



RAPPORT DE PROJET DE FIN D'ANNÉE

4^{ème} Année Réseaux Informatiques et Télécommunications

Conception d'un dispositif IoT et application mobile pour une évaluation de la qualité de l'eau basée sur l'apprentissage profond

Auteurs

**Ranim SAIDI
Asma JEBARI
Bilel KHELIFI**

Enacadrante

Dr. Rabaa YOUSSEF

Examinateuse

Dr. Wided MILED

Remerciements

On tient à exprimer notre gratitude et reconnaissance pour notre encadrante Dr. Rabaa YOUSSEF, qui nous a donné cette opportunité de travailler sur un tel projet aussi consistant. Elle nous a guidés durant toutes les phases du projet qui a duré plus que 4 mois. On la remercie pour la qualité de son encadrement, sa rigueur, sa disponibilité durant toute la période précédente et ses encouragements.

Table des matières

Introduction générale	2
1 Contexte général du projet	3
1.1 Contexte du projet	4
1.2 Autres travaux d'intérêt	6
1.3 Axes d'amélioration	8
2 Application mobile et IoT	9
2.1 Application mobile : Spécification et technologies d'implantation	10
2.1.1 Spécifications conceptuelles	10
2.1.2 Technologies utilisées	11
2.2 Plateforme IoT : Spécification et choix technologiques	15
2.2.1 Fonctionnement IoT	15
2.2.2 Choix du matériel	16
2.2.3 Traçabilité des données	17
3 Approche Computer Vision pour la classification de l'eau turbide	19
3.1 Machine Learning	20
3.1.1 Les descripteurs de texture	20
3.1.2 Les descripteurs de couleur	22
3.1.3 Classificateurs choisis	24
3.2 Deep Learning	25
3.2.1 Les réseaux de neurones convolutifs : Définitions et principes généraux	25
3.2.2 Concepts utiles en Deep Learning	27
3.2.3 Le modèle ResNet	27
3.2.4 Modèles profonds implémentés	28
4 Expérimentations et résultats	31
4.1 Acquisition des données et organisation du dataset	32
4.1.1 Dataset d'apprentissage	32

4.1.2	Nouveau dataset	33
4.1.3	Pré-traitement	34
4.1.4	Apprentissage	36
4.2	Résultats de classification	36
4.2.1	Machine Learning	36
4.2.2	Deep Learning	39
4.3	Test sur le nouveau dataset	46
	Conclusion générale et perspectives	48
	Bibliographie	50

Table des figures

1.1	Fonctionnement du détecteur d'un turbidimètre [3]	4
1.2	Motif utilisé pour l'acquisition des données [1]	5
1.3	Exemples de l'effet de tubidité sur le motif [1]	5
1.4	Mécanisme proposé par [7]	6
1.5	Disque de Secchi [11]	7
2.1	Cas d'utilisation de l'application mobile	10
2.2	Interface d'inscription	12
2.3	Interface de capture d'image	12
2.4	Interface de chargement de données	13
2.5	Hiérarchie des classes de turbidité	14
2.6	Exemple de hiérarchie des repertoires dans la classe de très faible turbidité	14
2.7	Visualisation des images chargées dans le Cloud Firestore	14
2.8	Architecture générale du système	15
2.9	Diagramme d'état de la plateforme	16
2.10	Fonctionnement d'un capteur Nephelometer [14]	16
2.11	Tableau de bord de la plateforme iot cloud	17
2.12	Extrait des résultats de google spread sheet	18
3.1	Caractérisation locale des régions par LBP [16]	21
3.2	Support Vector Machine [20]	24
3.3	Random Forest [21]	24
3.4	XGBoost [23]	25
3.5	Architecture des réseaux CNN [25]	26
3.6	Représentation d'un bloc résiduel [28]	28
3.7	Modèle Aquasight [1]	28
3.8	Modèle Aquasight modifié	29
4.1	Répartition des données d'apprentissage en fonction des classes	33
4.2	1ère proposition de motif	33

4.3	Forme du motif après inversion et dilatation	33
4.4	Turbidité = 5NTU	34
4.5	Turbidité = 58NTU	34
4.6	Turbidité = 129NTU	34
4.7	Matrice de confusion - SVM après rehaussement	37
4.8	Matrice de confusion - Random Forest après rehaussement	38
4.9	Matrice de confusion - XGBoost après rehaussement	38
4.10	Variation des valeurs de l'accuracy - InceptionResNet	39
4.11	Variation des valeurs du loss - InceptionResNet	39
4.12	Variation des valeurs de l'accuracy - ResNet18 - Avant augmentation	40
4.13	Variation des valeurs du loss - ResNet18 - Avant augmentation	40
4.14	Variation des valeurs de l'accuracy - ResNet18 - Après augmentation	40
4.15	Variation des valeurs du loss - ResNet18 - Après augmentation	40
4.16	Matrice de confusion - ResNet18	41
4.17	Variation des valeurs de l'accuracy - ResNet 2ème approche	41
4.18	Variation des valeurs du loss - ResNet 2ème approche	41
4.19	Matrice de confusion - ResNet 2ème approche	42
4.20	Variation des valeurs de l'accuracy - ResNet 3ème approche	42
4.21	Variation des valeurs du loss - ResNet 3ème approche	42
4.22	Matrice de confusion - ResNet 3ème approche	43
4.23	Variation des valeurs de l'accuracy - Architecture Aquasight modifiée	43
4.24	Variation des valeurs du loss - Architecture Aquasight modifiée	43
4.25	Matrice de confusion - Architecture Aquasight	44
4.26	Variation des valeurs de l'accuracy - Architecture ex nihilo	44
4.27	Variation des valeurs du loss - Architecture ex nihilo	44
4.28	Matrice de confusion - Architecture ex nihilo	45

Liste des abréviations

COmmunications **S**ignaux et **I**Mages

Convolutional **N**eural **N**etwork

Integrated **D**evelopment **E**nvironment

Internet **o**f **T**hings

Nephelometric **T**urbidity **U**nit

Hypertext **T**ransfer **P**rotocol

Deep **L**earning

Machine **L**earning

Root **M**ean **S**quared **E**rror

Central **P**rocessing **U**nit

Local **B**inary **P**atterns

Enhanement **M**easure **E**stimation

Gray **L**evel **C**o-occurrence **M**atrix

Histogram of **O**riented **G**radients

Hue **S**aturation **V**alue

Blue **G**reen **R**ed

Support **V**ector **M**achine

Light-**E**mmiting **D**iode

Wireless **F**idelity

Fully **C**onnected

Deep **N**eural à **N**etwork

Residual **N**etwork

Résumé

Maintes sont les caractéristiques physico-chimiques qui influencent la tâche de mesure de la qualité de l'eau. Étant donné que l'estimation de ces critères obéit à des protocoles expérimentaux, rigoureux et que le coût des équipements de laboratoire s'avère être très cher, un recours aux techniques de la vision par ordinateur est manifestement sollicité pour la démocratisation de ce processus. Dans ce projet, nous nous sommes inspirés des travaux précédents afin d'opter pour différentes approches computer vision dans le but d'estimer la classe de la turbidité. Ce rapport détaille les étapes de collecte de données grâce au développement d'une application mobile, suivie de l'utilisation d'une plateforme IoT pour la prise des valeurs de la turbidité et on finit par proposer des méthodes Machine Learning et Deep Learning implémentées pour la tâche de classification.

Abstract

Measuring water quality is an important means to ensure it is well suited for all types of usage. Since the task requires meticulous experimental protocols and the equipment needed is often expensive, turning to computer vision techniques is an evident alternative to normalizing the process. In this project, we have been inspired by previous works and proposed state of the art use of different approaches, architectures, and models of Machine Learning and Deep Learning to estimate the class of water turbidity. This report details the stages of data collection through the development of a mobile application, to the use of an IoT platform for measuring turbidity values, and finally to the testing of Machine Learning and Deep Learning methods deemed useful for Computer Vision.

Introduction générale

L'eau, étant la source de l'humanité et de toutes les vies, doit être en bonnes conditions pour être proprement exploitable. Cela étant dit, il est important qu'on contrôle sa qualité pour assurer qu'elle soit adéquate pour tout type d'utilisation. L'un des moyens de mesure de cette qualité est la turbidité qui désigne, selon l'usage courant du terme, la clarté optique de l'eau ou du liquide. Elle est due, en effet, aux particules flottantes dans l'eau car elles diffusent ou absorbent la lumière et réduisent par conséquent la visibilité dans l'image. Il est très important de pouvoir évaluer rapidement et en ligne la turbidité de l'eau pour réaliser un contrôle efficace, peu coûteux et précis de l'aquaculture.

À présent, le turbidimètre est une méthode de détection de la turbidité largement utilisée, qui comprend des méthodes de détection en laboratoire et des méthodes de détection par instrument en ligne. L'avantage des méthodes de détection en laboratoire est leur grande précision. Cependant, elles présentent des inconvénients évidents, tels que le coût élevé, le fonctionnement professionnel, le manque de performance en temps réel et l'impossibilité d'obtenir des échantillons précis dans de grands espaces et sur de longues périodes. Au contraire, les méthodes de détection en ligne sont garanties en temps réel car elles envoient des informations d'alerte lorsque la ressource en eau est polluée, sur la base de la spectrophotométrie ultraviolette (UV), mais leur précision est facilement affectée par les particules dans l'eau.

Ces méthodes populaires de détection de la turbidité présentent certaines limites en terme de coût, de commodité et de couverture spatiale. Sur la base des raisons citées ci-dessus et de l'explosion des applications de l'intelligence artificielle, les chercheurs se consacrent au développement des méthodes de détection de la turbidité basées sur la vision par ordinateur comme complément ou même alternative aux méthodes traditionnelles de détection de la turbidité. En particulier, l'application de cette technologie aux smartphones améliore la protection de l'environnement et des écosystèmes aquatiques par tout individu. Notre projet s'inscrit sous cette perspective pour concrétiser la démocratisation de l'évaluation, en particulier la classification, de la qualité d'eau. Ceci étant par le biais d'une application mobile conçue pour la collecte des images et d'une plateforme IoT liée au Cloud pour l'enregistrement en temps réel des valeurs de la turbidité. Par la suite, on opte pour différentes approches Machine Learning et Deep Learning pour réaliser la classification.

A cet effet, on consacre le premier chapitre pour l'exposition de l'état de l'art et des inspirations qu'on a eu pour ce projet ainsi que les axes d'améliorations. Le second chapitre est dédié aux spécifications conceptuelles de l'application mobile et de la plateforme IoT. Le troisième chapitre s'intéresse aux méthodes de classification Machine Learning et Deep Learning et on finit par présenter l'expérimentation et les résultats.

CONTEXTE GÉNÉRAL DU PROJET

Plan

1	Contexte du projet	4
2	Autres travaux d'intérêt	6
3	Axes d'amélioration	8

Introduction

Dans ce chapitre, on commence par définir le contexte du projet, en présentant le travail de M.Chaaben [1] qui sert de point de départ ainsi que d'autres lectures sur lesquelles on s'appuie pour élargir nos perspectives. Nous introduisons, à cet effet, les axes d'améliorations abordés.

1.1 Contexte du projet

La turbidité est une caractéristique d'un fluide causée par un grand nombre de particules en suspension qui sont généralement invisibles à l'œil nu. La mesure de la turbidité est un test clé de la qualité de l'eau. Les fluides peuvent contenir des matières solides en suspension constituées de particules de tailles très diverses. Ces petites particules solides donnent au liquide un aspect turbide.

Notre projet se base, ainsi, sur le travail réalisé par M.Chaaben [1] effectué au sein du laboratoire COSIM. Il s'agit de l'estimation de la turbidité de l'eau par les méthodes de la vision par ordinateur. Il commence par construire sa propre base d'images puis pour diminuer le surapprentissage, essaie la technique d'augmentation artificielle par l'ajout d'un bruit gaussien blanc et l'étirement du contraste. Certes, cette méthode n'a pas prouvé son utilité étant qu'elle génère un taux relativement élevé d'erreurs. Il est aussi à noté que les acquisitions effectuées de leur part sont déséquilibrées.

L'annotation du dataset est réalisée à l'aide d'un turbidimètre, où le détecteur convertit l'intensité lumineuse reçue de la lumière diffusée de l'eau en un signal électrique qui est transformée par la suite en une valeur de turbidité suivant cette corrélation [2] :

$$NTU = -1120.4v^2 + 5742.3v - 4352.9 \quad (1.1)$$

Comme le montre la figure 1.1, la diffusion est causée par les particules en suspension dans l'eau :

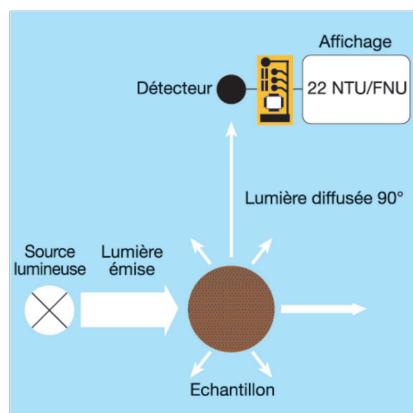


FIGURE 1.1 : Fonctionnement du détecteur d'un turbidimètre [3]

Ensuite, pour la collecte de données, il crée deux sous-bases, la première est sans motif et l'autre est avec un motif en arrière-plan. Ce motif est constitué de quatre rectangles noirs horizontaux de largeurs de plus en plus petites car au fur et à mesure que l'eau augmente en turbidité, les rectangles deviennent moins nets, augmentent en contraste et leurs bords deviennent moins acérés.



FIGURE 1.2 : Motif utilisé pour l'acquisition des données [1]

Pour prédire la valeur de turbidité, l'auteur de [1] recourt à deux approches : une estimation par régression et une autre par classification. Pour ce faire, on sollicite l'architecture AquaSight [4] qui est utilisée pour la détection automatique des impuretés aquatiques. Elle possède une conception peu profonde. En approfondissant l'architecture et en modifiant les hyperparamètres pour l'objectif de régression, il mesure la racine de l'erreur quadratique moyenne (RMSE) des deux bases qui étaient très élevées.

⇒ La régression s'avère être impossible à mettre en place puisqu'elle exige une base de données de taille plus grande.

Afin de réaliser une classification selon les niveaux de turbidité, les bases de données sont divisées en 4 classes en un premier temps, puis en 3 pour remédier au problème de confusion entre les 2 premières classes. Il constate l'efficacité du motif tout en observant la variation entre les bandes blanches et les bandes noires ainsi que le flou aux contours créé au fur et à mesure que la turbidité augmente comme le montre la figure 1.3 :

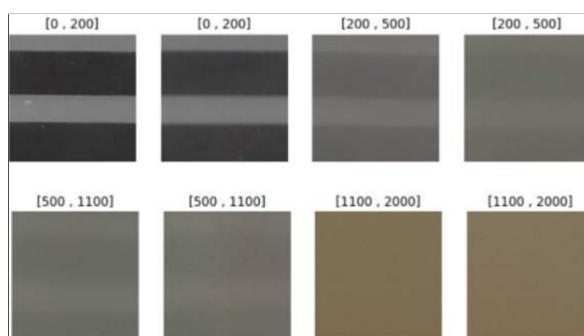


FIGURE 1.3 : Exemples de l'effet de turbidité sur le motif [1]

⇒ Il limite ainsi ses expérimentations à la classification et avec 3 classes au lieu de 4 en conservant le dataset avec un motif en arrière plan vu qu'il donne des résultats plus pertinents.

1.2 Autres travaux d'intérêt

On constate que parmi les turbidimètres professionnels standards figurent le Hach 2100N et le Thermo Fisher Scientific Orion AQ4500 [5]. Bien que ces instruments soient très précis, ils sont également assez chers. Bell et Russell [6] ont signalé que le coût élevé des instruments scientifiques nuit à la surveillance et à la protection efficaces de l'environnement dans les pays en développement.

En outre, on s'est inspiré de plusieurs papiers de recherches. D'abord, certains proposent une méthode pour mesurer la turbidité de l'eau avec un smartphone [7]. Il s'agit d'un dispositif qui se compose d'une diode électroluminescente infrarouge (IR), d'un détecteur à deux photodiodes. Le smartphone fournit l'alimentation à la LED et détecte le signal lumineux à travers le sténopé comme l'illustre la figure 1.4. Deux applications android ont été utilisées dans le système pour mesurer la lumière diffusée dans l'échantillon. Des valeurs élevées de coefficient de régression R^2 ont été obtenues par leur courbe caractéristique reliant l'intensité normalisée et la turbidité, indiquant que le turbidimètre proposé est précis pour mesurer la turbidité de l'eau.

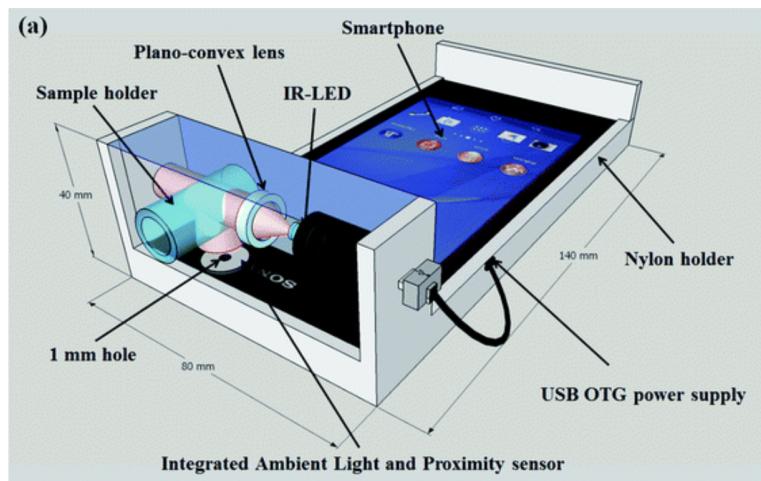


FIGURE 1.4 : Mécanisme proposé par [7]

En plus, [8], une autre étude utilise également l'histogramme d'intensité des images grises pour déterminer le degré de la turbidité. Cependant, l'étalonnage des systèmes proposés n'a pas été effectué dans cette étude, et par conséquent, la précision en termes de valeur R^2 n'est pas connue.

- NB : En statistique, le coefficient de détermination linéaire de Pearson, noté R^2 est une mesure de

la qualité de la prédiction d'une régression linéaire. Il est défini par [9] :

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

n est le nombre de mesures, y_i la valeur de la mesure n° i , \hat{y}_i la valeur correspondante et \bar{y} la moyenne des mesures.

Dans un autre travail, Lang [10] propose un système de détection de la turbidité basé sur un algorithme d'extraction d'images dans les domaines spatial et fréquenciel, combiné au réseau neuronal artificiel. Ce système améliore la précision de mesure de la turbidité de l'eau. En revanche, l'extraction des caractéristiques et la prédiction de la turbidité de cette méthode sont séparées, et la précision de l'extraction des caractéristiques affecte directement l'exactitude des résultats.

Une autre source d'inspiration pour notre projet est le rapport de master [11] réalisé au sein de "Aalborg University". Il traite en effet, la mesure de la turbidité dans un plan d'eau en contemplant les méthodes existantes de mesure de la turbidité, comme un turbidimètre électronique ou un disque de Secchi. La mesure basée sur le disque de Secchi consiste à utiliser un disque (Secchi) noir et blanc qui est plongé dans une masse d'eau, comme illustré à la figure 1.5. Au point où la visibilité est perdue, la profondeur du disque est enregistrée, et est connue sous le nom de profondeur de Secchi. Ils sont souvent rapides et peu coûteux, mais leur précision dépend de l'expertise de la personne qui les utilise.

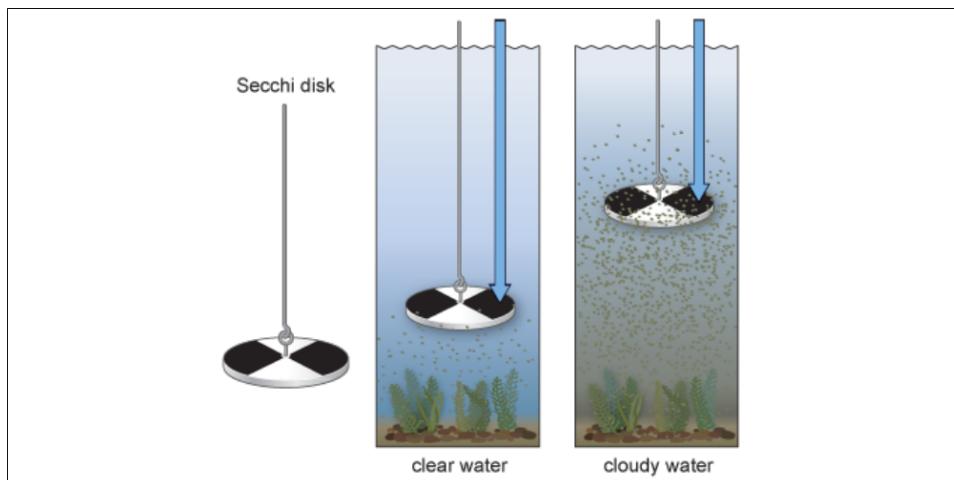


FIGURE 1.5 : Disque de Secchi [11]

Cette étude vient en effet appuyer la nécessité de l'intégration des méthodes de vision par ordinateur pour l'estimation de la qualité de l'eau.

Dans cette recherche, l'utilisateur remplit le récipient de l'eau et place la structure de mesure dans le récipient. Il prend ensuite une image avec l'appareil photo d'un téléphone portable à travers un trou dans le couvercle. Le logiciel envoie la photo à un serveur, qui analyse automatiquement les acquisitions.

1.3 Axes d'amélioration

Suite au travail effectué par [1] et aux lectures menées dans le cadre de ce projet, on constate que certaines modifications peuvent être utiles pour améliorer à la fois la précision du classifieur ainsi qu'une meilleure évolutivité et utilisabilité d'une telle solution pour l'estimation de la turbidité. Les axes d'amélioration à traiter peuvent se résumer dans les points ci-dessous :

- Création d'une nouvelle base de données :

La création d'une nouvelle base s'avère être indispensable étant donnée que le motif utilisé par [1] n'est pas assez complexe et la taille du dataset ancien est réduite. Une piste d'amélioration consiste à faire varier le sédiment pour une plus grande robustesse du modèle, le rendre plus général et adapté à différents types de sédiments. Dans le but d'augmenter la taille du dataset, nous avons appliqué un réhaussement sur les données du dataset déjà existant. Ceci a conduit à avoir un nouveau dataset de taille deux fois plus grand.

- Standardisation du protocole de l'acquisition de données :

Un premier axe d'amélioration consiste à développer une application mobile qui facilite l'acquisition des images tout en se focalisant sur la région d'intérêt. L'application permet le stockage en temps réel des images dans le cloud ainsi que les valeurs de turbidité correspondantes. Pour garder la traçabilité des valeurs de turbidité, une plateforme IoT est mise en place.

- Utilisation et implémentation de nouvelles architectures :

Une autre tâche consiste à tester la base existante et notre nouvelle base sur différents modèles Machine Learning et Deep Learning.

Conclusion

Les travaux réalisés sous cette thématique ont traité le sujet d'estimation du niveau de la turbidité selon différentes approches et ont été très utiles pour cerner la problématique. Néanmoins, ceci n'empêche qu'il existe d'autres pistes prometteuses à envisager et qui seront les points développés dans la suite de cette étude.

APPLICATION MOBILE ET IoT

Plan

1	Application mobile : Spécification et technologies d'implantation	10
2	Plateforme IoT : Spécification et choix technologiques	15

Introduction

En vue de faciliter la tâche de collecte et l'organisation des données, on opte pour le développement d'une application mobile qui permet de capturer la région d'intérêt et de saisir en temps réel la valeur de la turbidité correspondante.

2.1 Application mobile : Spécification et technologies d'implantation

2.1.1 Spécifications conceptuelles

Le besoin d'une estimation rapide et pratique de la qualité d'eau est devenu très important pour la réalisation efficace, peu chère et précise d'un système de contrôle.

L'introduction des systèmes utilisant la vision par ordinateur pour l'estimation de la turbidité requiert l'utilisation d'une application mobile pour la collecte et l'intégration de données et même le déploiement ultérieur d'un modèle d'intelligence artificielle. TurbiLearn, notre application, a pour objectif principal l'acquisition de données de manière à se focaliser sur la région d'intérêt et à les charger automatiquement vers le cloud.

Ci-dessous un diagramme de cas d'utilisation de l'application TurbiLearn :

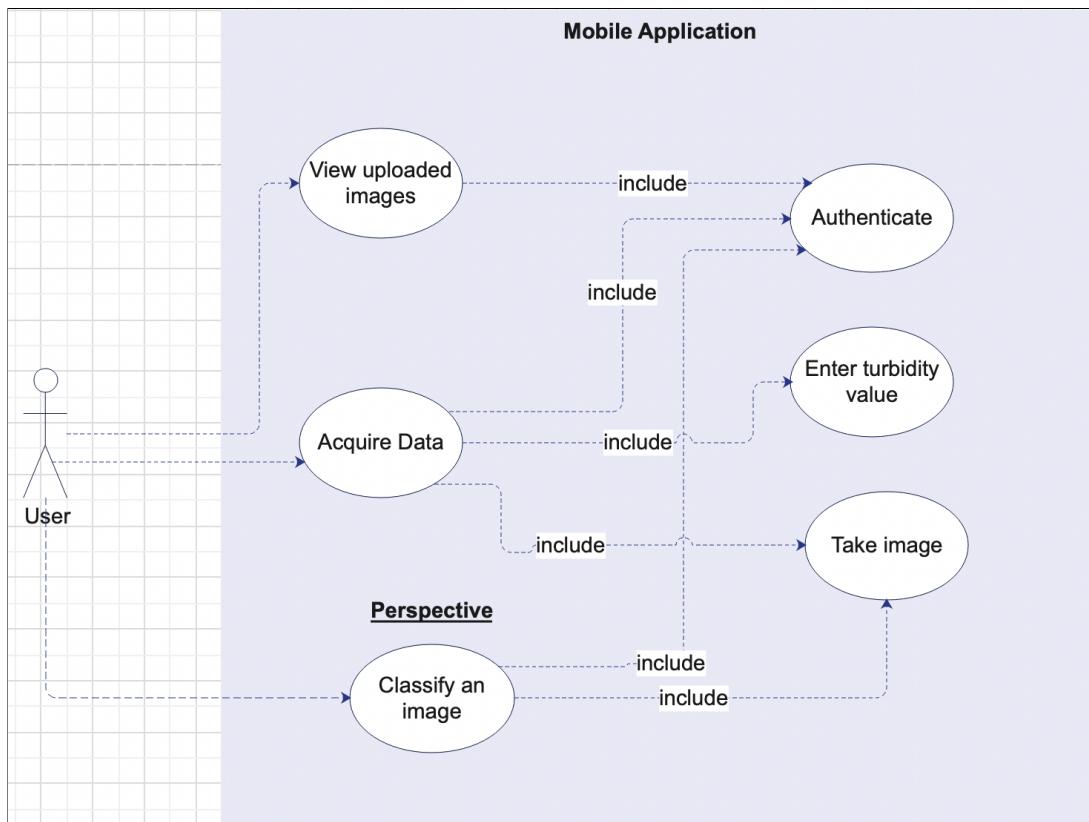


FIGURE 2.1 : Cas d'utilisation de l'application mobile

L'application mobile est développée avec le framework Flutter 3.0.1 qui utilise comme Backend Firestore, une base de données située dans le cloud qui sauvegarde les coordonnées des utilisateurs et leurs acquisitions.

Pour une première connexion d'un utilisateur, on lui demande de créer un compte en remplissant un formulaire classique contenant son email, mot de passe, nom prénom etc. Ce compte sera enregistré dans une base de données de Firebase. Ensuite, on a le choix entre créer une nouvelle acquisition ou consulter les images créées par cet utilisateur. Toute prise de nouvelle image sera suivie de la saisie de la valeur de turbidité correspondante en NTU, qui est le label de l'image.

Besoins fonctionnels :

- Permettre une acquisition facile et pratique des données ainsi que la saisie de la turbidité correspondante(label).
- Permettre une sauvegarde hiérarchique des images dans la base de données selon la valeur saisie qui correspond à la classe de turbidité(répertoire correspondant).
- Sauvegarder les coordonnées de chaque utilisateur dans le cloud.
- Pouvoir consulter les images chargées dans la base de données.

Besoins non fonctionnels :

- L'application doit permettre le chargement de chaque photo au bout de 3 secondes.
- L'interface est facile à utiliser.
- L'application doit être légère en terme d'espaces de stockage.

2.1.2 Technologies utilisées

2.1.2.1 Framework de développement Flutter

On choisit le framework Flutter pour le développement. Il s'agit d'un cadre de développement multi-plateformes créé par Google basé sur le language de programmation Dart. Ce choix peut être justifié par ces 4 raisons :

- (i) L'évolutivité du projet : En utilisant notre base de code actuelle, nous avons créé deux applications mobiles pour IOS et Android. Avec des modifications mineures, nous pourrions créer une application web. Potentiellement, Flutter sera également utilisé pour créer des applications de bureau.
- (ii) Le nombre impressionnant de paquets open source créés par la communauté. Dans notre application, nous avons utilisé trois paquets principaux : le paquetage de recadrage des photos, le paquetage de connexion à la base de données cloud-firebase et le camera-overlay pour faciliter la focalisation sur la région d'intérêt.

- (iii) Le développement mobile à l'aide de Flutter est plus rapide que l'utilisation d'autres frameworks comme React Native ou le développement natif à l'aide de Java ou Kotlin.
- (iv) Contrairement aux autres frameworks, il n'y a pas de compromis sur le plan des performances.

Comme environnement de développement intégré (IDE), on opte pour Android Studio. Cet IDE offre des fonctionnalités pour booster le développement, comme la possibilité de créer différents émulateurs avec des spécificités différentes, d'auto-compléter des mots-clés et de refactoriser le code. Pour le contrôle de version, on utilise Git et Github. Il facilite la collaboration entre les membres de notre équipe. En outre, il permet de créer des points de contrôle vers lesquels revenir en cas de problème.

Les figures 2.2, 2.3 et 2.4 ci-dessous présentent 3 interfaces de TurbiLearn, soit la première pour enregistrer un nouvel utilisateur dans l'application, la deuxième pour capturer une image en gardant uniquement la partie centrale (Région d'intérêt) qui sera envoyée à la base de données, et on finit par l'interface de confirmation des données avant de les charger vers le cloud.

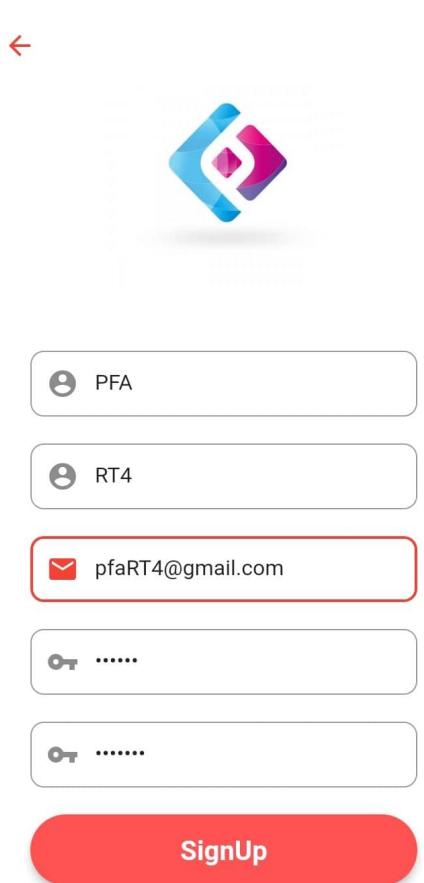


FIGURE 2.2 : Interface d'inscription

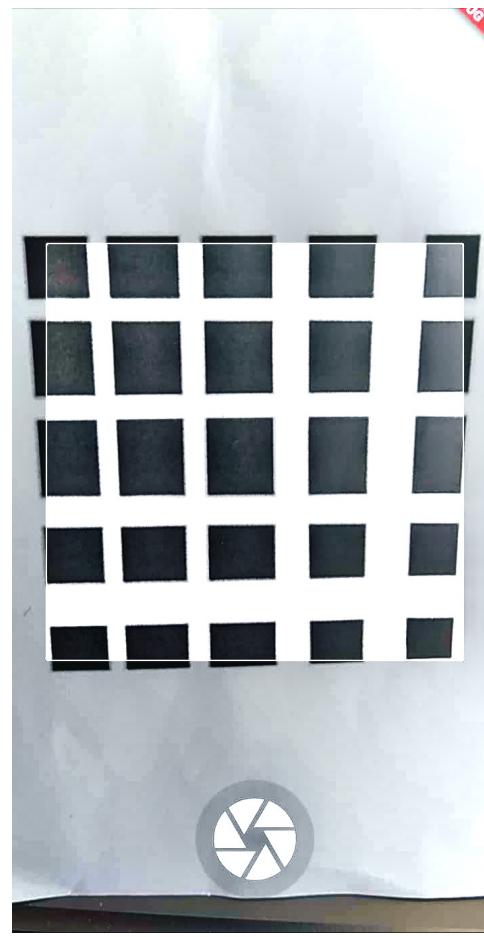


FIGURE 2.3 : Interface de capture d'image

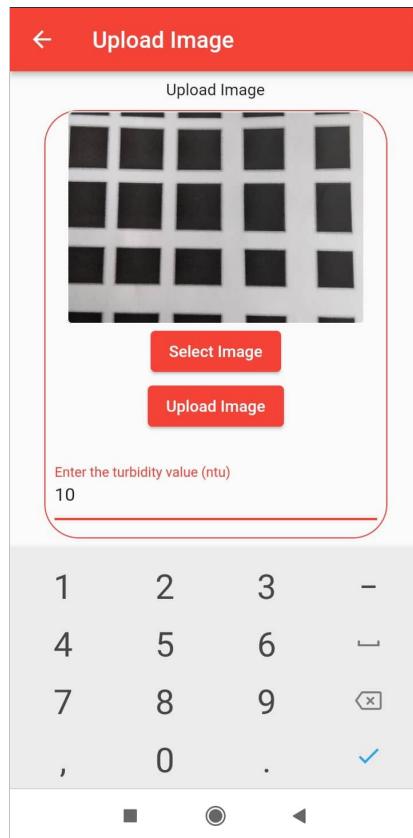


FIGURE 2.4 : Interface de chargement de données

2.1.2.2 Base de données Cloud Firestore

Firestore [12] est une base de données NoSQL flexible et évolutive conçue pour une mise à l'échelle automatique et pour la facilité de développement des applications. Flutter a un support officiel pour Firestore et la connexion se réalise à l'aide de deux Interfaces de programmation d'applications Firebase Core et Cloud Firestore.

Pour notre application, Firestore sert d'une solution cloud pour le stockage des acquisitions selon une hiérarchie de classes. On associe chaque image capturée selon la valeur de turbidité saisie au répertoire correspondant comme le montre la figure 2.5.

Pour plus de précision, chaque classe de turbidité définit des sous-hiéronymies en nombre de 3 (pour la première classe). C'est ce qui est mis en relief par la figure 2.6.

La figure 2.7 montre un exemple d'acquisition faite par l'application mobile et chargée dans le Firestore en portant comme nom la valeur de turbidité suivie de la date pour mieux faire la correspondance avec la plateforme IoT.

The screenshot shows the Google Cloud Storage interface for a project named "WaterTurbidity". The "Files" tab is selected. A banner at the top right says "Protégez vos ressources Storage des utilisations abusives telles que la fraude à la facturation et le hameçonnage" and "Configurer App Check". Below the banner is a breadcrumb navigation: gs://waterturbidity-940c6.appspot.com > images. There is a blue button "Importer un fichier" with a plus icon. A table lists five entries under the "Nom" column:

Nom	Taille	Type	Dernière modification
High_Turbidity/	—	Dossier	—
Low_Turbidity/	—	Dossier	—
Medium_Turbidity/	—	Dossier	—
very_Low_Turbidity/	—	Dossier	—

FIGURE 2.5 : Hiérarchie des classes de turbidité

This screenshot shows a similar view of the Google Cloud Storage interface. The breadcrumb navigation shows gs://waterturbidity-940c6.appspot.com > images > very_Low_Turbidi... . The table lists three entries under the "Nom" column:

Nom	Taille	Type	Dernière modification
1st_subclass/	—	Dossier	—
2nd_subclass/	—	Dossier	—
3rd_subclass/	—	Dossier	—

FIGURE 2.6 : Exemple de hiérarchie des répertoires dans la classe de très faible turbidité

This screenshot shows the Google Cloud Storage interface for the "1st_subclass" folder. The breadcrumb navigation is gs://waterturbidity-940c6.appspot.com > images > very_Low_Turbidi... > 1st_subclass. The table lists six image files:

Nom	Taille	Type	Dernière modification
turbidity=12_31-5-2022::13:11:21	349.26 KB	image/jpeg	31 mai 2022
turbidity=12_31-5-2022::13:11:5	368.29 KB	image/jpeg	31 mai 2022
turbidity=12_31-5-2022::13:11:58	375.1 KB	image/jpeg	31 mai 2022
turbidity=13_31-5-2022::13:11:43	371.49 KB	image/jpeg	31 mai 2022
turbidity=24_31-5-2022::13:15:46	312.02 KB	image/jpeg	31 mai 2022
turbidity=24_31-5-2022::13:15:57	304.38 KB	image/jpeg	31 mai 2022

To the right of the table, there is a preview of one of the images, showing a grid pattern. Below the preview, there is a summary of the file's properties:

- Nom: [turbidity=5_31-5-2022::10:39:26](#)
- Taille: 380 580 octets
- Type: image/jpeg

FIGURE 2.7 : Visualisation des images chargées dans le Cloud Firestore

2.2 Plateforme IoT : Spécification et choix technologiques

2.2.1 Fonctionnement IoT

Afin d'automatiser la tâche de mesure et d'enregistrement en temps réel des valeurs de la turbidité des différents essais, on conçoit une plateforme IoT dont le fonctionnement est dépendant de 3 composantes principales : une carte NodeMCU, un capteur de turbidité et une plateforme Arduino Cloud IoT. Cette architecture est mise en évidence par la figure 2.8.

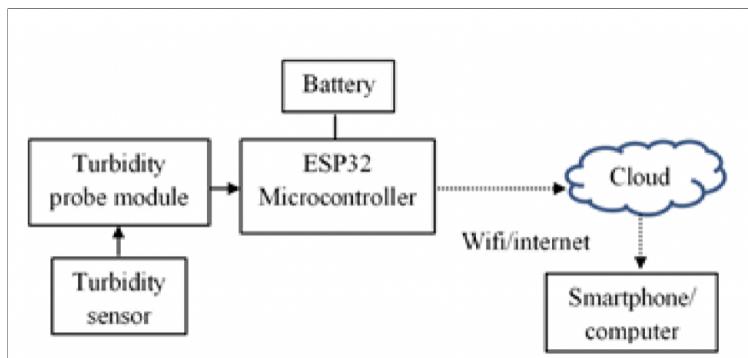


FIGURE 2.8 : Architecture générale du système

Initialement, le capteur de turbidité a une sortie comprise entre 0 et 1023. Le débit de communication est réglé sur 9600 bauds. Le microcontrôleur NodeMCU doit être connecté au réseau WiFi. Si la connexion est réussie, la carte ESP8266 est prête à envoyer les données via son module WiFi. La phase du calcul de la valeur de la turbidité est résumée dans le bloc de **Traitements des résultats** présent dans la figure 2.9. Le ratio de la lumière reçue doit être initialement converti en voltage et ensuite, avec une formule conventionnelle, on passe à l'échelle NTU.

Étant donné que les résultats retournés par le capteur sont parfois bruités, on calcule 100 valeurs à la fois et on détermine leur moyenne pour être plus précis. En fait, le bruit est essentiellement dû au mouvement des particules en suspension dans l'échantillon de l'eau qui influent directement sur la diffusion de la lumière et par la suite sur la valeur de la turbidité.

Selon la valeur reçue, on détermine la classe correspondante et on envoie les résultats ainsi que le voltage vers une feuille de calcul Google Spreadsheet et à la plateforme Arduino Cloud IoT à la fois.

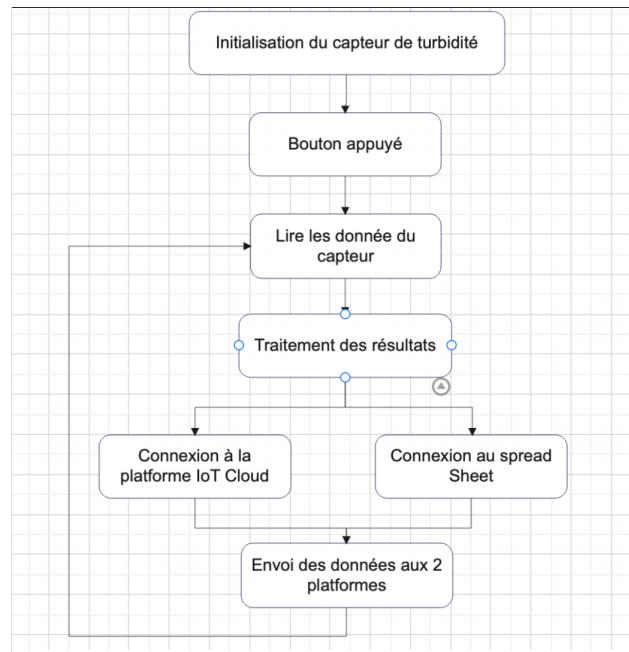


FIGURE 2.9 : Diagramme d'état de la plateforme

2.2.2 Choix du matériel

Une carte microcontrôleur Arduino est généralement associée au capteur pour effectuer les mesures de turbidité. Certes, on choisit de la remplacer par une carte NodeMCU [13] pour les raisons ci-dessous :

- Une meilleure performance en termes de mémoire et de temps de calcul CPU.
- Elle contient un module WiFi intégré ce qui va en adéquation avec les applications IoT.

En ce qui concerne le capteur de turbidité, il estime la qualité de l'eau en détectant les niveaux de turbidité. Pour ce faire, il utilise la lumière pour détecter les particules en suspension et ceci en mesurant le taux de transmission qui change en fonction de la quantité de solides dans l'eau. En effet, le niveau de turbidité de l'eau est proportionnel à cette quantité.

La figure ci-dessous illustre le fonctionnement entre les deux branches du capteur, chaque partie non dissoute dans l'eau affecte la transmission de la lumière émise qui diminue en fonction de la quantité des particules :

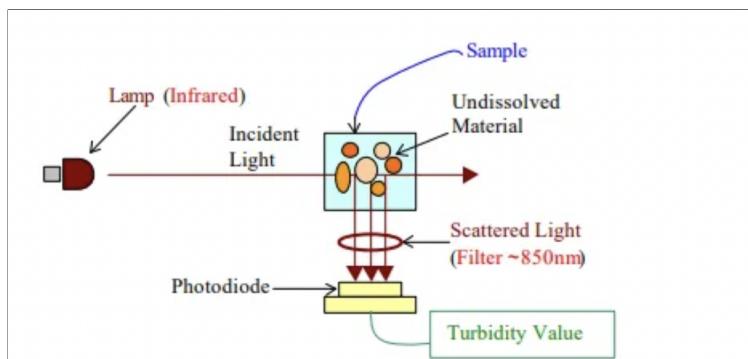


FIGURE 2.10 : Fonctionnement d'un capteur Nephelometer [14]

Limites du turbidimètre utilisé :

- Le turbidimètre est peu sensible aux particules de tailles infinitésimalles et donc il fournit des mesures moins précises.
- Le turbidimètre est incapable de différencier la lumière externe de celle émise par l'émetteur, ce qui mène à des lectures erronées de la valeur de la turbidité.
- Le turbidimètre est sensible aux bulles d'eau qui peuvent générer des faux signaux au détecteur.
- En plus, et en raison de la présence d'air intact dans l'échantillon, des bulles se forment ce qui génère des faux signaux aux détecteurs.

2.2.3 Traçabilité des données

Arduino Cloud IoT est une plateforme qui permet de créer des projets IoT, avec une interface conviviale. L'envoi des données est assuré par une requête HTTP à chaque appui sur le bouton, récupérée par un processus déployé au cloud. Dans notre cas, on peut surveiller les mesures effectuées par le turbidimètre grâce à un tableau de bord qui est généré automatiquement comme présenté dans la figure 2.11 :

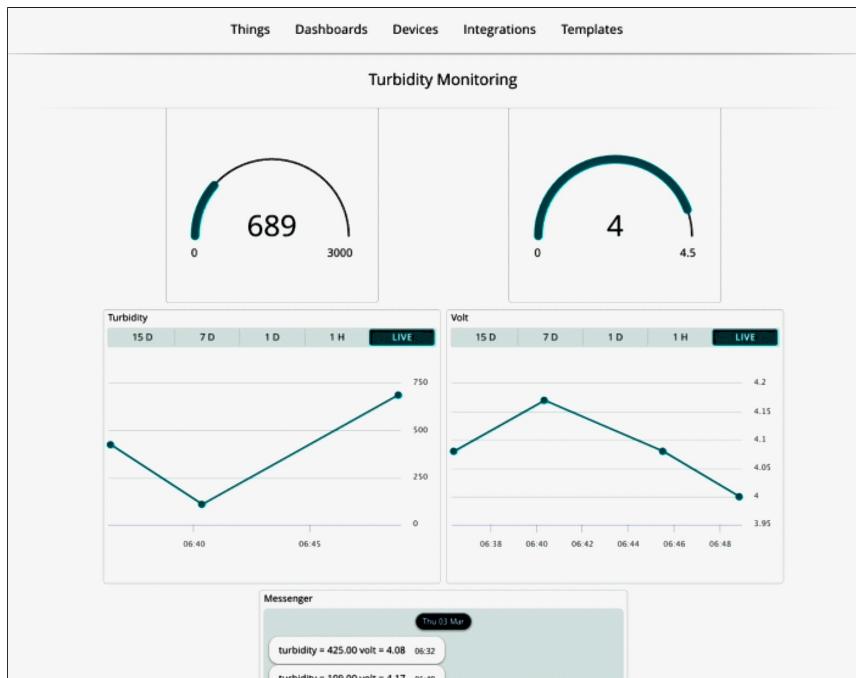


FIGURE 2.11 : Tableau de bord de la plateforme iot cloud

En outre, on recourt à une feuille de calcul dans google drive comme montré dans la figure 2.12, ce tableau sert à garder un historique des acquisitions pour une traçabilité et à faciliter la correction des mauvaises saisies surtout des valeurs de la turbidité dans la base données des images.

	A	B	C	D	E
1	Date	Time	Turbidity (ntu)	volt (v)	Label
2					
3	30/04/2022	12:51:41	4	4.20	very_Low_Turbidity
4	30/04/2022	12:57:24	77	4.18	very_Low_Turbidity
5	30/04/2022	1:00:08	41	4.19	very_Low_Turbidity
6	30/04/2022	1:09:39	52	4.19	very_Low_Turbidity
7	30/04/2022	1:11:48	102	4.17	very_Low_Turbidity
8	30/04/2022	1:29:27	174	4.15	Low_Turbidity
9	30/04/2022	1:51:22	4	4.20	very_Low_Turbidity
10	30/04/2022	1:51:39	4	4.20	very_Low_Turbidity
11	30/04/2022	10:52:32	0	4.20	very_Low_Turbidity
12	30/04/2022	10:56:10	0	4.20	very_Low_Turbidity
13	30/04/2022	11:00:23	0	4.20	very_Low_Turbidity
14	30/04/2022	12:24:40	0	4.20	very_Low_Turbidity
15	30/04/2022	12:38:43	287	4.12	Low_Turbidity
16	30/04/2022	12:39:13	287	4.12	Low_Turbidity
17	30/04/2022	12:39:43	287	4.12	Low_Turbidity
28	30/04/2022	1:53:40	1451	3.74	High_Turbidity
29	02/05/2022	6:21:05	73	4.18	very_Low_Turbidity

FIGURE 2.12 : Extrait des résultats de google spread sheet

Conclusion

En optant pour le développement de l'application mobile et la plateforme IoT, on rend le processus d'acquisition des images et de l'enregistrement de la valeur de la turbidité plus commode, et ceci avec des moyens peu cher comme la carte NodeMCU, le capteur de turbidité, et un outil gratuit open source qui est la plateforme Arduino Cloud IoT.

APPROCHE COMPUTER VISION POUR LA CLASSIFICATION DE L'EAU TURBIDE

Plan

1	Machine Learning	20
2	Deep Learning	25

Introduction

Le Deep Learning et en particulier les réseaux de convolutions sont très efficaces pour la classification des images car le concept de réduction de la dimensionnalité convient au grand nombre de paramètres d'une image. Cependant, et en raison de la taille réduite du dataset, on opte aussi pour l'implantation des approches automatiques supervisées utilisant des descripteurs de texture et de couleur en vue de comparer les résultats.

3.1 Machine Learning

Une image provenant d'une eau turbide présente un effet de flou, un faible contraste et des couleurs grisées en raison de l'effet de diffusion. En effet, plus l'eau est turbide, plus ses caractéristiques prennent de l'importance. Par conséquent, on suggère les caractéristiques ci-dessous comme descripteurs de couleur et de texture.

3.1.1 Les descripteurs de texture

La texture peut être définie comme une fonction de la variation spatiale de l'intensité de la luminosité des pixels. Elle est le principal terme utilisé pour définir les objets ou les concepts d'une image donnée. L'analyse de texture joue un rôle important dans des cas de vision par ordinateur tels que la reconnaissance d'objets, de formes, l'analyse d'images médicales. Nombreux descripteurs existent pour caractériser la texture. On s'intéresse dans notre projet à ces 3 caractéristiques :

- Haralick
- LBP (Local Binary Pattern)
- HOG (Histogram of Oriented Gradients)

3.1.1.1 Haralick

Les caractéristiques de texture de Haralick peuvent être utilisées pour distinguer les surfaces rugueuses des surfaces lisses. Elles sont dérivées de la matrice de cooccurrence des niveaux de gris (GLCM). Cette matrice enregistre le nombre de fois que 2 pixels de niveau de gris adjacents l'un à l'autre apparaissent dans une image. Puis, sur la base de cette matrice, Haralick propose 13 valeurs [15] qui peuvent être extraites de la GLCM pour quantifier la texture selon les 4 directions : Haut, bas, droite, gauche.

Dans notre approche, on va considérer toutes les directions parce qu'on s'intéresse aux intersections du motif selon les diagonaux pour extraire les informations pertinentes. On cite ci-dessous les formules

mathématiques utilisées pour extraire quelques indicateurs Haralick [15] :

$$Contrast : \sum_{n=0}^{N_g-1} n^2 \left\{ \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i, j) \right\}, |i - j| = n \quad Entropy : - \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i, j) \log(p(i, j))$$

$$Correlation : \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (ij)p(i, j) - \mu_x \mu_y}{\sigma_x \sigma_y} \quad Sum Variance : \sum_{i=2}^{2N_g} (i - f_8)^2 p_{x+y}(i)$$

3.1.1.2 Local Binary Pattern(LBP)

L'objectif du Local Binary Pattern (LBP) est de coder les caractéristiques géométriques d'une image en détectant les bords, les coins, les zones en relief ou plates et les lignes dures.

LBP sert à chercher l'information localement c'est ce qui font les couches convolutives des réseaux de neurones sauf que ça ne fonctionne pas avec un système de poids (d'apprentissage). On distingue deux paramètres clés du LBP :

- Rayon : à quelle distance on va chercher les points.
- Nombre de points voisins.

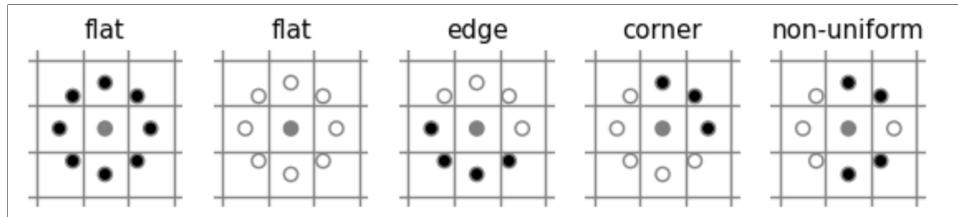


FIGURE 3.1 : Caractérisation locale des régions par LBP [16]

Le pixel central est utilisé comme seuil pour le pixel voisin, et le code LBP d'un pixel central est généré en codant la valeur de seuil calculée en une valeur décimale.

L'expression mathématique du LBP [17] est donnée comme suit :

$$LBP = \sum_{i=0}^{P-1} s(n_i - G_c) 2^i \quad s(x) = \begin{cases} 1, & \text{si } x > 0 \\ 0, & \text{sinon} \end{cases}$$

où P est le nombre de pixels voisins, n_i représente le i ème pixel voisin, et c représente le pixel central. Les caractéristiques de l'histogramme de taille $2P$ sont extraites du code LBP obtenu.

3.1.1.3 Histogram of Oriented Gradients(HOG)

L'histogramme des gradients orientés est utilisé en vision par ordinateur et en traitement d'images pour la détection d'objets. La technique compte les occurrences de l'orientation du gradient dans la partie localisée d'une image. HOG agit comme une compression de l'image condensant ainsi une grande quantité d'informations. Dans le descripteur de caractéristiques HOG, la distribution (histogrammes) des directions des gradients (gradients orientés) est utilisée comme caractéristique. Les gradients (dérivés x et y) d'une image sont utiles parce que l'ampleur des gradients est importante autour des bords et des coins (régions où l'intensité change brusquement) et nous savons que les bords et les coins contiennent beaucoup plus d'informations sur la forme de l'objet que les régions plates.

3.1.2 Les descripteurs de couleur

Pour les descripteurs de couleur, on choisit les canaux BGR et HSV pour déterminer la moyenne et l'écart-type de chaque canal dans ces deux représentations, ainsi que les histogrammes de chaque image. Nous travaillons également sur les points suivants : la distribution de chaque couleur, la netteté, la coloration, le contraste, ainsi que la luminosité de l'image.

3.1.2.1 Représentations RGB et HSV

On choisit de travailler sur les canaux RGB ainsi que HSV puisque cette dernière est moins sensible aux changements de luminosité. Nous calculons la moyenne et l'écart-type de chaque canal pour ces deux représentations.

3.1.2.2 Détermination de la distribution de chaque couleur

La proportion de la couleur par rapport à la somme de toutes les couleurs aide à mettre l'accent sur les couleurs dominantes dans l'image.

3.1.2.3 Détermination de la netteté de l'image

La mesure de la netteté [11] dans l'image permet d'estimer la quantité de turbidité qui disperse la lumière. La netteté étant un attribut lié à la préservation des détails fins et des bords, elle est mesurée aux bords de l'image. Pour ce faire, l'algorithme de détection de contours Sobel est d'abord appliqué à l'image en niveaux de gris, ce qui produit l'image des bords.

Si la turbidité de l'eau est très faible, les bords apparaîtront plus nets. On sait que le calcul d'estimation de la mesure d'amélioration (EME) convient aux images dont le fond est uniforme et qui ne présentent pas de motifs périodiques. Par conséquent, la mesure EME est utilisée pour mesurer la netteté des bords. L'image

e est divisée en k_1k_2 blocs et la mesure EME est définie comme suit :

$$EME = \frac{2}{k_1k_2} \sum_{m=1}^{k_1} \sum_{n=1}^{k_2} \log\left(\frac{e_{max,m,n}}{e_{min,m,n}}\right)$$

Où $e_{max,m,n}$ et $e_{min,m,n}$ sont le minimum et le maximum de l'image à l'intérieur du bloc. Pour chaque bloc de (m, n) dans (k_1, k_2) blocs, le rapport de contraste sous forme logarithmique est défini comme suit :

$$CR_{k,l} = \frac{e_{max,m,n}}{e_{min,m,n}}$$

3.1.2.4 Détermination de la coloration de l'image

La couleur peut devenir grise car elle ne peut pas parcourir une longue ou même courte distance, dans une eau turbide. Plus l'eau est turbide, moins l'image est colorée par rapport à une image à faible turbidité. Comme la couleur rouge disparaît en premier parce qu'elle possède la longueur d'onde la plus courte, les images d'eau ont généralement un aspect bleuâtre ou verdâtre.

Par conséquent, les deux composantes de couleur opposées liées à la chrominance RG et YB sont [18] :

$$RG = R - G \quad YB = \frac{R + G}{2} - B$$

Nous procédons ensuite au calcul de la moyenne et de l'écart-type du RG et du YB , pour ensuite calculer la racine moyenne, la racine de l'écart-type et la métrique de couleur de la façon suivante [18] :

$$\sigma_{rgyb} = \sqrt{\sigma_{rg}^2 + \sigma_{yb}^2} \quad \mu_{rgyb} = \sqrt{\mu_{rg}^2 + \mu_{yb}^2} \quad C = \sigma_{rgyb} + 0.3 * \mu_{rgyb}$$

3.1.2.5 Détermination du contraste de l'image

Lorsque le signal de l'image est atténué et que l'effet de rétrodiffusion domine, la structure originale de l'image est naturellement perdue ce qui mène à une dégradation du contraste. La mesure du contraste peut également donner des informations sur la turbidité puisque le contraste sera faible en cas de forte turbidité et élevé en cas de faible turbidité. Nous pouvons mesurer le contraste de l'image en calculant l'écart type de la version en niveaux de gris de l'image.

3.1.2.6 Détermination de la luminosité de l'image

Nous pouvons mesurer la luminosité de l'image en calculant la moyenne de la version en niveaux de gris de l'image.

3.1.3 Classificateurs choisis

3.1.3.1 Support Vector Machine

Le principe des SVM [19] consiste à ramener un problème de classification ou de discrimination à un hyperplan (feature space) dans lequel les données sont séparées en plusieurs classes dont la frontière est la plus éloignée possible des points de données.

La figure 3.2 montre séparation par un hyperplan entre deux classes selon SVM.

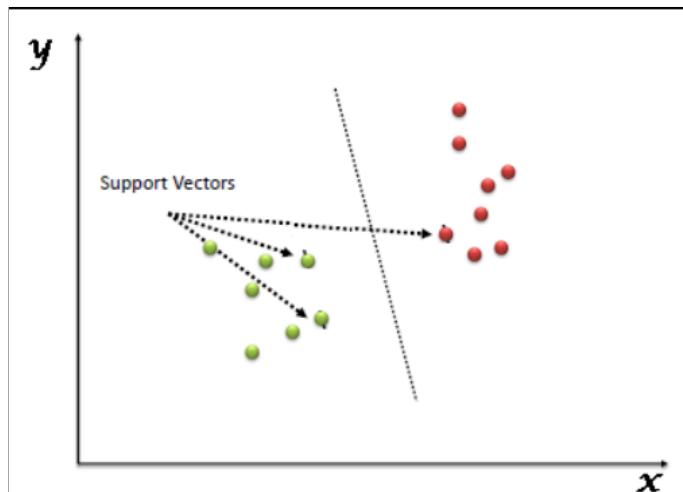


FIGURE 3.2 : Support Vector Machine [20]

3.1.3.2 Random Forest

Random Forest est une méthode d'apprentissage ensembliste pour la classification ou régression qui génère une multitude d'arbres de décision au cours d'apprentissage. Pour les tâches de classification, la sortie du Random Forest est la classe sélectionnée par la plupart des arbres. Ceci est mis en relief par la figure 3.3 :

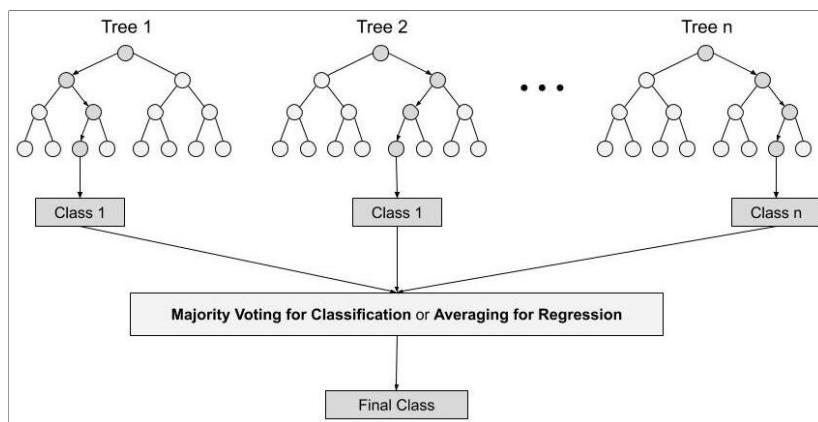


FIGURE 3.3 : Random Forest [21]

3.1.3.3 XGBoost

La bibliothèque XGBoost [22] met en œuvre l'algorithme d'arbre de décision par boosting de gradient. Le boosting est une technique d'ensemble où de nouveaux modèles sont ajoutés pour corriger les erreurs commises par les modèles existants. Les modèles sont ajoutés séquentiellement jusqu'à ce qu'aucune amélioration ne puisse être apportée. Le fonctionnement de XGBoost est illustré par la figure 3.4.

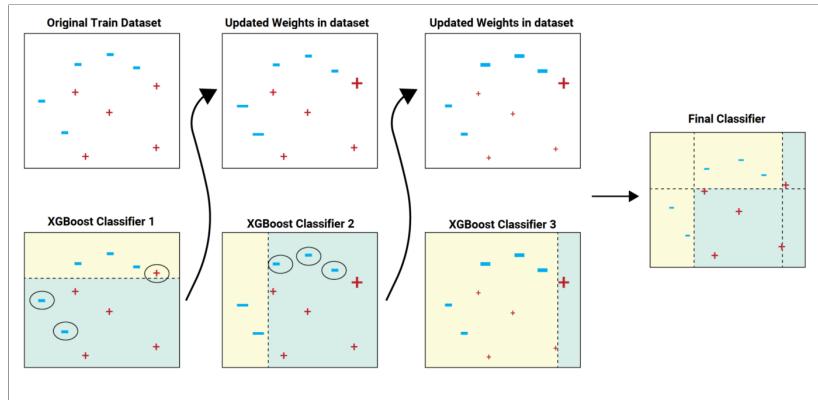


FIGURE 3.4 : XGBoost [23]

3.2 Deep Learning

L'apprentissage profond, qui est basé sur les réseaux de neurones est une forme d'intelligence artificielle, dérivée de Machine Learning. Ce n'est que récemment, grâce à l'avancée des performances de calcul des ordinateurs, que ce concept s'est développé. Chaque type de réseau de neurones excelle dans la résolution d'un domaine spécifique de problèmes, et chacun est réglé avec des hyper-paramètres qui optimisent les performances.

Passons maintenant à l'explication du fonctionnement de l'un des types des réseaux de neurones dédié aux applications de computer vision, les réseaux de convolution CNNs.

3.2.1 Les réseaux de neurones convolutifs : Définitions et principes généraux

Les CNNs sont souvent utilisés en cas de traitements d'images, car les chercheurs se sont rendu compte de leur aptitude d'extraire précisément les caractéristiques des images. Dans un exemple célèbre d'utilisation des CNNs, Krizhevsky et al [24] ont mis en œuvre un réseau avancé qui a atteint une précision record sur le jeu de données ImageNet composé de 1,2 million d'images .

3.2.1.1 Principe général des CNNs

L'architecture d'un réseau CNN comprend deux parties principales :

- Un outil de convolution qui sépare et identifie les différentes caractéristiques de l'image.
- Des couches entièrement connectées (à la fin de l'architecture) pour la tâche de classification basée sur les caractéristiques déduites.

Les réseaux de neurones convolutifs sont basés sur les calculs de convolution et ont parfois une structure profonde, comprenant des couches convolutives, de pooling et des couches entièrement connectées(fully connected layers).

L'objectif de la convolution est d'extraire les différentes caractéristiques de l'entrée, ce qui réduit la quantité de mémoire occupée par le réseau profond, définie par la complexité spatiale. La couche de pooling peut être très efficace pour réduire la taille de la matrice de poids, comprimer les données pour la réduction des caractéristiques, et ainsi réduire le nombre de paramètres dans la couche finale entièrement connectée.

Un processus classique d'application de CNN est présenté ci-dessous :

$$r = \text{FC}(\text{Activate}(\text{Pool}(\text{Conv}(X))))$$

où Conv désigne la couche de convolution, Pool désigne la couche de pooling, Activate désigne la fonction d'activation telle que ReLu ou Softmax, FC désigne la couche entièrement connectée.

Les 3 premières couches font correspondre les données originales à l'espace des caractéristiques cachées à extraire, tandis que la couche entièrement connectée fait correspondre la représentation des caractéristiques appris pour but de classification. En d'autres termes, elle joue le rôle de classifieur dans l'ensemble du réseau neuronal convolutif.

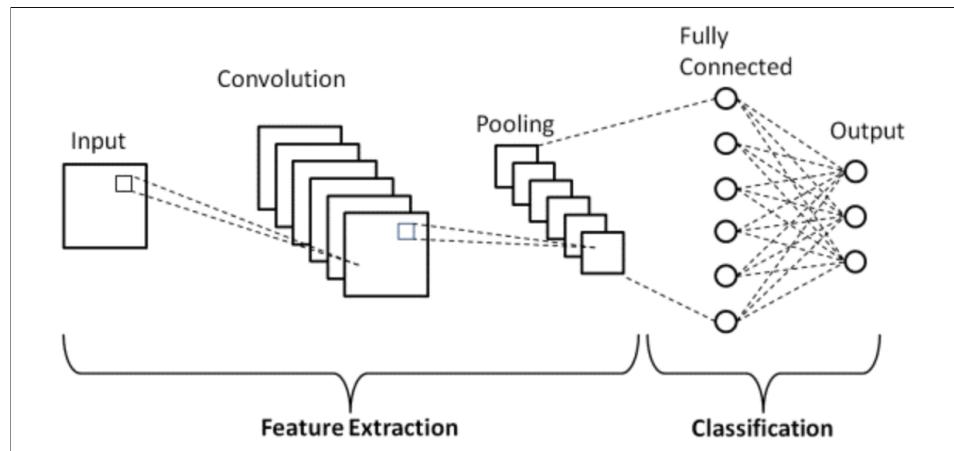


FIGURE 3.5 : Architecture des réseaux CNN [25]

3.2.1.2 Avantages des CNNs

Les CNNs visent à limiter le nombre d'entrées tout en conservant la forte corrélation spatiale et locale dans l'image. En plus, le partage des paramètres de filtrage entre les différents neurones d'un même filtre permet d'avoir la propriété d'invariance de traitement par translation.

- ⇒ Les valeurs du filtre (le patch) doivent être partagées par tous les neurones de la couche de convolution.
- ⇒ Cela signifie également que le calcul de la sortie nécessite moins d'opérations, ce qui réduit encore davantage les exigences de stockage du modèle.

3.2.2 Concepts utiles en Deep Learning

Quelques couches et techniques d'optimisation seront indispensables pour améliorer l'apprentissage :

- **Couche Batch Normalization**

Batch normalization est une technique qui normalise les entrées d'une couche pour chaque lot. Cela a pour effet de stabiliser le processus d'apprentissage et de réduire considérablement le nombre d'époques d'apprentissage nécessaires pour former des réseaux profonds efficaces.

La normalisation consiste à redimensionner les données pour qu'elles aient une moyenne de 0 et un écart-type de 1 par exemple pour une gaussienne standard.

- **Couche Dropout**

En général, lorsque toutes les caractéristiques extraites sont connectées à la couche FC, cela peut résulter en un sur-apprentissage. En effet, il se produit lorsqu'un modèle fonctionne si bien sur les données d'apprentissage mais non plus sur celles de test.

Pour surmonter ce problème, on utilise une couche de Dropout dans laquelle quelques neurones sont éliminés du réseau neuronal pendant le processus d'apprentissage, ce qui réduit la taille du modèle.

Une variante de ce type de couche est **Gaussian Dropout** [26]. Au lieu d'éliminer des nœuds pendant l'apprentissage, le poids de chaque nœud est éliminé en ajoutant du bruit pendant cette phase, ce qui réduit la complexité de calcul au test. Pour ce cas, le bruit suit une forme gaussienne.

- **Early stop**

C'est une technique d'optimisation utilisée pour réduire le surapprentissage sans affecter la précision du modèle. Si les performances du modèle sur l'ensemble de données de validation commencent à se dégrader (par exemple la perte commence à augmenter), le processus d'apprentissage est arrêté.

3.2.3 Le modèle ResNet

Les réseaux convolutifs traditionnels et profonds souffrent de la perte d'information, de la disparition du gradient (vanishing gradient) ou de l'explosion du gradient pendant le transfert d'information, ce qui rend impossible l'entraînement de réseaux très profonds. Le ResNet (Residual Neural Network), proposé par l'équipe de Kaiming He [27], est une bonne solution à ce problème. L'idée principale est d'utiliser des unités résiduelles pour permettre aux informations d'entrée de passer directement à la couche suivante, comme le montre la figure 3.6 ci-dessous, afin de protéger l'intégrité des informations.

⇒ Le réseau entier n'a besoin d'apprendre que les différences d'entrée et de sortie, ce qui simplifie les objectifs et la difficulté d'apprentissage, tout en accélérant l'apprentissage et en améliorant la précision du modèle.

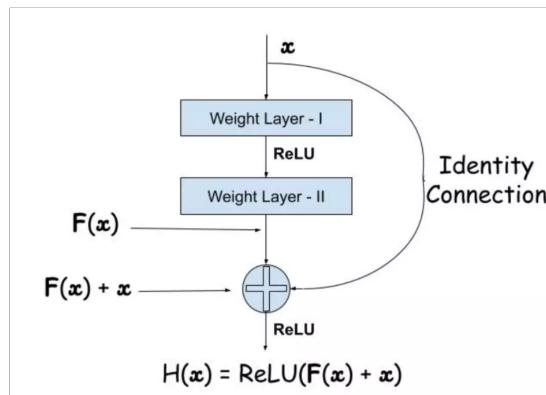


FIGURE 3.6 : Représentation d'un bloc résiduel [28]

Souvent, et étant que quelques types de ces réseaux sont très utilisés et très repandues, un type particulier d'apprentissage est généralement utilisé surtout lorsqu'on est dans le cas de données réduites, on parle de l'apprentissage par transfert.

Apprentissage par transfert :

Il s'agit de transférer le poids de chaque noeud du réseau d'un réseau pré-entraîné à un tout nouveau réseau, au lieu de commencer de zéro. En fait, la plupart des tâches sont liées d'une problématique à une autre et grâce à l'apprentissage par transfert, nous pouvons partager les paramètres du modèle appris sur une grande base d'apprentissage avec le nouveau modèle, afin qu'il puisse s'adapter plus rapidement, ce qui peut accélérer la convergence du modèle et améliorer l'efficacité de l'apprentissage.

3.2.4 Modèles profonds implémentés

On essaie des différentes architectures CNN et on compare les résultats obtenus.

3.2.4.1 Modèle Aquasight modifié

On recourt à une autre approche basée sur les résultats de [1], ci-dessous deux architectures profondes qui illustrent le nouveau modèle inspiré de celui utilisé précédemment :

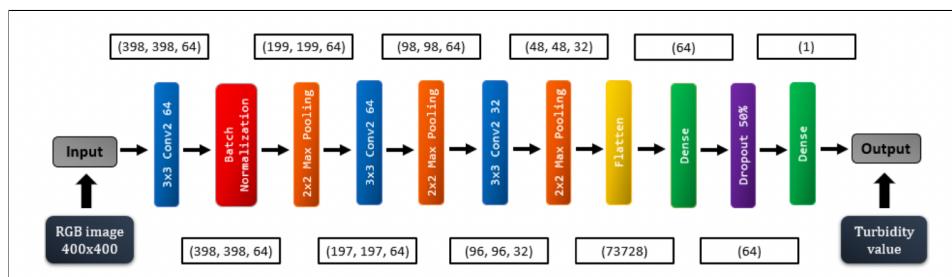


FIGURE 3.7 : Modèle Aquasight [1]

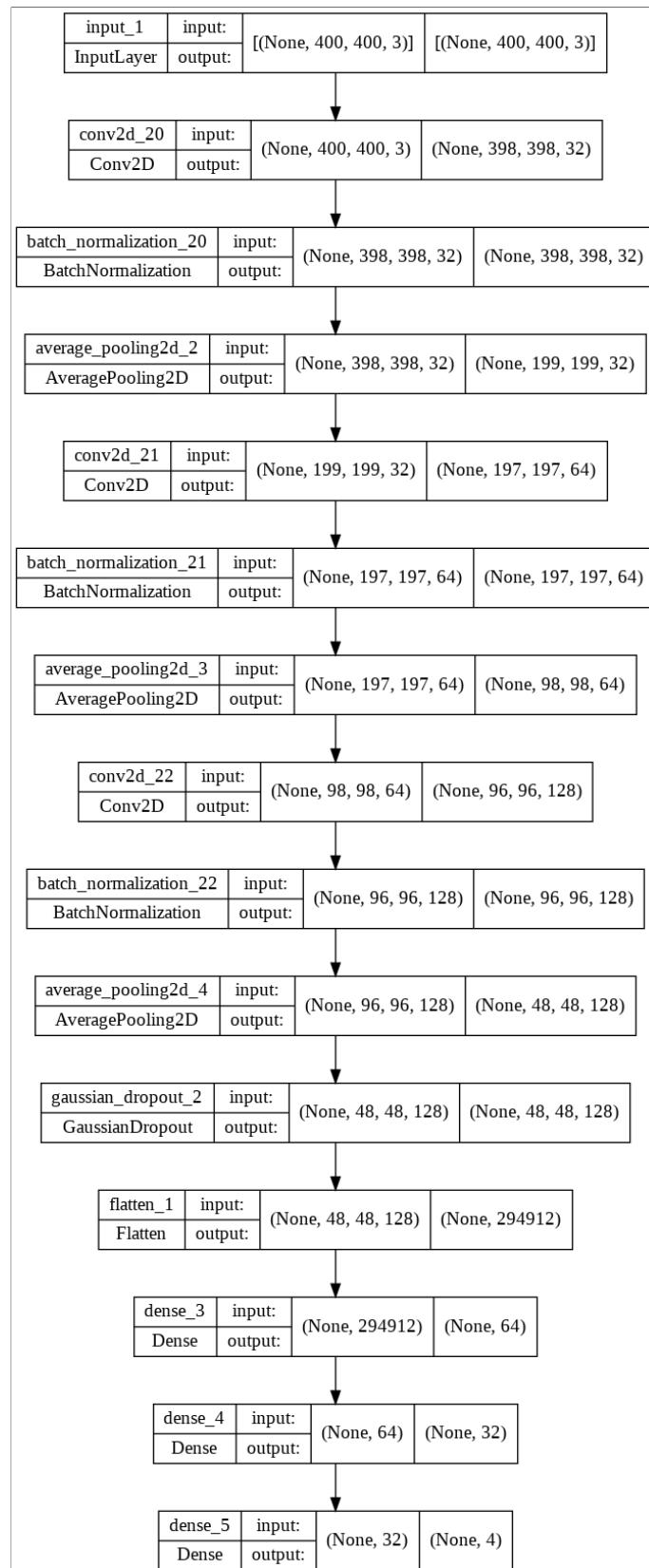


FIGURE 3.8 : Modèle Aquasight modifié

La clé pertinente de ces changements peut se résumer dans les couches de batch normalization qui suivent chaque couche de convolution.

3.2.4.2 Modèle ResNet

Par la suite, on utilise le modèle Resnet, et on étudie la méthode de la classification basée sur l'apprentissage par transfert. Trois techniques différentes d'apprentissage par transfert ont été élaborées, décrites comme suit :

- Apprentissage classique : L'entraînement du modèle Resnet 18 à partir de zéro sans utiliser les poids pré-entraînés.
- Couches non entraînées : Lors de l'entraînement du modèle, toutes les couches de convolution sont figées et on met à jour seulement les paramètres des couches de classification.
- Couches entraînées : On utilise les poids pré-entraînés comme **initialisation**. Les couches de convolution sont entraînées sur nos données de base ainsi que la couche entièrement connectée, et on met à jour les poids, dans le but de permettre au modèle de mieux extraire les caractéristiques spécifiques de nos données.

3.2.4.3 Modèle Ex nihilo

On crée une architecture formée de 4 couches de convolution suivies des couches de Average Pooling pour l'extraction des caractéristiques. Une couche de Gaussian Dropout avec un paramètre de 0.4 sera très utile pour minimiser l'effet de sur-apprentissage qui est très fréquent pour des bases de petites tailles, et qui est le cas ici. Pour la partie du classifieur, 3 couches fully connected sont utilisées pour prédire la classe.

Conclusion

Durant nos travaux de recherche, on a décelé les descripteurs de l'image les plus significatifs en ce qui concerne la texture et la couleur pour les utiliser en l'apprentissage automatique. On a aussi essayé certaines architectures Deep Learning dont les résultats obtenus seront dévoilés dans le chapitre suivant.

EXPÉRIMENTATIONS ET RÉSULTATS

Plan

1	Acquisition des données et organisation du dataset	32
2	Résultats de classification	36
3	Test sur le nouveau dataset	46

Introduction

Dans ce chapitre, on présente l'organisation des deux datasets, celle du travail de [1] et la nôtre. Par la suite, on passe aux techniques de prétraitement adoptées ainsi que les résultats obtenus lors de l'expérimentation des modèles Machine Learning et architectures Deep Learning.

4.1 Acquisition des données et organisation du dataset

4.1.1 Dataset d'apprentissage

Le dataset utilisé pour l'apprentissage est le résultat de l'acquisition faite par [1]. Il est composé de 338 images utilisant un motif. Toutefois, la correction de mauvaises saisies et l'élimination des duplicates résultent en une base de 286 images annotées et divisées comme suit :

- La classe 1 : Contient 54 images qui couvrent le spectre de turbidité de 0 à 150 UTN où la turbidité est très faible.
- La classe 2 : Contient 55 images de 151 à 400 UTN, l'eau est plutôt turbide, il est même
- La classe 3 : Contient 61 images variant de 401 à 700 UTN, l'eau est assez turbide, les lignes du motif ainsi que leurs croisements ne sont pas claires.
- La classe 4 : Contient 116 images couvrant le spectre de turbidité d'au-delà de 700 UTN, une très forte turbidité où le motif devient invisible à l'œil nu.

Puisque l'augmentation artificielle de données avait un effet négatif sur les performances des modèles comme mentionné par [1], on recourt à la technique de **rehaussement** des images pour avoir duplication du nombre des images. Ceci étant dans le but d'avoir des résultats plus pertinents. La technique utilisée pour la restauration est fastNLMeansDenoisingColored de OpenCV. Ceci étant le point de départ, l'application d'un rehaussement d'images résulte en une base de 676 images partitionnées comme suit :

classe1 : 108 classe2 : 110 classe3 : 122 classe4 : 232

On note que 70% de cette base sera allouée à l'apprentissage des modèles, 15% pour la validation et 15% pour le test.

Soit la figure 4.1 ci-dessous qui illustre la répartition des images d'apprentissage de chaque classe :

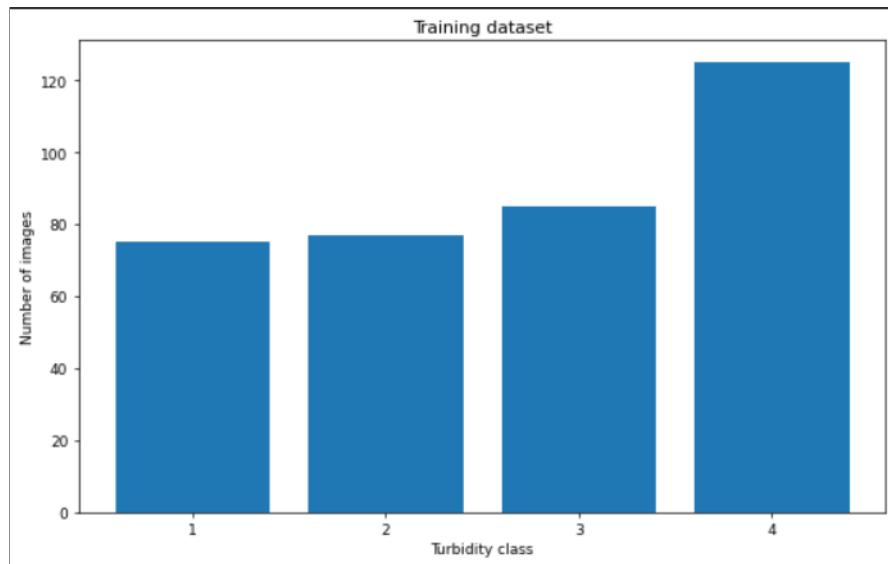


FIGURE 4.1 : Répartition des données d'apprentissage en fonction des classes

4.1.2 Nouveau dataset

Dans le but d'enrichir la base de données existante, on procède à l'ajout de nouvelles acquisitions grâce à l'application mobile et à la plateforme IoT.

4.1.2.1 Le motif : Modifications apportées

La proposition finale du motif est passée par plusieurs étapes. En premier lieu, on suggère une première vision comme la montre la figure 4.2 pour le motif qui renferme des lignes horizontales et verticales qui se croisent, et dont la taille diminue de bas en haut et de droite à gauche. On s'intéresse aux intersections parce qu'on estime que le modèle apprend les attributs les plus significatifs grâce à la progression sur les diagonaux. Certes, il n'était pas aussi efficace puisque le blanc domine et il est facilement caché. Pour y remédier, on inverse les couleurs pour diminuer la surface blanche et on opère une dilatation pour augmenter les lignes comme présenté par la figure 4.3 :

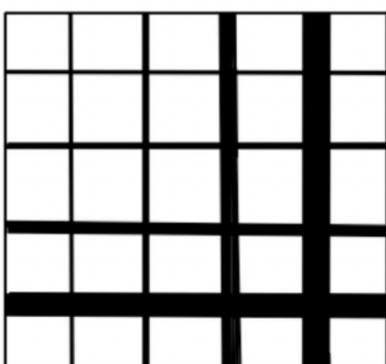


FIGURE 4.2 : 1ère proposition de motif

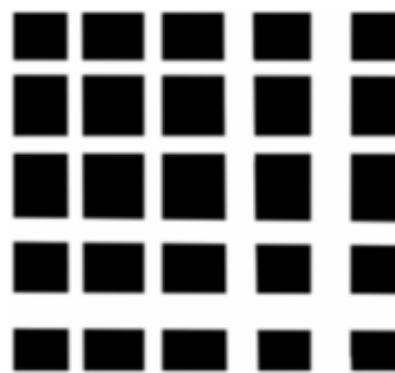


FIGURE 4.3 : Forme du motif après inversion et dilatation

4.1.2.2 Protocole d'acquisition et dataset obtenu

L'acquisition est faite en ajoutant minutieusement une variété de sédiments à l'eau. Puis, on note la valeur de la turbidité, et on capture une image de l'eau avec le motif placé en arrière-plan (avec l'application mobile) simultanément. Le processus se fait comme suit :

- Ajouter des sédiments à l'eau, principalement de l'argile, ou bien du lait ou du café.
- Mélanger le résidu et attendre 2 minutes, le temps que l'échantillon se stabilise, pour que le capteur n'enregistre pas de hautes valeurs.
- Faire plusieurs prises de photos pour la même valeur enregistrée.
- Ajouter des faibles quantités de sédiments et faire plusieurs acquisitions pour chaque ajout.
- Vérifier l'exactitude des résultats des acquisitions et leur correspondances avec les images.

Les 3 figures suivantes illustrent un extrait des images obtenues :

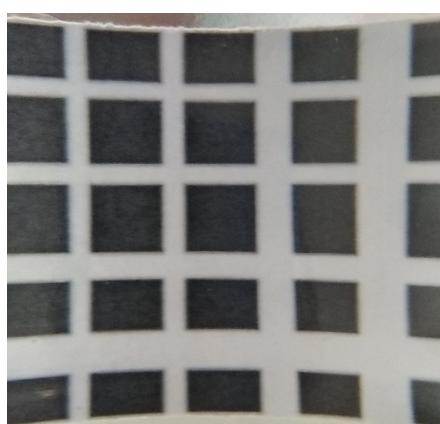


FIGURE 4.4 : Turbidité = 5NTU



FIGURE 4.5 : Turbidité = 58NTU



FIGURE 4.6 : Turbidité = 129NTU

4.1.3 Pré-traitement

4.1.3.1 Machine Learning

Les descripteurs de texture comme LBP et HOG, et les descripteurs de couleur comme l'histogramme BGR et l'histogramme HSV génèrent un nombre énorme de features. Afin de le réduire, on compare l'histogramme de l'image en question dans les deux représentations à celui de la première image de chaque classe de turbidité qui sert de référence (la première image de chaque classe est la plus claire et représente la meilleure qualité comparée aux autres membres de la classe). On utilise un score de similarité entre les histogrammes assuré par la fonction `compareHist` [29] de OpenCV qui prend en argument l'histogramme de référence, l'histogramme de l'image en question et une métrique. On choisit comme métrique la corrélation qui réalise d'une façon

efficace le calcul du score dont la formule est [29] :

$$d(H_1, H_2) = \frac{\sum_I (H_1(I) - \bar{H}_1)(H_2(I) - \bar{H}_2)}{\sqrt{\sum_I (H_1(I) - \bar{H}_1)^2 \sum_I (H_2(I) - \bar{H}_2)^2}} \quad \bar{H}_k = \frac{1}{N} \sum_J H_k(J)$$

Notre dataframe renferme ainsi un ensemble de 87 attributs dont les indicateurs sont mentionnés dans le tableau suivant :

Descripteurs de texture	Descripteurs de couleur
Haralick : Second Moment Angulaire	Moyenne des canaux RGB
Haralick : Contrast	Ecart type RGB
Haralick : Corrélation	Moyenne des canaux HSV
Haralick : Variance	Ecart type des canaux HSV
Haralick : Différence Inverse du Moment	Similarités RGB Classes 1,2,3,4
Haralick : Moyenne de la somme	Similarités HSV Classes 1,2,3,4
Haralick : Somme des variances	Proportions RGB
Haralick : Somme des entropies	Coloration
Haralick : Entropie	Luminance
Haralick : Difference de la Variance	Netteté
Haralick : Difference de l'Entropie	
Haralick : Measure de correlation 1	
Haralick : Measure de correlation 2	
Similarités classes 1,2,3,4 LBP	
Similarités classes 1,2,3,4 HOG	

4.1.3.2 Deep Learning

Afin d'obtenir une meilleure classification, un prétraitement des images est nécessaire, notamment l'exécution d'un processus de normalisation.

L'opération spécifique est la suivante : la valeur moyenne de l'intensité de l'image est soustraite de chaque valeur de pixel. Dans de nombreux cas, on n'est pas sensible à la luminosité de l'image mais on prête davantage attention à son contenu. Dans la tâche de classification des images, la luminosité globale n'affecte

pas les objets qui existent, il est donc utile de soustraire la valeur moyenne d'intensité de chaque pixel. La normalisation des données consiste à mettre à l'échelle les données en proportion d'un petit intervalle spécifique. La méthode de traitement d'un échantillon utilise les fonctions suivantes :

$$\mu = \frac{i=1}{n} \sum_{i=1}^n x_i \quad \sigma = \frac{i=1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

où x_i désigne la valeur du i ème pixel de l'échantillon. n est le nombre total de pixels, μ est la moyenne et σ^2 est la variance.

La formule est présentée par :

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2}}$$

\hat{x}_i désigne la valeur normalisée du i ème pixel de l'échantillon.

4.1.4 Apprentissage

(i) Environnement d'exécution :

Pour exécuter l'apprentissage des modèles choisis, on utilise Google Colaboratory, qui est un produit de Google Research. Google Colab permet d'écrire et d'exécuter du code python via le web, et est particulièrement bien adapté à l'apprentissage automatique et à l'analyse de données.

(ii) Les hyperparamètres choisis :

Lors de l'apprentissage des modèles, les hyperparamètres, explicitement spécifiés, contrôlent le processus d'entraînement et agissent directement sur les résultats de la validation du modèle. Après plusieurs essais, ils sont fixés à ces valeurs qui donnent globalement la meilleure performance :

- Optimizer : adagrad
- Loss : categorical_crossentropy
- Learning rate : 0.01
- batch_size = 20

4.2 Résultats de classification

4.2.1 Machine Learning

Comme mentionné dans le chapitre 3, on a besoin d'extraire des descripteurs numériques de texture et de couleur pour entraîner les modèles d'apprentissage automatique. Ceci est réalisé à l'aide des bibliothèques

conçues pour la vision par ordinateur comme OpenCV, Scikit-image, PIL et Mahotas.

Il est à noter que les modèles entraînés au dataset initial ne montrent pas une bonne performance et n'ont pas dépassé un score d'accuracy 0.82.

⇒ Pour le reste des expérimentations, on opte pour le dataset rehaussé.

4.2.1.1 Support Vector Machine

Après l'augmentation des données, SVM donne un score d'accuracy de 0.77 et 0.78 par validation croisée de fold=20. Comme le montre la matrice de confusion ci-dessous, contrairement au cas où on a un dataset non augmenté, une confusion apparaît lors de la classification des images de très faible turbidité avec un pourcentage de 10%, une autre pour les images de faible turbidité avec un pourcentage de 33% ainsi que 51% pour les images de turbidité moyenne et 14% pour les images de haute turbidité.

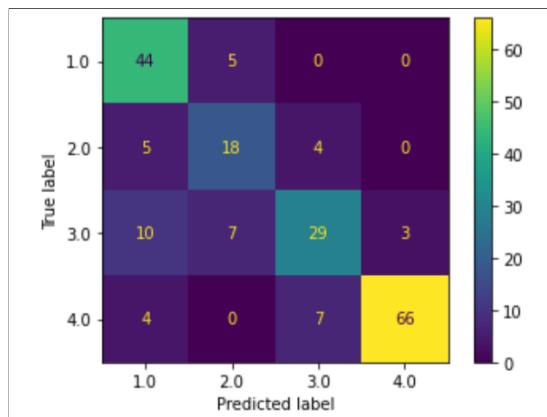
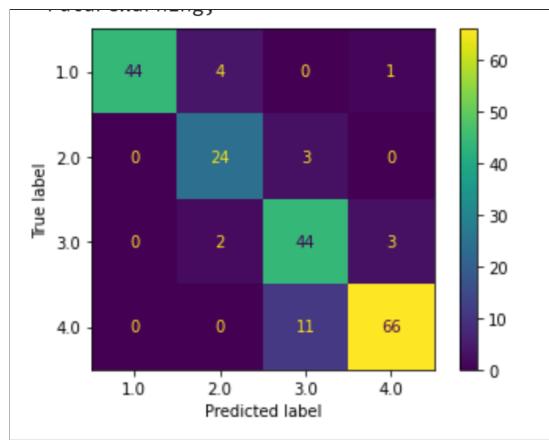


FIGURE 4.7 : Matrice de confusion - SVM après rehaussement

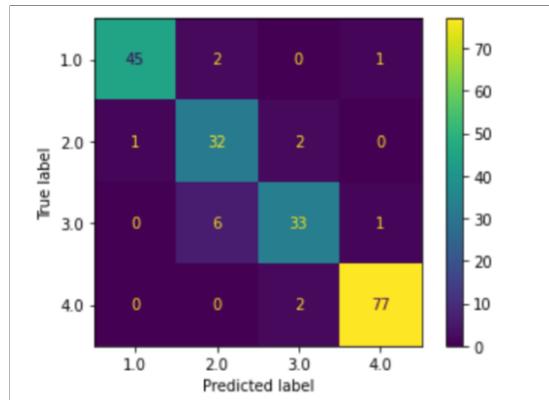
4.2.1.2 Random Forest

On obtient après dédoublement des données 0.88 comme score d'accuracy qui devient égal à 0.90 après une validation croisée pour k= 20 folds. Comme le montre la matrice de confusion ci-dessous, l'augmentation de données diminue la confusion dans la distinction des images de la 2ème classe bien qu'il y a une confusion dans la classification des images de haute turbidité de 14%.

**FIGURE 4.8 :** Matrice de confusion - Random Forest après rehaussement

4.2.1.3 XGBoost

Avec XGBoost, on obtient une accuracy de 0.92 et 0.9 avec une Cross Validation de fold=20. La matrice de confusion de ce modèle montre une bonne performance dans la classification des images de 1ère et 2ème classes. Cependant, on remarque une confusion dans la de classification des images de turbidité moyenne de 17%.

**FIGURE 4.9 :** Matrice de confusion - XGBoost après rehaussement

4.2.1.4 Comparaison des performances des classifieurs

Le tableau ci-dessous résume les résultats finaux de test des modèles Machine Learning sur les données augmentées proposés selon 3 métriques : Accuracy, Recall et F1-Score :

Modèle	Accuracy	Recall	F1-Score
SVM	0.78	0.75	0.74
Random Forest	0.9	0.89	0.88
XGBoost	0.92	0.91	0.91

On peut déduire de l'analyse des matrices de confusion et des résultats proposés par ce tableau que le modèle SVM n'est pas tout à fait adéquat pour la tâche de classification des images des eaux turbides. En effet, il propose un taux d'accuracy et de recall relativement faibles par rapport aux deux autres modèles. On décèle une confusion dans la classification par SVM des images de haute turbidité et de turbidité moyenne. C'est ce qui réduit le niveau de confiance en ce modèle.

En ce qui concerne Random Forest et XGBoost, ils présentent des performances comparables et un bon score d'accuracy. Toutefois, XGBoost est le modèle le plus adéquat à cette classification des images turbides. En effet, il présente le score Recall le plus élevé, c'est à dire qu'il confond moins les résultats que Random Forest.

4.2.2 Deep Learning

Pour toutes les architectures ci-dessous, on choisit d'unifier la taille des images d'entrée, soit donc 400x400 pixels.

4.2.2.1 Architecture InceptionResNet

Puisque les expériences dans [30] on conclu que InceptionResNet est le modèle d'apprentissage profond le plus approprié pour la classification de la turbidité de l'eau, on opte pour l'essayer sur notre dataset avant l'augmentation des données.

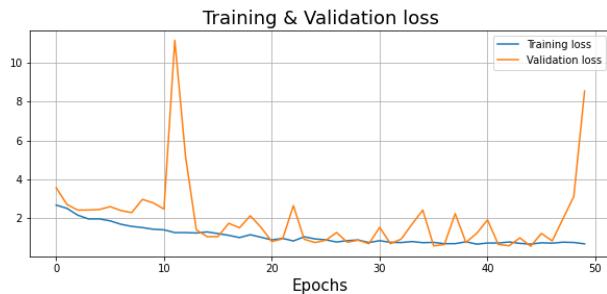


FIGURE 4.10 : Variation des valeurs de l'accuracy - InceptionResNet

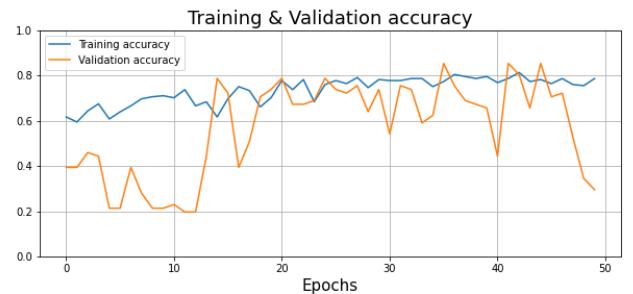


FIGURE 4.11 : Variation des valeurs du loss - InceptionResNet

Cependant, les figures 4.10 et 4.11 montrent que le modèle présente des fluctuations lors de l'apprentissage. La raison de cette instabilité de l'accuracy de la validation est que ce modèle est très profond par rapport aux données sur lesquelles il est entraîné.

⇒ Ainsi, InceptionResNet n'est pas adéquat pour ce dataset réduit, car sa taille est trop petite de sorte que des petits changements dans la sortie entraînent de grandes variations dans l'erreur de la validation.

4.2.2.2 Architecture ResNet18 : 1ère approche

- Avant le rehaussement du dataset

Après l'essai de plusieurs modèles d'apprentissage profond sur le dataset avant restauration des images, on

trouve que ResNet18 est le plus performant.



FIGURE 4.12 : Variation des valeurs de l'accuracy - ResNet18 - Avant augmentation

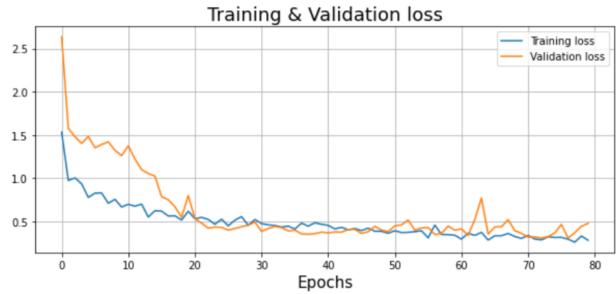


FIGURE 4.13 : Variation des valeurs du loss - ResNet18 - Avant augmentation

On présente maintenant les résultats d'apprentissage de ce modèle sur le dataset après le rehaussement des images pour voir les modifications apportées.

- **Après le rehaussement du dataset**

Les courbes de l'accuracy (plus élevée dans ce cas) et du loss sont plus stables et présentent moins de fluctuations et sont similaires pour l'apprentissage et la validation => Pas de surapprentissage.

D'après la matrice de confusion de la figure 4.16, le modèle prédit les classes avec de bonnes valeurs et ne fait pas beaucoup de confusion entre les classes lors du test. Néanmoins, la classe 1 se confond parfois avec la classe 2 adjacente : soit 19% des images de test de la première classe sont prédites fausses.



FIGURE 4.14 : Variation des valeurs de l'accuracy - ResNet18 - Après augmentation

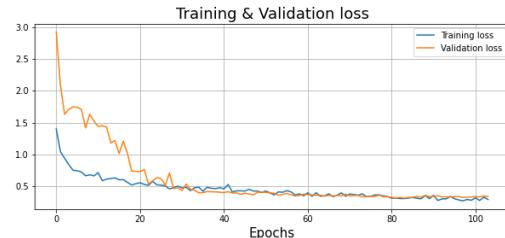


FIGURE 4.15 : Variation des valeurs du loss - ResNet18 - Après augmentation

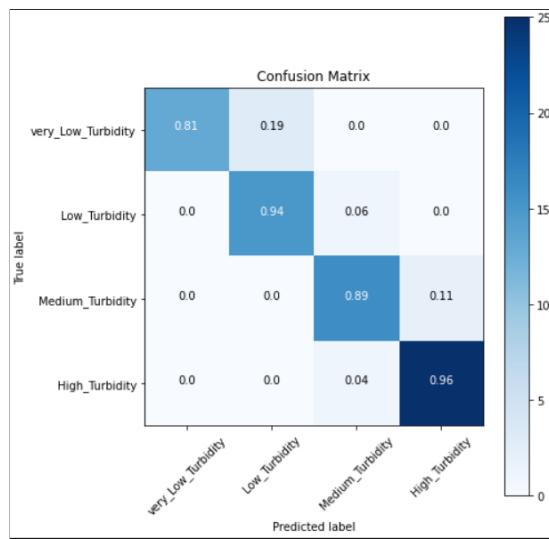


FIGURE 4.16 : Matrice de confusion - ResNet18

4.2.2.3 Apprenstissage par transfert : ResNet 2ème approche

Avec un apprentissage par transfert qui bloque la mise à jour des poids de toutes les couches de convolution, la courbe de l'accuracy représente peu de variations et tourne autour de 0.86. En outre, la courbe du loss est quasiment stable et la courbe de validation et celle de l'apprentissage sont similaires. Par contre, il y a une confusion considérable lors de la prédiction de la classe 3, soit 56% seulement des images de test sont correctement prédites.

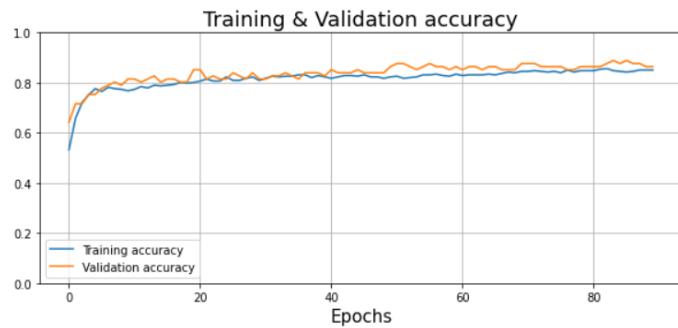


FIGURE 4.17 : Variation des valeurs de l'accuracy - ResNet 2ème approche

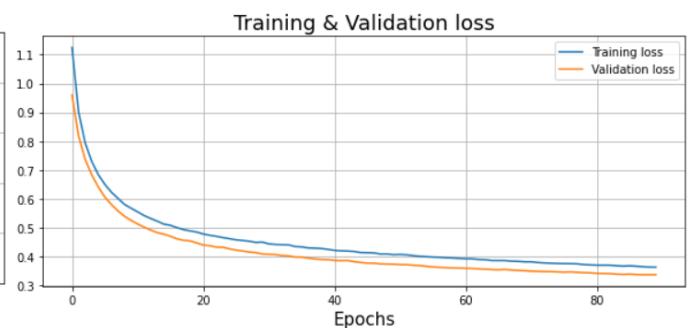


FIGURE 4.18 : Variation des valeurs du loss - ResNet 2ème approche

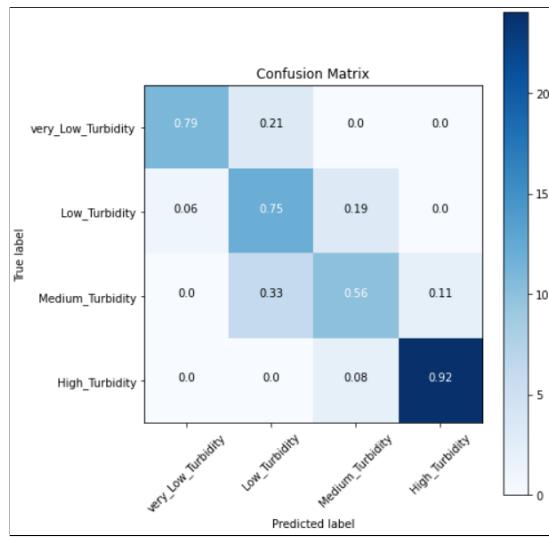


FIGURE 4.19 : Matrice de confusion - ResNet 2ème approche

4.2.2.4 Apprentissage par transfert : ResNet 3ème approche

La performance pour l'accuracy sur l'apprentissage et la validation est bonne mais les valeurs du loss sont instables. Cela indique un faible degré de confiance aux prédictions.

En plus, lors du test et d'après la matrice de confusion de la figure 4.22, on remarque que le modèle parvient à prédire les classes avec une bonne précision à quelques exceptions près.

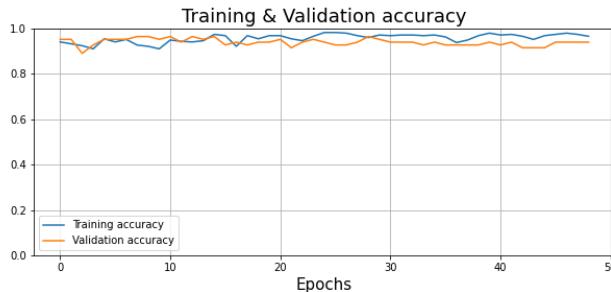


FIGURE 4.20 : Variation des valeurs de l'accuracy - ResNet 3ème approche



FIGURE 4.21 : Variation des valeurs du loss - ResNet 3ème approche

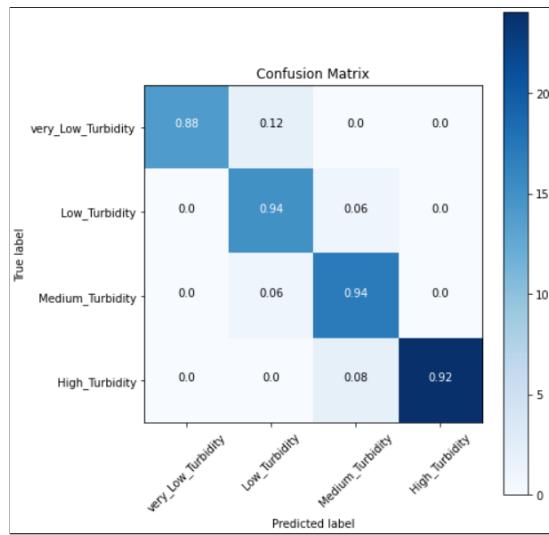


FIGURE 4.22 : Matrice de confusion - ResNet 3ème approche

4.2.2.5 Architecture Aquasight modifiée

Les figures 4.23 et 4.24 montrent que les courbes de l'apprentissage et de la validation ont un même sens de variation pour l'accuracy et le loss. Néanmoins, on note que l'apprentissage est interrompu avec un early stopping après 70 epochs. Cela indique que les valeurs du loss pour la validation ne progressent pas. Par ailleurs, le modèle fait une confusion un peu particulière notée pour la classe 4 dont 15% des images de test sont mal prédites, vu que cette classe est supposément la plus simple à prédire.

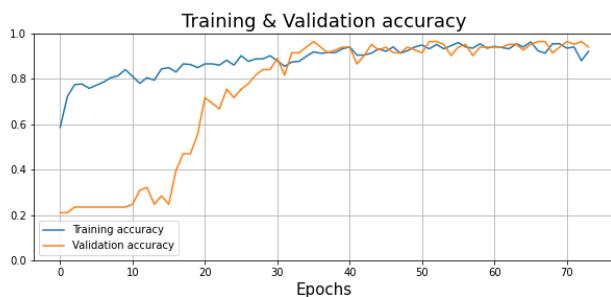


FIGURE 4.23 : Variation des valeurs de l'accuracy - Architecture Aquasight modifiée

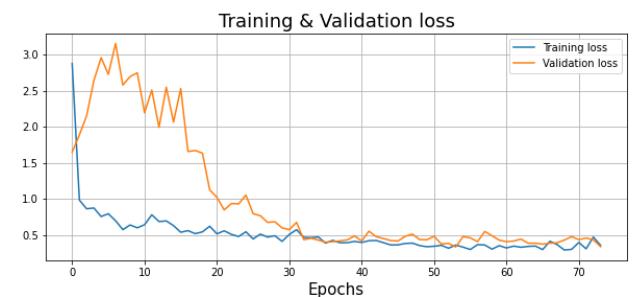


FIGURE 4.24 : Variation des valeurs du loss - Architecture Aquasight modifiée

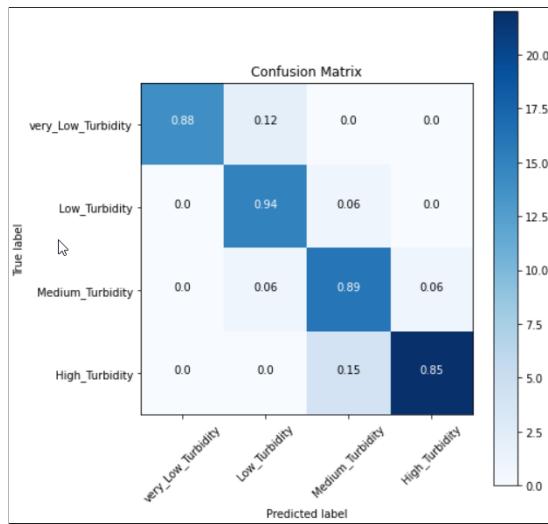


FIGURE 4.25 : Matrice de confusion - Architecture Aquasight modifiée

4.2.2.6 Architecture ex nihilo

En terme de complexité spatiale, ce modèle présente un nombre de poids comparable à celui des architectures ResNet mais beaucoup plus inférieur par rapport à l'architecture Aquasight modifiée (1/6). Toutefois, ceci n'empêche qu'il en finit avec la meilleure valeur d'accuracy pour les données de test, plus précisément, on atteint 96.05%. Cela indique que ce modèle se généralise efficacement face à des données non vues précédemment, ce qui est aussi marqué par le graphe 4.26 où les 2 courbes sont similaires. Quant aux courbes de pertes, elles commencent à se stabiliser dès la 50ème époque.

Ces bonnes performances paraissent être confirmées par la matrice 4.28 où on compte 3 images seulement qui sont mal prédites au total.



FIGURE 4.26 : Variation des valeurs de l'accuracy - Architecture ex nihilo



FIGURE 4.27 : Variation des valeurs du loss - Architecture ex nihilo

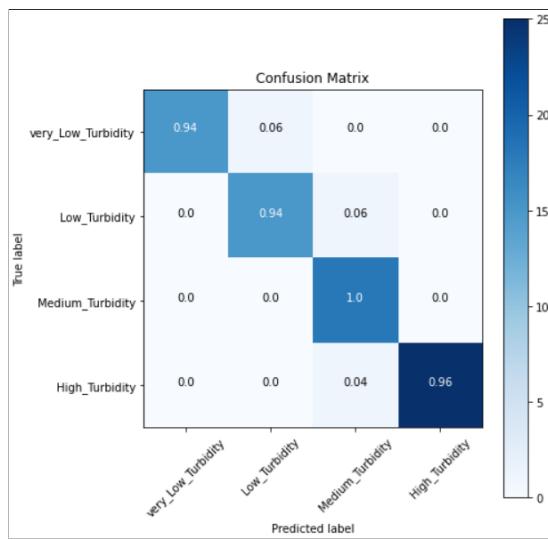


FIGURE 4.28 : Matrice de confusion - Architecture ex nihilo

4.2.2.7 Comparaison des architectures

Le tableau ci-dessous résume les résultats finaux de test des modèles Deep Learning proposés selon 3 métriques : Accuracy, Recall et F1-Score :

Architecture	Accuracy	Recall	F1-Score
ResNet18 : 1ère approche	0.9079	0.9	0.9
ResNet : 2ème Approche	0.7703	0.75	0.76
ResNet : 3ème Approche	0.9211	0.92	0.92
Architecture Aquasight modifiée	0.8816	0.89	0.88
Architecture ex nihilo	0.9605	0.96	0.96

Le problème de sur-apprentissage est globalement peu présent même si le dataset est réduit. Ceci peut être dû au bon calibrage des hyper-paramètres qui agissent directement sur la validation des modèles. On commence par les modèles ResNet, les résultats montrent qu'il existe une différence considérable de l'accuracy lors de la phase du test pour les 3 scénarios. A ce niveau, la méthode d'apprentissage par transfert de la 3ème approche donne la meilleure classification. La 1ère approche, qui s'appuie seulement sur l'architecture ResNet (sans utiliser les poids générés par le dataset "imagenet"), retourne des résultats appréciables principalement lorsqu'on les compare avec les valeurs pour le cas du dataset sans augmentation. Dans le cas où les couches de convolution sont figées, les résultats des métriques sont faibles, se situant aux alentours de 78%. On peut parler d'un sous-apprentissage étant donné que ce modèle n'a pas bien retenu les relations qui relient les images d'entrée.

Pour le scénario où on commence l'apprentissage à base des poids initialisés à partir de l'apprentissage

par transfert, les performances demeurent optimales, avec une valeur d'accuracy, f1-score et recall égales à 0.92, ce qui indique que l'utilisation des modèles pré-entraînés pour initialiser les poids, et le réglage fin des paramètres des couches de classification par rétropropagation, peuvent rendre les modèles et les paramètres plus adaptés à cette tâche.

⇒ Selon les résultats expérimentaux, le mode d'apprentissage de transfert qui se base sur une initialisation faite sur des données d'un énorme dataset peut obtenir de meilleures performances pour cette tâche.

Concernant le modèle Aquasight après modification, l'ajout des couches de Bach Normalization, des régularisations et la modification de quelques autres paramètres permettent d'atteindre des résultats au-delà de 86%. Il est important de noter que la classe la plus confondue aux autres lors du test est la 4ème classe. Une situation à éviter absolument, puisqu'elle est la classe la plus importante et la plus facile à détecter : On ne doit pas classifier des niveaux fortement turbides dans des classes de faible turbidité.

Passons maintenant à l'architecture ex nihilo, les valeurs de recall et f1-score sont marquantes. La classe 3, celle qui est problématique dans plusieurs cas, est parfaitement prédite(100%) même si le nombre de paramètres est environ 3.721.620 : très inférieur par rapport à plusieurs autres modèles.

⇒ **Gain en complexité et en performances des résultats.**

4.3 Test sur le nouveau dataset

Lors de l'acquisition du nouveau dataset, les valeurs de la turbidité acquises paraissent visuellement différentes des résultats obtenus par [1]. Pour notre cas et au-delà de 100ntu, le motif n'est plus visible. Par contre, le seuil dans le dataset précédent atteint 700ntu. On propose quelques hypothèses qui peuvent expliquer cette différence considérable, notamment :

- Un problème technique dans le capteur ou le matériel utilisé pour l'acquisition.
- Les types des sédiments utilisés ne sont pas adéquats et doivent donc être changés.
- Le fonctionnement du capteur n'est pas approprié pour la mesure de la turbidité, qui se base sur les particules en suspension mélangées dans l'eau. Donc la position du capteur dans l'eau peut influer facilement sur les résultats.
- Les résultats obtenus sont corrects et le problème se trouve dans l'ancien dataset.

On procède à présent à tester les données du nouveau dataset sur les modèles Ex nihilo et XGBoost. Le dataset est formé de 51 images allant de 5ntu à 143ntu, donc toutes les images doivent appartenir à la première classe.

Le problème, mentionné ci-dessus, concernant l'annotation est confirmé. Les résultats étant prévus : le réseau CNN ne classifie que 38% de ces images dans la classe de très faible turbidité, quant au XGBoost, il atteint

seulement 66%.

⇒ On suggère pour repérer l'erreur, de commencer par changer le capteur et essayer d'autres types plus sophistiqués pour avoir une annotation de dataset plus précise et adéquate.

Conclusion

Après avoir effectué nos expérimentations et comparé les résultats, nous arrivons à la conclusion que bien que les résultats des modèles ML et DL soient relativement proches, en particulier pour les modèles Ex nihilo et XGBoost, l'approche DL présente un meilleur résultat globalement. Nous nous rendons également compte que la performance de ces modèles sur l'ensemble des données augmentées s'est améliorée de manière significative, ce qui illustre l'importance de la procédure du rehaussement des données. Cependant, lorsque nous testons ces modèles sur notre dataset nouvellement acquis, nous constatons que nos spéculations sont effectivement confirmées et que les modèles ont faussement prédit les résultats en raison de la différence entre les deux datasets. Ce problème doit être étudié et résolu afin de perfectionner le fonctionnement du système.

Conclusion générale et perspectives

Durant ce projet, nous faisons une étude de l'existant, des travaux d'intérêt et des papiers de recherche réalisés au sujet de la caractérisation du niveau de turbidité de l'eau par des techniques de la vision par ordinateur. Ceci étant dans le but de cerner la problématique et d'avoir une vision globale sur les équipements, les protocoles d'acquisition et les différentes méthodologies de travail.

D'abord, on consulte le travail réalisé par [1] et on suit l'évolution de son étude : le protocole d'acquisition des images, la récupération de la valeur de turbidité correspondante ainsi que l'architecture du modèle d'apprentissage profond adoptée. Cette observation nous sert de point de départ ainsi qu'une incitation vers des pistes d'amélioration. A titre d'exemple, il estime qu'une approche de prédiction directe et précise est pratiquement impossible avec un dataset réduit. On s'inspire également de l'idée du motif et on lui apporte des modifications.

On arrive à réaliser une classification avec 4 classes vu qu'on recourt à une nouvelle technique qui permet à nos modèles de mieux s'entraîner : l'augmentation de données par débruitage. En outre, dans le but de tester les algorithmes de Machine Learning, on extrait des descripteurs de couleur et de texture. Cette approche présente des résultats comparables aux architectures Deep Learning, notamment avec le modèle XGBoost. En ce qui concerne le protocole d'acquisition suivi pour avoir un nouveau dataset, on apporte des modifications sur les étapes, principalement, l'attente de plus de temps (2 minutes au lieu de 30 secondes) pour permettre au sédiment de plus se stabiliser. De plus, pour augmenter les features sur lesquels le modèle apprend, on fait varier les types de sédiments introduits à l'eau.

On développe, parallèlement, une application mobile qui permet de se focaliser directement sur la région d'intérêt afin d'éviter le découpage des images dans la phase de pré-traitement. Cette application permet un chargement des images vers le cloud accompagnées de la valeur de turbidité dont on peut faire le traçage grâce à la plateforme IoT qu'on a conçue. On parvient finalement à achever des résultats importants avec les données anciennes augmentées que ce soit avec l'approche Machine Learning notamment l'algorithme XGBoost ou bien avec l'approche Deep Learning avec notre modèle ex nihilo.

Cependant, on se trouve face à une obstruction lors de l'expérimentation avec le dataset nouvellement construit. On observe qu'au bout de 100 NTU le motif n'est plus visible, or que cette valeur dans l'ancienne dataset appartient à la classe de très faible turbidité. Ceci est probablement dû soit au capteur parce qu'il peut ne pas être pas adéquat à la mesure de la turbidité, soit aux types des sédiments utilisés. Cela ne permet pas de confirmer l'effet du changement du motif à cause de la différence dans les annotations des 2 datasets.

En contrepartie, on croit fermement que ce projet est susceptible d'évoluer dans la mesure où on

Conclusion générale et perspectives

pourrait réaliser une connexion directe entre l'application mobile et la plateforme IoT lors de l'acquisition des données. On pourrait aussi envisager une intégration du modèle de l'intelligence artificielle dans l'application mobile pour prédire le niveau de la turbidité automatiquement. A cet effet, on assure un apprentissage continu du modèle où chaque donnée de test serait par la suite intégrée dans le dataset d'apprentissage pour améliorer ses performances.

Bibliographie

- [1] M.CHAABEN. « Conception d'une chaîne de traitement pour l'évaluation de la qualité de l'eau : du capteur à l'évaluation basée sur l'apprentissage profond. » (Aug. 2021), adresse : <http://drive.google.com/drive/folders/1-QzK4HXHF0hBJXLTs-fwWxXZtMJJxzY4>.
- [2] « Turbidity sensor SKU SEN0189 Series, » DFRobot. (), adresse : http://wiki.dfrobot.com/Turbidity_sensor_SKU_SEN0189.
- [3] « Mesure de turbidité dans l'eau de la piscine. » (), adresse : <https://www.lovibond.com/fr/1-analyse-de-l-eau/support-service/pour-en-savoir-plus/mesure-de-la-turbidite/mesure-de-turbidite-dans-l-eau-de-la-piscine>.
- [4] E. R. ANKIT GUPTA. « AquaSight : Automatic Water Impurity Detection Utilizing Convolutional Neural Networks. » (2020), adresse : <https://arxiv.org/pdf/1907.07573.pdf>.
- [5] Z. A. FARAH HAMIDI Mohamad Faiz. « Low Cost and Simple Procedure to Determine Water Turbidity with Image Processing. » (2017), adresse : <https://dl.acm.org/doi/abs/10.1145/3132300.3132302>.
- [6] C. R. RUTH GREENSPAN BELL. « Environmental Policy for Developing Countries. » (), adresse : <https://issues.org/greenspan-environmental-policy-developing-countries/>.
- [7] K. A. I. HUSSAIN et P. NATH. « Water turbidity sensing using a smartphone. » Oxford university press. (2016), adresse : https://www.academia.edu/33455542/Water_turbidity_sensing_using_a_smartphone.
- [8] V. KARNAWAT et S. L. PATIL. « Turbidity detection using Image Processing, In International Conference on Computing, Communication and Automation. » IEEE Xplore. (Apr.2016), adresse : <http://ieeexplore.ieee.org/document/7813877>.
- [9] WIKIPEDIA. « Coefficient de détermination. » IEEE Xplore. (Apr.2016), adresse : https://fr.wikipedia.org/wiki/Coefficient_de_d%C3%A9termination.
- [10] X. R. YING Y. « Design of Aquaculture Water Quality Monitoring System Based on LoRa J. » Chinese Journal of Fisheries. (2020), adresse : https://www.researchgate.net/publication/349012265_Research_of_Water_Body_Turbidity_Classification_Model_for_Aquiculture_Based_on_Transfer_Learning.
- [11] L. I. N. HANSEN. « Turbidity measurement based on computer vision. » (2019), adresse : <https://projekter.aau.dk/projekter/files/306657262/master.pdf>.
- [12] « Firebase Documentation. » (), adresse : <https://firebase.google.com/docs/firestore>.

- [13] « NodeMCU ESP8266 Vs. Arduino UNO Board Comparison. » (), adresse : <https://www.makerguides.com/nodemcu-esp8266-vs-arduino-uno-board/>.
 - [14] « ESP32 Turbidity Sensor. » (), adresse : <https://www.teachmemicro.com/esp32-turbidity-sensor>.
 - [15] M. V. BOLAND. « Haralick texture features. » (), adresse : https://murphylab.web.cmu.edu/publications/boland/boland_node26.html.
 - [16] « Local Binary Pattern for texture classification. » (), adresse : https://sharky93.github.io/docs/gallery/auto_examples/plot_local_binary_pattern.html.
 - [17] D. M. KAVIYA ELAKKIYA. « Local Binary Pattern. » (), adresse : <https://www.sciencedirect.com/topics/engineering/local-binary-pattern>.
 - [18] « Computing image “colorfulness” with OpenCV and Python. » (), adresse : <https://pyimagesearch.com/2017/06/05/computing-image-colorfulness-with-opencv-and-python/>.
 - [19] N. BELAIDI. « SVM : Support Vector Machine. » (2022).
 - [20] « Understanding Support Vector Machine(SVM) algorithm from examples. » (), adresse : <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>.
 - [21] S. E. R. « Understanding Random Forest. » (), adresse : <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest>.
 - [22] J. BROWNLEE. « Ideal ratio mask estimation using deep neural networks for robust speech recognition. » (2016), adresse : <https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/>.
 - [23] SUMANTA. « XGBoost by Heart. » (2021), adresse : <https://medium.com/almabetter/xgboost-by-heart-b494a471845e>.
 - [24] S. I. H. G. E. KRIZHEVSKY A. « Imagenet classification with deep convolutional neural networks. » In Advances in neural information processing systems. (2012), adresse : <https://dl.acm.org/doi/10.5555/2999134.2999257>.
 - [25] M. GURUCHARAN. « Basic CNN Architecture : Explaining 5 Layers of Convolutional Neural Network. » (2020), adresse : <https://www.upgrad.com/blog/basic-cnn-architecture/>.
 - [26] R. SHARMA. « Understanding Gaussian Dropout. » (Nov.2020), adresse : <https://www.mygreatlearning.com/blog/understanding-gaussian-dropout/>.
 - [27] Z. X. HE K et R. S. et AL. « Deep residual learning for image recognition Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. » (2016), adresse : <https://ieeexplore.ieee.org/document/7780459>.
-

- [28] A. SACHAN. « Detailed Guide to Understand and Implement ResNets. » (), adresse : <https://cv-tricks.com/keras/understand-implement-resnets>.
- [29] « Histogram Calculation. » (), adresse : https://docs.opencv.org/3.4/d8/dc8/tutorial_histogram_comparison.htmls.
- [30] H. QIN. « Comparison of Convolutional Neural Networks in Real-Time Monitoring of Aquaculture Water State. » (2021), adresse : <https://iopscience.iop.org/article/10.1088/1742-6596/2026/1/012020>.

