Université de Strasbourg

Master 2 CSMI

Détection d'arcs électriques par des méthodes d'intelligence artificielle

Céline Lorentz

Tuteurs : **Patrick Schweitzer et Christophe Bonnet**

Stage de fin d'études du 21 février 2022 au 29 juillet 2022 à l'Institut Jean Lamour de Nancy

Année scolaire 2021-2022

Table des matières

Remerciements.

Je voudrais commencer ce rapport en adressant toute ma gratitude à mes maîtres de stage, M. Patrick Schweitzer et M. Christophe Bonnet, pour leur confiance, leur disponibilité et surtout pour leur patience et les précieux conseils qu'ils ont pu m'apporter.

Ce stage m'a permis d'affiner certaines pistes pour bâtir mon projet professionnel et signe l'aboutissement de mon cursus universitaire, c'est pour cela que je vous remercie de m'avoir accueillie.

Je remercie également les autres stagiaires de l'IJL, avec qui j'ai pu partager des moments chaleureux mais également échanger des conseils sur nos travaux respectifs.

Introduction du mémoire.

Ce mémoire de stage va me permettre de résumer les différentes étapes de mon stage. Je vais commencer par présenter le laboratoire dans lequel j'effectue mon stage, ainsi que mon sujet de stage et les différentes missions qui m'ont été confiées. Par la suite, je présenterais l'ensemble du travail que j'ai effectué et les différents résultats obtenus. Pour finir, je tirerais des conclusions de mon étude.

1 Introduction générale.

1.1 Présentation de l'institut.

Le stage que je réalise cette année a lieu à l'institut Jean Lamour (IJL). Commençons par présenter cet institut. L'IJL est un laboratoire de recherche fondamentale et appliquée en science des matériaux. C'est une unité mixte du CNRS et de l'Université de Lorraine.

Ce laboratoire multi-thématique couvre les matériaux, la métallurgie, les nanosciences, les plasmas, les surfaces et l'électronique en réponse aux enjeux sociétaux que sont l'énergie, l'environnement, l'industrie du futur, la mobilité, la préservation des ressources et la santé.

Cet institut possède un effectif d'environ 500 personnes et a été créé en 2009. Sur les 500 personnes que constituent cet établissement il y a 37 chercheurs, 134 enseignants-chercheurs, 98 ingénieurs, techniciens et personnels administratifs, 150 doctorants, 35 post-doctorants et environ 50 stagiaires. Les travaux de recherche de l'IJL vont de la conception du matériau jusqu'à ses applications industrielles et sont menés au sein de 25 équipes organisées en 4 départements scientifiques qui sont :

- Physique de la Matière et des Matériaux (P2M)
- Chimie et Physique des Surfaces et Solides (CP2S)
- Science et Ingénierie des Matériaux et Métallurgie (SI2M)
- Nanomatériaux, Électronique et le Vivant (N2EV)

Ces équipes de recherches peuvent s'appuyer sur 8 centres de compétences qui regroupent équipements et savoir-faire sur les thématiques suivantes :

- Conception et Fabrication (CC-HERE)
- Diffraction, Diffusion, fluorescence et tomographie X, spectroscopie Mössbauer (CC-XGAMMA)
- Dépôts et Analyses sous Ultravide de Nanomatériaux (CC-DAUM)
- Informatique et Calcul (CC-ERMIONE)
- Magnétisme et Cryogénie (CC-MAGCRYO)
- Micro et Nanotechnologie (CC-MINALOR)
- Microscopies, microsondes et métallographie (CC-3M)
- Optique-Lasers (CC-OL)

L'IJL est basé à Nancy, sur le campus Artem mais plusieurs de ses équipes sont localisées sur d'autres campus nancéiens ainsi qu'à Metz et Epinal. Pour mon stage, je suis restée uniquement sur le campus Artem de Nancy. L'IJL est classé en zone à régime restrictif (ZRR) au titre de la protection du patrimoine scientifique et technique de la nation. L'arrivée des stagiaires est par conséquent encadrée, et il faut une autorisation spéciale pour être admis.

FIGURE 1 – L'institut Jean Lamour de Nancy
Source : https://www.usinenouvelle.com/article/
a-nancy-l-institut-jean-lamour-inaugure-le-nouvel-ecrin-devolu-a-ses-recherches-
N827160

1.2 Présentation du laboratoire de recherche associé au stage.

Comme mentionné dans la partie précédente, l'IJL est organisé en différents départements et équipes. Je réalise mon stage au sein de l'équipe Mesures et Architectures Électroniques (MAE) dirigée par Patrick Schweitzer qui est mon tuteur lors du stage avec Christophe Bonnet qui est quant à lui ingénieur d'études.

L'équipe MAE est spécialisée dans la conception de capteurs et systèmes électroniques de mesure intelligents et autonomes. Les domaines d'application sont multiples et répondent à des enjeux sociétaux : santé, efficacité énergétique, renouveau industriel. L'expertise des membres de l'équipe permet d'aborder les problématiques de conception de systèmes électroniques embarqués à différents niveaux : depuis le capteur (notamment biocapteurs) jusqu'au système intégrant des traitements du signal complexes et rapides.

Leurs recherches prennent en compte les aspects d'apprentissage automatique, de tolérance aux fautes et de problèmes énergétiques et sont orientées autour de deux axes complémentaires :

- Capteurs, instrumentation et mesure
- Architectures de circuits et systèmes électroniques

Depuis 2007, l'équipe travaille sur la mise au point de systèmes de détection de défauts d'arcs électriques dans le domestique (230V, 50Hz), l'aéronautique (115V, 400Hz) et (270VDC) et dans le domaine des panneaux solaires photovoltaïques. Pour ma part, mon projet est orienté sur la détection d'arcs électriques avec une approche relevant de l'intelligence artificielle.

1.3 Présentation du sujet.

1.3.1 Qu'est-ce qu'un arc électrique ?

Un arc électrique se définit comme un phénomène physique qui se produit lorsqu'un milieu généralement isolant est ionisé et laisse ainsi passer le courant. Ce phénomène est visible à l'oeil nu. De manière plus concrète, nous pouvons donner un exemple : un arc peut se produire par exemple quand on retire un câble d'une prise électrique, il est possible de voir une étincelle qui signifie qu'un arc a eu lieu.

Un défaut d'arc électrique se définit quant à lui comme l'apparition d'un arc non intentionnel dans le circuit. C'est ceci que l'on aimerait réussir à détecter pour éviter que cela se produise.

Figure 2 – Exemple d'arc électrique
Source : `https://www.mesdepanneurs.fr/blog/arc-electrique`

Un arc peut se produire de deux manières différentes dans le circuit, en parallèle ou en série. Quand il se produit en parallèle sur l'alimentation du circuit, il produit une surintensité. Les disjoncteurs actuels sont capables de déceler une telle surintensité et de couper immédiatement le courant. Les arcs produits en parallèle ne sont donc pas ceux qui nous intéressent car leur détection est déjà assez bien maîtrisée.

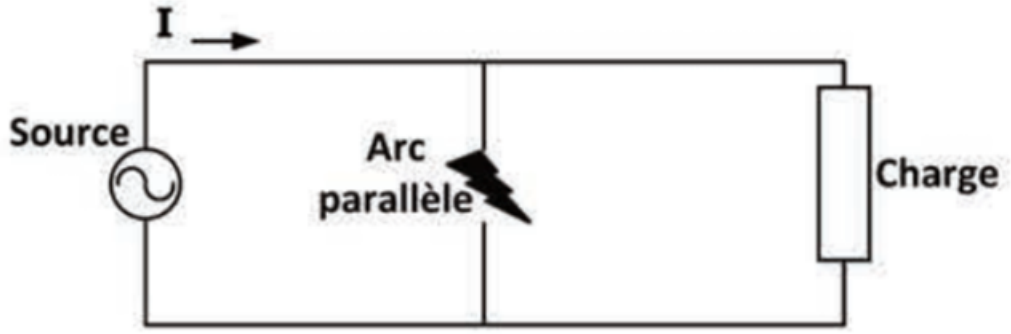

FIGURE 3 – Schéma électrique de la création d'un arc en parallèle
Source : Thèse [1]

En revanche, quand l'arc se produit en série avec les charges, il n'y a pas d'augmentation d'intensité dans le circuit. En effet, pour comprendre ceci, il faut connaître la loi d'Ohm $U = R * I$. La tension U_alim de l'alimentation est calculée en faisant la somme de la tension U_arc de l'arc et de la tension U_load de la charge variable que l'on a placée dans le circuit. Or, en l'absence d'arc, U_arc $= 0$, puis quand l'arc se produit, U_arc augmente. La valeur de la tension de l'alimentation est une constante fixée à $270V$. Pour conserver cette valeur constante malgré l'augmentation de la tension d'arc, il faut que U_load $=R * I$ diminue donc forcément l'intensité du courant I diminue. Le schéma électrique ci-dessous symbolise le circuit électrique tel que l'on va le considérer.

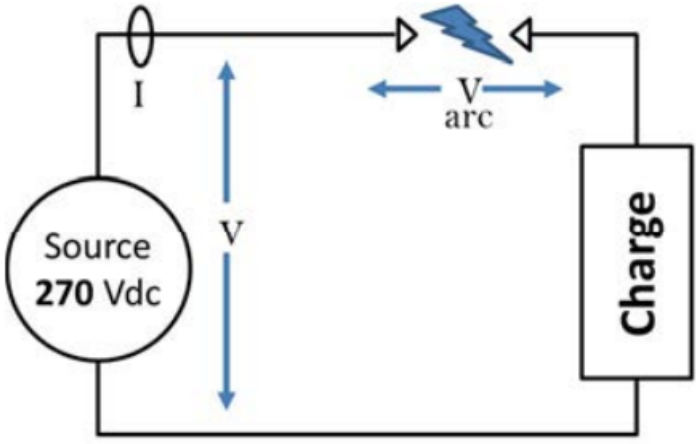

FIGURE 4 – Schéma électrique de la création d'un arc en série
Source : Thèse [1]

Or, les disjoncteurs détectent en général des sur-courants et des sur-tensions donc ils ne détectent pas ces arcs. C'est donc sur ce type d'arc que nous allons axer notre travail. C'est dans ce contexte que s'inscrit le projet, nous allons détailler ce contexte dans la partie qui suit.

1.3.2 Contexte du stage.

Les défauts d'arcs électriques dans les réseaux électriques, que ce soit au niveau domestique mais également aéronautique ou automobile, peuvent causer de graves incendies et provoquer en conséquence des accidents plus ou moins graves. Plusieurs crashs d'avion ont déjà été causés par des arcs électriques. En effet, lorsqu'un arc se produit, la température peut atteindre jusqu'à 5000 degrés localement.

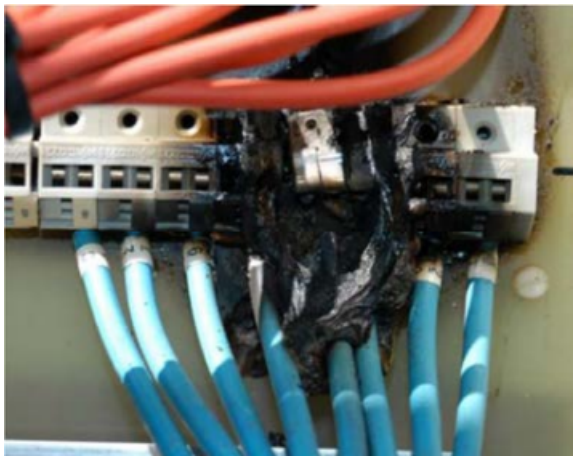

Figure 5 – Dégâts causés par un arc électrique
Source : Thèse [1]

Nous aimerions éviter au maximum qu'un arc électrique ne se produise et donc nous aimerions mettre au point des méthodes de détection d'arcs, qui détectent suffisamment tôt la présence d'un arc avant même que l'incendie ne se produise et qui nous permettraient de mettre en place des systèmes de protection des installations électriques.

Le laboratoire dans lequel j'effectue mon stage ayant déjà travaillé par le passé sur le domestique, nous nous intéresserons uniquement aux arcs apparaissant dans les avions électriques, donc avec un courant continu (DC). En effet, des études sur l'alimentation alternative (AC) ont déjà été menées, avec des résultats concluants. Il n'est par contre pas possible de reprendre les mêmes méthodes pour le circuit continu car la signature électrique des arcs produits par ces deux méthodes est très différente. C'est la raison pour laquelle nous allons nous tourner vers des méthodes d'intelligence artificielle qui sont actuellement les méthodes les plus prometteuses. Nous allons illustrer la différence de signature des signaux entre l'alimentation AC et l'alimentation DC.

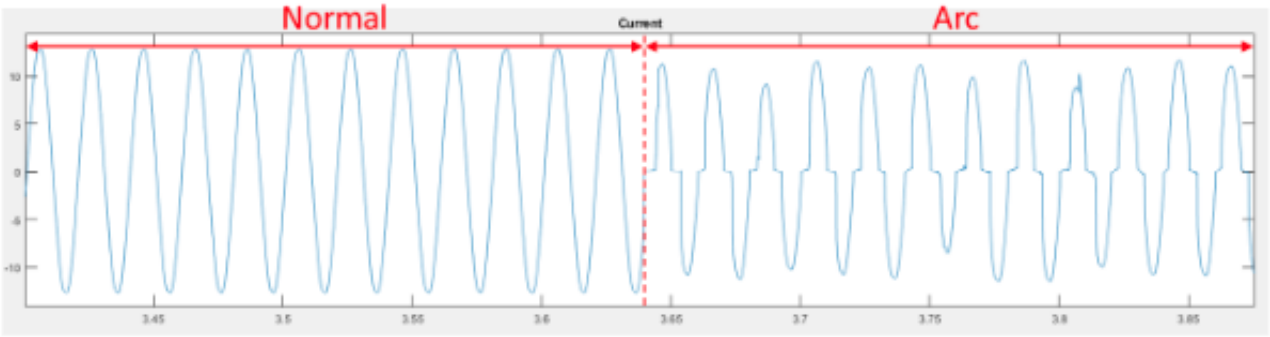

Figure 6 – Signature du signal en AC
Source : Rapport autre stagiaire

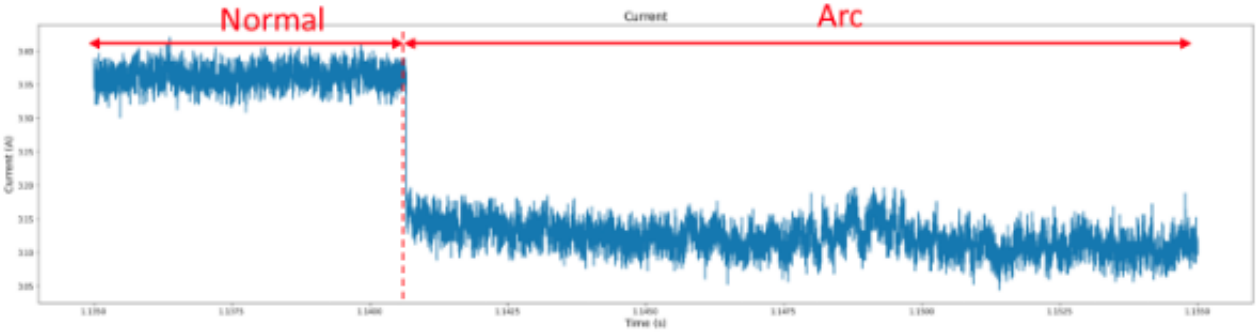

Figure 7 – Signature du signal en DC
Source : Rapport autre stagiaire

De ces deux courbes nous pouvons faire deux constats. Tout d'abord, nous voyons effectivement que les courbes ne se ressemblent pas du tout donc nous ne pouvons pas appliquer la même méthode de détection d'arcs. Par ailleurs, le deuxième constat que nous pouvons faire est que pour un même type d'alimentation (AC ou DC), les signaux avec arc et sans arc diffèrent, c'est pourquoi nous allons pouvoir utiliser des algorithmes d'intelligence artificielle basés sur l'analyse de la forme de ces signaux.

La détection des défauts d'arcs électriques sur un réseau continu est un enjeu important car le nombre de véhicules électriques se multiplie notamment les avions pour lesquels l'impact d'un accident n'est pas le même qu'avec une voiture.

1.3.3 Objectifs du stage.

Connaissant le contexte du stage, les objectifs principaux sont de développer des algorithmes d'intelligence artificielle qui permettent de détecter ces défauts. Nous pourrons ensuite comparer les performances entre ces différents algorithmes et tenter d'optimiser leurs performances. Un autre point important est d'analyser la bibliographie existante pour déterminer les algorithmes qui s'adapteraient le mieux à notre problème de détection d'anomalies sur des données échantillonnées. Enfin, nous disposons déjà d'un jeu de données d'arcs électriques pour entraîner et tester nos algorithmes, qui a été mis au point par un précédent stagiaire. Cependant, ce jeu de données est trop homogène et les données sont souvent mal labellisées. Nous irons

donc en laboratoire faire de nouvelles expérimentations pour acquérir de nouvelles données, en changeant par exemple le mode de production de l'arc (modification des charges,...). Cela nous permettra alors de créer une nouvelle base de données que nous allons labelliser de manière automatique. Cela nous permettra d'avoir une méthode de labellisation fonctionnelle que nous pourrons utiliser à chaque fois que nous aurons un nouveau signal.

2 Organisation du stage.

2.1 Outils utilisés.

Pour mener à bien mon stage, j'utilise différents outils et principalement les notebooks de Google Colab qui permettent de coder en Python de manière optimisée pour des projets d'intelligence artificielle. Au niveau des librairies, j'utilise beaucoup les librairies Keras et Tensorflow qui permettent de créer des architectures variées de réseaux de neurones. Quant au rapport, je l'ai écrit intégralement en LaTeX sous Overleaf.

J'ai également utilisé GitHub pour l'aspect organisationnel. En effet, j'ai créé un projet sur GitHub, en utilisant la méthode Kanban automatisée qui permet d'organiser efficacement le projet. Un projet contient 3 colonnes qui sont : To Do (à faire), In Progress (en cours), et Done (effectué). Ensuite, au sein du projet j'ai créé des Milestones qui sont des sous-objectifs du projet et qui permettent de séparer le projet en différentes étapes. Dans notre cas, on peut définir deux sous-objectifs principaux : la présentation intermédiaire du stage (V1) qui a eu lieu en avril et qui est une évaluation de stage à mi-parcours avec une première version du rapport et une présentation orale, et la version finale (Vfinale) qui aura lieu en août et qui sera l'évaluation finale du projet avec le mémoire final et la soutenance.

Puis, pour finir, j'ai créé des Issues qui sont toutes les tâches à réaliser pour mener à bien le stage. J'ai décidé de séparer chaque tâche de telle sorte que cela soit bien clair et détaillé. Quand elles sont créées, ces tâches sont alors placées automatiquement dans la colonne "à faire" du projet et quand je ferme une tâche elle est placée automatiquement dans la colonne "effectué". Cette automatisation permet de gagner du temps. Toutes les tâches sont liées à un sous-objectif (V1 ou Vfinale) de telle sorte à ce que l'on voit les objectifs de chaque sous-partie du projet.

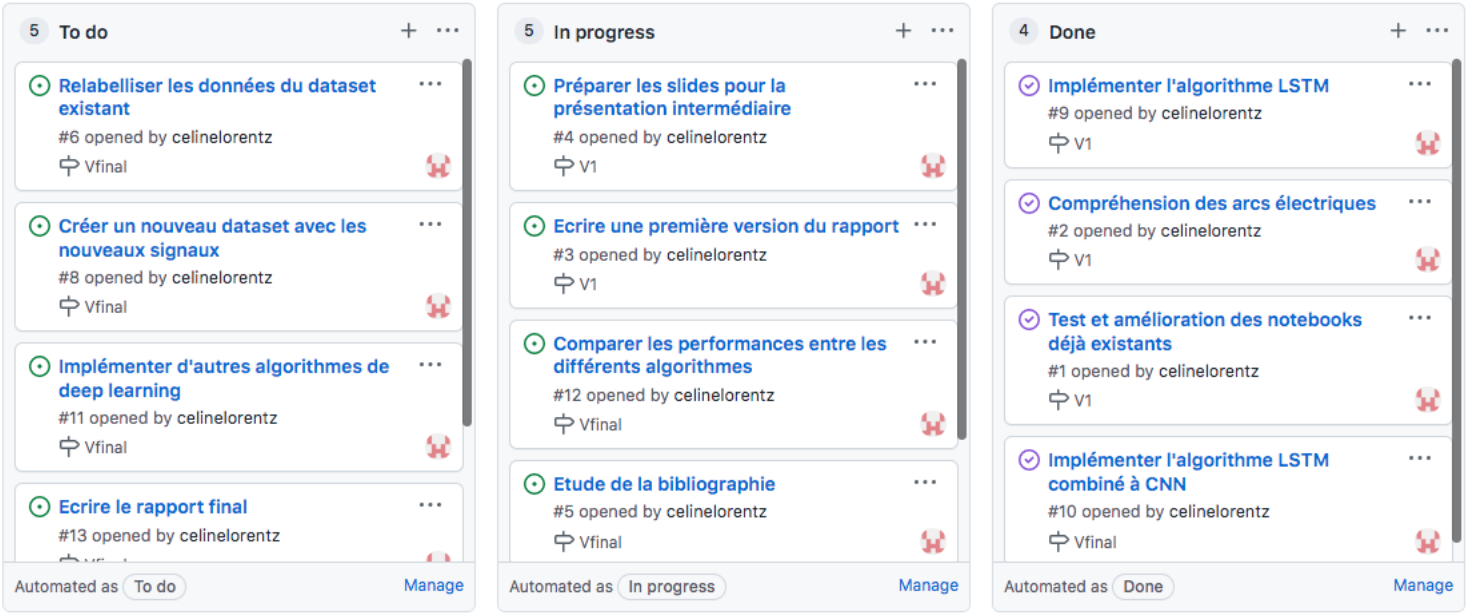

FIGURE 8 – Projet Kanban automatisé sous GitHub
Source : Répertoire GitHub personnel

Le diagramme de Gantt qui sera présenté par la suite permet de résumer toutes les tâches du projet avec un aspect temporel en plus. Une période de temps pour réaliser chaque tâche est fixée, période qui sera sûrement amenée à changer en fonction des aléas du projet.

2.2 Organisation des fichiers.

Les notebooks de codes sont principalement stockés sur Google Colab par soucis de simplicité. J'ai créé différents dossiers pour bien organiser mon travail. A côté de cela, j'ai également créé un répertoire GitHub privé qui regroupe l'intégralité de mon travail. Je vais détailler les différents dossiers créés pour que cela soit plus simple à comprendre :

- **.github/workflows** contient un fichier qui permet au rapport en latex de s'autocompiler et de générer automatiquement la version pdf du rapport
- **Code python** contient une copie de tous les notebooks que j'ai écris pendant le stage
- **Pictures** contient les images utilisées dans le rapport et les slides
- **Point hebdo** contient les diapositives en format pdf que je présente à mes tuteurs toutes les semaines
- **V1** contient le rapport et le diaporama de la V1 au format pdf

Pour ce qui est du rapport, la version la plus récente est placée à la racine du répertoire car cela est plus simple pour la génération automatique du pdf. De même pour les slides.

2.3 Road Map.

Au début du stage, j'ai créé un diagramme de Gantt qui permet de résumer tout le travail à accomplir au cours du stage. Bien sûr, il sera amené à évoluer au cours du stage, si des problèmes se posent ou des

nouveaux objectifs se dessinent. Ce type d'outil est très utile pour permettre d'organiser efficacement son travail, de toujours savoir ce qu'il reste encore à faire et ce qui a déjà été fait.

Tâche	Date de début	Date de fin	Durée (en nombre de jours)	Pourcentage complété
1ère partie du projet (V1)				
Travail bibliographique	21/02/2022	01/04/2022	30	100%
Compréhension des arcs électriques	21/02/2022	01/03/2022	7	100%
Test et amélioration des notebooks déjà existants	28/02/2022	18/03/2022	15	100%
Implémenter l'algorithme LSTM	01/04/2022	04/04/2022	2	100%
Implémenter l'algorithme LSTM+CNN	01/04/2022	15/04/2022	11	100%
Préparer la V1 du rapport et des slides	21/02/2022	15/04/2022	40	70%
2ème partie du projet (Vfinale)				
Travail bibliographique	18/04/2022	29/07/2022	75	10%
Reprendre le dataset existant et relabelliser les données	21/03/2022	06/05/2022	35	20%
Faire l'acquisition de nouveaux signaux en labo pour avoir plus de données différentes	01/04/2022	02/05/2022	22	50%
Créer un nouveau dataset avec les nouveaux signaux et les labelliser	02/05/2022	04/07/2022	46	0%
Implémenter d'autres algorithmes d'intelligence artificielle	25/04/2022	29/07/2022	70	0%
Comparer les performances entre les différents algorithmes	01/04/2022	29/07/2022	86	30%
Ecrire le rapport de stage	21/02/2022	15/08/2022	126	10%
Préparer les slides pour la soutenance	04/07/2022	15/08/2022	31	0%

FIGURE 9 – Diagramme de Gantt du projet initial
Source : `https://www.teamgantt.com/free-gantt-chart-excel-template`

Étant désormais en fin de stage, je vais mettre à jour cette version pour voir ce qui a évolué entre le début et la fin du stage. Cela sera une bonne illustration de la gestion d'un projet, car souvent tout ne fonctionne pas du premier coup.

Tâche	Date de début	Date de fin	Durée (en nombre de jours)	Pourcentage complété
1ère partie du projet (V1)				
Travail bibliographique	21/02/2022	01/04/2022	30	100%
Compréhension des arcs électriques	21/02/2022	01/03/2022	7	100%
Test et amélioration des notebooks déjà existants	28/02/2022	18/03/2022	15	100%
Implémenter l'algorithme LSTM	01/04/2022	04/04/2022	2	100%
Implémenter l'algorithme LSTM+CNN	01/04/2022	15/04/2022	11	100%
Préparer la V1 du rapport et des slides	21/02/2022	15/04/2022	40	100%
2ème partie du projet (Vfinale)				
Travail bibliographique	18/04/2022	10/05/2022	17	100%
Faire l'acquisition de nouveaux signaux en labo pour avoir plus de données différentes	01/04/2022	20/06/2022	57	100%
Créer plusieurs datasets avec les nouveaux signaux	01/06/2022	24/07/2022	38	100%
Implémenter d'autres algorithmes d'intelligence artificielle	25/04/2022	29/07/2022	70	100%
Comparer les performances entre les différents algorithmes	01/04/2022	29/07/2022	86	100%
Ecrire le rapport de stage	21/02/2022	15/08/2022	126	80%
Préparer les slides pour la soutenance	01/08/2022	25/08/2022	19	50%

FIGURE 10 – Diagramme de Gantt du projet final
Source : `https://www.teamgantt.com/free-gantt-chart-excel-template`

Nous constatons que les objectifs n'ont pas fondamentalement changé. Au sein de certains objectifs, nous avons rajouté certains éléments (par exemple, nous avons rajouté des algorithmes à implémenter, mais cela n'apparaît pas sur le diagramme car le diagramme est général et ne rentre pas dans les détails). La seule chose que nous avons abandonné est de modifier la labellisation de l'ancienne base de données, étant donné que nous ne savons pas exactement comment elle a été créée et il serait trop difficile de la modifier en l'état. Il vaut mieux repartir de zéro.

Par ailleurs, un autre point qui a été amené à être modifié est la durée allouée à chaque tâche que j'ai rallongée ou raccourcie en fonction de mon avancée.

2.4 Communication avec mes tuteurs.

Toutes les semaines ou éventuellement dans certains cas toutes les deux semaines, j'effectue une réunion avec mes deux tuteurs, souvent le vendredi. Cette réunion me permet de leur montrer ce que j'ai fait pendant la semaine et de leur montrer l'avancée du projet. Je prépare à chaque fois un diaporama d'une dizaine de slides. Ainsi, ils ont une vue d'ensemble du travail effectué et je peux également leur faire part de mes problèmes ou blocages. Ensuite, après leur avoir montré mon travail de la semaine, nous échangeons ensemble sur la suite du projet, sur ce que nous pouvons faire désormais, sur les points à travailler pour la semaine suivante. En fin de stage, étant donné que mon travail était surtout du code, je leur présentais directement mes notebooks avec des commentaires.

En plus de ces réunions, j'échange souvent avec Mr Schweitzer au cours de la semaine pour lui montrer ce sur quoi je suis en train de travailler et il m'aide à trouver des pistes en cas de blocage.

3 Techniques expérimentales de création d'arcs.

3.1 Description du banc expérimental.

Nous avons généré des arcs électriques en laboratoire. Pour ce faire, nous avons utilisé deux méthodes : l'écartement d'électrodes et les chemins carbonés. Ces deux méthodes permettent de créer des arcs et vont alors nous permettre d'obtenir une base de données variée et précise. Comme expliqué plus haut, les arcs générés sont en série entre l'alimentation et la charge, et non pas en parallèle.

L'équipe dans laquelle je fais mon stage possède un banc expérimental qui permet de générer des arcs électriques dans une enceinte fermée pour travailler en toute sécurité. Cette enceinte est en plexiglas et ses dimensions sont de 40x20x25cm et 15mm d'épaisseur.

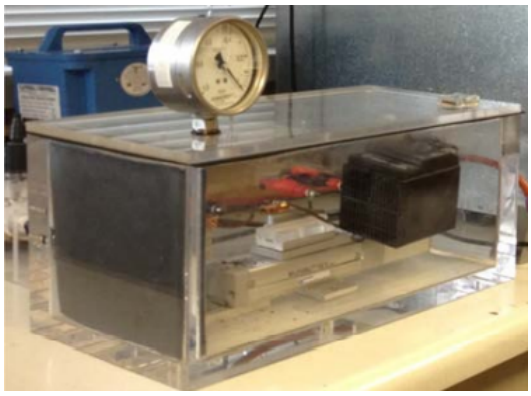

FIGURE 11 – Enceinte climatique du laboratoire
Source : Thèse [1]

Le circuit est alimenté par une source continue qui délivre $270V$ (constant). Il y a différentes charges qui sont présentes afin de pouvoir les ajouter dans nos tests. Par ailleurs, il y a un oscilloscope qui permet de faire les mesures de courant et de tension, et de les sauvegarder au format csv, pour pouvoir ensuite créer la base de données.

Voici une photo du banc expérimental avec les différents éléments que je viens de vous décrire.

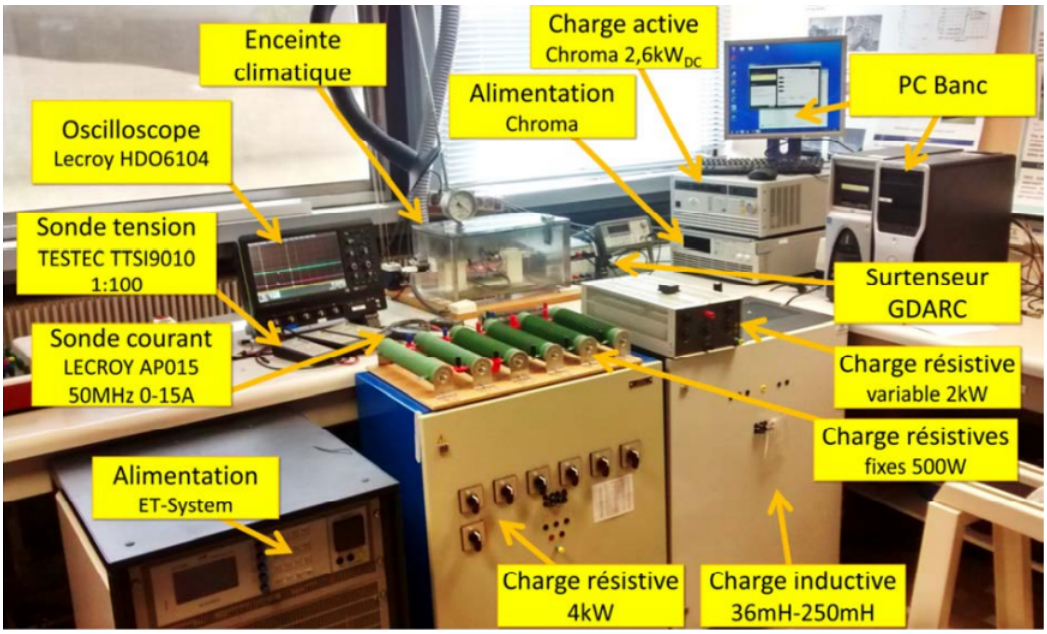

FIGURE 12 – Banc expérimental utilisé pour nos mesures
Source : Thèse [1]

3.2 Méthodes de génération d'arcs

Nous avons commencé par générer des arcs électriques par écartement de deux électrodes, c'est la méthode qui est utilisée en majeure partie dans le laboratoire. On utilise des électrodes qui peuvent être composées de différents matériaux (par exemple cuivre, acier, carbone, ...). On met ces électrodes sur notre banc et

on les écarte tout en faisant passer du courant. L'écartement est géré manuellement par une vis. Cet écartement va provoquer l'apparition d'un arc électrique, qui est visible à l'oeil nu. Immédiatement après, on rapproche à nouveau les électrodes et on coupe le courant pour ne pas provoquer un incendie.

Le principal inconvénient de cette méthode est que les mesures expérimentales peuvent être longues à faire. En effet, du fait de la chaleur intense et selon l'intensité du courant, il arrive fréquemment que lorsqu'on rapproche à nouveau les électrodes après que l'arc se soit produit, les électrodes restent collées. Il est alors difficile de les décrocher et elles s'abîment, il faut donc changer d'électrodes. Malgré tout, cela reste quand même une méthode assez simple pour générer des arcs.

Figure 13 – Électrodes cuivre/carbone
Source : Thèse [1]

La deuxième méthode consiste à utiliser un chemin carboné. Pour créer un chemin carboné, les câbles de deux conducteurs sont partiellement dénudés (striés avec un cutter) puis accolés et maintenus par du ruban isolant. Une tension de plusieurs kV est appliquée. Un chemin carboné est alors créé entre les deux câbles. Après carbonisation, on peut alors utiliser un tel chemin pour créer un arc électrique. Nous n'allons toutefois pas utiliser d'arc généré par chemin carboné dans notre base de données, car nous avons remarqué que les signaux générés par chemin carboné présentent un pic un peu avant le début de l'arc et ce pic risquerait d'être mal interprété par nos modèles. Nous allons donc plutôt nous intéresser à l'écartement d'électrodes, et pourquoi pas rajouter par la suite les chemins carbonés.

Figure 14 – Exemple de chemin carboné
Source : Thèse [1]

3.3 Acquisition des signaux pour la base de données

Nous avons généré nous-même l'ensemble des signaux pour la base de données que je vais créer et détailler dans une partie ultérieure. Cette étape fut assez longue à réaliser car il fallait aller en salle de manipulations, puis choisir le type d'électrodes que nous voulions utiliser. Il fallait également modifier les charges entre chaque essai. De plus, en raison de la température élevée, nous avions régulièrement des électrodes ou des fils qui prenaient feu, nous ne pouvions alors pas faire beaucoup d'essais avec une même électrode. Nous avons quand même enregistré la plupart de nos essais même ceux pour lesquels un problème a été rencontré (arc éteint trop tôt, trop d'étincelles, ...) et j'ai ensuite fait le tri des signaux à garder et ceux à éliminer. Cette étape de data cleaning fut totalement nécessaire. En effet, sans ce nettoyage de données, les résultats de nos tests auraient été faussés en raison de données non pertinentes qui auraient été mal labellisées par l'algorithme de labellisation automatique et qui auraient induis le modèle en erreur.

A la fin après tri, nous avons environ 250 signaux d'arcs et 45 signaux de leurres (saut de courant dû à une modification de charge semblable à un arc mais n'est pas un arc). Concernant l'aspect temporel, il nous a fallu environ 8 demi-journées pour faire toutes les acquisitions.

4 État de l'art.

4.1 Résumé du travail accompli par d'autres stagiaires.

Deux stagiaires ont travaillé avant moi sur le projet. J'ai eu accès à leur rapport de stage pour voir le travail qu'ils ont accompli. Comme ils ont suivi des études de mécatronique, une partie de leur stage était plus axée sur des prototypes de détection ne relevant pas de l'intelligence artificielle. Ils ont cependant également effectué un début de travail sur la détection par intelligence artificielle en utilisant des réseaux de neurones convolutifs. Ils ont créé une première base de donnée qui contient différents signaux qu'ils ont labellisé à la main. Ils ont ensuite transformé ces signaux en images de taille 28x28 ou 64x64. Par la suite, ils ont créé un réseau de neurones convolutifs avec plusieurs couches pour permettre la détection d'arc. Ce réseau sera une base pour le premier objectif de mon stage qui est d'implémenter un réseau de type CNN couplé à un réseau de type LSTM.

4.2 Résumé bibliographique.

Au début de mon stage, mon tuteur m'a donné quelques articles à lire pour me familiariser avec les arcs électriques et leur détection. Certains de ces articles présentent des méthodes de détection d'arc utilisant l'intelligence artificielle assez similaires à ce sur quoi je vais travailler durant le stage. Nous nous inspirerons ensuite des méthodes de détections mises au point sur des données séries dans d'autres domaines d'application pour lesquels la détection d'anomalies, de défauts peut être appliquée à notre problématique.

J'ai également fait mes propres recherches et trouvé divers cours et articles qui m'aident à développer des algorithmes et à comprendre les enjeux du sujet. Je vais résumer assez brièvement ces articles.

Dans l'article [2], il est question de créer un algorithme qui combine les réseaux de neurones convolutifs (CNN) et les réseaux de type LSTM pour faire de la détection de tumeurs en analysant des signaux. Cet article nous intéresse fortement car, dans notre cas, on veut aussi faire de la détection en analysant des signaux. Par ailleurs, l'architecture qui combine CNN et LSTM est une des architectures qui nous intéresse fortement. Dans l'article, il est expliqué que le modèle CNN combiné avec LSTM créé un modèle convolutif profond à mémoire à long terme.

Par ailleurs, on apprend que généralement l'utilisation d'une de ces deux méthodes de manière isolée permet seulement de faire de la classification binaire (2 classes), alors que la combinaison des deux permet

de faire de la classification multi-classes (5 classes dans le cas de l'article). Cela sera utile dans notre cas, car on voudra sûrement rajouter des classes supplémentaires. Pour l'instant, nous avons uniquement 2 classes (début d'arc et pas d'arc) mais il est prévu d'en rajouter d'autres donc il est utile de trouver un algorithme qui s'y adapte bien.

De plus, l'article nous apprend que ce type d'algorithme combiné présente une meilleure précision face au bruit. Cela est également une bonne chose dans notre cas car les signaux de notre base de données sont relativement bruités.

L'article présente également des architectures de réseaux types, à partir desquelles on pourra construire nous-même notre algorithme.

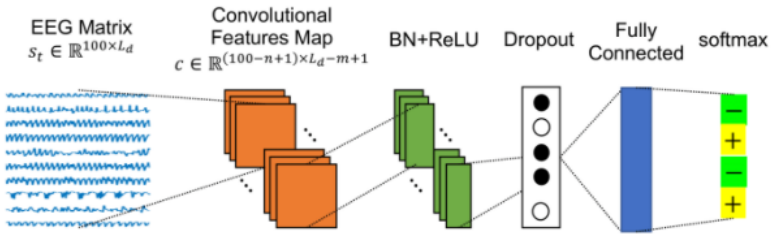

FIGURE 15 – Structure d'un réseau de neurones convolutif classique pour la classification
Source : Article [2]

On constate que le réseau de type CNN est constitué au début par des couches convolutives. Ces couches sont suivies d'une batch normalisation qui permet de joindre les entrées et les sorties et de normaliser les caractéristiques calculées indépendamment. Cette méthode permet d'augmenter la précision du réseau. Ensuite, ils utilisent une fonction d'activation de type ReLU qui est une fonction d'activation non linéaire, pour permettre l'apprentissage de limites de décision non linéaires. L'article nous indique que ce type de fonction d'activation permet d'améliorer la précision et d'accélérer les calculs. La couche de drop out est utile pour empêcher le sur-ajustement.

L'article présente ensuite l'architecture du réseau qui combine CNN et LSTM.

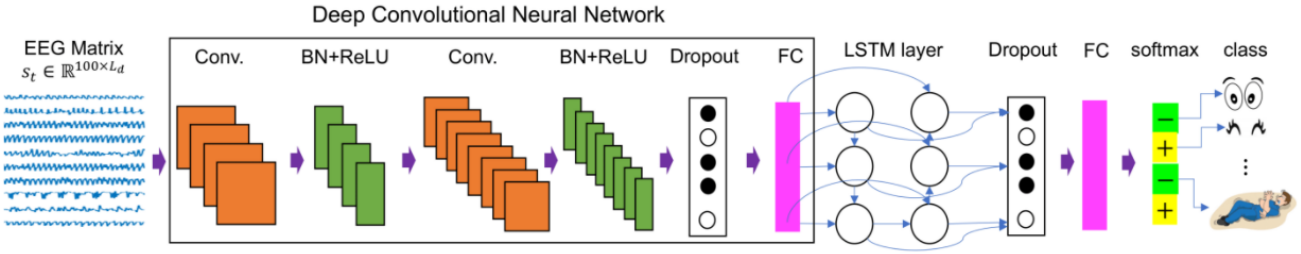

FIGURE 16 – Structure d'un réseau de neurones convolutif combiné à des couches LSTM pour la classification multi-classes
Source : Article [2]

On constate que la partie convolutive est pratiquement identique à celle présentée juste au-dessus. Puis on ajoute des couches LSTM, qui sont utiles d'après l'article car comme les signaux de l'article constituent une séquence temporelle (tout comme nos signaux), l'état actuel est étroitement lié à l'environnement précédent. Le modèle LSTM est le meilleur choix pour résoudre ce problème.

Dans l'article [3], une méthode de type CNN/LSTM est également présentée. Je ne vais pas autant rentrer dans les détails que pour l'article précédent car beaucoup d'éléments sont similaires. Il y a cependant un résultat intéressant que j'ai extrait qui montre l'accuracy des différents algorithmes. J'expliquerais plus en détail à quoi correspond l'accuracy dans une partie ultérieure, mais retenons pour l'instant que c'est un score entre 0 et 1 qui nous permettra de vérifier la fiabilité de notre modèle.

	Algorithm Type	Input Size		
		32 × 32	**64 × 64**	**128 × 128**
Accuracy (Recognition Time (t/ms))	**LSTM**	0.85 (0.017)	0.88 (0.033)	0.8 (0.167)
	CNN	0.88 (0.02)	0.89 (0.06)	0.92 (0.233)
	CNN-3	0.90 (0.04)	0.91 (0.099)	0.94 (0.33)
	CNN–LSTM	0.92 (0.033)	0.94 (0.067)	0.95 (0.2667)

FIGURE 17 – Accuracy et temps de reconnaissance pour différents algorithmes et différentes tailles d'entrée
Source : Article [3]

On constate que sur cet exemple, le réseau CNN-LSTM est effectivement le plus précis (mais pas le plus rapide, cela semble logique car il possède plus de couches). Le réseau LSTM seul est quant à lui le moins précis pour cet exemple.

4.3 Affichage des signaux et visualisation.

Avant de commencer à faire de la détection d'arc électrique par des méthodes d'intelligence artificielle, j'ai commencé par afficher les signaux issus de l'oscilloscope que nous avons enregistré au cours de nos manipulations, afin d'avoir une idée de ce qu'il se passe quand un arc a lieu. Tout d'abord, il faut savoir que lorsqu'on enregistre un signal depuis l'oscilloscope, on peut l'enregistrer au format csv ce qui est très pratique dans notre cas. On choisit donc ce format.

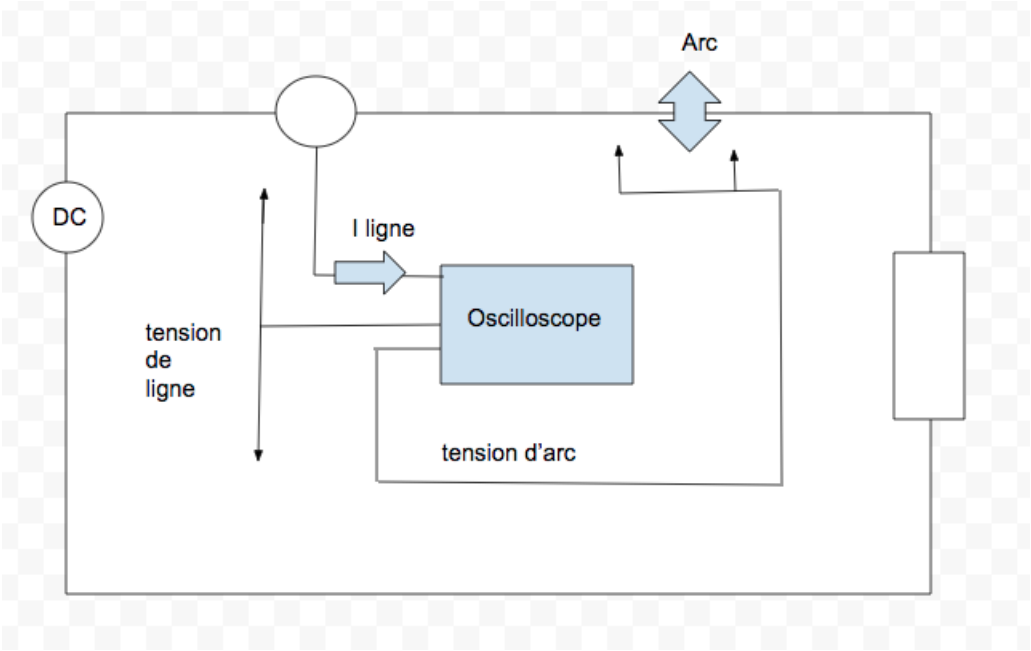

Figure 18 – Schéma de l'acquisition de la tension et du courant par l'oscilloscope
Source : Schéma personnel

L'oscilloscope enregistre 3 voies différentes nommées C1, C2 et C3. C'est essentiellement la voie C3 qui nous intéresse car elle mesure le courant de ligne. C1 mesure la tension de ligne et C2 mesure la tension d'arc, mais on ne va pas utiliser ces données pour nos algorithmes de détection. Cependant, la voie C2 sera utilisée pour la labellisation des données car une hausse de la tension permet de nous assurer qu'il y a effectivement un arc. On va commencer par afficher C3 pour pouvoir observer l'arc.

Pour observer ceci, j'ai créé un notebook qui affiche les signaux. Lorsque j'ai écrit ce code, la première difficulté que j'ai rencontrée est que les fichiers csv créés par l'oscilloscope ne sont pas destinés à être utilisés pour être analysés. En effet, quand j'ai ouvert le fichier correspondant au premier signal, j'ai constaté que les colonnes ont été créées assez étrangement. Le temps est placé en index dans la première colonne et le courant est dans cette même colonne, j'ai donc dû adapter le code pour pouvoir le lire. Après avoir extrait le temps et le courant, j'ai mis toutes ces valeurs dans 2 listes pour pouvoir afficher les graphiques. J'ai d'abord fait une courbe qui affiche simplement le courant. Voici un exemple d'une telle courbe sur un des signaux.

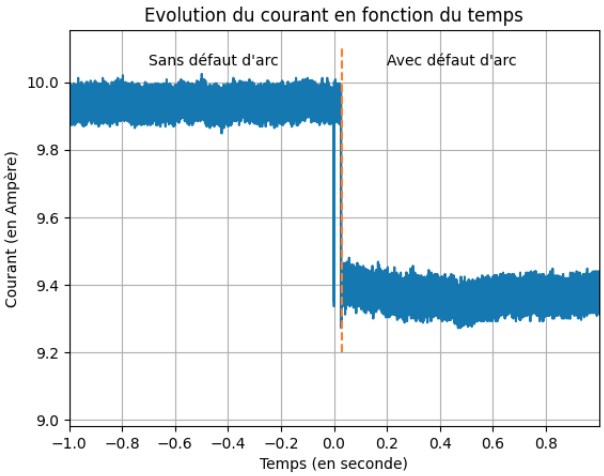

FIGURE 19 – Affichage du courant d'un signal
Source : Notebook personnel

On peut observer l'arc électrique qui se produit au temps $t = 0s$, le courant passe de $10A$ à $9.4A$, donc on constate une chute de courant lorsque l'arc débute. Sur ce signal, on peut bien séparer la partie sans défaut d'arc et avec défaut d'arc. Cependant, dans certains cas la transition est moins nette et plus bruitée ce qui rend plus difficile la détection de l'arc.

Mes tuteurs m'ont ensuite demandé de créer un code qui permet d'afficher le signal par fenêtres (un graphique du courant du signal en entier, un graphique du courant avant l'arc, un graphique de la transition quand l'arc se produit et un graphique du courant après l'arc). J'ai rajouté une option qui permet de choisir l'espacement des points en abscisse et en ordonnée, et également des paramètres à choisir pour définir nous-même les fenêtres avant arc/après arc. Mes tuteurs peuvent alors réutiliser aisément cet algorithme en modifiant seulement les valeurs de ces paramètres et sans avoir besoin de coder quoi que soit par eux-même. Voici un exemple d'un tel graphique ainsi créé, dont on génère automatiquement une version en png grâce à Python.

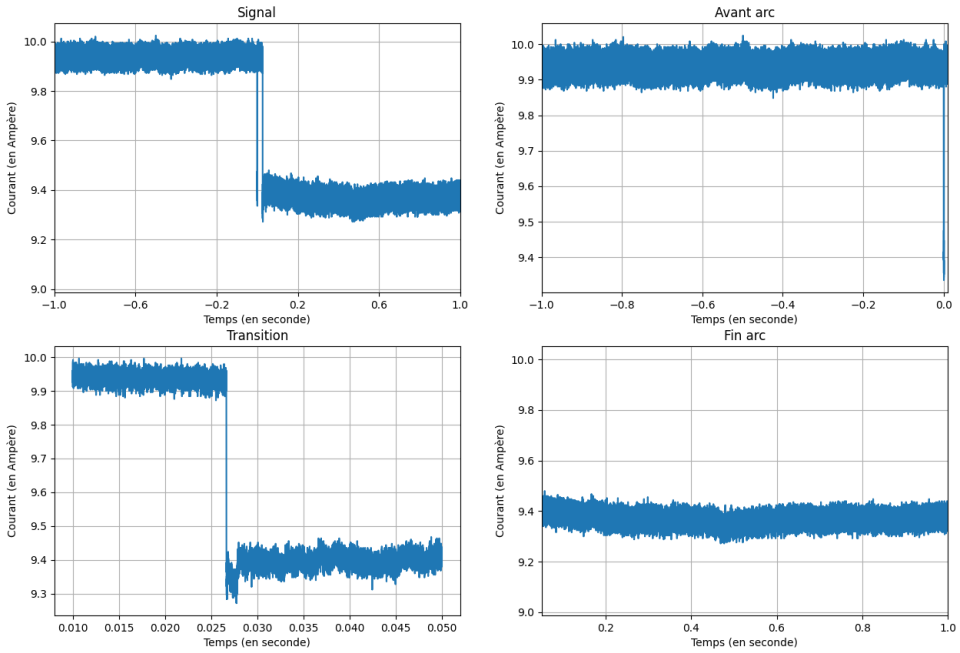

FIGURE 20 – Affichage du signal découpé en différentes parties avec fenêtres choisies par l'utilisateur
Source : Notebook personnel

Nous pouvons également afficher la tension d'arc (voie C2) pour nous assurer que ce n'est pas une anomalie du signal mais bien un arc. Si tel est le cas, alors la tension d'arc devra augmenter à l'instant de début présumé de l'arc (t=0).

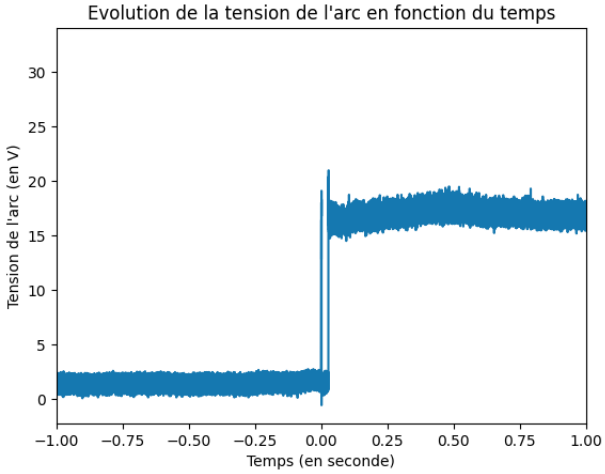

FIGURE 21 – Affichage de la tension d'arc en fonction du temps
Source : Notebook personnel

Nous observons effectivement une augmentation de la tension d'arc qui coïncide avec le début supposé de l'arc, ce qui nous permet de confirmer la présence d'un arc.

5 Création d'une nouvelle base de données.

5.1 Motivation à créer une nouvelle base de données

Quand on analyse la base de données déjà existante, on peut constater que plusieurs signaux ont été mal labellisés ce qui rend le travail compliqué. On pourrait peut être relabelliser ces signaux, mais le plus simple est de simplement recréer une base de données avec les nouveaux signaux et d'y inclure directement les nouvelles classes avec les nouvelles notations pour que tout soit correctement labellisé. Ce travail a été effectué dans la suite du rapport et il a pris un certain temps car malgré que nous pouvons mettre en place un algorithme de labellisation automatisé, la différence de signature entre les signaux ainsi que les anomalies présentes sur certaines courbes vont complexifier le travail. Nous allons voir un exemple d'anomalie.

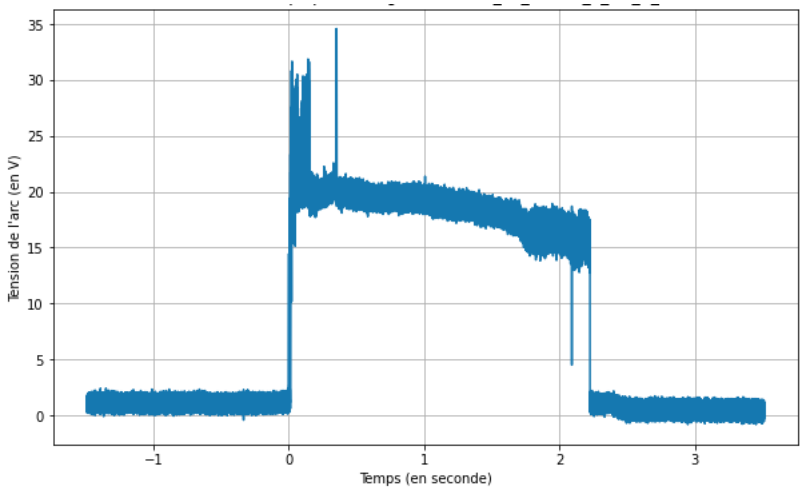

Figure 22 – Exemple d'anomalies sur un signal
Source : Notebook personnel

Sur le signal ci-dessus, nous constatons qu'il y a différentes anomalies. Tout d'abord, au déclenchement de l'arc aux alentours de t=0, la tension augmente fortement puis baisse et se stabilise. Ce pic de tension pourrait être mal interprété par l'algorithme et il faudra y faire attention. De plus, on constate que le signal présente deux autres pics de tensions (un pic de tension plus haute et un pic inversé de tension plus basse). Ce sont ces ensembles de phénomènes qui sont pour nous des anomalies et qu'il faudra tenter de maîtriser.

L'objectif des nouvelles mesures que nous avons effectuées est d'obtenir des données plus hétérogènes en variant à chaque fois les charges placées dans le circuit ce qui implique une modification du courant. Le fait d'avoir des données hétérogènes (ce qui signifie des données aux conditions de mesures différentes : charges, type d'électrodes,...) est primordial dans un projet de deep learning car cela permet à l'algorithme d'avoir vu différents cas de figures et ainsi d'être mieux entraîné. Nous allons donc créer une population significative de signaux avec des arcs électriques et également des signaux sans anomalie.

Nous allons développer plusieurs nouvelles bases de données qui correspondent mieux à nos attentes. Nous souhaitons que la labellisation de ces nouvelles bases de données soient automatisées, en effet la base de

données déjà existante a été labellisée à la main en regardant les valeurs de tension de déclenchement de l'arc propres à chaque signal, ce qui n'est pas cohérent avec autant de signaux. En automatisant la labellisation, dès qu'on aura un nouveau signal qui s'ajoute, il suffira de faire tourner l'algorithme avec le nouveau signal et la labellisation sera effectuée. Nous souhaitons également que la labellisation soit plus précise et puisse prendre en compte des classes supplémentaires du fait des transitions non franches entre le déclenchement de l'arc et la non présence de l'arc.

5.2 Présentation des données.

Les données utilisées lors de mon stage sont de nature très variées, elles proviennent toutes du laboratoire de recherche de mon tuteur. Les mesures sont réalisées pour un circuit dont la tension d'alimentation vaut 270V et les courants peuvent atteindre 16A, c'est pour cette raison que pour des raisons de sécurité, seuls mes tuteurs pouvaient faire les essais, et je pouvais uniquement faire varier les charges ou enregistrer les signaux.

Au niveau des bases de données que nous allons créer, les données sont des matrices 2D obtenues à partir de segment du courant numérisé. On transforme les signaux en images de différentes tailles. Il aurait également été possible d'analyser directement les signaux mais nous avons fait une étude de faisabilité à l'aide de la bibliographie et nous avons retenu la méthode qui les analyse sous forme d'images (méthode utilisée pour la détection de défaut dans des roulements à bille).

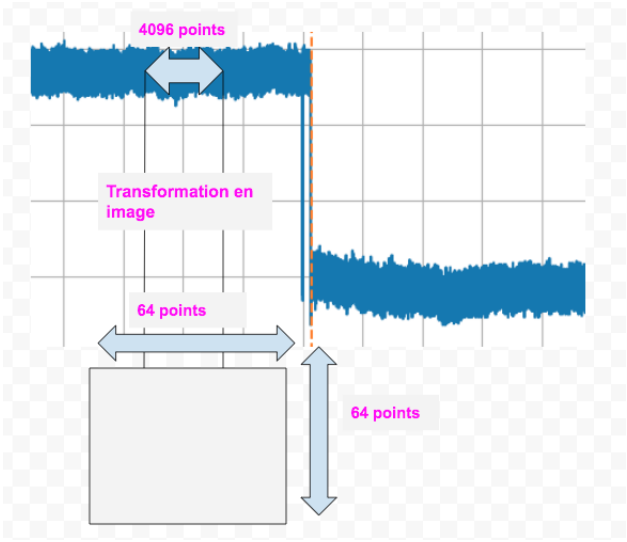

Figure 23 – Transformation de segment en image
Source : Image personnelle

5.3 Lissage et leurres

Tout d'abord, nous avons constaté que le bruit présent sur nos signaux ainsi que la présence de certains pics (étincelle, saleté sur l'électrode,...) entravent notre labellisation, étant donnée que celle-ci se base essentiellement sur les modifications de valeurs de tension. C'est pourquoi nous avons décidé de lisser les signaux par moyenne mobile exponentielle, de cette manière les pics apparus par erreur disparaissent et le signal reste quand même fidèle à ce qu'il était avant lissage. La moyenne mobile exponentielle est un filtre linéaire passe-bas du premier ordre tout à fait classique.

Par ailleurs, dans la réalité, certaines variations de courant peuvent être détectées comme étant un arc alors qu'elles n'en sont pas un. Pour permettre à l'algorithme d'apprendre le mieux possible en condition réelle, nous décidons d'ajouter des signaux de leurre, qui correspondent à des variations de charges et donc à des baisses de courant similaires à la baisse de courant qui apparaît en présence d'un arc. Cependant, nous labellisons ces signaux comme n'étant pas des arcs (classe 0) étant donné que ce sont des leurres, et le modèle devra les différencier par lui-même.

5.4 Labellisation des données point par point.

Nous avons commencé par labelliser les données point par point. En effet, c'est la manière la plus simple de labelliser des données quand les données sont sous la forme d'une suite de points. Après avoir analysé les signaux, nous avons décidé de faire notre labellisation grâce à la tension d'arc et non pas avec le courant, car la tension varie de manière plus uniforme entre les signaux quand un arc se produit. En revanche, on enregistre uniquement les valeurs de courant dans le dataset labellisé car c'est le courant qui sera utilisé par nos modèles de détection d'arcs.

Pour pouvoir faire une labellisation automatique, nous avons commencé par déterminer des seuils de valeurs de tensions communs à tous les signaux, ces valeurs de tensions correspondent à la présence d'un arc ou non et également à la transition au début de l'arc. Après avoir fait différents essais pour voir quelles valeurs convenaient le mieux, nous avons choisi des seuils et labels que vous retrouverez dans le tableau ci-dessous :

Conditions	Label	Signification
Entre 0V et 4.7V	0	Pas d'arc
Au dessus de 4.7V et t<0s	1	Début d'arc
Au dessus de 4.7V et t>0s	2	Arc

Table 1 – Détail des labels

En me basant sur ces valeurs, j'ai alors implémenté un algorithme de labellisation qui créé ensuite un fichier (au format csv) de labellisation point par point pour chaque signal. Ce fichier contient 3 colonnes (le temps, le courant non lissé et le label associé). En voici un exemple :

	Temps	Courant	Label
0	-0.500001	10.988	0
1	-0.500000	10.988	0
2	-0.499999	10.996	0
3	-0.499998	10.992	0
4	-0.499997	10.996	0
...	...	...	...
999997	0.499996	10.212	2
999998	0.499997	10.188	2
999999	0.499998	10.196	2
1000000	0.499999	10.180	2
1000001	0.500000	10.200	2

Figure 24 – Exemple d'un fichier csv généré par notre algorithme de labellisation point par point
Source : Notebook personnel

Nous allons maintenant voir un exemple concret de notre labellisation point par point sur des signaux.

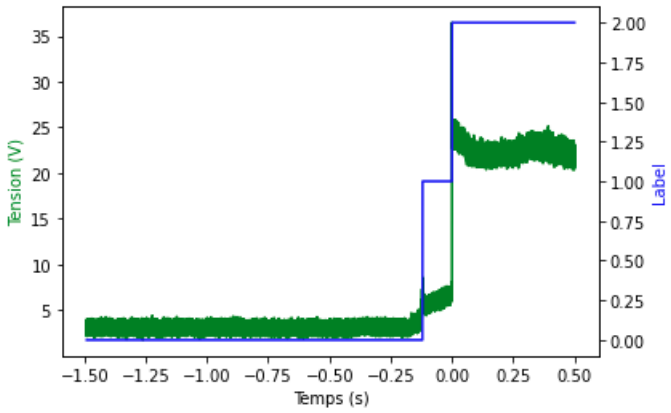

FIGURE 25 – Exemple de labellisation point par point avec affichage de la tension
Source : Notebook personnel

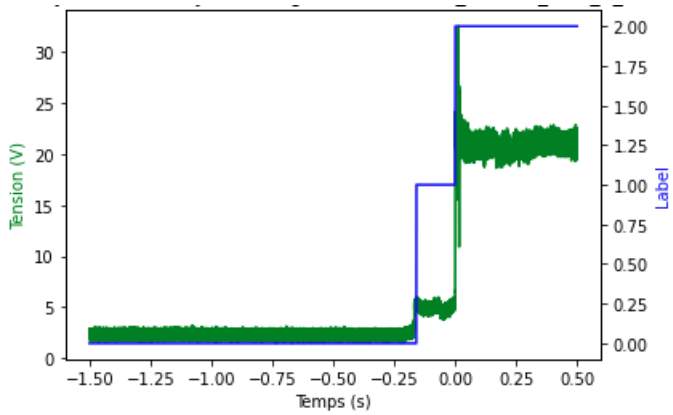

FIGURE 26 – Exemple de labellisation point par point avec affichage de la tension
Source : Notebook personnel

On voit bien les différentes étapes de la labellisation, avant l'arc chaque point est labellisé 0 (non arc), puis sur les deux signaux on observe une transition qui correspond au label 1 (début d'arc). Cette transition n'est pas systématique et certains signaux ne possèdent pas de transition. Enfin, quand l'arc est déclenché les points sont labellisés 2. Sur ces exemples, l'arc ne s'éteint pas mais dans certains exemples l'arc s'éteint avant la fin de l'enregistrement et le label repasse alors à 0.

5.5 Transformation du fichier de points en fenêtre de labellisation.

A ce stade-là, nous avons en notre possession les fichiers contenant les labels pour chaque signal (1 fichier par signal), nous voulons désormais sélectionner des fenêtres sur chacun des signaux et les labelliser. L'objectif est de sélectionner différentes fenêtres pour avoir une vue représentative sur chacun des signaux, à différents instants.

Nous allons tout d'abord sélectionner des fenêtres sur la partie du signal où l'arc n'a pas encore débuté. Pour ce faire, nous devons sélectionner les points labellisés 0 dans le fichier de labellisation par point. Comme

nous voulons faire des fenêtres, nous sélectionnons uniquement les endroits où il y a suffisamment de 0 consécutifs (un nombre égal ou supérieur à la taille de nos fenêtres). Nous sélectionnons ensuite un nombre conséquent de fenêtres (nombre en paramètre à choisir par l'utilisateur), par exemple dans notre cas 30 fenêtres. Nous faisons ensuite de même pour le label 2 (arc), nous sélectionnons des fenêtres de points labellisés 2 consécutifs.

En revanche, au niveau des transitions, nous allons procéder différemment. En effet, nous aimerions sélectionner des fenêtres glissantes au niveau de la transition. Ces fenêtres contiendraient un certain pourcentage de la partie avant l'arc (donc soit des point labellisés 0 = pas d'arc ou des points labellisés 1 = début d'arc) et un certain pourcentage de la partie qui contient l'arc (les points labellisés 2). Pour avoir suffisamment de fenêtres, on sélectionne toutes les répartitions de pourcentages possibles de 5 en 5 (5%-95%, 10%-90%, ..., 95%-5%). De cette façon, nous sélectionnons de nombreuses fenêtres de transition.

Voici un schéma qui illustre la manière dont nous procédons.

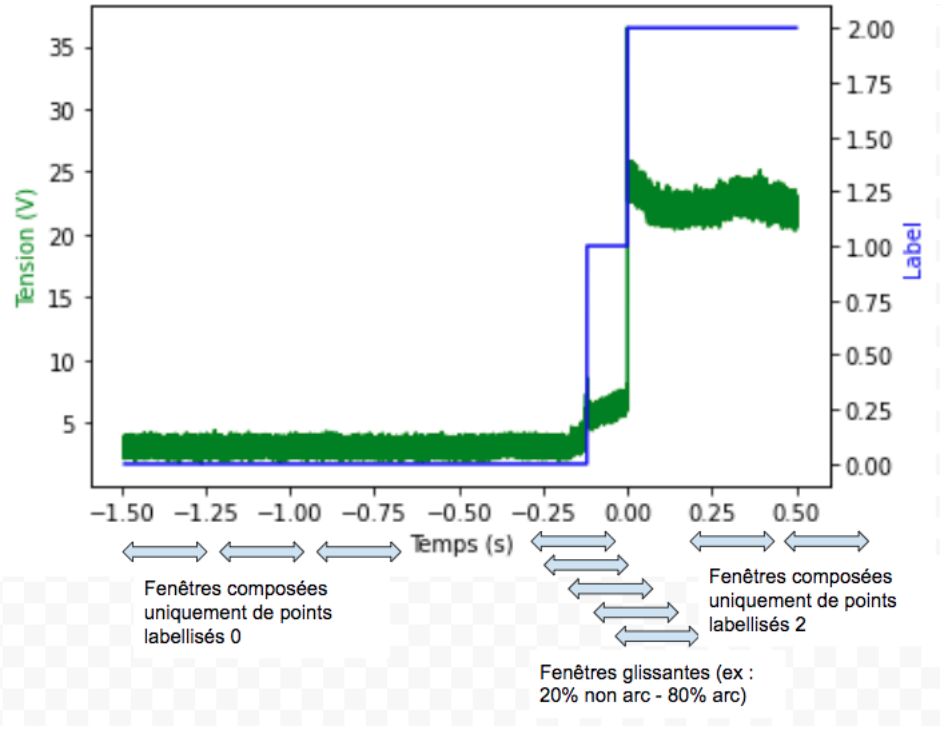

FIGURE 27 – Création des segments de labellisation
Source : Image personnelle

Afin d'avoir différents éléments de comparaison, nous allons créer 4 bases de données différentes composées de différentes classes, qui correspondent à 4 cas différents :

— **Cas 1 : 3 classes (non arc et début arc et arc)**

Ce cas est le seul cas de classification non binaire. Il est le cas le plus ambitieux car bien qu'il soit celui qu'on aimerait retenir, il est fortement probable qu'il ne soit pas le plus performant en raison du déséquilibre entre les classes, des ressemblances entre certaines classes et également car les algorithmes que nous avons écrit sont adaptés à la classification binaire. De plus, nous ne pourrons pas calculer la précision et le rappel (et donc le score F1) pour cette base de données car une telle métrique ne se

définit pas de la même manière quand il y a 3 classes ou plus. Nous allons donc uniquement afficher la matrice de confusion.

- **Cas 2 : 2 classes (non arc et début arc)**
  Ce cas est un cas similaire à l'ancienne base de données mais labellisée automatiquement et avec moins d'erreurs. Dans ce cas, nous nous intéressons uniquement à la partie avant l'arc et à la transition lors du déclenchement de l'arc.
- **Cas 3 : 2 classes (non arc et début arc+arc)**
  Le cas 3 considère les trois parties (avant l'arc, transition, après l'arc), mais au lieu de les séparer en 3 classes, nous les séparons seulement en 2 classes en regroupant la transition et l'arc car au niveau de la transition l'arc débute donc il n'est pas faux de considérer que la transition correspond déjà à l'arc.
- **Cas 4 : 2 classes (non arc et arc)**
  Le cas 4 ne prend pas en compte la transition afin de voir si la transition est un élément qui pourrait être problématique pour l'algorithme et voir si les performances sont meilleures sans la transition.

De plus, nous allons créer différentes tailles de bases de données pour voir l'influence de la taille sur les résultats. En effet, la taille des segments est à choisir par l'utilisateur. Dans notre cas, nous allons essentiellement considérer 3 tailles de segment qui sont 28*28, 64*64 et 128*128.

6 Présentation des modèles de deep learning utilisés lors du stage.

Nous allons implémenter des algorithmes d'intelligence artificielle qui vont nous permettre de faire de la détection d'arcs électriques. L'objectif sera de minimiser le nombre de fausses détections car si on veut ensuite utiliser ces algorithmes dans des cas concrets (exemple : dans un avion), des fausses alertes pourraient impliquer des conséquences graves tel qu'un atterrissage d'urgence qui s'avèrerait inutile. A l'inverse, une non-détection d'un arc serait encore plus grave car un incendie pourrait se propager rapidement dans l'avion. Le but sera donc d'être le plus fiable possible, c'est pourquoi nous mesurerons les performances de nos algorithmes pour voir lesquels sont les plus aptes à être utilisés.

6.1 Les réseaux de neurones convolutifs (CNN).

Le premier type d'algorithme que nous allons étudier est un algorithme très classique en intelligence artificielle appelé réseau de neurones convolutifs (CNN : convolutional neural network). Ils sont très puissants pour analyser des signaux (images, sons, vidéos).

Nous allons présenter le réseau LeNet5 qui est l'un des premiers réseaux de type CNN et qui a été inventé en 1998. Son principe est d'enchaîner des convolutions d'images. Une convolution est l'application d'un filtre à une entrée, ce filtre entraîne une activation. Donnons un exemple d'un tel enchaînement. On part d'une image d'entrée à 3 canaux (rouge, vert, bleu). Par convolutions, on obtient une image avec 30 canaux dont chaque pixel est un neurone de la première couche. Ensuite, on obtient une image avec 60 canaux et ainsi de suite. Ceci est le principe des convolutions d'images qui est un processus créé en observant le cerveau humain. En effet, il existe un processus d'élargissement du champ de vision dans les différentes couches de neurone.

Grâce à cette architecture, le réseau se concentre sur les motifs de bas niveau dans les premières couches (exemple : détection de contours), puis il les assemble en des motifs de haut niveau (détails plus précis) dans les couches élevées. Cette structure hiérarchique se retrouve naturellement dans les images issues

de la nature. C'est une des raisons pour laquelle les réseau convolutifs sont si efficaces pour analyser des images. Puis, après chaque convolution on ajoute une fonction d'activation non linéaire (le plus souvent ReLU).

Par ailleurs, entre certaines couches de convolutions, on peut ajouter des couches de max-pooling qui réduisent la taille spatiale des images. On réduit ainsi le nombre de paramètres, ce qui diminue le sur-apprentissage. Pour mieux comprendre, chaque neurone après un max-pooling 2*2 a un champs de vision deux fois plus étendu sur l'image initiale. De plus ils ne retiennent que l'essentiel (à cause du max). Par contre, à chaque max-pooling on se permet d'augmenter le nombre de canaux. Les layers ont ainsi plus de couleurs pour se représenter les caractéristiques importantes de l'image. La fin d'un réseau LeNet est constituée d'une ou de plusieurs couches denses (fully connected) qui synthétisent les derniers features pour les transformer en un vecteur de probabilité, correspondant aux classes que l'on veut prédire. Nous allons voir un schéma récapitulatif de ce réseau.

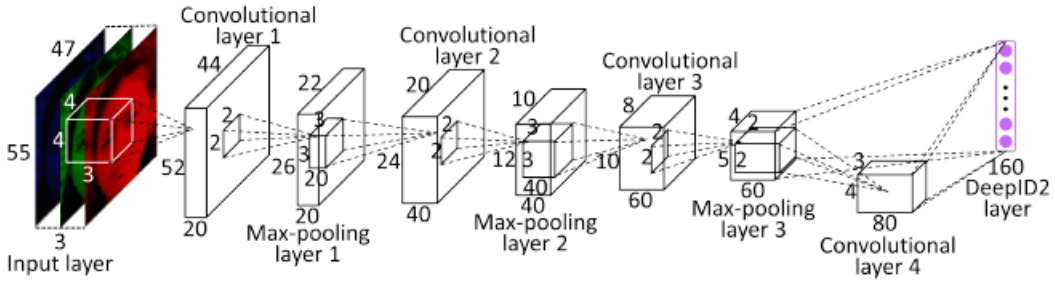

FIGURE 28 – Schéma d'un réseau de neurones convolutif
Source : Livre "Hands on Machine Learning", A Géron

Pour le stage, on va travailler avec la librairie Keras pour implémenter le CNN. En utilisant cette librairie, on doit placer les images dans un tenseur de dimension 4, il faudra bien faire attention à cette dimension. La taille du tenseur correspond à $(batch\_size, height, width, nb\_channels)$ avec

- **batch_size** le nombre d'images que l'on traite en même temps
- **height, width** les dimensions des images
- **nb_channels** le nombre de canaux

Par ailleurs, nous avons choisi d'utiliser le réseau LeNet5 comme base et non pas un autre modèle plus récent car selon plusieurs articles un tel réseau est plus facilement implantable dans un dispositif électronique, ce qui est l'objectif à long terme de notre travail.

Nous allons maintenant voir plus en détail le code du modèle que j'ai implémenté et voici tout d'abord le schéma du modèle.

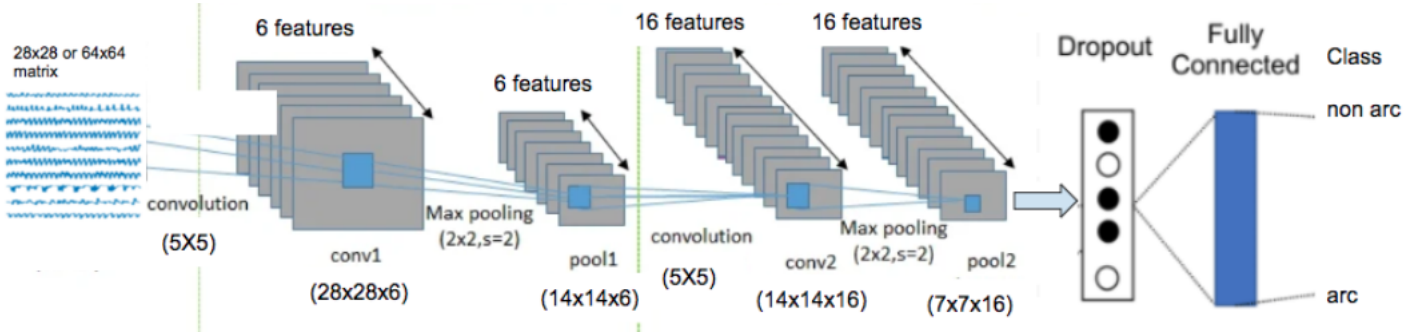

FIGURE 29 – Détail du modèle CNN que j'ai implémenté
Source : Notebook personnel

Au niveau du code, je vais vous expliquer en détail l'architecture de mon notebook. Cette architecture sera à peu de choses près la même pour les autres modèles donc je vais uniquement la détailler pour ce modèle.

Au début du notebook, il y a un certain nombre de variables qui sont à définir et qui seront des paramètres pour la suite. Tout d'abord, il faut choisir la répartition entre les données d'entraînement et de test. En général, c'était 80% pour l'entraînement et 20% pour le test mais ce nombre est modifiable si on veut faire d'autres tests.

Ensuite, il faut entrer la taille des données en entrée, taille qui dépend de la base de données choisie et on choisit également le nombre de classes pour la classification (en principe 2 classes, excepté pour le cas 1 qui contient 3 classes).

Après avoir choisi ces paramètres, je passe au brassage des données qui a pour but de réduire la variance et de s'assurer que le modèle reste général et s'adapte moins.

La prochaine étape est la transformation des données en image donc en matrice. Pour ce faire, on créé une matrice vide de taille $(nb\_segment, hauteur, largeur)$ avec $hauteur$ et $largeur$ la taille des segments (par exemple $hauteur = largeur = 64$ pour des segments de taille $64 * 64$). On fait ensuite une triple boucle avec ces 3 dimensions et on remplit la matrice à l'aide du dataset en entrée.

On normalise ensuite les données pour avoir la même gamme de valeurs de features. Puis, on redimensionne les données car comme vu précédemment, les tenseurs en entrée de nos couches convolutives doivent être de dimension 4 alors qu'actuellement nos matrices sont de taille 3.

Enfin, la dernière étape avant l'entraînement du modèle est la binarisation des labels. En effet, les labels sont quantitatifs (0, 1, 2...) et cela n'aurait aucun sens de laisser ces valeurs numériques, il vaut mieux les binariser.

Maintenant, après avoir fait toutes ces étapes, passons au modèle. Le modèle que j'ai écrit est composé tout d'abord d'une couche convolutive (qui constitue toujours la première couche d'un réseau de neurone convolutif), puis d'une couche de pooling pour réduire la dimension des features/images tout en gardant leurs caractéristiques importantes. Ensuite, il y a à nouveau une couche convolutive et une couche de pooling. Ensuite, on fait un flatten qui convertit nos données en un tableau uni dimensionnel pour pouvoir les intégrer dans la couche suivante. Pour finir, on a trois couches denses (qui constituent toujours la dernière couche d'un réseau de neurones).

Entre toutes les étapes, on fait du dropout pour réduire l'overfitting (manque de généralisation) du modèle.

On entraîne ensuite le modèle et on ajoute un callback qui permet d'interrompre prématurément l'entraînement si les performances ne cessent de s'améliorer. On peut choisir le nombre d'époques pour l'entraînement, nombre qui varie fortement selon les modèles. Pour celui-ci, nous choisirons une cinquantaine d'époques.

Pour finir, nous évaluons les performances du modèle en affichant différents scores que nous détaillerons par la suite.

6.2 Les réseaux LSTM (Long Short Term Memory).

Nous allons maintenant étudier un autre algorithme d'intelligence artificielle appelé réseau de type LSTM, qui signifie mémoire longue à court terme. Ce type de réseau de neurone possède une mémoire interne appelée cellule qui permet de maintenir un état aussi longtemps que nécessaire. La mémoire est simplement représentée par une valeur numérique. Nous allons voir un schéma qui va nous permettre de mieux comprendre les LSTM.

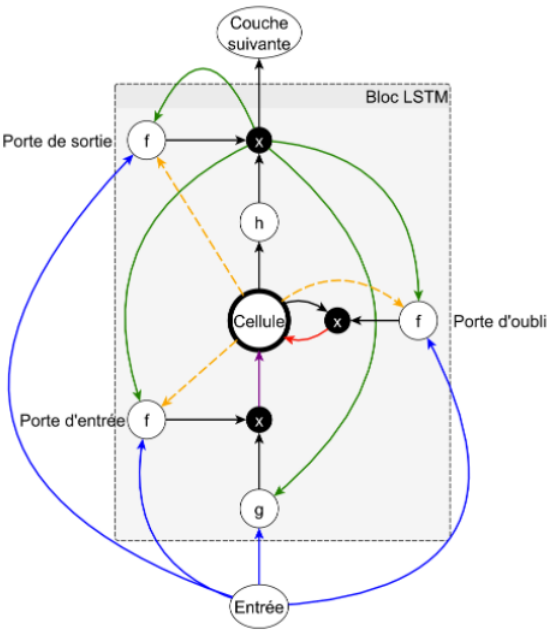

FIGURE 30 – Schéma d'un réseau de neurones LSTM
Source : Site web [4]

On constate que les LSTM sont constitués de 3 portes.

La porte d'entrée décide si l'entrée du réseau doit modifier le contenu de la cellule. En quelque sorte, des propositions d'ajout de nouvelles informations dans la mémoire sont faites et l'algorithme trie parmi les propositions d'ajout, lesquelles vont être réellement ajoutées à la mémoire, selon si l'information est utile ou non.

La porte d'oubli peut remettre à 0 le contenu de la cellule, si une information n'est plus utile et qu'il faut l'oublier.

La porte de sortie donne l'état de la cellule à l'instant t. Cela sert pour les neurones de la couche suivante.

Il existe une variante aux LSTM appelée GRU qui est plus ou moins la même chose mais sans la porte d'oubli. Il a été prouvé que les performances entre ces deux algorithmes sont pratiquement similaires mais GRU nécessite moins de paramètres donc tester une telle méthode pourrait être une bonne piste d'ouverture pour la suite.

Pour l'écriture du modèle, ce réseau est constitué de 4 couches LSTM et entre chaque couche LSTM, il y a une couche de dropout pour réduire le manque de généralisation du modèle. Pour finir le modèle contient une couche dense de sortie.

Par ailleurs, on fixe le learning rate à 0.001 qui est une valeur conseillée et il sera possible de modifier cette valeur si on voit que les performances ne sont pas suffisantes. Pour cela, j'ai ajouter un callback qui modifie automatiquement le learning rate après un certain nombre d'époques.

6.3 La combinaison de ces deux méthodes (CNN-LSTM).

L'algorithme qui va nous intéresser tout particulièrement dans le cadre du stage est un algorithme qui combine les deux méthodes vues ci-dessus. En effet, comme en témoignent les articles que j'ai décrit dans la partie étude bibliographique, on combine souvent LSTM et couche convolutionnelle pour augmenter les performances des algorithmes. Cela aide pour améliorer la précision de notre réseau. Le détail de l'architecture d'un tel réseau a déjà été vu plus haut dans la partie bibliographie, je ne vais donc pas trop réexpliquer tout ça. Ce qu'il faut principalement retenir est qu'un tel réseau combine les couches convolutives et les couches LSTM. Nous allons quand même revoir l'architecture d'un tel réseau, ici dans le cas de la détection du COVID-19.

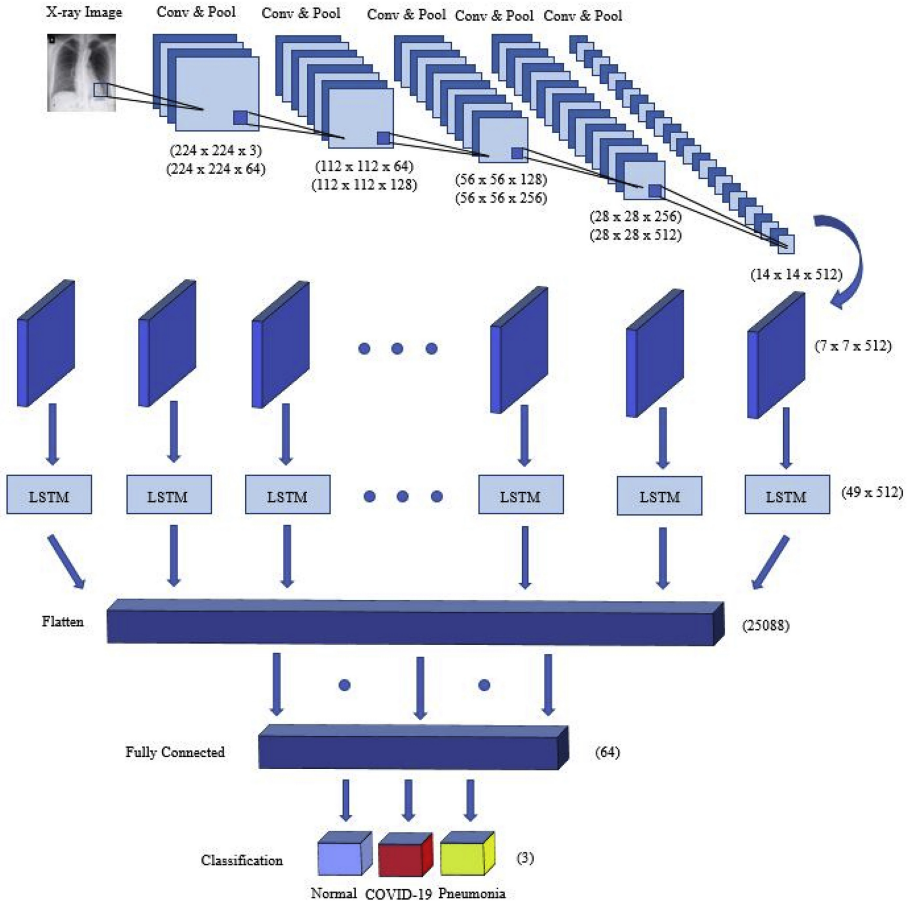

Figure 31 – Schéma d'un réseau de neurones CNN-LSTM
Source : Article [5]

Dans cet exemple, le CNN est utilisé pour extraire les caractéristiques complexes des images et le LSTM est utilisé comme classificateur. Le réseau hybride est composé de plusieurs couches convolutionnelles et d'une couche LSTM, avec entre d'autres couches dont je vous ai déjà parlé précédemment.

Je vais m'inspirer de cet article pour construire le réseau de neurones adapté à notre problème.

Après avoir implémenté les réseaux de neurones de type CNN et de type LSTM, ce qui nous intéresse vraiment est d'implémenter un algorithme qui mêle réseau de type convolutifs et réseau de type LSTM. Notre objectif est d'implémenter ce modèle et de voir s'il s'adapte bien à nos données et si ses performances sont meilleures que celles de chacune de 2 méthodes isolées.

J'ai testé différentes manières d'écrire le modèle en inversant certaines couches jusqu'à avoir un algorithme performant. Par ailleurs, il faut bien faire attention à changer la dimension des tenseurs car on a des couches convolutives qui prennent en entrée des tenseurs de dimension 4 comme expliqué plus haut. Mais, les couches LSTM prennent des entrées de dimension 1 donc il faut ajouter des couches de réduction de dimension. Voici le détail du modèle que j'ai implémenté.

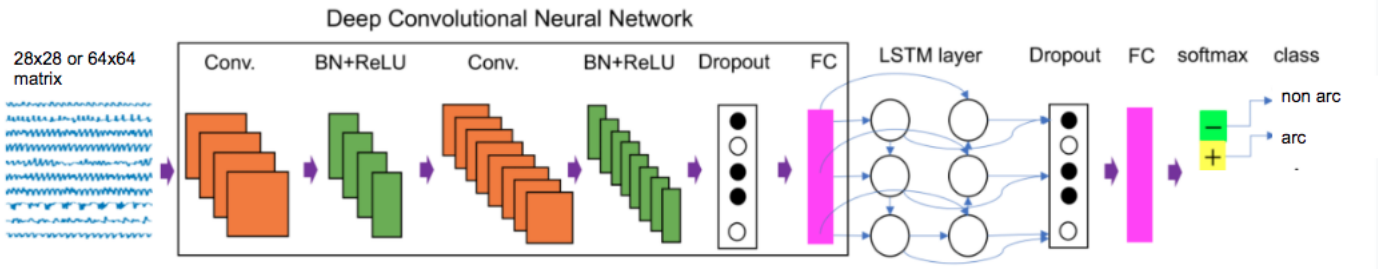

Figure 32 – Détail du modèle CNN-LSTM que j'ai implémenté
Source : Notebook personnel

On peut voir que le modèle débute par 2 couches de convolution suivies chacune d'une batch normalisation. Puis, il y a une couche de max-pooling et une couche de flatten qui servent à réduire la dimension des entrées. On poursuit avec une couche LSTM et enfin deux couches denses concluent le modèle.

6.4 Le principe de self-attention et son application à la détection d'arcs électriques.

6.4.1 Qu'est-ce que le principe de self-attention ?

Les mécanismes d'attention sont apparus assez récemment dans le domaine de l'intelligence artificielle et sont rapidement devenus omniprésents, initialement dans le domaine du langage et de la traduction. En effet, auparavant les LSTM étaient les architectures leaders dans le domaine. Les LSTM prennent en compte les interdépendances entre les mots et sont par conséquent fiables, mais le problème est qu'ils sont lents à entraîner et peu parallélisables. C'est pour cette raison que l'architecture de type transformer a été créée, elle conserve l'interdépendance des mots d'une séquence sans utiliser de réseau récurrent mais seulement en utilisant le mécanisme d'attention qui est au centre de son architecture.

L'idée du mécanisme d'attention est de déterminer à quel point deux éléments issus de deux séquences sont liés. On obtient alors une matrice d'attention qui nous indique à quel point ces éléments sont liés.

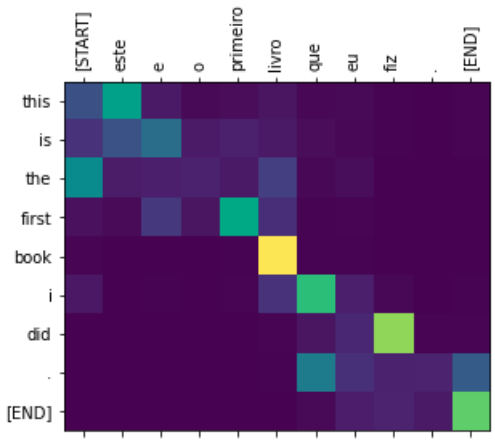

FIGURE 33 – Matrice d'attention entre 2 séquences
Source : Notebook étudié en cours

Dans cet exemple, on fait de la traduction. Plus une cellule est claire et plus les deux mots auxquels elle correspond sont liés. On constate que les mots ont un lien fort avec leur traduction littérale.

Maintenant que nous avons compris ce qu'était l'attention, définissons le principe de self-attention. Le self-attention correspond au mécanisme d'attention appliqué à une seule séquence et non pas entre deux séquences. La couche de self-attention détermine en effet l'interdépendance entre les éléments d'une même séquence afin de l'encoder de manière pertinente sous forme d'un vecteur. Ce vecteur d'attention est calculé en considérant 3 autres vecteurs :

- le vecteur de requête appelé q (query)
- le vecteur de clé appelé k (key)
- le vecteur de valeur appelé v (value)

On obtient ces vecteurs en multipliant l'encodage de la séquence d'entrée par les 3 matrices (qu'on appelle Q, K et V) qui sont entraînées pendant le processus d'entraînement du transformer.

Ces calculs sont faits en parallèle par plusieurs blocs d'attention différent, cette méthode est appelée "Multi-head Attention" et a pour but d'avoir plusieurs sous-espaces de représentation qui empêchent que la représentation soit totalement biaisée si une couche d'attention l'est.

6.4.2 L'architecture des transformers.

Un transformer est un modèle séquence à séquence (seq2seq) basé sur le mécanisme d'attention et non sur un réseau de neurones récurrent comme c'était le cas pour les modèles précédents. Son architecture est composée d'un encodeur et d'un décodeur. L'encodeur créé une représentation vectorielle d'une séquence de mots et le décodeur retourne quant à lui une séquence de mots à partir d'une représentation vectorielle. La partie encodage est constituée de 6 encodeurs montés les uns sur les autres et la partie décodage est constituée de 6 décodeurs montés les uns sur les autres et prenant chacun comme entrée la sortie du sixième encodeur.

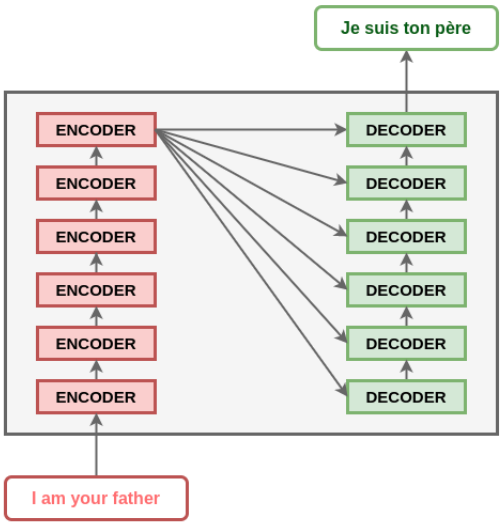

FIGURE 34 – Architecture d'un transformer
Source : `https://ledatascientist.com/a-la-decouverte-du-transformer/`

Les transformers sont très performants, ils sont donc rapidement devenus omniprésents. Dans notre cas, nous allons coder une architecture qui utilise le mécanisme d'attention en se basant sur le transformer mais nous n'allons pas coder un transformer dans sa totalité car en extrayant seulement une partie, cela semble suffisant pour avoir des bonnes performances.

Ainsi, pour coder mon modèle, j'ai écrit une fonction qui crée un bloc d'encodeur, ainsi on pourra faire appel à cette fonction plusieurs fois et donc avoir plusieurs blocs d'encodeurs. Le nombre de bloc d'encodeur sera choisi en paramètres et pourra être de 6 comme dans le cas du transformer ou différent. La partie décodeur ne sera pas implémentée car elle n'est pas nécessaire pour notre classification, elle est plus utile dans le cas de la traduction par exemple. J'ai ensuite rajouté un couche dense de sortie ainsi qu'une couche de pooling pour réduire la dimension du tenseur.

7 Les métriques de score.

7.1 L'accuracy et la loss.

L'accuracy est une métrique qui permet de connaître la proportion de bonnes prédictions par rapport à toutes les prédictions. Notons TP les vrais positifs, FP les faux positifs, FN les faux négatifs et TN les vrais négatifs. Pour comprendre ce que cela veut dire, dans notre cas si nous avons deux classes : début d'arc et pas d'arc. Si on dit que la classe début d'arc est la classe des positifs et la classe pas d'arc correspond aux négatifs, alors par exemple les faux positifs sont tous les éléments prédits positifs (donc prédits comme étant un arc) alors qu'ils ne l'étaient pas réellement (ce ne sont pas des arcs). On obtient cette formule pour l'accuracy

$$Accuracy = \frac{TN + TP}{TN + FN + FP + TP}$$

On peut résumer cette formule comme étant le nombre de bonnes prédictions divisé par le nombre total de prédictions. Cette métrique permet d'avoir une idée de la précision de notre algorithme mais son problème est qu'elle n'indique pas les forces et faiblesses de notre modèle.

La loss est quant à elle une valeur qui représente la somme des erreurs de notre modèle. Elle mesure la qualité (ou la faiblesse) de notre modèle. Si les erreurs sont nombreuses, la loss sera élevée, ce qui signifie que le modèle ne fait pas un bon travail. Dans le cas contraire, plus elle est faible, plus notre modèle fonctionne bien.

Pour calculer la loss, on utilise une fonction de perte ou de coût. Il existe plusieurs fonctions de coût différentes à utiliser. Chacune pénalise les erreurs de différentes manières. L'entropie croisée et l'erreur quadratique moyenne sont les plus couramment utilisées pour les problèmes de classification et de régression.

Cependant, le fait que la loss soit élevée ou faible n'est pas ce qui nous intéresse le plus. Pour que cela soit mieux interprétable, il faut tracer les valeurs de la loss dans le temps et nous pouvons alors voir si notre modèle apprend, et à quelle vitesse. En effet, en apprentissage profond, la fonction de loss est utilisée par le modèle pour apprendre. L'objectif du modèle est de minimiser la valeur de la loss. Pour ce faire, on utilise des techniques telles que la descente de gradient, qui modifie les paramètres du modèle en utilisant les informations du résultat de la loss.

Si la loss diminue dans l'ensemble d'apprentissage et oscille dans l'ensemble de validation, il est possible que le modèle soit surajusté. En d'autres termes, il peut être entrain de sur-apprendre à partir des exemples d'apprentissage.

D'une manière générale, si l'accuracy est élevée et la loss faible, le modèle ne commet des petites erreurs que sur certaines des données, ce qui serait le cas idéal.

7.2 La matrice de confusion.

La matrice de confusion est constituée des prédictions qui sont placées en colonne et des vraies classes qui sont placées en ligne. En reprenant les mêmes notations que précédemment, la matrice de confusion peut s'écrire

	$\hat{-}$	$\hat{+}$
$-$	TN	FP
$+$	FN	TP

Donc, par exemple l'intersection entre la première ligne et la première colonne indique le nombre de négatifs bien classés. Cet outil permet de mesurer la fiabilité de notre modèle d'une manière générale, il permet de voir si d'une manière globale notre algorithme est performant ou non mais il ne permet pas de faire de test de fiabilité précis ni de comparer différents algorithmes entre eux. On va alors l'utiliser essentiellement pour avoir une idée de la fiabilité de notre modèle.

7.3 La précision et le rappel.

Nous allons définir deux métriques de scores essentielles en classification, la précision et le rappel.

La précision est l'accuracy des prédictions positives, définie par

$$\text{precision} = \frac{TP}{TP + FP} = \frac{+ \cap \hat{+}}{\hat{+}}$$

Le rappel est le nombre d'instances positives correctement détectées, défini par

$$\text{rappel} = \frac{TP}{TP + FN} = \frac{+ \cap \hat{+}}{+}$$

Si la précision est proche de 1, la plupart des prédictions sont bonnes. C'est donc important d'avoir une bonne précision dans le cas de la détection de spams par exemple, en effet dans ce cas-là ce n'est pas très

grave si un spam est omis par l'algorithme, par contre quand on en détecte un on aimerait que la détection soit correcte.

Si le rappel est proche de 1, la plupart des positifs sont détectés. C'est donc important d'avoir un bon rappel dans le cas de la détection de cancer ou de maladie par exemple, car on ne peut pas se permettre d'omettre un cas.

7.4 Le score F1.

Le score F1 est une combinaison entre la précision et le rappel, c'est leur moyenne harmonique et il est défini par :

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{rappel}}}$$

Le modèle a un bon score F1 quand précision et rappel sont tous les deux grands. Le score F1 combine parfaitement la précision et le rappel donc c'est l'élément qui va le plus nous intéresser.

7.5 La courbe ROC.

La courbe ROC (Receiver Operating Characteristic) est une mesure de la performance d'un classificateur binaire, c'est-à-dire d'un système qui a pour objectif de catégoriser des éléments en deux groupes distincts sur la base d'une ou plusieurs des caractéristiques de chacun de ces éléments, elle est très utilisée pour observer l'effet du seuillage. Graphiquement, on la représente souvent sous la forme d'une courbe qui donne le taux de vrais positifs (nombre des positifs qui sont effectivement détectés, c'est le rappel) en fonction du taux de faux positifs (nombre des négatifs qui sont incorrectement détectés).

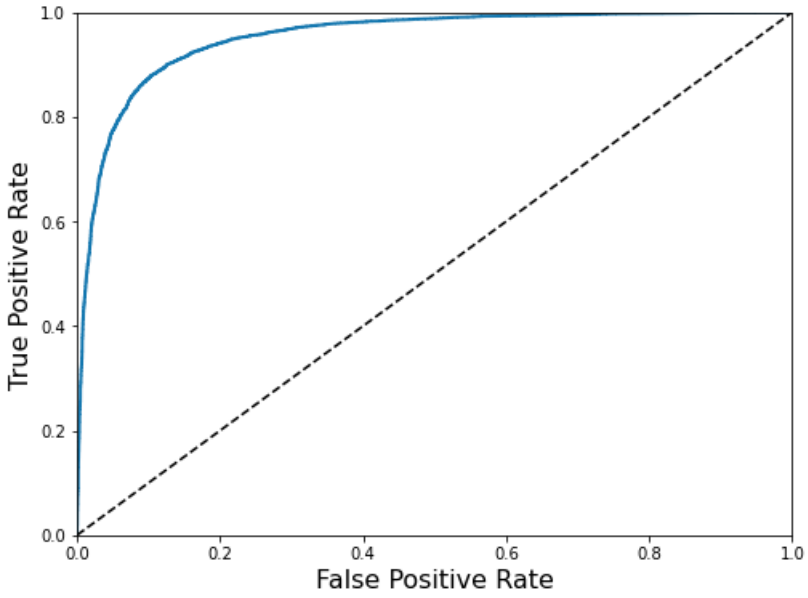

Figure 35 – Exemple de courbe ROC
Source : Notebook personnel

Pour comparer les performances entre plusieurs modèles à l'aide de leurs courbes ROC, il faut se baser sur l'aire sous la courbe (Area Under the Curve). Plus cette aire est grande et meilleur est le modèle. En effet,

au point $(0, 1)$, on a 100% de vrais positifs, c'est notre objectif, mais dans les faits on cherchera plutôt le meilleur équilibre.

Pour trouver le seuil idéal, il faut trouver le meilleur compromis entre le taux de vrais positifs et le taux de faux positifs qui correspond au point à la fois le plus proche de l'idéal (0,1) et à la fois le plus loin de la diagonale.

8 Tests de nos bases de données et comparaison des performances.

8.1 Notations pour les tests

Pour l'ensemble de nos tests, nous allons utiliser des notations afin de raccourcir certains termes. Tout d'abord pour les 4 bases de données, la notation Ci correspondra au cas i, donc par exemple C2 correspond au cas 2.

Ensuite, nous allons également donner une notation pour les différents algorithmes à tester, ces notations sont résumées dans le tableau ci-dessous :

Algorithme	**Notation**
CNN	A1
LSTM	A2
CNN+LSTM	A3
Attention	A4

Au niveau de la taille de la matrice de données, nous allons utiliser la notation Mi avec i la taille des segments, donc par exemple la base de données 128*128 sera notée M128.

Enfin, un dernier paramètre sera la fréquence d'échantillonnage, qui est fixée à $fs = 1MHz$ par l'oscilloscope. Nous allons commencer nos essais en gardant cette fréquence d'échantillonnage, puis nous passerons à $fs = 100kHz$.

Voici un tableau qui récapitule le nombre de segments de signaux pour chaque base de données :

	Cas 1	**Cas 2**	**Cas 2 (100kHz)**	**Cas 3**	**Cas 4**
M28	19460	12073	12073	19460	14767
M64	19506	12170	10545	19506	14813
M128	/	12028	6363	/	/

Table 2 – Détail du nombre de segments pour chaque base de données

8.2 Affichage de quelques matrices de confusion

Nous allons commencer par afficher les matrices de confusion pour certains cas afin d'avoir une première idée des résultats.

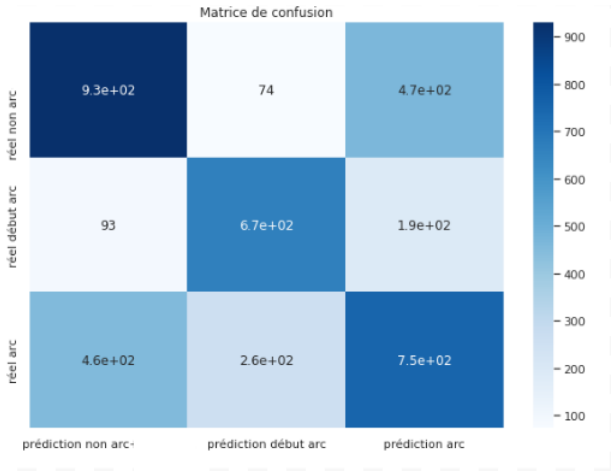

FIGURE 36 – Matrice de confusion dans le cas C1-A3-M28-fs=1MHz
Source : Notebook personnel

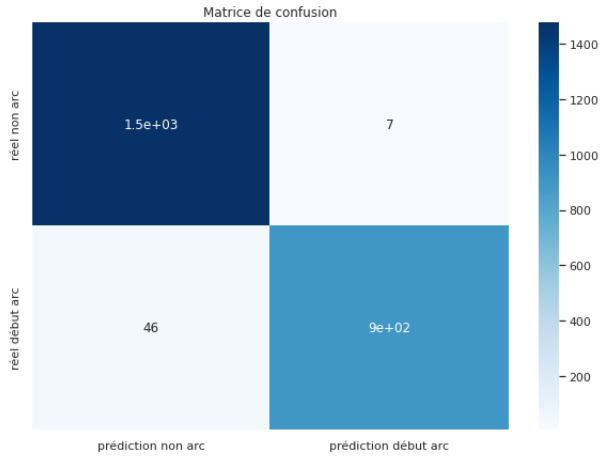

FIGURE 37 – Matrice de confusion dans le cas C2-A4-M64-fs=1MHz
Source : Notebook personnel

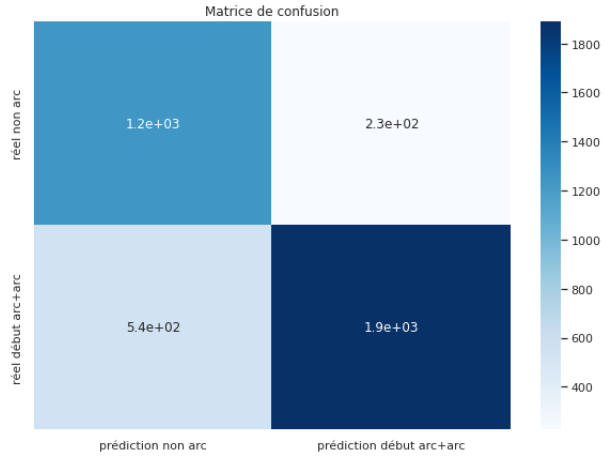

FIGURE 38 – Matrice de confusion dans le cas C3-A1-M64-fs=1MHz
Source : Notebook personnel

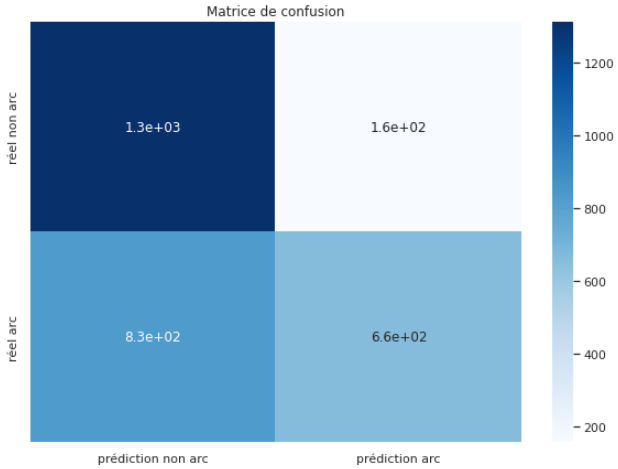

FIGURE 39 – Matrice de confusion dans le cas C4-A2-M64-fs=1MHz
Source : Notebook personnel

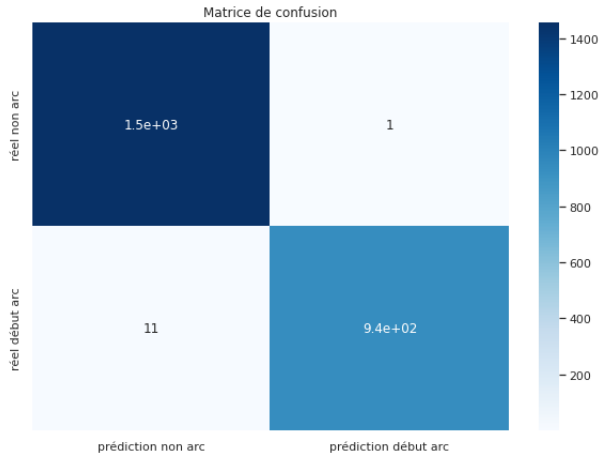

Figure 40 – Matrice de confusion dans le cas C2-A3-M128-fs=1MHz
Source : Notebook personnel

Une première tendance semble se dégager : le cas 2 semble être celui qui présente le moins d'erreur, cependant il est difficile de conclure à ce stade car nous n'avons affiché que 5 matrices de confusion. Or, il y a vraiment beaucoup de tests à faire entre les différents modèles, tailles de matrice et bases de données. Nous allons par contre d'ores et déjà écarter le cas 1, qui semble vraiment trop peu précis. L'algorithme a du mal à faire la distinction entre la classe non arc et la classe arc, il semble se focaliser uniquement sur la transition et ce peu importe le modèle donc nous allons éliminer ce cas, d'autant plus que nous ne pouvons pas calculer le score F1 étant donné qu'il y a 3 classes.

8.3 Comparaison des performances entre les bases de données

Nous allons maintenant comparer les scores F1 entre nos différents cas, en laissant pour l'instant la fréquence d'échantillonnage fixée à $fs = 1MHz$.

	C2	**C2**	**C2**	**C3**	**C3**	**C4**	**C4**
A1	0.87	0.98	0.99	0.78	0.83	0.60	0.70
A2	0.83	0.94	0.99	0.77	0.76	0.67	0.67
A3	0.86	0.98	1.00	0.75	0.80	0.66	0.66
A4	0.83	0.97	0.97	0.81	0.83	0.69	0.64
	M28	**M64**	**M128**	**M28**	**M64**	**M28**	**M64**

Table 3 – Comparaison du score F1 pour nos différents algorithmes et nos différentes bases de données

Nous allons commencer par comparer les 3 cas entre eux pour voir si nous pouvons choisir quelle est la meilleure base de données et travailler uniquement avec celle-ci. Les cas 3 et 4 semblent assez peu convaincants, leurs scores F1 sont compris entre 0.6 et 0.83 selon les modèles et tailles de matrices, ce qui est trop peu précis dans le cas d'un avion par exemple, on ne peut pas se permettre d'avoir autant d'erreurs (par contre, dans d'autres cas, un score F1 qui atteint 0.7 est déjà considéré comme un bon résultat). En revanche, le cas 2 obtient d'excellents résultats avec des scores F1 allant jusqu'à 1.00 et des taux d'erreurs

très faibles dans les matrices de confusion. C'est alors ce cas que nous allons retenir et avec lequel nous allons terminer notre étude.

Concentrons nous maintenant uniquement sur la partie gauche du tableau (celle où sont présentés les résultats pour le cas 2). Nous constatons de manière assez flagrante que plus la taille de la matrice augmente et plus les résultats s'améliorent. Les résultats avec la base de données 28*28 sont en effet assez mauvais, nous pouvons donc en conclure qu'il est préférable de travailler avec des segments de données de taille 68*68 ou encore mieux de taille 128*128.

Au niveau du modèle, nous constatons que l'algorithme LSTM semble être le moins bon, même si ses performances restent tout de même très bonnes. Ensuite le modèle d'attention semble être le deuxième moins bon, bien que ses performances sont très élevées. Les deux modèles qui semblent les meilleurs sont le CNN et le CNN+LSTM. S'il fallait en choisir un seul, il serait préférable de garder le CNN+LSTM car il est possède plus de couches que le CNN et il est plus complexe donc il est plus probable qu'il soit le meilleur pour détecter des signaux plus compliqués. Par ailleurs, le modèle d'attention n'est pas à écarter totalement car ses performances sont presque similaires à celles de nos meilleurs modèles alors que j'ai fait les tests avec seulement 4 blocs d'encodeurs. En augmentant ce nombre, les performances pourraient encore s'améliorer mais le soucis est que ce modèle est extrêmement long à entraîner donc je ne l'ai pas fait avec plus de blocs.

8.4 Sous-échantillonnage de la base de données

Nous avons ensuite décidé de sous-échantillonner la base de données. En effet, le but à long terme du projet est d'implémenter nos algorithmes sur des systèmes électroniques embarquées qui ont des fréquences d'échantillonnage plus faibles que celles des oscilloscopes. Cela permet de réduire le coût pour un industriel par exemple, étant donné que les oscilloscopes sont des matériaux assez chers. Nous allons passer de $1MHz$ à $100kHz$. Il faut donc que l'on prenne 1 point sur 10 dans nos bases de données créées précédemment. L'enjeu est de déterminer si en enlevant autant d'informations, nos modèles arrivent toujours à être performants. Observons les résultats uniquement sur le cas 2 que nous avons sélectionné auparavant.

	C2	**C2**	**C2**
A1	0.99	1.00	1.00
A2	0.98	1.00	1.00
A3	0.99	1.00	1.00
A4	0.98	0.98	0.98
	M28	**M64**	**M128**

Table 4 – Comparaison du score F1 pour nos différents algorithmes et nos différentes bases de données

Nous constatons que les résultats sont meilleurs avec le sous-échantillonnage, ce qui est de prime abord surprenant mais qui est explicable par le fait que le modèle recevait peut être trop d'informations qu'il avait du mal à distinguer et en sous-échantillonnant, on réduit l'information. Nous constatons qu'on atteint des scores F1 maximaux dès la taille 64*64 donc il n'est pas nécessaire de faire de segments plus grands, d'autant plus que le code met plus de temps à s'exécuter quand la taille est trop grande. Au niveau des résultats, les résultats sont similaires entre les algorithmes à part le mécanisme d'attention qui est légèrement moins performant avec le sous-échantillonnage.

Pour conclure sur la comparaison entre les algorithmes, la base de données qui a le meilleur équilibre entre le niveau de performance et le temps de calcul semble être la base de données pour le cas 2, avec des

segments 64*64, une fréquence d'échantillonnage de $100kHz$ et entraînée sur l'algorithme CNN+LSTM. Les résultats sont en tout cas très concluants, et pour confirmer ceci, nous allons afficher les métriques de score que nous avons défini plus haut, excepté le score F1 que nous avons déjà donné.

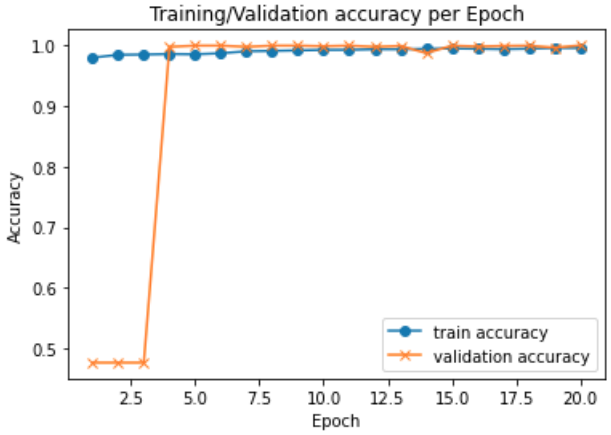

FIGURE 41 – Accuracy en fonction du nombre d'époque dans le cas C2-A3-M64-fs=100kHz
Source : Notebook personnel

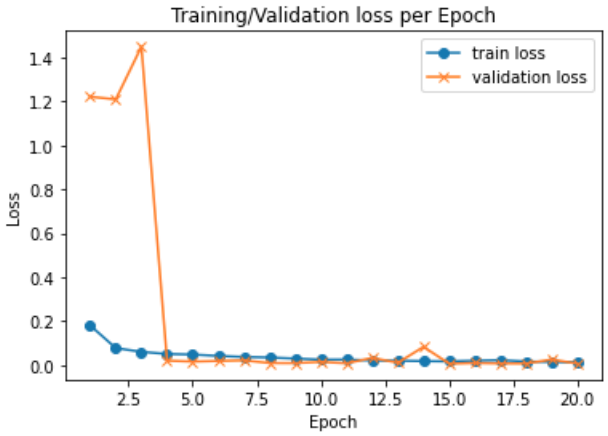

FIGURE 42 – Loss en fonction du nombre d'époque dans le cas C2-A3-M64-fs=100kHz
Source : Notebook personnel

Nous pouvons constater que l'accuracy augmente jusqu'à atteindre une précision maximale tandis que la loss diminue jusqu'à atteindre 0, cela traduit que le modèle ne commet pratiquement aucune erreur et est très fiable, nous allons voir le nombre exact en observant la matrice de confusion.

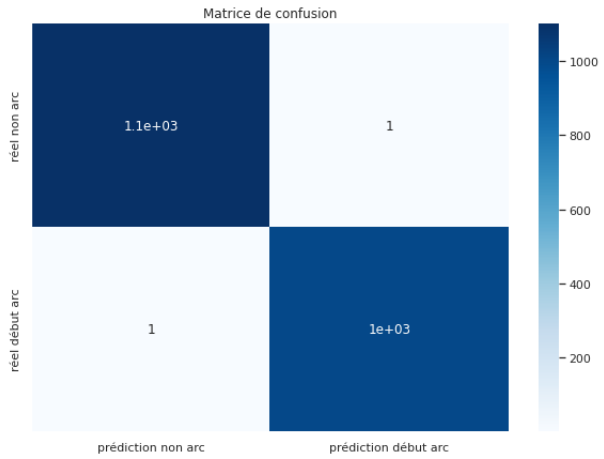

FIGURE 43 – Matrice de confusion dans le cas C2-A3-M64-fs=100kHz
Source : Notebook personnel

Nous constatons que l'algorithme se trompe seulement 2 fois, une fois pour chaque classe donc il est extrêmement précis au vu du nombre de segments qu'on lui a donné en entrée (10545 pour cette base de données). Affichons maintenant ces deux erreurs pour les analyser.

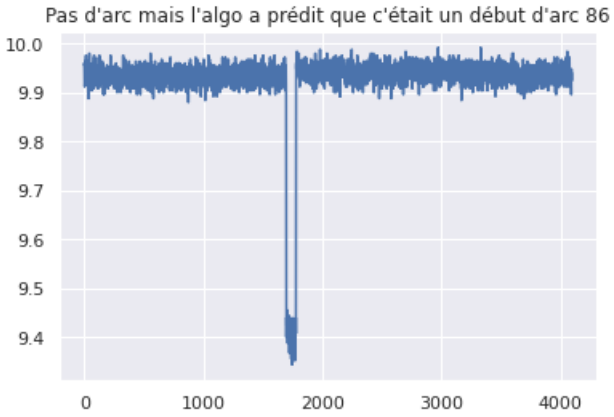

FIGURE 44 – Première erreur de l'algorithme dans le cas C2-A3-M64-fs=100kHz
Source : Notebook personnel

Ce segment a été interprété comme un début d'arc par l'algorithme alors qu'en réalité il n'y a pas d'arc et seulement un pic qui est un défaut, peut être une étincelle. Ces pics ont été lissés lors de la labellisation mais ils sont enregistrés dans le dataset et le modèle semble les comprendre plutôt bien vu qu'il n'y a qu'une seule erreur qui comporte un tel pic.

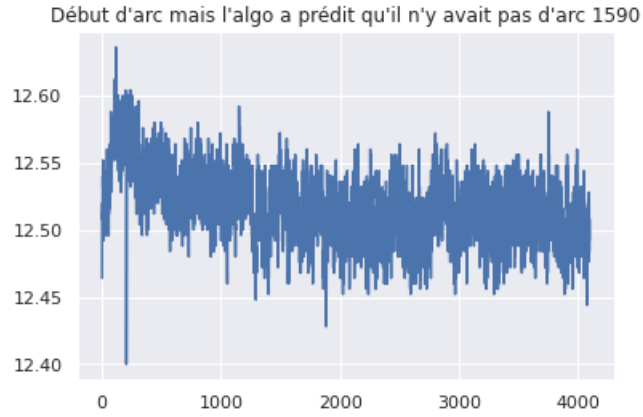

FIGURE 45 – Deuxième erreur de l'algorithme dans le cas C2-A3-M64-fs=100kHz
Source : Notebook personnel

Ce segment correspond à un début d'arc mais la transition semble très légère et le modèle ne l'a pas détectée et l'a classifiée comme s'il n'y avait pas d'arc.

Ainsi, dans l'ensemble nous constatons que le modèle commet très peu d'erreurs, et le fait d'afficher les erreurs nous permettra de faire des ajustements dans nos codes par la suite en voyant ce qui ne va pas.

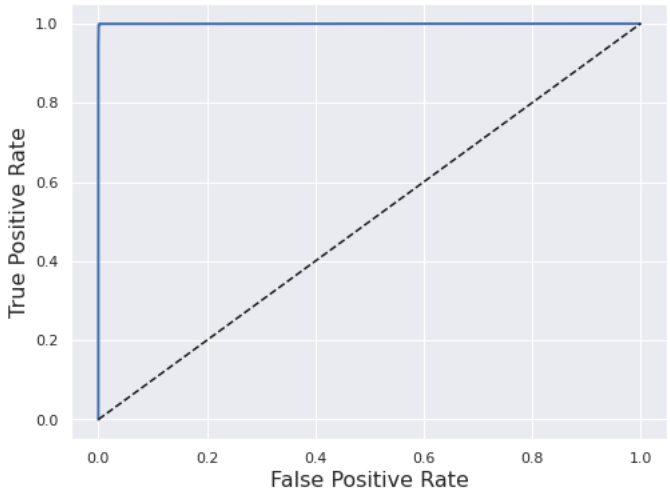

FIGURE 46 – Courbe ROC dans le cas C2-A3-M64-fs=100kHz
Source : Notebook personnel

A première vue, on constate que la courbe ROC semble avoir une aire sous la courbe très élevée, il y a pratiquement un angle droit. J'ai calculé directement cette aire avec Python, et j'obtiens 0.9997.

Nous avons donc toutes nos métriques de score qui présentent des résultats excellents.

Conclusion et poursuites envisageables.

Pour conclure, j'ai lors du stage travaillé sur différents aspects d'un projet d'intelligence artificielle, allant de l'écriture de modèles de classification à la création de nouvelles bases de données. Ce stage m'a donc, de par ses tâches multiples, permis de gagner en autonomie et en connaissances.

J'ai commencé par analyser les signaux en ma possession, les trier, les comprendre, les observer afin d'avoir une première idée du travail à effectuer. J'ai ensuite codé différents algorithmes d'intelligence artificielle en commençant par les plus basiques (convolutions) jusqu'aux plus sophistiqués (transformer). Afin d'avoir un grand panel de données pour tester mes algorithmes, j'ai créé différentes bases de données en faisant varier leur taille, leur labellisation ou encore leur fréquence d'échantillonnage. Tous ces tests m'ont permis de faire des comparaisons entre les différents algorithmes et les différentes bases de données. Pour cela, j'ai utilisé des métriques de score qui m'ont permis de comparer de différentes façons les performances.

Au cours du stage, tout ne s'est pas toujours passé comme prévu et certains résultats étaient moins bons qu'attendus tandis que d'autres résultats sont excellents. En tout cas, certains algorithmes atteignent la précision maximale donc nous pouvons nous en réjouir. Nous avons conclu que la meilleure base de données qui combine le mieux performances et faible temps de calcul est la base de données pour le cas 2, avec des segments 64*64, une fréquence d'échantillonnage de $100kHz$ et entraînée sur l'algorithme CNN+LSTM.

J'ai rempli tous les objectifs qui m'avaient été confiés mais le projet n'est pas terminé pour autant et une autre personne pourra continuer mon travail en rajoutant des signaux plus complexes dans la base de données afin de voir si les résultats sont toujours bons. Il faudra également implanter le modèle dans un système électronique pour le tester en condition réelle et embarquée.

Références

[1] J.-B. Humbert, *Étude et détection des défauts d'arcs électriques dans un réseau électrique aéronautique 270V HVDC*. Theses, Université de Lorraine, June 2018.

[2] Y. Liu, Y.-X. Huang, X. Zhang, W. Qi, J. Guo, Y. Hu, L. Zhang, and H. Su, "Deep c-lstm neural network for epileptic seizure and tumor detection using high-dimension eeg signals," *IEEE Access*, vol. 8, pp. 37495–37504, 2020.

[3] T. Liu, J. Bao, J. Wang, and Y. Zhang, "A hybrid cnn–lstm algorithm for online defect recognition of co2 welding," *Sensors*, vol. 18, no. 12, 2018.

[4] R. Herault and C. Chatelain, "Découvrez les cellules à mémoire interne : les LSTM." `https://openclassrooms.com/fr/courses/5801891-initiez-vous-au-deep-learning/5814656-decouvrez-les-cellules-a-memoire-interne-les-lstm`, 2021.

[5] M. Z. Islam, M. M. Islam, and A. Asraf, "A combined deep cnn-lstm network for the detection of novel coronavirus (covid-19) using x-ray images," *Informatics in Medicine Unlocked*, vol. 20, p. 100412, 2020.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[7] S. V. Thiruloga, V. K. Kukkala, and S. Pasricha, "Tenet : Temporal cnn with attention for anomaly detection in automotive cyber-physical systems," 2021.

[8] J.-B. Cordonnier, A. Loukas, and M. Jaggi, "On the relationship between self-attention and convolutional layers," 2019.

[9] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.