medium* et *Large CNN*, on effectue un ou plusieurs calculs de pooling<sup>28</sup>. Cette opération prendrait un temps considérable puisqu'il faudrait effectuer une moyenne de deux valeurs une centaine de fois. Nous avons choisi de premièrement se baser sur un modèle assez simple, avec deux couches de 24 neurones en mode *Relu*<sup>29</sup>. Il s'est avéré que, pour les trois labels différents, le modèle convenait, donnant une précision de 100% à chaque apprentissage.

**2.3.3.2 La compatibilité de TensorFlow et le module IA de STM32CubeMx** Ne sachant pas quel type de réseau de neurones nous devons implémenter dans notre carte, nous

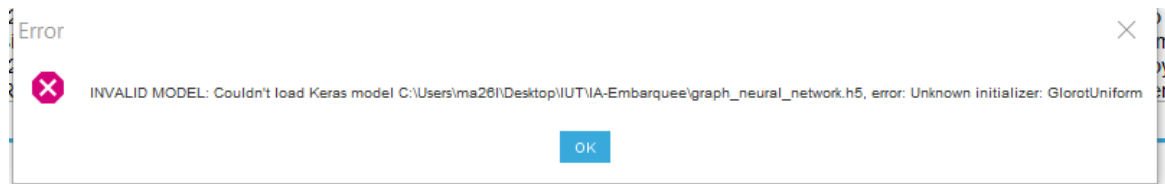
26. <https://www.arduino.cc/en/Main/arduinoBoardDuemilanove>

27. <https://www.microchip.com/Developmenttools/ProductDetails/DM163022-1>

28. Ici, réduire le nombre de données à intégrer en entrée du réseau de neurones

29. Couches de correction dans un réseau de neurones

pensions que mettre un réseau TensorFlow pourrait fonctionner. Or, le module AI de l'IDE de STM32 nous autorise seulement Keras, TensorFlow Lite, et d'autres moins parlant. Keras étant un module utilisé dans la librairie TensorFlow 2.0, nous pensions que sauvegarder le réseau sous le Keras utilisé par TensorFlow pourrait fonctionner. Hélas, une erreur apparaît.



*Figure ? : Erreur apparue lors du transfert du réseau sur STM32CubeMX*

Une solution serait de dissocier TensorFlow et Keras pour que cela puisse fonctionner, mais cela impliquerait que nous n'utilisions plus TensorFlow, ce qui nous aurait dérangé car c'est ce que nous pensions utiliser depuis le départ. Heureusement, TensorFlow 2.0 (version que nous utilisons) peut convertir les modèles en format exploitable par TensorFlow Lite. Le fichier binaire enregistré a pu être ouvert dans le module AI de l'IDE de STM32, avec les paramètres des couches de neurones renseignés depuis l'apprentissage.

### **2.3.3.3 Les difficultés liées à l'utilisation de STM32CubeMx**

**2.3.3.4 La prise en compte du réglage de l'oscilloscope** résumé : problèmes car le réseau a du mal

## **3 Bilan et perspectives**

### **3.1 L'état final du projet**

#### **3.1.1 Description des fonctionnalités du produit final**

##### **3.1.1.1 Facultés de l'IA pour la différenciation**

##### **3.1.1.2 Formatage des données sur carte**

##### **3.1.1.3 Résultats de l'analyse des données de la carte**

#### **3.1.2 Description de tâches complexes**

##### **3.1.2.1 Apprentissage d'anomalies**

##### **3.1.2.2 Réduction du temps de calcul**

#### **3.1.3 Description des tâches à réaliser**

##### **3.1.3.1 Récupération des données en temps réel**

3.1.3.2 Précision du réseau de neurones à améliorer

## 3.2 Les apports du projet

### 3.2.1 Apports humains et professionnels

#### 3.2.1.1 Apports professionnels

#### 3.2.1.2 Apports humains

### 3.2.2 Apports techniques

#### 3.2.2.1 Nouvelle méthode (Agile)

#### 3.2.2.2 Se familiariser avec STM32CubeMx et la librairie TensorFlow

#### 3.2.2.3 Notions d'électronique et de programmation embarquée

## 3.3 Perspectives de l'évolution du projet

### 3.3.1 Amélioration de l'IA

#### 3.3.1.1 Détection d'anomalies et de leur type

#### 3.3.1.2 Réduction du temps de calcul

#### 3.3.1.3 Aspect "temps réel"

### 3.3.2 Portabilité de l'application

#### 3.3.2.1 Rendre possible l'import de banques personnalisées

#### 3.3.2.2 Rendre possible l'import direct sur carte

4 Annexes (début sur chaque page impaire)

4.1 annexes

Stories	tâches associées	Durée	Total
Installer TensorFlow	Installer Conda	2	4
	Installer les modules	2	
Recherches sur l'IA et les réseaux de neurones	Recherches sur l'IA	5	20
	Recherches sur les réseaux de neurones	5	
	Recherches sur TensorFlow	5	
	Utilisation et fonctionnement de TensorFlow	5	
Tester les neurones avec des vecteurs	Savoir utiliser TensorFlow	10	25
	Essayer de faire reconnaître des vercteurs personnalisés	10	
	Tester la rapidité	5	

Stories	tâches
Convertir le signal analogique en signal numérique	Apprendre le fo
	osci
	Convertir les don
Faire apprendre les spécificités de deux cartes différentes au réseau de neurones	données
	Savoir lire des d
	prog
Tester le réseau de neurones	Observer les dif
	C
	Pouvoir analyse
	Ten
	Regarder le temp
	et optimisation
	Faire
	Résoudre

Stories	tâches associées
Mettre le réseau de neurones sur la cartes	Faire connaître la carte
	Finaliser le réseau de n
	Implémenter le réseau de
Rédiger la documentation	la carte
	Documentation sur l'In
	artificielle
Convertir un signal analogique en signal numérique	Documentation sur Te
	Apprendre le fonctionne
	oscilloscope
Faire apprendre le réseau de neurone	Convertir les données ana
	données numériq
	Savoir lire des données Ex
Tester le réseau de neurones	programme C
	Pouvoir analyser les diffé
	TensorFlow
	Regarder le temps de rec
	et optimisation du résea
	Faire des tests
	Résoudre les erre

Stories	tâches associées	Durée	Total
Création d'erreurs pour préparer le réseau de neurones	Débrancher la carte	2	6
	Baisser le voltage	2	
	Utiliser une carte défaillante	2	
Tester le réseau de neurones virtuels	Prendre des données sur deux cartes différentes	5	15
	Comparer les deux courbes	10	

Sprint	Stories	Durée en heures
Cycle 1 – 2 semaines – 49h	Installer TensorFlow	4
	Recherches sur l'intelligence artificielle	20
	Tester les neurones avec des vecteurs	25
Cycle 2 – 4 semaines – 77h	Convertir le signal analogique en signal numérique	12
	Faire apprendre les spécificités de deux cartes différentes au réseau de neurones	35
	Tester le réseau de neurones	30
Cycle 3 – 2 semaines – 21h	Création d'erreurs pour préparer le réseau de neurones	6
	Tester le réseau de neurones	15
Cycle 4 – 4 semaines – 120h	Mettre le réseau de neurones sur la carte	21
	Rédiger la documentation	20
	Convertir le signal analogique en signal numérique	4
	Faire apprendre le réseau de neurones	25
	Tester le réseau de neurones	50
Total en heures		267

## 4.2 table des figures

## 4.3 Accronymes

## 4.4 définitions

## 4.5 sources et bibliographies