



Reconnaissance de la forme unidimensionnelle d'un objet en développant un sonar et et réseau de neurones

PHY473O : MODAL d'Electronique “Son et Parole”

Jia Jean LAW, Alexandre MULLER

Promotion X2020

18 mai 2022

Remerciements

Durant les dix séances au cours de ces quatres derniers mois, nous avons pu acquérir de nouvelles connaissances et mener un projet du début à la fin. Pour cela, nous souhaitons remercier les personnes qui nous ont aidé et accompagné durant ce MODAL.

Nos remerciements vont d'abord à nos professeurs, à M. Louis-Joseph, pour la clarté de ses explications à propos de l'électronique analogique, à Mme Beaujean pour ses conseils sur la façon de comprendre les FPGA et à M. Chakar pour les idées et propositions qu'il nous a données afin d'améliorer notre modèle de réseaux de neurones.

Nous remercions aussi M. Chandèze pour son aide constante et pour le tracé des pistes de la carte électronique.

Introduction

La technologie du SONAR (Sound Navigation And Ranging) est au cœur de la compréhension des environnements où la vision est inefficace, comme les fonds marins ou encore pendant la nuit. On s'intéresse ici au SONAR actif, c'est-à-dire un appareil qui émet une onde sonore en direction d'une cible et qui analyse l'écho reçu afin d'obtenir des informations sur cette cible. Les applications des SONAR sont multiples : on peut accéder à la distance, au gisement, à la taille, ou encore à la forme de l'objet. Ainsi, c'est un domaine en constant développement, que ce soit dans l'industrie ou dans la recherche. Fort des connaissances acquises en électronique et dans le domaine de l'intelligence artificielle durant ce MODAL, nous avons cherché à combiner ces deux champs de connaissances pour créer un sonar capable de reconnaître la forme d'objets.

Ainsi, nous avons réussi à développer la carte électronique d'un SONAR. Cette carte électronique se base sur l'utilisation de NE555 pour effectuer la synthèse d'une onde ultrasonore à 25 kHz puis sur un traitement analogique de l'écho reçu pour déterminer le temps mis par l'onde sonore à parcourir la distance jusqu'à la cible. Le seuil de détection du SONAR est de l'ordre de la dizaine de centimètres et va jusqu'à 50 cm, sa précision est de l'ordre du centimètre. Nous avons également réussi à utiliser le SONAR actif HC-SR04 connecté à une carte Arduino Nano 33 BLE Sense pour reconnaître la forme unidimensionnelle d'un objet, en prenant des mesures de 20 objets différents, tous de forme cylindrique ou rectangulaire, et développant un modèle d'un réseau de neurones convolutifs qui fait une classification binaire entre les données d'un objet cylindrique et d'un pavé droit. Ce modèle a un taux de précision de 99,5% sur les images générées numériquement dans notre jeu de test; sur les 10 données d'objets qui ont été gardées et préalablement filtrées et traitées, il a un taux de précision d'entre 70% et 80%.

Le présent document s'intéresse d'abord à l'état de l'art des recherches sur ce sujet, puis à la réalisation de la carte électronique et enfin à celle du réseau de neurones. Ces deux axes ont été étudiés séparément, la carte électronique ayant été réalisée par A. Muller et le réseau de neurones par J. Law, avant d'être testés ensemble.

1. Reconnaissance d'objet avec SONAR et réseau de neurones

Les SONARs sont couramment utilisés pour la localisation d'objets : dans la plupart des technologies de SONAR, le système récolte des données sur les caractéristiques des échos des cibles afin de produire une image acoustique de l'environnement, qui peut ensuite être analysée. L'avantage d'utiliser des ondes acoustiques pour détecter des objets par rapport aux techniques qui se basent sur les ondes électromagnétiques (e.g. dans un LIDAR) est que les ondes sonores, ayant une longueur d'onde plus grande, peuvent pénétrer les conditions de visibilité sont très basses (par exemple, la longueur d'onde d'une onde ultrasonore à 20 kHz est au moins 10^3 fois plus grande que celle de la lumière visible). L'utilisation d'un SONAR pour détecter les formes de l'objet est ainsi pertinent dans les conditions marines (la pêche, les conflits militaires) ou les conditions de basse luminosité (comme dans l'écholocation) [1].

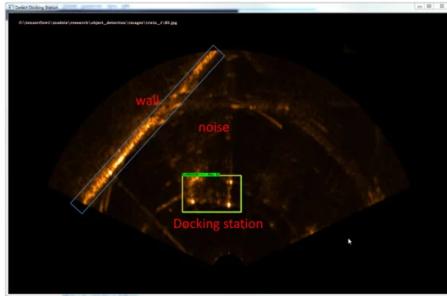


Figure 1 : Un exemple d'une image acoustique avec les objets sur l'image identifiés [2]

Cependant, la détection automatique des objets dans les images acoustiques a deux limitations principales : l'image est souvent bruitée et a une faible résolution [3]. L'intelligence artificielle joue ainsi souvent un rôle pour classifier, détecter et segmenter les images. Les réseaux de neurones convolutifs ont une meilleure précision que d'autres méthodes de comparaison à un modèle standard (*template matching*) dans la détection d'objets et peuvent mieux détecter des parties d'image qui sont facilement négligées. Grâce aux techniques de deep learning, l'état de l'art dans le traitement des images a beaucoup progressé ces derniers temps, par exemple dans la reconnaissance automatique des cibles du SONAR [4]. Les algorithmes de détection d'objet les plus connus incluent AlexNet [5] et DetNAS [6].

En effet, presque tous les systèmes de détection d'objet modernes utilisent les réseaux de neurones convolutifs [7], car ces derniers réussissent à convertir une image en un vecteur de caractéristique (*feature vecteur*), que la plupart des algorithmes d'apprentissage ont du mal à faire.

Pour donner un exemple de l'utilisation de CNN dans le domaine de l'imagerie par SONAR, nous allons nous baser sur un article écrit par Valdenegro-Toro en 2018 [8]. Il a réussi

à développer et valider expérimentalement un modèle de CNN pour détecter des objets et les classer dans 10 catégories différentes à l'aide d'images audio fournies par un *forward-looking SONAR* (FLS). La précision de son modèle est de 99,2%, ce qui dépasse l'état de l'art de la reconnaissance d'objets dans les images FLS (entre 92.4% et 97.6%). Le CNN peut non seulement reconnaître des objets complexes avec une facilité qui ressemble à celle de l'homme, mais de plus, il a besoin de moins de paramètres que le *template matching* et est 40% plus rapide. Il peut donc être implémenté en temps réel.

Il a utilisé un ARIS Explorer 3000 pour obtenir au total 2627 images de FLS dans un réservoir d'eau contenant des débris, et l'architecture de son CNN est la suivante : conv32-5, MP2, conv32-5, MP2, FC64, FC10 (conv_kn-w fait référence à une module de convolution qui a n filtres de taille wxw, MP-s est le max-Pooling avec la taille de subsampling (s,s), FC n est une couche qui est complètement connectée avec n neurones de sortie). Son modèle est optimisé avec les paramètres suivants :

1	Batch size	32
2	Nombre d'epochs	10
3	Rapport d'entraînement - test	0,7 : 0,3
4	Taux d'apprentissage du départ	0,1
5	Optimizer	Adam
6	Fonction d'activation	RELU
7	Fonction de classification	SoftMax
8	Fonction de perte	Multinomial cross-entropy loss

1. *Batch size* : nombre d'images utilisées dans une itération du réseau. Les *batch size* en puissance de deux sont utilisées pour des raisons d'efficacité liées au GPU.
2. Nombre d'epochs : nombre de fois que le réseau utilise toutes les images dans le jeu d'entraînement. Le nombre d'epochs utilisé dans cet article est plutôt faible (des centaines ou de milliers nécessaires souvent).
3. Rapport d'entraînement-test : entre les 2627 images, 70% (1838 images) ont été utilisées pour l'entraînement et 30% (789 images) pour le test post-entraînement. Il n'y a pas de jeu de validation (*validation set*) dans cette étude.
4. Taux d'apprentissage (*learning rate*) : fait référence à l'ampleur du changement aux poids dans le modèle afin de minimiser la fonction de perte à chaque itération.
5. *Optimizer* : L'algorithme d'optimisation Adam se base sur deux variations de la descente de gradient stochastique (Adaptive Gradient Algorithm et Root Mean Square Propagation) et utilise des estimations des moments de premier et deuxième ordre du gradient. Il est empiriquement l'un des meilleurs *optimizers* dans l'état de l'art aujourd'hui.
6. Les fonctions d'activation, de classification et de perte utilisées dans cette étude correspondent à celles qui produisent empiriquement les meilleurs résultats. Elles ont les équations suivantes :
 - a. Fonction d'activation ReLU :

$$f(x) = x^+ = \max(0, x)$$

b. Fonction de classification SoftMax :

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

c. Fonction de perte categorical cross entropy :

$$\text{Loss} = - \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

Cet article nous montre l'avancement de la recherche dans ce domaine, mais notamment tous les paramètres sur lesquels il est possible de jouer pour optimiser un réseau de neurones.

2. Réalisation d'un sonar à l'aide d'une carte électronique

2.1. Conception de la carte

2.1.1 Inspiration

La première partie du projet a consisté en la création d'un module de sonar actif totalement contrôlé par une carte Arduino.

Ce projet s'inspire du fonctionnement du module HC-SR04 (Figure 2), un télémètre ultrasonore construit pour fonctionner avec Arduino, disponible dans le commerce. Il permet de mesurer des distances à l'aide d'ondes sonores. Ayant déjà pu utiliser ce module, nous avons cherché à comprendre son fonctionnement et à reconstituer l'électronique du traitement du signal pour avoir un produit fonctionnant de manière similaire.

Le module possède 4 broches : VCC, GND, TRIG et ECHO (Pulse sur la Figure 2). Il est alimenté entre 0V et +5V sur les broches VCC et GND par l'Arduino. Pour le faire fonctionner, la carte maintient un état logique haut durant 10µs sur la broche TRIG afin d'activer le lancement d'une salve ultrasonore. Sur la broche ECHO, en mesurant la durée d'état haut, on accède au temps de trajet des ondes ultrasonores et donc à la distance à l'objet, selon la formule, avec la distance du capteur à la cible et la célérité du son dans les conditions de l'expérience :

$$d = \frac{c\Delta t}{2}$$

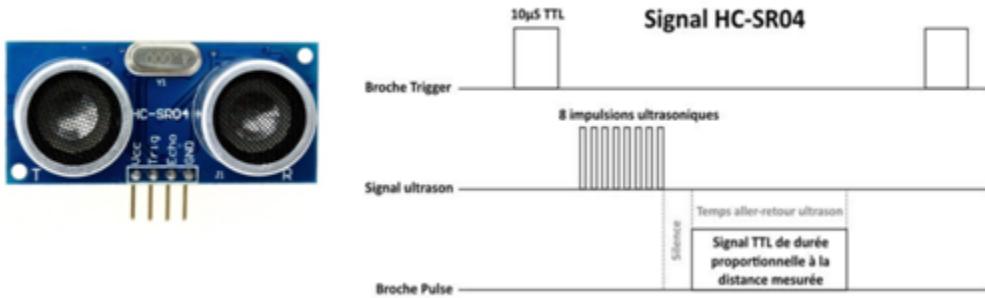


Figure 2: Module HC SR-04 et son fonctionnement

Le fonctionnement interne du capteur est en partie géré par un microcontrôleur, ainsi le traitement du signal n'est pas accessible. Le parti pris ici a été d'effectuer la synthèse du signal à l'aide d'un circuit intégré, le NE 555, puis tout le traitement du signal de manière analogique et de faire remonter les données à l'Arduino à l'aide portes logiques.

Enfin, une contrainte du projet est que l'alimentation ne se fait que par l'Arduino. Ainsi tous les composants actifs seront alimentés entre 0 et 5 V. Cela ne pose pas de problèmes pour les NE et pour les portes logiques, par contre, pour les amplificateurs opérationnels, cela nécessite de : 1) bien les choisir, on va utiliser des TL71 et des OP27 et 2) de placer le signal sur lequel on va travailler à une valeur moyenne non nulle, sinon le filtrage et l'amplification risquent de l'écrêter.

2.1.2. Dimensionnement

La carte électronique va donc être composée de deux parties principales, une ligne d'émission et une ligne de réception.

Pour la partie émission, il s'agit de créer une salve d'impulsions à 25 kHz et de les convertir en ondes sonores. Pour choisir la taille des salves, il faut considérer la borne inférieure des mesures que l'on souhaite avoir. Ici, on choisit d'effectuer ses mesures autour de 10 cm pour ne pas avoir trop de perte de puissance de l'onde sonore, tout en conservant une durée d'émission suffisante. Cela permet de pallier à l'éventuelle perte de signal due à des réflexions qui ne renvoient pas l'onde incidente dans la direction d'où elle vient. Or, un aller-retour vers un objet à 10 cm à la vitesse du son prend environ 600 µs. C'est le temps maximal d'émission, car la carte Arduino ne peut pas contrôler la durée d'émission et mesurer le temps de parcours en même temps. La période d'un signal à 25 kHz est de 40 µs. On prend donc un signal carré de période 120 µs (8300 kHz) pour pouvoir moduler les ultrasons et créer une salve. Cette salve peut être réitérée entre deux et cinq fois, les tests montreront quel est le nombre de répétitions adéquat (Figure 3).

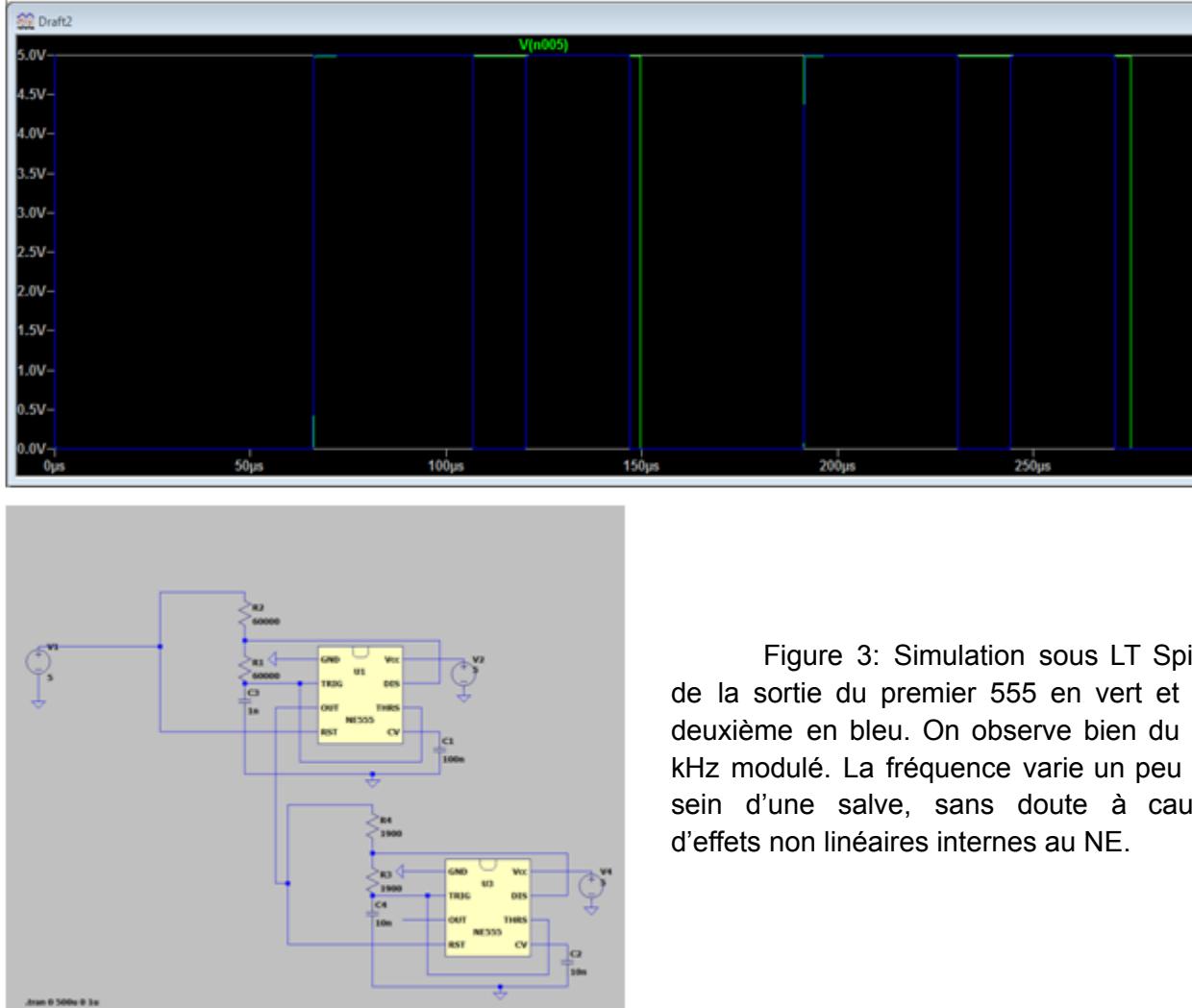


Figure 3: Simulation sous LT Spice de la sortie du premier 555 en vert et du deuxième en bleu. On observe bien du 25 kHz modulé. La fréquence varie un peu au sein d'une salve, sans doute à cause d'effets non linéaires internes au NE.

2.1.3 Principe de fonctionnement

On s'intéresse d'abord à la ligne d'émission (Figure 5).

Pour synthétiser un signal carré à une fréquence fixée, on utilise le NE 555 en montage astable (Figure 4). En plaçant la broche RESET à l'état haut, on obtient le signal voulu sur la broche OUT. Pour fixer la fréquence, on utilise la formule (d'après la datasheet) :

$$f = \frac{1.44}{(R_a + 2R_b)C}$$

Avec R_a , R_b en Ohm, C en Farad et f en Hertz

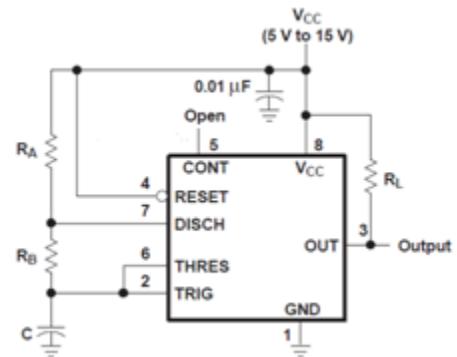


Figure 4: Montage astable du NE 555

Ainsi pour créer une salve, on utilise un premier montage astable à 1kHz qui va piloter le RESET du second à 25 kHz. On va ensuite placer un ampli en mode suiveur pour que l'entrée sur l'émetteur à ultrason se fasse sans perte de puissance.

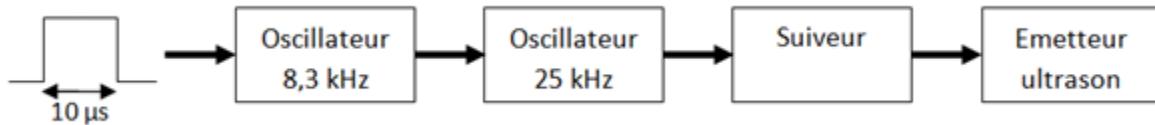


Figure 5: Principe de fonctionnement de la ligne d'émission

On établit ensuite la ligne de réception (Figure 6 et 11). L'idée est de ne pas prendre en compte un signal arrivant sur la ligne de réception si celui-ci arrive trop tôt, car il peut provenir d'un couplage électromagnétique entre les pistes de la carte et dès lors, il ne correspond pas à l'aller-retour des ultrasons. Ainsi, en attendant un délai raisonnable après la fin de l'émission (2 µs), l'Arduino va mettre une bascule RS à l'état logique Haut (par la broche CONTROL) et la sortie (sur la broche ECHO) de cette bascule ne passera à l'état Bas que lorsque le signal ultrasonore sera reçu. En utilisant la fonction `pulseIn` de l'Arduino, on peut mesurer la durée de cet état Haut et accéder au temps d'aller retour. Ainsi, à la différence du HC SR-04 et pour simplifier la conception, cette carte possède 5 broches : VCC, GND, TRIG, CONTROL, ECHO.

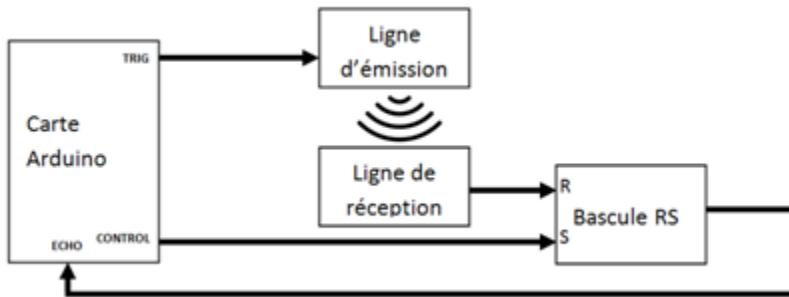


Figure 6 : Principe de fonctionnement du sonar composé de la carte Arduino et de la carte électronique

Tout d'abord, le signal est reçu par le transducteur à ultrasons, à la sortie duquel on place un ampli en montage suiveur (Figure 7). Pour être sûr que le signal qui va être traité est bien celui qui a été émis par la carte, on va filtrer le signal reçu puis l'amplifier.

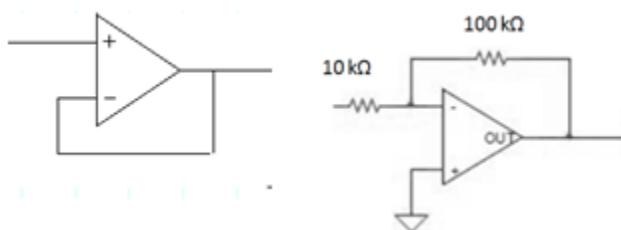
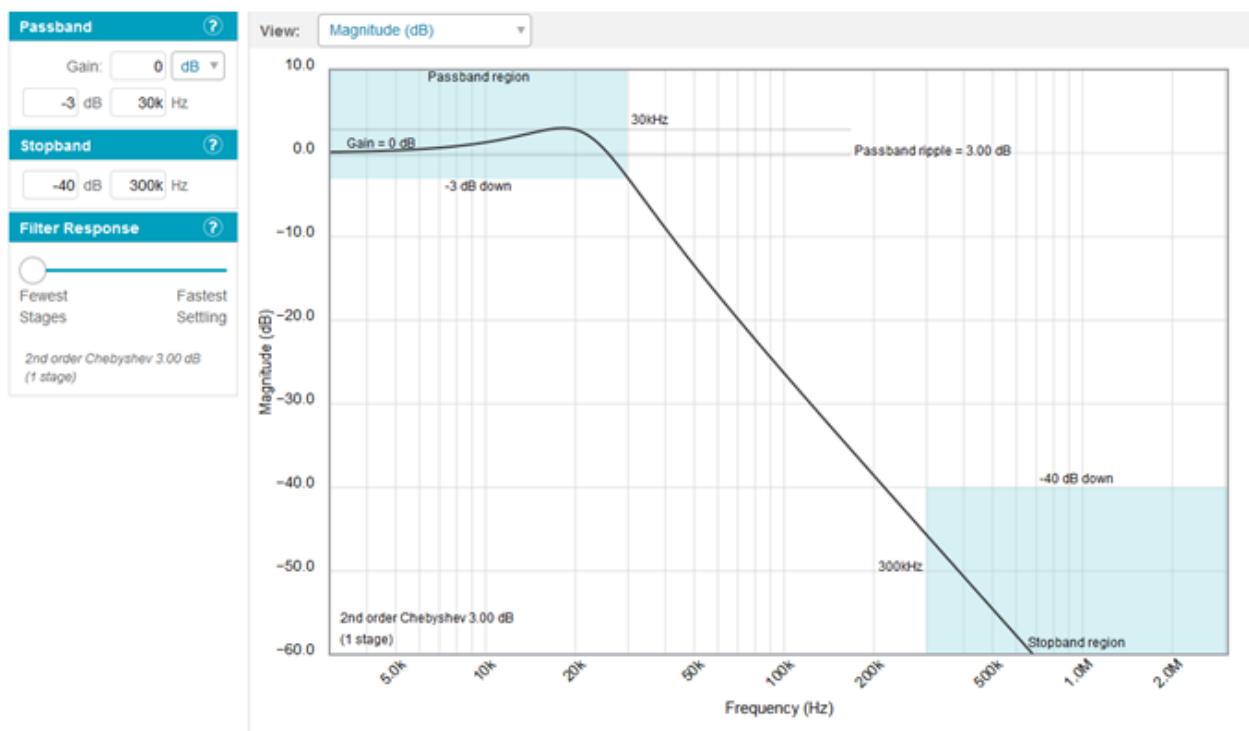


Figure 7: Ampli en mode suiveur (à g.) et en mode inverseur (à d.)

Pour le filtrage, on utilise un filtre passe-bas car : 1) Les tests menés au préalable avec les récepteurs ultrasons montrent que les seules fréquences susceptibles de se propager dans le circuit à la sortie du récepteur sont supérieures à 25 kHz et 2) un filtre passe-bande avec amplificateur opérationnel nécessite deux étages (deux amplis).

On utilise le logiciel en ligne Analog Filter Wizard qui permet de calculer les valeurs de composants pour des filtres de type de Sallen et Key en ayant spécifié les valeurs de la fréquence de coupure et de la fréquence d'arrêt (Figure 8). Ce type de filtre est particulièrement adapté pour le traitement d'ondes sonores car la présence de composants actifs permet d'éviter d'utiliser des inductances qui fonctionnent mal à basse fréquence.



Stage A
2nd order
Low-Pass
Sallen Key

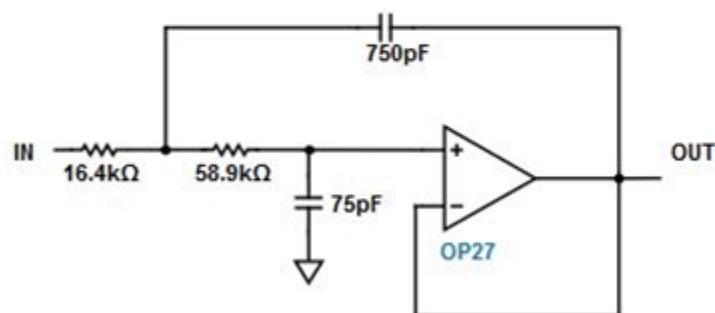


Figure 8: Spécifications et composants pour la réalisation d'un filtre de Chebyshev de type Sallen et Key.

Pour pouvoir comparer le signal avec un signal de référence on va d'abord l'amplifier : les mesures à l'oscilloscope de tension en sortie d'un récepteur ultrason montrent que pour un signal de 25 kHz émis avec une tension de 5V, on capte un signal à 500mV à 10 cm et à 100 mV à 30 cm. On utilise donc un amplificateur en montage inverseur avec un gain de 11 (Figure 7) : on utilise $R_2 = 100 \text{ k}\Omega$ et $R_1 = 10 \text{ k}\Omega$.

Ensuite, afin de pouvoir détecter un signal stable et éviter que des pics de tensions aléatoires déclenchent la mesure du temps, on va utiliser un détecteur d'enveloppe avec une constante de temps assez élevée, pour être sûr que la sortie de la carte mesurée par l'Arduino soit lisible. On utilise une diode 1N4148 (Figure 9). On calcule la constante de temps de manière à ce qu'elle soit grande devant la période du signal (40 μs) et petite devant la période de l'enveloppe (1 ms). On peut prendre la moyenne géométrique soit 200 μs .

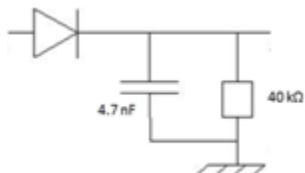


Figure 9 : Montage détecteur d'enveloppe

Enfin, on utilise un ampli en mode comparateur pour comparer la valeur de la tension de la crête avec une valeur de tension réglable, adaptée grâce à des tests. Ainsi si le 25 kHz arrive sur la ligne, il est amplifié, passe le comparateur et peut déclencher l'Arduino. Si c'est un autre signal, rien ne se passe. La suite logique est d'utiliser une bascule RS pour que le signal arrivant puisse être converti en signal logique.

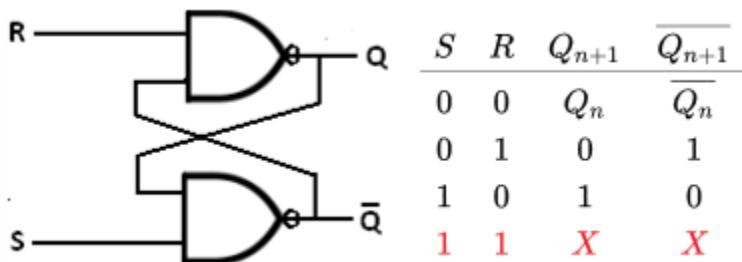


Figure 10 : Bascule RS réalisée à l'aide de portes NAND et table de vérité associée. L'indice $n+1$ désigne le changement de la sortie par rapport à l'instant précédent n si les valeurs de R et de S changent

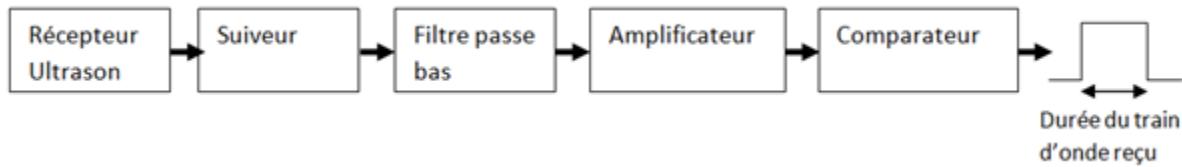


Figure 11: principe de fonctionnement de la ligne de réception

2.2 Réalisation de la carte électronique

Pour réaliser la carte électronique, nous avons utilisé le logiciel Eagle, qui nous permet de placer les composants électroniques puis d'effectuer le routage en vue de la réalisation des pistes de la carte.

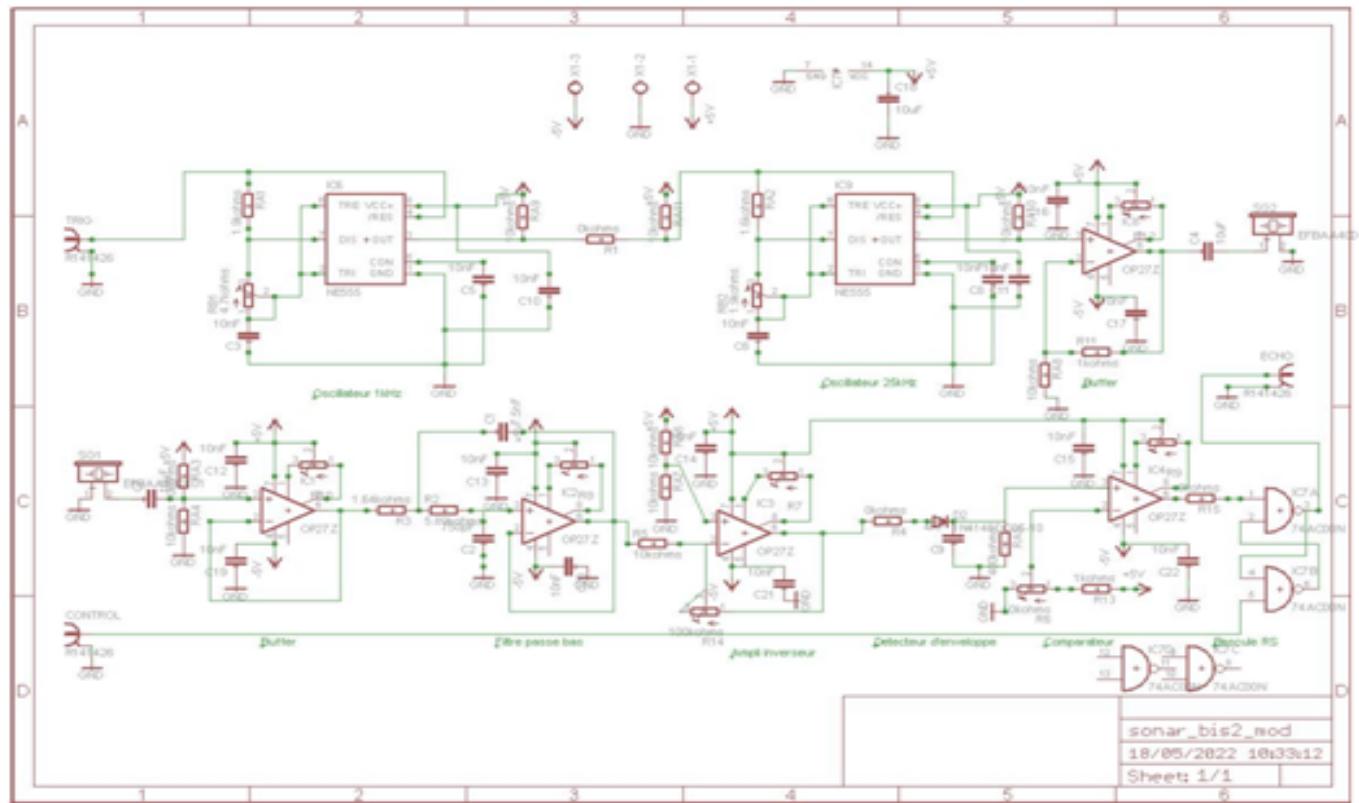


Figure 12: schéma électrique de la carte du SONAR sous Eagle. La version détaillée est en annexe avec le routage des pistes

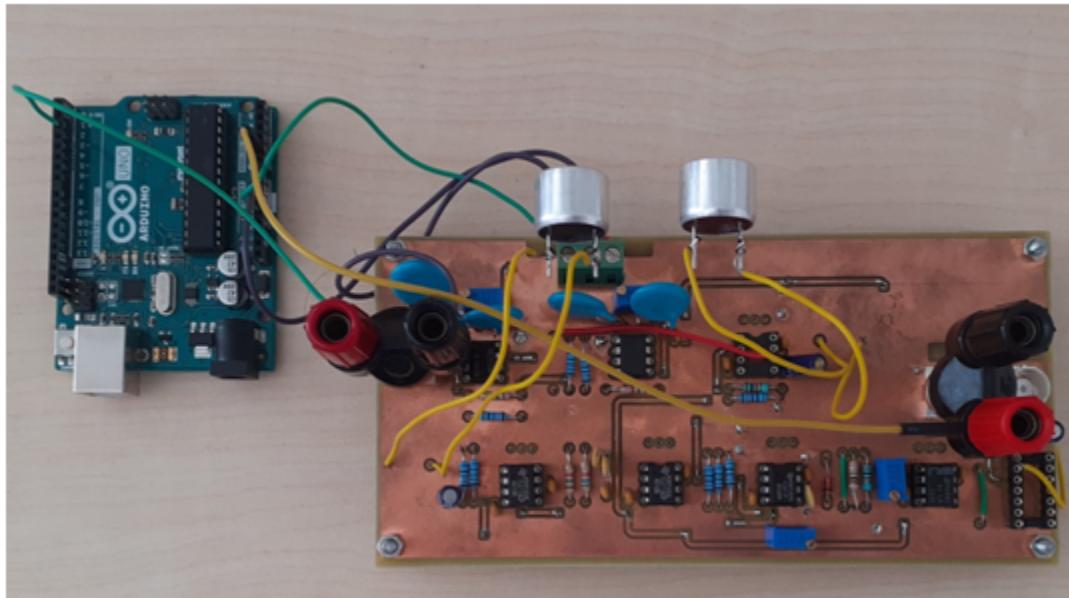


Figure 13: Carte électronique dans sa version finale branchée à l'Arduino

On a ensuite effectué des tests à chacune des étapes afin de vérifier le bon fonctionnement des étages et des composants, mais aussi de régler les potentiomètres (Figures 14 et 15).

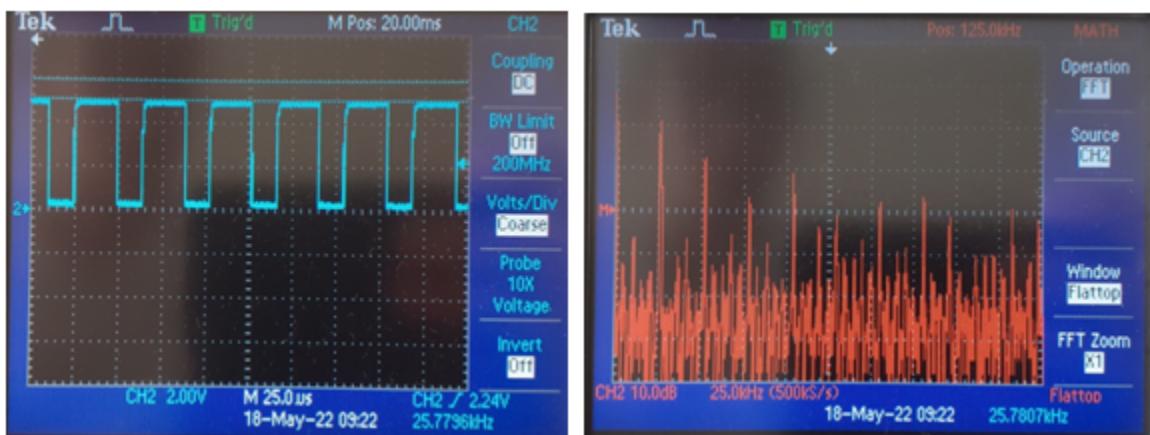


Figure 14: signal émis par le NE 555 et spectre associé. On observe le fondamental à 25 kHz, mais aussi les harmoniques multiples

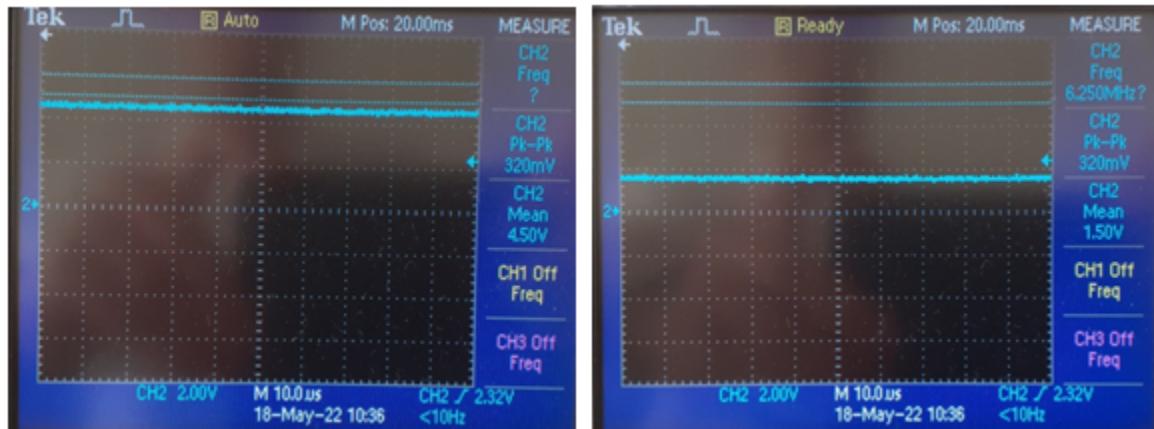


Figure 15 : Sortie du comparateur lorsque le signal à 25 kHz est détecté sur la ligne d'émission (à g.) et lorsque rien n'est détecté (à d.). On observe une tension de déchet : la sortie du comparateur n'est pas nulle lorsque rien n'est détecté.

L'étape suivante de la bascule RS n'a pas fonctionné sans doute à cause du composant. On va donc shunter ce composant et brancher directement la sortie du comparateur sur l'Arduino. En effet, les entrées analogiques de l'Arduino lui permettront de discriminer les résultats. Ainsi, le fonctionnement change un peu : un chronomètre est lancé au moment de l'émission et il est arrêté au moment de la réception, quand l'Arduino détecte une tension d'entrée supérieure à 3V. On accède ainsi à la durée de l'aller retour.

2.3 Tests du SONAR

Les mesures montrent que le signal est trop faible pour un objet à plus de 50 cm. De plus, l'utilisation de la fonction `micros()` pour mesurer le temps limite la précision de la mesure à 4 μ s. Cela est toutefois largement acceptable car correspond à 1 mm à la vitesse du son. Ce SONAR est donc utilisable pour la détection de formes comme on va le voir dans la partie suivante. On trouvera le code du SONAR en Figure 16.

```
sonar_final

/* Constantes pour les broches */
const byte TRIGGER_PIN = 2; // Broche TRIGGER
const byte ECHO_PIN = A1;    // Broche ECHO
unsigned long t1 = 0;
unsigned long t2 = 0;

void setup() {
    /* Initialise le port série */
    Serial.begin(9600);

    /* Initialise les broches */
    pinMode(TRIGGER_PIN, OUTPUT);
    digitalWrite(TRIGGER_PIN, LOW); // La broche TRIGGER doit être à LOW au repos
    pinMode(ECHO_PIN, INPUT);
}

void loop() {
    t1 = micros();
    digitalWrite(TRIGGER_PIN, HIGH);
    delayMicroseconds(200);
    digitalWrite(TRIGGER_PIN, LOW);
    delayMicroseconds(100);

    while (analogRead(ECHO_PIN) < 3) {
    }
    t2 = micros();
    float distance = (t2 - t1)*340/2000000;
    Serial.println(t2 - t1);
    delay(500);
}
```

Figure 16 : code du SONAR

3. Implémentation d'un système de reconnaissance de forme à l'aide de réseaux de neurones

3.1 Récolte des données par le SONAR

3.1.1 Conception du montage

Les réseaux neurones convolutifs pourraient révolutionner l'analyse des images acoustiques. Nous avons ainsi voulu réaliser un tel projet, l'application du Deep Learning à la reconnaissance de la forme d'un objet à partir des données récoltées par un SONAR. Pour ce faire, nous avons utilisé une carte Arduino Nano 33 BLE Sense, connecté à un SONAR actif HC-SR04 qui pour chaque mesure envoie huit impulsions d'ultrason de fréquence de 40 kHz pour mesurer le temps que met une onde sonore à parcourir l'aller retour entre le SONAR et la cible. Ce SONAR, qui mesure des distances entre 2cm et 400cm, est de taille 45 x 20 x 15mm, a une précision de 3mm et un angle de mesure de 15° [9].

Dans le cadre du MODAL et en prenant en compte les contraintes sur la précision de notre SONAR, nous avons réalisé une simplification quant à la reconnaissance de la forme d'un objet en considérant seulement deux types d'objet : les cylindres et les pavés droits. Notre but était ainsi de développer un modèle de réseau neuronal convolutif qui peut reconnaître un objet en tant que cylindre ou pavé droit. Pour ce faire, nous avons tout d'abord fait un étalonnage de notre SONAR pour déterminer de façon plus précise sa précision et ses limites. D'après nos résultats (Annexe A), le SONAR est incapable de mesurer une distance à moins de 10 cm, mais est assez précis entre 10 cm et 25 cm (incertitude maximale de 0,3 cm). Nous nous sommes ainsi toujours placés dans cet intervalle de mesure pour le positionnement de l'objet cible.

Nous avons ensuite utilisé le protocole suivant pour obtenir les données :

1. Placer le SONAR au côté gauche de l'objet, qui correspond à l'alignement du SONAR au marquage "0.0 cm" sur la règle
2. Déplacement manuel du SONAR à une vitesse de 0,25 cm par seconde (par un métronome de 120 bpm). Nous faisons le choix de déplacer le SONAR et non pas l'objet pour que les frottements solides soient constants pour toutes les expériences. Le SONAR est aussi collé sur des plaques de verre afin de minimiser les frottements associés.
3. Le déplacement s'arrête lorsque le SONAR atteigne le marquage "40.0cm" sur la règle

4. Un code Python que nous avons écrit transforme ces données en un graphique noir et blanc de $640 * 400$ pixels qui pourra ensuite être exploité par les réseaux de neurones

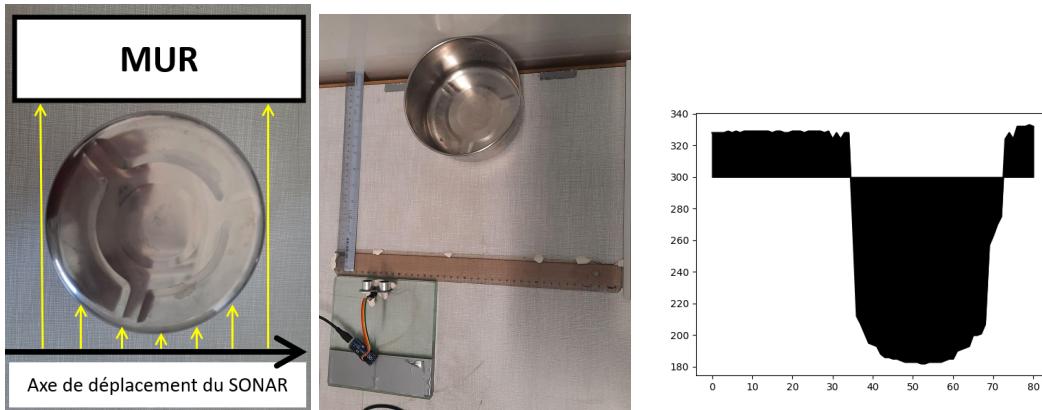


Figure 16 : Le montage pour obtenir les données d'un cylindre (gauche et centre) et les données que nous avons obtenues pour ce cylindre (droite)

3.1.2 Les données récoltées et leurs limites

Nous avons tout d'abord fait 10 premières mesures pour nous assurer du bon fonctionnement du montage : notamment, nous avons pris plusieurs mesures sans objet. D'après les données récoltées, lorsqu'il n'y a pas d'objets, les distances mesurées (qui devraient théoriquement être constantes) peuvent avoir des fluctuations qui vont jusqu'à 5 mm. Nous attribuons ces fluctuations aux échos des ondes sonores avec l'environnement. En effet l'alignement du SONAR n'étant pas continuellement constant (incertitude de type A, d'origine humaine), les ondes émises peuvent se réfléchir sur le plan de travail, et effectuer un trajet plus court que si elles se réfléchissaient sur le mur. Dans l'autre cas extrême, si le SONAR vise une direction où un objet est à plus de 4m, il ne détecte rien et renvoie 0 mm comme distance.

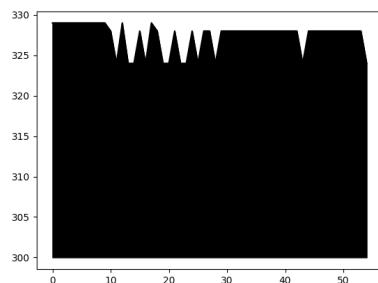


Figure 17 : Une mesure sans objet a une incertitude de 5 mm

Après cet étalonnage, nous avons au total récolté des données sur 20 objets différents, 14 formes cylindriques et 6 pavés droits (ces images se trouvent dans l'Annexe B). Ces objets diffèrent par leur forme (cylindrique ou pavé droit), taille (hauteur, longueur, diamètre) et matériau (plastique, métal ou carton).

En effet, nous n'avons pas réussi à trouver des objets de même matériau. Néanmoins, la différence du matériau de l'objet a un impact limité sur les mesures effectuées par notre SONAR, comme l'explique une étude faite par Tarulescu et al. en 2014 [10] sur l'impact du matériau lors de la détection par ondes ultrasonores. L'erreur associée à la mesure est entre 0,3% et 1,4% pour les objets métalliques, en carton, plastique et en caoutchouc. D'autant plus que pour tous nos objets, le matériau utilisé est homogène sur tout l'objet. Ainsi la seule différence que le matériau apportera à nos mesures c'est que l'objet apparaîtrait jusqu'à 2% plus grand ou plus petit. Autrement dit, la forme de l'objet mesuré ne devrait pas être impactée par son matériau.

Entre les 20 mesures, les mesures pour 10 objets sont extrêmement bruitées, avec un nombre non négligeable de distances mesurées étant jusqu'à trois fois plus loin que le mur. Le SONAR détecte parfois aussi une distance de "0 mm" : ceci correspondrait aux distances trop lointaines pour être détectées, que nous trouvons expérimentalement à être des distances au-delà de 1000 mm. Ces fluctuations extrêmes restent même après avoir refait les mesures une ou deux fois sur le même objet, et malgré le fait que nous avons essayé de réduire la présence des échos supplémentaires en enlevant le plus d'objets possibles du montage.

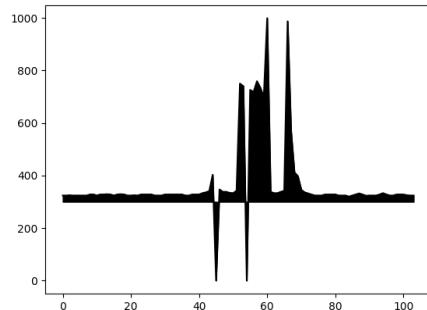


Figure 18 : Une donnée extrêmement bruitée dans laquelle toutes les distances mesurées sont plus lointaines que le mur

Pour ces 10 signaux très bruités, nous avons remarqué un lien entre la petite taille de l'objet et la quantité de bruit sur l'objet. En effet, ce phénomène semble ne concerner que les objets de diamètre en dessous de 15 cm. Ceci pourrait s'expliquer par le fait que les objets de petit diamètre ont une ellipticité très élevée: c'est ainsi que les réflexions des ondes ultrasons sur ces surfaces s'éloignent trop du SONAR et ne sont pas repérées dans le 15 degré d'angle de mesure. Quant aux objets de pavé droit, nous remarquons aussi que le bruit est d'autant plus élevé que la surface n'est pas lisse. C'est ainsi que nous établissons l'hypothèse que les distances de mesure aberrantes se produisent lorsque les échos de l'onde ultrasonore ne sont pas repérés par le SONAR, du aux caractéristiques de la surface.

Nous reconnaissons également les facteurs limitants suivants comme des possibles raisons derrière le bruit qui se trouve dans toutes les mesures :

1. L'utilisation de seulement un capteur par rapport à une dizaine dans d'autres études similaires [11]. La précision latérale de notre SONAR n'est pas notée dans les documentations et donc pas prise en compte.
2. L'incertitude de mesure liée à l'irrégularité du déplacement : il existe une incertitude d'environ 0,25 cm par déplacement puisqu'il est fait à la main, en prenant en compte l'erreur de parallaxe et les mauvaises estimations de distance.
3. Manque de standardisation de la hauteur de l'objet qui pourrait influencer les échos reçus par le SONAR.

C'est ainsi que, dans cette partie de récolte des données, nous n'avons pu retenir au total que 10 données (6 cylindres et 4 pavés droits) pour la suite du projet. Ces données proviennent des cylindres de diamètre de plus de 15 cm, ainsi que les pavés droits qui sont suffisamment lisses. Toutefois, ces signaux restent toujours assez bruité et nous avons dû faire un traitement de ces données en deux étapes :

1. Dans un premier temps, nous avons éliminé toutes les données qui ont une valeur supérieure à 300 mm (valeurs impossibles).
2. Nous avons, dans un deuxième temps, appliqué un filtre gaussien d'écart type 2 à ces données afin de lisser les contours des graphes.

Les résultats après chaque étape du traitement des données se trouvent dans l'Annexe C. Nous avons utilisé les 10 images à l'issue de l'étape 1 et aussi les 10 images à l'issue des étapes 1 et 2 comme des images tests pour évaluer la précision de notre réseau de neurones.

3.2 Développement du réseau de neurones convolutif

3.2.1 Génération numérique du jeu d'entraînement

Prenant en compte la grande quantité de données nécessaires pour entraîner notre modèle de réseau de neurones, nous avons décidé de générer nos jeux d'entraînement, de validation et de test numériquement.

Pour faire ressembler ces jeux à nos données réelles, nous avons utilisé l'équation de l'ellipse $(x/a)^2 + (y/b)^2 = 1$ pour générer des points qui correspondent à la mesure d'un cylindre (que nous allons par la suite appeler " cercle"), puis l'équation de $x^4 + y^4 = r^4$ (que nous allons appeler "rectangle courbé") pour modéliser les données d'un pavé droit, comme dans la figure suivante :

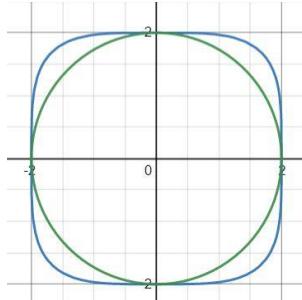


Figure 19 : Equation de $x^4 + y^4 = r^4$ (en bleu) comparé à celle d'un cercle

Au fur et à mesure du développement de notre modèle de réseau de neurones, nous avons modifié les paramètres du bruit, le nombre de points et de l'équation mathématique afin de modéliser au mieux possible les images SONAR. C'est ainsi que nous avons généré au total 4 bases de données, appelées *dataset 1 à 4* (les détails sur les 4 bases de données se trouvent dans l'Annexe D). Voici les choix finaux de nos paramètres (dans le *dataset 4*) :

1. Rayon de nos objets : entre 3 cm et 10 cm sur une échelle de 25cm, donc nous prenons a , r et b des variables aléatoires uniformes sur [1,4] sur une échelle de 10.
2. Bruit : variable aléatoire uniforme. Le bruit est très élevé pour les pavé droits par rapport aux cylindres. C'est ainsi que pour les rectangles courbés, il est à valeurs dans [0,1] et est amplifié par un facteur r . Pour les ellipses, il est à valeurs dans [0,1, 0,3].
3. Nombre de points par image (N) = 100 qui modélise bien le bruit dans nos images.
4. Nombre total d'images : 2400 (1600 pour l'entraînement, 400 pour la validation, 400 pour le test), avec 1200 cercles et 1200 rectangles courbés

3.2.2 Tentative de classification des données par la méthode des moindres carrés

Il est intéressant de vérifier dans un premier temps si la classification cylindre-pavé droit pourrait s'établir avec une méthode moins coûteuse que les réseaux de neurones convolutifs, et si oui, d'évaluer sa précision. En effet, l'utilisation d'un réseau de neurones est très coûteuse en termes de puissance de calcul, nécessitant un GPU. D'autre part, l'établissement d'un modèle mathématique qui puisse classifier les objets pourrait aussi servir en tant qu'un niveau de référence (*baseline*) auquel nous pouvons comparer notre modèle de réseau de neurones.

Pour ce faire, nous avons tout d'abord appliqué un filtre sur les données afin d'enlever le bruit présent sur les images. Nous avons empiriquement trouvé que le filtre de Savitzky-Golay au degré 9 produisait les meilleurs résultats (par rapport ce filtre aux autres degrés et le filtre gaussien). Un code Python [12] qui implémente la méthode des moindres carrés pour un cercle est ensuite appliqué aux données, et ce code nous permet d'évaluer à quel point nos données semblent correspondre à l'équation d'un cercle ($x^2 + y^2 = R^2$). Nous avons commencé cette

analyse par rapport à la base de données *dataset 2*, dans laquelle les images “cercles” n’étaient pas encore modélisées par l’équation de l’ellipse, mais par celle d’un cercle.

Le code prend en entrée un ensemble de N points $\{p_1, p_2, \dots, p_n\}$ avec $p_i = (x_i, y_i)$, et renvoie des estimation des coordonnées de son centre (x_E, y_E) , son rayon R_E , ainsi qu’une fonction de résidu qui représente l’écart entre les données et l’équation du cercle :

$$\text{Residu} = \sum (R_i - R_E)^2 \text{ pour } R_i = \sqrt{(x_i - x_E)^2 + (y_i - y_E)^2}$$

Nous avons ensuite tracé trois diagrammes de dispersion : 1. le résidu de chaque image en fonction de son bruit (gauche) 2. Le résidu en fonction du rayon calculé par la méthode des moindres carrés (centre) 3. Le rayon calculé en fonction du rayon réel (droit). Sur tous les diagrammes, les points en jaune font partie des données du “cercle” et les points en violet font partie des données du “rectangle courbé”.

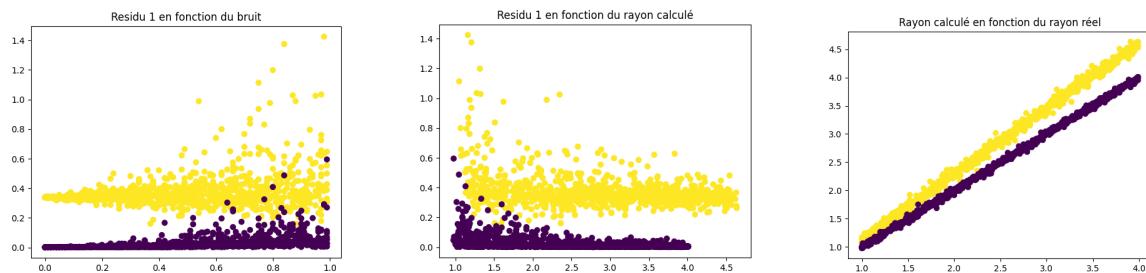


Figure 20 : Les trois diagrammes de dispersion pour $N = 100$ nous permettent de distinguer les cercles et les rectangles courbés

Cependant, nous ne pouvions pas utiliser ces diagrammes de dispersion directement pour analyser nos données réelles car chaque image du *dataset 2* a été générée avec un total de 100 points, tandis que dans les données réelles, chaque image a un nombre de points différent après filtrage (il varie entre 20 et 80). Nous avons ainsi essayé d’appliquer la même démarche au *dataset 3*, dans lequel chaque image a été tracée pour $n = 20$ points.

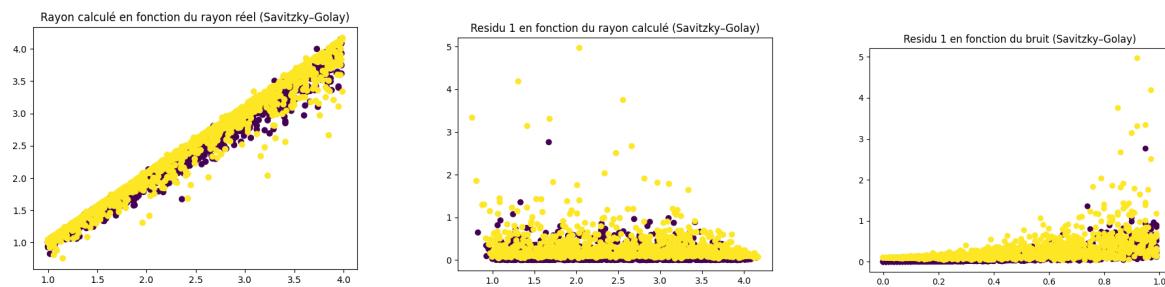


Figure 21: Les trois diagrammes de dispersion pour $N = 20$ ne permettent plus de faire la classification

Cependant, dans ce cas, le résidu ne peut plus faire la distinction entre les deux catégories de formes. Cela est sans doute dû au fait que le nombre de points est trop faible par rapport au bruit sur les images. Changer le degré du filtre Savitsky-Golay et utiliser le filtre gaussien ne permettait pas non plus d’améliorer le résultat.

C'est ainsi que nous n'avons pas réussi à faire une classification de nos données par la méthode des moindres carrés.

3.2.3 Architecture et paramètres du modèle de CNN

L'architecture du réseau de neurones convolutif utilisé s'inspire de celle proposée dans un tutoriel de *Keras for engineers* [13], et cette dernière est une version simplifiée des architectures qui sont souvent utilisées (plusieurs couches de *convolution-max pooling* qui se terminent par un *dense classifier*). Nous avons optimisé notre modèle en fonction du *batch size* et du nombre d'*epochs*, et fixé l'architecture et tous les autres paramètres du modèle. Voici l'architecture et les paramètres du modèle final que nous avons développé après avoir testé 8 versions différentes du modèle (les détails de chaque essai se trouvent dans l'Annexe D) :

1. Nombre total d'images : 2400 (1600 en entraînement, 400 en validation, 400 en test), toutes provenant du *dataset 4*
2. Dimension des images d'entrée : 640 * 400 pixels, noir et blanc
3. *Batch size* : 64
4. Nombre d'*epochs* : 30
5. Pré-traitement d'image : Recadrement de 400*400 à partir du centre, normalisation de chaque pixel
6. Architecture du modèle : 18,882 paramètres avec 7 couches :
 - Couche 1 : Convolution 2D de 3*3, avec 32 filtres (activation : ReLU)
 - Couche 2 : MaxPooling2D de 3*3
 - Couche 3 : Convolution 2D de 3*3, avec 32 filtres (activation : ReLU)
 - Couche 4 : MaxPooling2D de 3*3
 - Couche 5 : Convolution 2D de 3*3, avec 32 filtres (activation : ReLU)
 - Couche 6 : GlobalAveragePooling2D
 - Couche 7 : Dense classifier (activation : SoftMax)
7. Fonction d'optimisation : Adam
8. Fonction de perte : Cross-entropy

Ce modèle a une précision de 99,75% sur les images du jeu de validation et 99,5% sur les images du jeu de test.

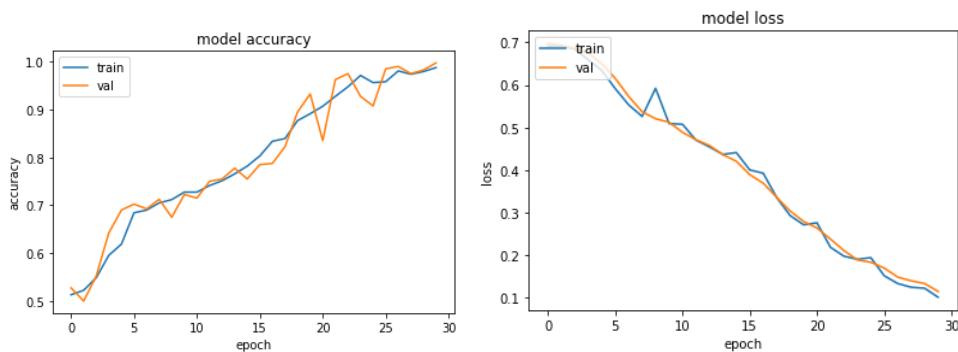


Figure 22 : L'évolution de la précision du modèle et la fonction de perte montre qu'il n'y a pas eu d'*overfitting* des données

Ce modèle a réussi à catégoriser les données obtenues par le SONAR avec une précision de 80% sur les 10 images finales (sans filtre gaussien). La précision est diminuée jusqu'à 70% s'il prend en entrée les images lisées par le filtre gaussien. Les résultats de classification par image se trouvent dans l'Annexe D.

Nous pouvons à partir de ces images voir que la relativement faible précision du modèle, lorsqu'il est appliqué à nos données, proviennent du bruit sur ces données. En effet, avec seulement 10 images provenant de l'expérience, nous ne possédons pas suffisamment d'informations pour modéliser le bruit élevé qui se trouve sur les images de pavé droit.

4. Conclusion et prolongement du projet

Ainsi, ce projet nous a permis de faire travailler de concert de l'électronique analogique et des réseaux de neurones très modernes.

La partie électronique, bien qu'ayant nécessité une légère réarticulation, réussit toutefois à être fonctionnelle, grâce à la flexibilité de l'arduino et aux astuces mises en place sur la carte pour court-circuiter les étages défectueux. Cette carte permet d'adapter la durée de l'émission à la distance supposée de l'objet.

Dans le cadre du réseau de neurones, nous avons réussi à utiliser un SONAR actif HC-SR04, connecté à un Arduino Nano 33 BLE Sense, pour récolter des données qui tracent le contour de la forme unidimensionnelle des objets cylindriques et celle d'un pavé droit, sous la condition que la surface de ces objets soient assez lisses et une ellipticité suffisamment faible (diamètre d'au moins 15 cm) pour que ses échos soient détectés par le SONAR. Nous avons ensuite réussi à entraîner et valider un réseau de neurones qui puisse distinguer ces deux formes avec un taux de précision d'entre 70% et 80%. Ce taux de précision est assez faible par rapport à celui dans la littérature (de plus de 95%), et il s'explique par le bruit fort et, à première vue, irrégulier qui se trouve sur toutes nos données.

On pourrait prolonger ce projet suivant divers axes : utiliser un microcontrôleur pour être plus précis dans l'émission et le traitement du signal, développer un réseau de neurones convolutifs créant une classification multiclasse (les pyramides ou même les formes plus irrégulières, en plus des cylindres et des pavé droits) et essayer de trouver des équations mathématiques qui peuvent mieux modéliser les données des pavés droit. Cela permettrait de générer une base de données plus représentative de ces objets et de développer un modèle mathématique servant de référence pour la précision de notre CNN.

Bibliographie

- [1] Alexis Mours. Localisation de cible en sonar actif. Océanographie. Université Grenoble Alpes, 2017. Français. NNT : 2017GREAU016. tel-01456797v2
- [2] Karimanzira D, Renkewitz H, Shea D, Albiez J. Object Detection in Sonar Images. Electronics. 2020; 9(7):1180. <https://doi.org/10.3390/electronics9071180>
- [3] Jiang, Longyu & Cai, Tao & Qixiang, Ma & Xu, Fanjin & Wang, Shijie. (2020). Active Object Detection in Sonar Images. IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.2999341.
- [4] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. Nature 521, 436–444 (2015). <https://doi.org/10.1038/nature14539>
- [5] Krizhevsky, A.; Sutskever, I.; Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. Neural Inf. Process. Syst. 2012, 25.
- [6] Chen, Y.; Yang, T.; Zhang, X.; Meng, G.; Xiao, X.; Sun, J. DetNAS: Backbone Search for Object Detection. arXiv 2019, arXiv:1903.10979.
- [7] Karimanzira D, Renkewitz H, Shea D, Albiez J. Object Detection in Sonar Images. Electronics. 2020; 9(7):1180. <https://doi.org/10.3390/electronics9071180>
- [8] Valdenegro, Matias. (2016). Object recognition in forward-looking sonar images with Convolutional Neural Networks. 1-6. 10.1109/OCEANS.2016.7761140.
- [9] Cdn.sparkfun.com. 2022. Ultrasonic Ranging Module HC - SR04. [online] Available at: <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>
- [10] Agence Web Kernix. 2017. A toy convolutional neural network for image classification with Keras - Agence Web Kernix. [online] Available at: <<https://www.kernix.com/article/a-toy-convolutional-neural-network-for-image-classification-with-keras/>>
- [11] Teodora Dimitrova-Grekow, Marcin Jarczewski. Identification Effectiveness of the Shape Recognition Method Based on Sonar. 14th Computer Information Systems and Industrial Management (CISIM), Sep 2015, Warsaw, Poland. pp.244-254, https://doi.org/10.1007/978-3-319-24369-6_20.
- [12] Scipy-cookbook.readthedocs.io. 2022. Least squares circle — SciPy Cookbook documentation. [online] Available at: https://scipy-cookbook.readthedocs.io/items/Least_Squares_Circle.html
- [13] Team, K., 2022. Keras documentation: Introduction to Keras for Engineers. [online] Keras.io. Available at: <https://keras.io/getting_started/intro_to_keras_for_engineers/> [Accessed 17 May 2022].

Annexe A : Étalonnage et test des limites du SONAR Arduino

Montage : Le SONAR est mis face à une mur. Une règle de 30cm est utilisée pour connaître de façon précise la distance entre le SONAR et le mur. Pour chaque mesure, la distance détectée par le SONAR est comparée avec la vraie distance entre le SONAR et le mur indiqué par la règle.

Distance sur règle /cm	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Distance mesurée sur SONAR	0.0	0.0	0.0	0.0	0.0	5	5	5	5	0

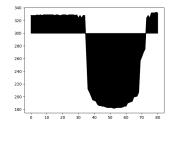
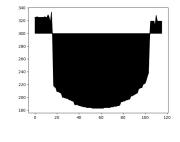
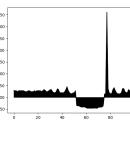
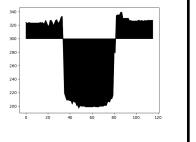
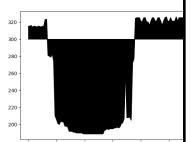
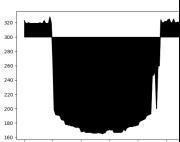
Distance sur règle/cm	0.0	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
Distance mesurée	0.0	0.7	1.3	1.3	0.7	0.7	0.7	0.7	0.7	0.7

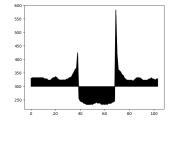
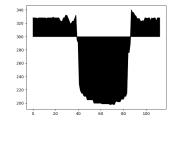
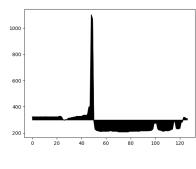
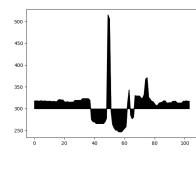
Distance sur règle/cm	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0	14.0	15.0
Distance mesurée	6.7	7.7	9.1	9.3	10.3	11.2	12.2	13.2	14.2	14.9

Distance sur règle/cm	16.0	17.0	18.0	19.0	20.0	21.0	22.0	23.0	24.0	25.0
Distance mesurée	15.9	17.0	18.1	18.9	20.0	21.2	22.3	23.8	24.8	24.7

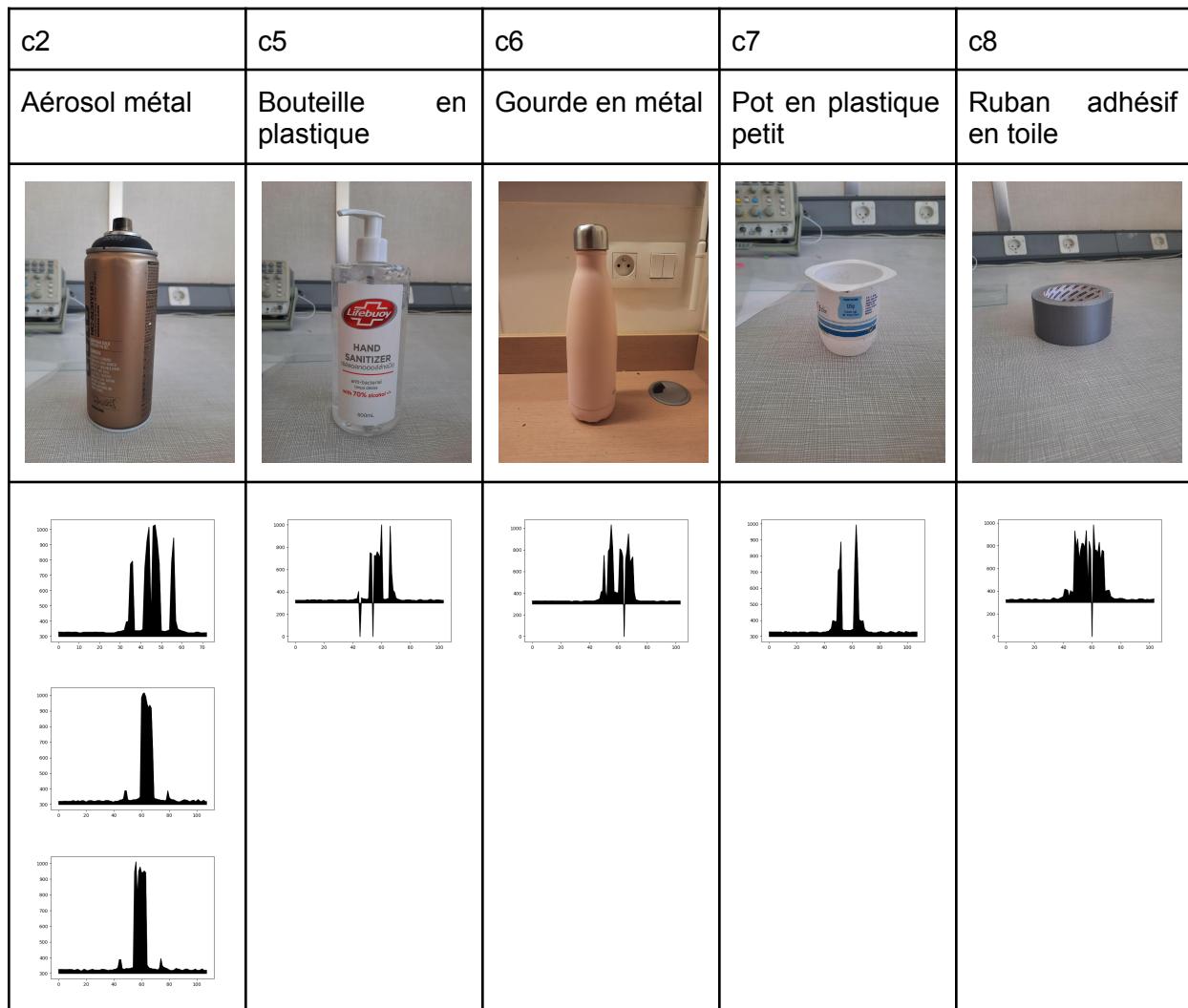
Annexe B : Tableau des images des échantillons d'objets

Donées retenues

c1	c3	c4	c13	c14	c15
Bol métal 1	Pot plastique grand	Bocal verre	Bol métal 2	Bol en métal 4	Bol en métal 5
					
					

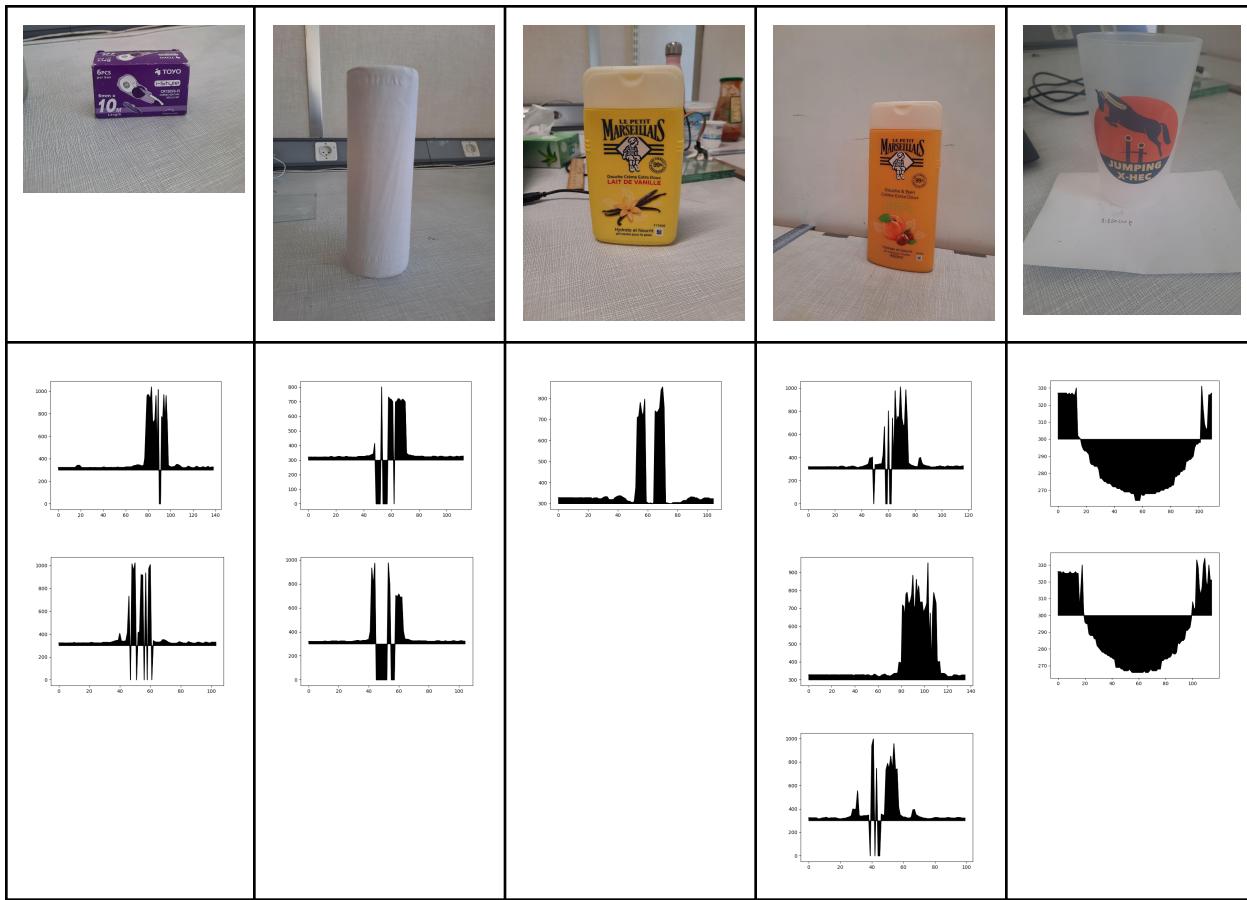
f2	f3	f7	f8
Boîte en plastique	Boîte en carton 1	Bûche en carton 2	Boîte en carton 3
			
			

Données inexploitables car trop bruitées

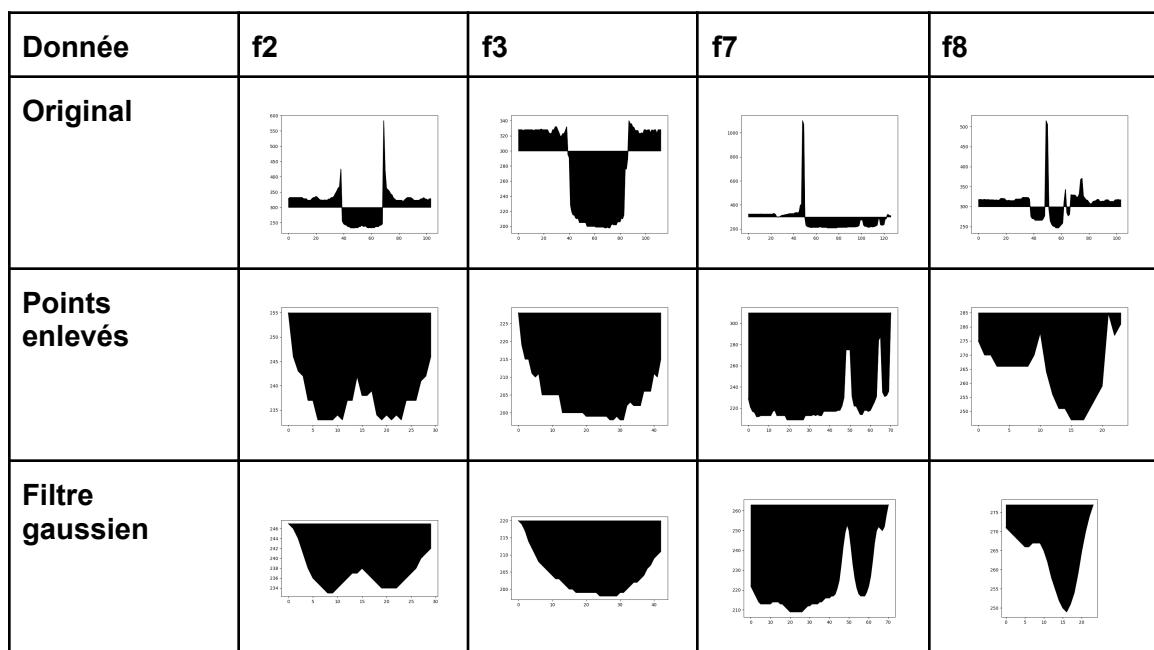
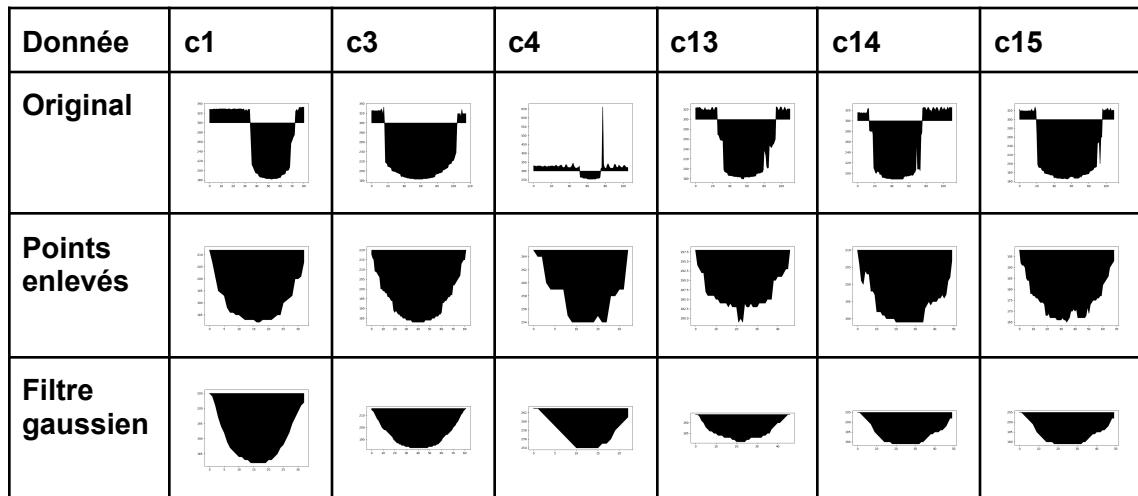


c10	c12	f5	f6	c11
Conteneur en papier	Sopalin	Conteneur en plastique petit	Conteneur en plastique grand	Ecocup

RAPPORT DE MODAL, PHY473O, 2022



Annexe C : Traitement des données



Annexe D : *Datasets et Réglages des paramètres du réseau de neurones*

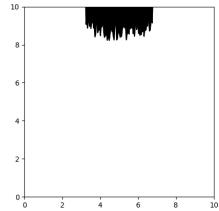
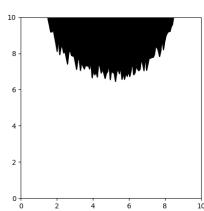
4 datasets utilisés :

Dataset 1:

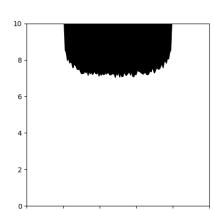
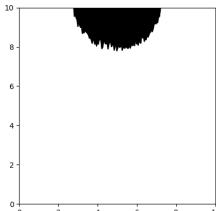
N (le nombre de points générés pour chaque image) = 100

Amplitude du bruit = 1

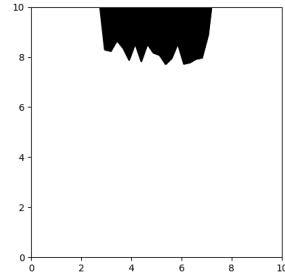
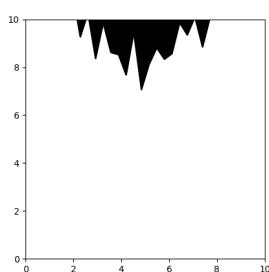
rayon : variable aléatoire uniforme à valeurs dans [1,4]



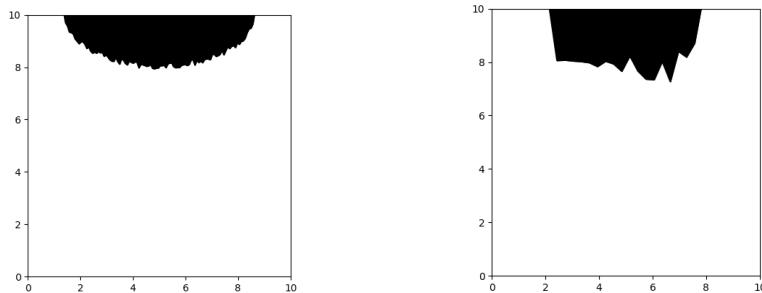
Dataset 2 : pareil que Dataset 1, mais l'amplitude du bruit n'est plus fixe à 1 : il est déterminé par une variable aléatoire uniforme à valeurs dans [0,1]



Dataset 3 : pareil que Dataset 2, mais la valeur de N est réduite = 20 et le bruit est amplifié par un facteur égal au rayon de l'échantillon



Dataset 4 : les rectangles courbés sont exactement les mêmes images que Dataset 3. Pour les cercles, on utilise N = 100 et on utilise aussi l'équation d'une ellipse avec a,b des variables aléatoires uniformes à valeurs dans [1,4]



Modèle de réseaux de neurones

Trial, utilisé	dataset	Batch Size, Total epochs	Epochs selected	Loss / val_loss/ test_acc	acc/val_ acc/test_ acc	Confusion matrix	Test data
1	dataset1	32,10	10	0.6747	0.55	-	-
2 (CNN2)	dataset2	32,100	~20	0.2416 /0.2657 /0.27	0.9156/ 0.8650/ 0.89	[[197 3] [42 158]]	-
3 (CNN3)	dataset2	64,30	30	0.2033/0 .2166 /0.20	0.9513 /0.9475 /0.96	[[199 1] [16 184]]	[[0 6] [0 4]]
4 (CNN3)	dataset2	128,30 128,60 128,90	30 (not enough) 60 (not enough) 90 (possibl e overfit after 63)	0.6045 /0.6322 /0.60 0.2730/0 .2960 /0.29 0.1828 /0.2229/ 0.21	0.6856 /0.6275/ 0.70 0.9250 /0.8700/ 0.90 0.9519 /0.9250/ 0.93	[[179 21] [100 100]] [[199 1] [39 161]] [[174 26] [2 198]]	-
5 (CNN3)	dataset2	128,63		0.28/-/-	0.89/-/-	[[200 0] [44 156]]	-
6 (CNN3)	dataset2	64,30	30	0.3271/0 .3201/-	0.8856/ 0.8175/-	[[193 7] [51 149]]	[[0 6] [0 4]]
7 (CNN3)	dataset3	64,50 64,60		0.5920 /0.6166 /0.57 0.5643	0.6833 /0.6166 /0.66 0.6914	[[95 105] [49 151]] [[158 42]]	[[0 6] [0 4]]

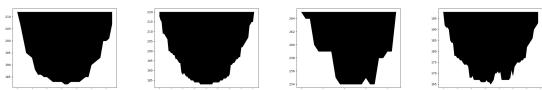
RAPPORT DE MODAL, PHY473O, 2022

			/0.5657 /0.56	/0.6983/ 0.72	[92 180]]	[0 4]] Raspberryg 2 - [[1 5] [1 3]]
8 dataset4	64,30		0.1146 /0.1009 /0.1026	0.9875 /0.9975/ 0.9950	[[200 0] [2 198]]	Raspberry g2: [[6 0] [3 2]] Raspberry data: [[4 2] [0 3]]

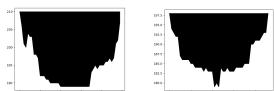
Résultats de la classification des images de l'expérience

Données sans filtre gaussien (80% précision) :

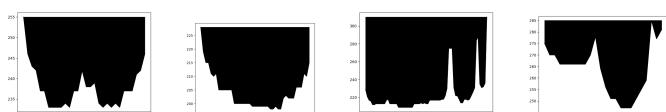
1. Quatre cercles ont été correctement identifiés.



2. Deux cercles n'ont pas été correctement identifiés.

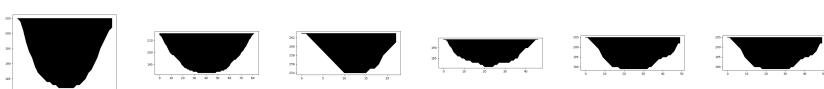


3. Quatre rectangles courbés ont été correctement identifiés.



Données avec filtre gaussien (70% précision) :

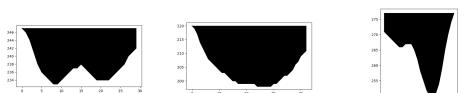
Six cercles ont été correctement identifiés.



Un rectangle courbé a été correctement identifié.



Trois rectangles courbés n'ont pas été correctement identifiés.



Annexe E : Schéma et routage sous EAGLE

