الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

Filière Électronique
Spécialité Électronique Des Système Embarquée

Présenté par :

KAABECHE OUSSAMA

&

AINOUZ TAHAR ABDURRAHMEN

EXTRACTION DES ILOTS PAR DEEP LEARNING

Proposé par : Mme BOUGHERIRA KHETIB NADIA

Année Universitaire 2019-2020

Remerciements

Au nom de dieu le tout puissant, le clément, le miséricordieux qui nous a inciter à acquérir le savoir et les sciences et nous a doter de toutes les facultés et moyens pour y parvenir. Nous rendons grâce au tout Puissant pour la volonté, le courage, la patience et la force qu'il nous a donné durant toutes les longues années d'étude.

En seconde lieu, nous tenons à exprimer nos vifs remerciements :

- *A **Mme**. **BOUGHERIRA KHETIB Nadia**, pour avoir Dirigé ce travail, et pour la confiance et l'intérêt qu'il a témoigné tout au long de la réalisation De ce travail, son expérience et sa connaissance ont contribué à notre formation scientifique.*
- *A Tous les enseignants du département d'Electronique de l'université Saad Dahleb Blida pour tout le savoir qu'ils ont su nous transmettre durant notre cursus universitaire.*
- *Aux Membres du jury pour l'intérêt qu'ils ont porté à notre projet en acceptant d'examiner notre travail et de l'enrichir par leurs propositions.*

Enfin nous remercions tous ceux qui de près ou de loin ont participé à l'élaboration de ce Travail.

Dédicaces

Louange à notre ALLAH qui nous a dotés de la merveilleuse faculté de raisonnement. Louange à notre Créateur qui nous a incités à acquérir le savoir.

A ma mère pour m'avoir soutenu, accompagné et surtout encouragé tout au long de ce travail.

A mon père, pour son soutien inconditionnel, encouragements, et pour m'avoir permis de réaliser mes études dans les meilleures conditions.

A mes deux frères : Michael Nabil et Adem.

A mes amis : Abdelmoumene, Amire, Ousseynou, abdelwahab et sans oublier mon binôme Oussama.

Je dédie ce modeste travail.

-0-0-0-(Ainouz tahar abdurrahmen) -0-0-0

Dédicaces

Je dédie ce modeste travail à mes très chers parents qui sont la source de sagesse et de tendresse, et qui ont toujours été à mon côté par leur encouragement, leur soutien et leur compréhension, aucun mot ne peut traduire le profond amour et le grand respect que j'ai pour vous, que ALLAH vous protège pour nous.

A mes très chères frères, Rabah et Abdeljalil et ma sœur Maria.

A toute ma famille Kaabeche, Aouire, Cheboui, Abikchi et Kadik.

A mes amies : Cheboui Ishak, Hammana Ishak, Zaid, Amire, Housseyno, Abdelwahab, Mourad, Ahcen, Mouhcin, Nassim, Abderrasek, Abderraouf, Mahdi ,Saad ,melloul et Ishak .

A tous ceux qui me connaissent Kaabeche Oussama.

A mon binôme Abdurrahmen pour les excellents et les durs moments passés ensemble, ainsi que sa famille.

A tous les étudiants d'électronique particulièrement les ESE.

Et à tous ceux que j'aime et ceux qui m'ont aidé, je dédie ce travail.

-0-0-0-(KAABECHE Oussama) -0-0-0

ملخص:

يتكون هذا المشروع في خطوته الأولى حول مفهوم الذكاء الاصطناعي (Intelligence Artificiel) والتعلم الآلي (Machine Learning) وكذلك التعلم العميق (Deep Learning)، ثم تعريف عن نموذجYOLO ، وفي الخطوة الثانية، إنشاء وتدريب هذا الأخير (نموذج YOLO) باستخدام الأداة(Google Colab ، Labelimg) الموجهة لمجال التعرف الخاضعة، تحت إشراف شبكات CNN .

سيكون النموذج الذي تم إنشاؤه قادرًا على استخراج الجزر في الخرائط الحضرية وخرائط الأقمار الصناعية.

الكلمات المفتاحية: الجزر، الخرائط الحضرية، خرائط الأقمار الصناعية، شبكات CNN،Google Colab ، YOLO،Labelimg، Machine Learning، Deep Learning، Intelligence Artificiel.

Résumé :

Le présent projet consiste à faire dans une première étape une présentation de l'Intelligence Artificiel et le Machine Learning, ainsi que le Deep Learning, puis définir le modèle YOLO, et, dans une seconde étape, à créer et entrainer ce dernier (modèle YOLO), en utilisant des outils (Google Colab, Labelimg) destinés aux domaines de la reconnaissance supervisée a base des réseaux CNN. Le modèle créé va être capable de faire une extraction d'ilots dans des cartes urbaines et satellitaires.

Mots clés : ilots, cartes urbaines, cartes satellitaires, réseaux CNN, Google Colab, Labelimg YOLO, Machine Learning, Deep Learning, Intelligence Artificiel.

Abstract:

The present project consists of a presentation of the Artificial Intelligence and the Machine Learning and the Deep Learning , then define the YOLO model, and, in a second step, to create and train the (YOLO model), using tools like (Google Colab, Labelimg…),designed for the fields of supervised recognition based on CNN networks. The model created will be able to extract islets in urban maps and Satellite.

Keywords: islets, urban maps, satellites maps, networks CNN, Google Colab, Labelimg YOLO, Machine Learning, Deep Learning, Artificial Intelligence.

Liste des acronymes et abréviations

BP : Back propagation.

CNN : Convolutional Neural Network (Réseau neuronal convolutif).

CPU : Central Processing Unit (unités centrales de traitement).

CUDA: Compute Unified Device Architecture.

DL : Deep Learning.

GPU : Graphics Processing Unit (unités de traitement graphique).

IA: Intelligence Artificielle.

IH : Intelligence humaine.

ML : Machine Learning.

MLP : Multilayer perceptron (Réseaux à plusieurs neurones).

R-CNN: Region with Convolutional Neural Network (Réseau neuronal convolutif régionale).

RNN : Recurrent Neural Network (Réseau à connexions récurrentes).

SSD : Single Shot MultiBox Detector.

YOLO : You Only Look Once (Vous ne regardez qu'une fois).

Table des matières :

Liste des figures

Liste des tableaux

Introduction générale

a) **Contexte**

L’intelligence artificielle avec sa popularisation et son émergence progressive en tant que technologie de base qui entraîne des développements considérables dans un grand nombre de domaines, l'utilisation de l'apprentissage automatique et de l'apprentissage approfondi a connu un essor considérable. D'après de nombreuses enquêtes et études, l'intelligence artificielle (IA) et l'apprentissage machine devraient figurer parmi les filières professionnelles les mieux rémunérées et les plus lucratives dans les années à venir.

Dans le IA les projets les plus demandes sont sur les domaines de vision comme la reconnaissance des objets alors les chercheurs ont développer des algorithmes de détection d'objets qui sont construits à base réseaux neurones artificiels, utiliser dans le domaine médical comme la détection de cancer et comme la détection d’immatriculation et après les implémenter dans des capteurs, tels que le Lidar ou les cameras ou les cartes FPGA.

Comme tous les autres domaines qui se développe constamment et se préparent pour l'avenir, le domaine d’urbanisme et analyse des cartes satellitaires et les cartes urbaines profite aussi de (DL).

Pour les urbanistes qui s'appuient sur l'interprétation d'images satellites ainsi que les cartes urbaines, l'apprentissage approfondi peut être une solution pour la détection et la classification des objets, la localisation des caractéristiques topographiques et géographiques

b) Objectifs

Le but de notre projet est d’entrainer un modèle à base de (DP) afin qu’il puisse reconnaitre les ilots (quel que soit leurs formes) dans le centre-ville de blida (ville cas d’étude) sur deux types de cartes satellitaires et urbaines, car ces derniers (les ilots) sont des éléments importants pour l’étude de la typo-morphologie des villes.

La reconnaissance des ilots est importante car elle permet d'analyser le tissu urbain selon la théorie de Muratori et Caniggia en contribuant à la détermination les axes structurants dans une ville (centre anciens).

Il est à noter que l'automatisation de la lecture des cartes de villes est d'une grande aide pour les urbanistes pour traiter un nombre important de cartes [en un temps réduit].

Il est à noter que ce travail peut être également exploité dans d'autres applications telle que l'application d'optimisation utilisée dans les GPS (par exemple).

Pour réaliser ce projet, on a choisi comme un modèle de détection d'objet le modèle YOLO construit à base de réseaux CNN, qui a la capacité de traiter les données en temps réel, aussi on a généré une base de données composée d'une variations d'ilots pour les deux types de cartes(urbaines et satellitaires), cette base de données représente une alimentation de l'entrée du modèle, le langage d'exécution utilisé est python, et l'environnement d'entrainement choisi est Google Colab, ce dernier permet d'entrainer le modèle en utilisant son GPU. Ceci va nous permettre d'avoir un modèle qui peut reconnaitre les ilots.

Il est à noter que e modèle t peut être embarqué sur les cartes FPGA (par exemple).

c) Organisation du mémoire

Pour décrire notre projet nous avons organisé ce mémoire comme suit :

Chapitre I

Dans le premier chapitre on présente des généralités sur la morphologie de la ville, et aussi on va présenter sur la méthode d'interprétation Muratori et Caniggia,et aspects historique sur le centre de Blida et les types de cartes .

Chapitre II

Dans le second chapitre on s'approfondit dans l'Intelligence Artificielle notamment dans le Machine Learning et le Deep Learning et définir le modèle YOLO.

Chapitre III

Le dernier chapitre est consacré à l'application de plusieurs outils (Google Colab, Google Drive…), pour la création et l'entrainement du modèle YOLO sous Python, pour qu'il puisse reconnaitre les objets d'intérêt à savoir, les ilots.

Chapitre I :

L’aspect urbain de la ville et extraction d’objets urbain.

I.1 Introduction :

Le travail que nous allons réaliser représente la reconnaissance des ilots dans les espaces urbains anciens inclus dans une carte urbaine en utilisant les techniques du Deep Learning.

Ce chapitre a pour objectif de donner les grandes lignes de la méthode Muratorienne et son développement à travers le modèle interprétatif de Caniggia. Afin de répondre à ce dernier, nous allons concevoir et réaliser une partie d'un système qui fait la reconnaissance des ilots urbains.

I.2 La morphologie urbaine :

La morphologie urbaine vise à étudier les tissus urbains au-delà de la simple analyse architecturale des bâtiments et à identifier les schémas et structures sous-jacents. La morphologie urbaine étudie les formes et les caractéristiques de la ville (la voirie, le parcellaire, le découpage du sol, les densités, les usages), et les phénomènes qui en sont à l'origine: topographie, histoire, influence culturelle, économie, règles d'urbanisme, contexte technologique ou encore énergétique. Elle s'appuie sur les différentes échelles constitutives du monde urbain : le bâtiment, l'îlot, le tissu urbain, la ville, l'agglomération. Elle est interdisciplinaire.

Dans ce qui suit, nous allons présenter deux méthodes de des tissus urbains.

I.3 La méthode de Muratori : [1] [2]

La méthode de lecture du tissu urbain examiné pour calculer les paramètres (calcul du nombre de parcelles et mesure des côtés de l'îlot) pour la modélisation et la méthode typo-morphologique développée par l'école Muratorienne. Cette méthode revient à la naissance d'anthropologue italien l'architecte Saverio Muratori.

Contrairement à la méthode de traiter l'architecture comme de l'art pur, cette dernière ne peut pas être définie par la loi, car elle dépend essentiellement de la personnalité et de l'inspiration de l'architecte. La méthode muratorienne propose de définir des disciplines spécifiques à l'architecture à partir d'observations permanentes basées sur l'observation objective de l'évolution des phénomènes urbains.

Cette hypothèse s’appuie sur deux postulats :

1-il n’y a pas de réalité non structurée.

2-la genèse de l’habitat est une genèse logique, ou logiquement reconstructible.

Depuis les années 1950, Muratori s'intéresse à la critique du mouvement moderne en observant les carences et dérives du mouvement moderne et les qualités qui font défaut dans les villes contemporaines : racines, mémoire collective, génie de lieu.

Si Muratori prend position contre les objectifs antihistoriques et anti-urbains du mouvement moderne, c'est parce qu'il a maîtrisé les intérêts culturels fondamentaux de la continuité des établissements humains, et il doit également utiliser les vieux bâtiments comme infrastructure pour que de nouveaux bâtiments puissent apparaître et se développer.

Donc, les attitudes individualistes des architectes, des souverains et des créateurs solitaires ont causé la perte du sens du complexe architectural. Un exemple est la production urbaine de banlieues sans vie et de zones résidentielles composées de dortoirs.

Afin d'éviter cette situation, il est revenu sur les réalités spécifiques des villes historiques, et il a commencé à comprendre ces réalités lors de sa transformation.

Muratori et ses élevés ont souligné la relation entre la typologie des bâtiments et la forme urbaine pour comprendre les phénomènes urbains. Ils traitent ces problèmes à travers une méthode structurée (basée sur la reconnaissance des structures), selon laquelle le niveau structurel global de la ville peut être observé.

Selon cette méthode, en première lecture, les bâtiments ne sont pas considérés comme des objets isolés, mais avec des non-bâtiments (terrains, parcelle, lieux publics). La parcelle sera considérée comme l'élément de base de notre reconnaissance de forme.

Au deuxième niveau de lecture, le regroupement des parcelles permet de considérer la structure caractéristique des éléments du tissu en fonction de leur localisation dans la ville, de leur période de formation et de leur mode de croissance.

De ces analyses Muratori tire trois leçons essentielles :

a) Le type bâti ne se caractérise pas en dehors de son tissu.
b) Le tissu urbain à son tour ne se caractérise pas en dehors de l'ensemble de la structure urbaine.
c) L'étude de la structure urbaine ne se conçoit que dans sa dimension historique car sa réalité se fonde dans le temps sur une succession de réactions et de croissances à partir d'un état antérieur.

Ces observations marquent largement les idées architecturales contemporaines et ont apporté de nombreuses années de contribution à la reconstruction des valeurs effacées par le mouvement moderne : la relation entre l'architecture et les villes et l'histoire.

I.4 Le modèle interprétatif de Gianfranco Caniggia : [2]

Gianfranco Caniggia était un ancien assistant de Muratori du Département d'Architecture de l'Université de Rome, et était un continuateur de sa réflexion. Il a développé un modèle explicatif pour lire l'analyse de la structure urbaine. C'est sur ce modèle que nous pouvons développer notre projet.

L'hypothèse globale de Caniggia est de considérer l'organisation interne de l'environnement bâti comme une influence de son processus de formation.

Il a construit sa théorie sur plusieurs échelles d'observation. Après avoir examiné la relation entre les emplacements des villes, nous sommes partis du noyau d'origine. Il a commencé par une comparaison de divers cas de mutation dans des types de bâtiments italiens et a approfondi ses recherches sur les structures urbaines de base. Il a observé la formation et la rénovation des maisons de Florence de la basilique romaine.

On peut dire que cette ville est un immense laboratoire, et sa forme urbaine a été testée pendant des centaines d'années. Le processus de construction progressive des structures urbaines est appelé processus de transformation structurelle.

Pendant des siècles, en raison de la dynamique de la vie humaine, il a connu une série de mouvements continus. Cela se traduit par la dynamique de la forme urbaine.

Surtout dans les anciennes villes, cette dynamique apparaît au niveau des trois lois fondamentales de la structure urbaine :

- La modularité.
- La nodalité.
- La hiérarchie.

I.4.1 Notion de modularité :

Le concept de modularité ou d'agrégation modulaire de la structure de l'habitat est très important car il peut être dérivé de plusieurs ordres de grandeur à partir du multiple ou de la division du module de base. Ensuite, des opérations simples peuvent être utilisées pour décrire le processus de construction de la croissance et de la diversification : répétition continue, séries unifiées d'agrégation et fusion.

Dans la terminologie de Caniggia le module de base a la double signification de module élémentaire et de module origine.

I.4.2 Notion de nodalité :

Un nœud fait référence à tout point qui représente un objet continu, généralement déterminé par l'intersection entre deux objets continus ou le bijou d'un objet continu et d'un autre objet : un nœud de corde ou deux nœuds de corde, ce qui est une bonne expression Cette fonctionnalité. Par conséquent, si une personne comprend un itinéraire à travers des objets continus, le nœud sera l'intersection entre les deux itinéraires, ou la collision entre deux objets continus, par exemple, un pont ou Ford, qui est liée à un parcours.

I.4.3 Notion de hiérarchie :

Les parcours dans la ville permettent de relier les différents points de cette dernière. On peut les classer selon l’ordre d’importance que leur conférant leur position et les fonctions qui il desservent : nous aurons ainsi des vois principales, des voies secondaire et des voies tertiaires ; dans les terminologie de Caniggia, ces voies sont appelé : parcours matrice , qui est à l’origine de la création de la ville ; puis parcours d’implantation, qui positionnera les nouvelles extensions perpendiculairement à la ville ; puis parcours de liaison, qui relient les parcours d’implantation

entre eux . Et enfin desserte, qui sont les voies et impasse menant aux maisons dans les quartiers résidentiels.

Pour notre travail, nous en retenons a partie de la théorie de G. Caniggia :

a) Que la structure du tissu soit modulaire et que le module de base est la parcelle.
b) Que les parcours sont hiérarchisés.
c) La relation entre les parcours et les parcelles nous permet de définir les parcours structurants suivant la loi qui stipule que le parcours structurant présente un feu grand nombre de parcelles.

I.4.3 Définition de paramètres issus du modèle interprétatif :

Détection des ilots pris en état avancée à la détection de réseaux routière qui séparent les déférent bloc urbain que les parcelles sont incluses dans elle (fig I.1).

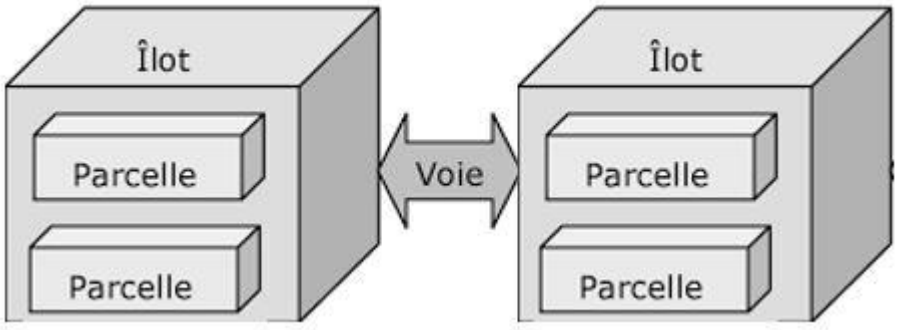


Figure I.1 : *Relation entre ilots et voirie dans un tissu urbain.*

a) Définition d'un Ilot :

Aussi appelé pâté de maisons ou block en anglais, est un ensemble de parcelles, bâties ou non, constituant une unité élémentaire de l'espace urbain, il possède trois frontières (des cotés) ou plus, chacune est délimitée par une voie du réseau de voirie (des rues). Donc c'est une partie du tissu urbain limitée par l'intersection des voies autour de ses cotés (fig I.2).

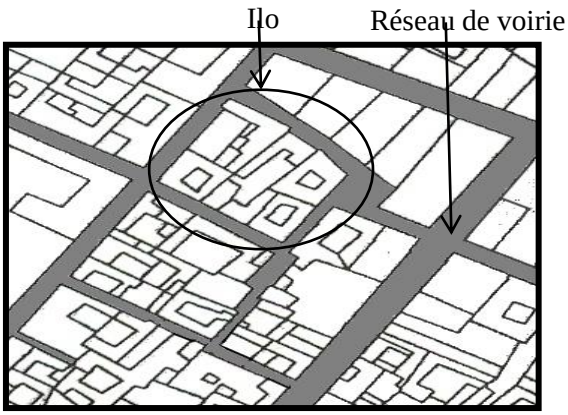


Figure I.2: *Représentation d'un ilot dans un bloc urbanisme.*

b) Définition d'une Parcelle :

C'est un élément de base d'un ilot, qui peut prendre de nombreuses formes. De la quantité et de la taille, on peut déduire le développement de la structure urbaine. Un ensemble des parcelles peut être désigné comme un « parcellaire » (fig I.3).

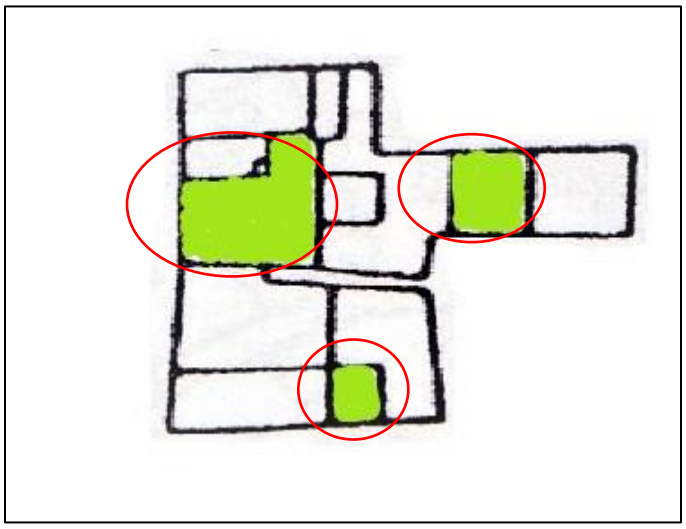


Figure I.3: *Exemple de placement des parcelles dans un ilot.*

c) Définition d'un Réseau de voirie :

Un réseau routier (ou de voirie) est un ensemble de voies et d'intersections, dont les côtés sont composés par les frontières des ilots. Il relie les différentes régions de la ville entre elles (fig I.4).

Figure I.4 : *représentation de réseaux routière indiqué par le gris.*

I.5 Présentation centre ancien de blida : [3] [4]

I.5.1 Situation géographique :

La ville de Blida est située dans la partie sud-ouest d'Alger et s'est développée selon sa position stratégique dans le plan régional de la région. En effet, c'est une ville carrefour, vitale pour l'échange Est-ouest de l'Algérie.

C'est la capitale de la plaine de Mitidja et l'agriculture a toujours été le fondement de son développement.

Blida a été fondée par Sidi Ahmed EL Kébir pendant la période ottomane, c'est une ville de garnison et représente le pouvoir turc dans la laine de Mitidja. Et également un important relais entre Alger et Titrerai (la zone de Médéa). Une petite ville (petite ville) appelée « el Blidah » est entourée d'un mur de la ville, qui est entouré d'un mur percuté de 3 à 4 mètres de haut et percé de 06 portes.

Quelques années plus tard, la ville se développe rapidement et la place d'Armes devient le centre des pouvoirs européens (théâtre, mairie, poste, banque, construction Ets), symbolisant le nouvel ordre administratif et économique, à savoir la façade architecturel de du 11éme siècle Les bâtiments ont remplacé les murs extérieurs aveugles des maisons mauresques (andalouses) le long de la route goudronnée, et la ville a finalement entouré toutes les surfaces libres à l'intérieur de la clôture (à l'exception des installations militaires, des barrières de croissance).

Actuellement Blida ville des roses est un grand centre administratif, industriel, ses offres de services elle dépasse ses limites administratives pour atteindre ou dépasser les wilayets limitrophes.

I.5.2 Croissance urbaine du centre ancien de Blida :

a) Période pré turque :

Selon le colonel Trumelet près de Blida, les tribus berbères sont habitées, les plus importantes étant Beni-Khalil au sud et Hajar Sidi Ali au nord, et certains La tribu vit dans les montagnes comme les Beni. La tribu Salah est divisée en plusieurs parties, divisées en villages. Les résidences d'Oued sultan et Hedjar Sidi Ali sont des gourbis en pierre, entourées de maraîchage, tandis que les montagnards sont placés dans de petits villages sur les pentes de la vallée.

b) Période médiévale :

Blida a été fondée par une vieux Marabout de l'Est. Il a beaucoup voyagé dans les pays islamiques, en particulier en Andalousie (Espagne). Son nom est Sidi Ahmed · Kibir (sidi Ahmed elkébir), qui vivait dans l'oued Taberkachent et le chaàbet Ar Roman vers 1519, aujourd'hui appelé "oued sidi El Kabîr".

L'ermitage de Sidi Ahmed El kebir, bientôt entouré d'une mosquée, est rapidement devenu un lieu de pèlerinage visité par de nombreux citoyens, attiré par les enseignements du saint et la renommée de ses vertus et ses miracles.

En 1533, après la capitulation de la Grenade, Le Pacha Kheir Eddine déporta des milliers de Maures d'Espagne et l'Algérie fut dispersée entre Alger et Cherchell.

La ville se protège de remparts en prisé de 3ou 4 mètres de hauteur et percés de 4 pins, de 6 portes :

- Au sud : Bab Errahba (la porte du marché).
- A l’est : Bab Eddzair (porte d'Alger).
- A l’ouest : bab el kébour (porte des tombeaux).
- Au nord : ab Essebt (porte du marché du samedi).
- Au nord-est : Bab Ezaouia (ouvrant sur la route de la zaouïa de sidi madjébeur).

- Au sud- est : Bab el khouikha (la poterne).

La structure résidentielle dansée se compose de petites maisons avec une cour intérieure et un toit en terrasse. À cette époque, la ville traversait 30 rues larges et était reliée aux artères principales.

c) Période coloniale :

Le 7 février 1839, l'armée française occupe la ville et occupe fermement le château. En 1842, à la fin de la guerre de Mitidja, les premières familles européennes s'installent dans la ville, qui compte une population importante. Des maisons de style européen ont commencé à être construites dans le centre, juxtaposées à des maisons de style mauresque. Les bâtiments coloniaux construits dans les villes européennes d'origine peuvent être résumés en trois signes de ville : Les ouvrages de défense, les ouvrages religieux et les espaces vides.

En 1846, le mur de pierre a remplacé l'ancien entrepôt gravement endommagé par la guerre et a été remplacé par le mur de pierre, percé par la porte et situé bien au-delà du plan d'origine. Par conséquent, la périphérie de la ville a presque été agrandie au profit de l'armée. Il représente la moitié de la zone urbaine et 20 hectares de 48 hectares.

Par la suite, l'occupation du centre-ville de Blida a également été caractérisée par des bâtiments à haute densité, des bâtiments commerciaux, des bâtiments administratifs ou religieux (marchés, hôtels de ville, mosquées), qui coïncidaient avec le premier noyau de la ville.

I.6 Présentation de la carte urbaine : [5]

La carte traite deux informations fondamentales : la position et l'information qui y est rattachée. Les informations rattachées (attributs), peuvent être un nombre, une activité, un taux, une quantité et elles évoluent avec le temps.

Différentes sortes de cartes offrent un échantillon de ce large choix, et les cartes remplissent ainsi la fonction de présenter ces faits d'une façon réalisable.

Les cartes varient en fonction de l'échelle, de son usage ou de son contenu et peuvent être regroupées selon la typologie suivante :

a) Les cartes topographiques : qui présentent les relations spatiales existant entre différents objets géographiques, comme les bâtiments, les routes, les frontières et les cours d'eau. La plupart des communes produisent aussi des plans de ville. Les cartes topographiques existent également pour la randonnée ou les activités nautiques. De nombreux systèmes embarqués de navigation routière ainsi que des services internet mettent également à disposition des cartes topographiques (fig I.5).

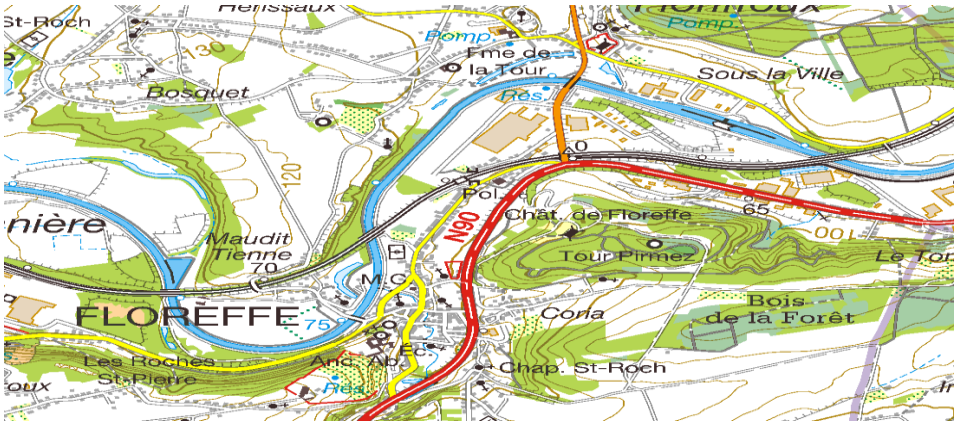


Figure I.5 : *Carte topographies.*

b) Les cartes spéciales : comme par exemple les cartes maritimes ou aéronautiques. Ces cartes relèvent d'un usage professionnel et respectent les règles définies par l'Organisation des Nations Unies. Il existe également des cartographies spécifiques à la navigation de plaisance ainsi que des cartes dédiées à l'orientation définies par la Fédération internationale de course d'orientation (fig I.6).

Figure I.6: *Carte maritime.*

c) Les cartes thématiques : décrivent des phénomènes géographiques qui appartiennent par exemple à la géologie (sol et substrat rocheux essentiellement), à l'occupation du sol ou à la végétation. Les cartes statistiques font aussi partie des cartes thématiques. Elles présentent la répartition géographique d'une variable statistique (fig I.7).

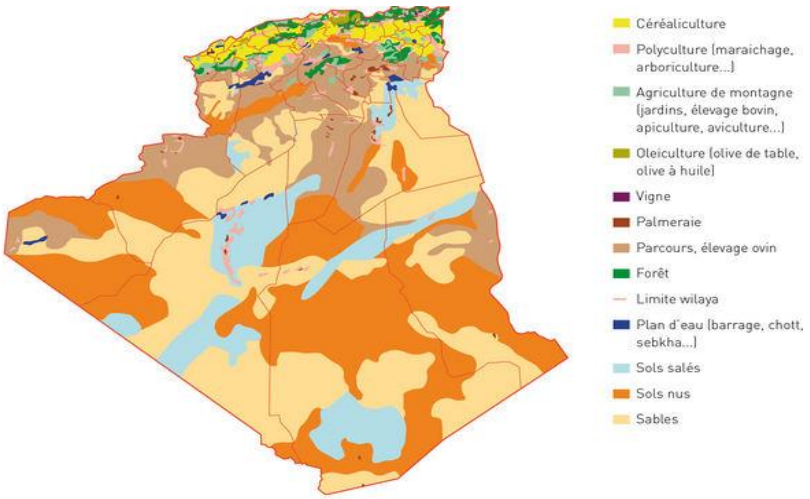


Figure I.7 : *Carte thématique de sol.*

d) Les cartes météorologiques : sont les cartes thématiques les plus répandues. Elles sont diffusées quotidiennement à la télévision pour présenter les conditions météorologiques du jour et les prévisions pour les jours suivants (fig I.8).

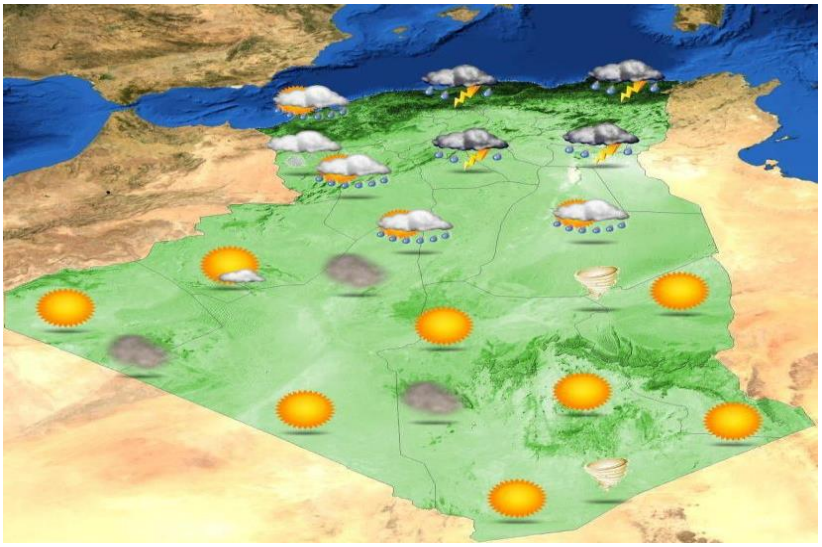


Figure I.8: *Carte de météo.*

e) Les cartes géologiques : font partie des cartes thématiques et sont inestimables pour la prospection de minerais ou de pétrole, ou pour analyser le potentiel agricole. Elles combinent des informations plutôt compliquées et les résultats d'études universitaires en géologie sont accompagnés de plusieurs planches cartographiques (fig I.9).

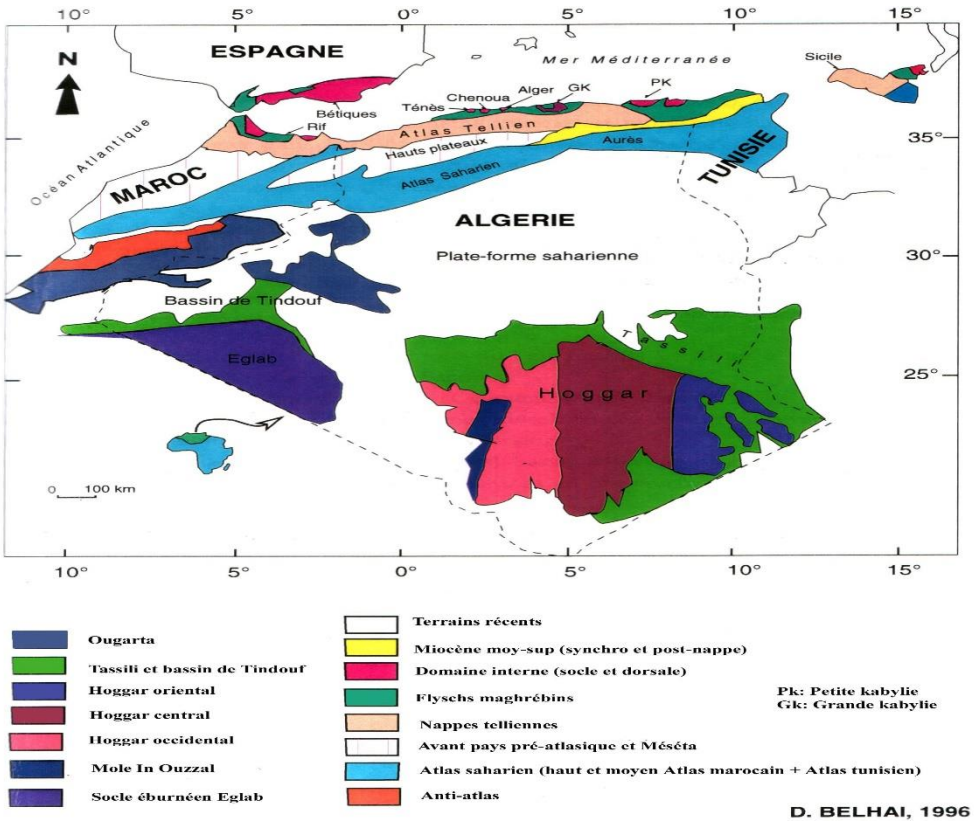


Figure I.9 : *Carte géologique de Algérie.*

I.6.1 Cartes urbaines cas d'étude :

La figure (I.10) donne un exemple de cartes qui sont susceptibles d'être utilisées pour faire de la reconnaissance d'objets urbains, pour notre cas on cherche à extraire des ilots pour être exploités ultérieurement pour la réalisation d'un système d'aide de centres anciens.

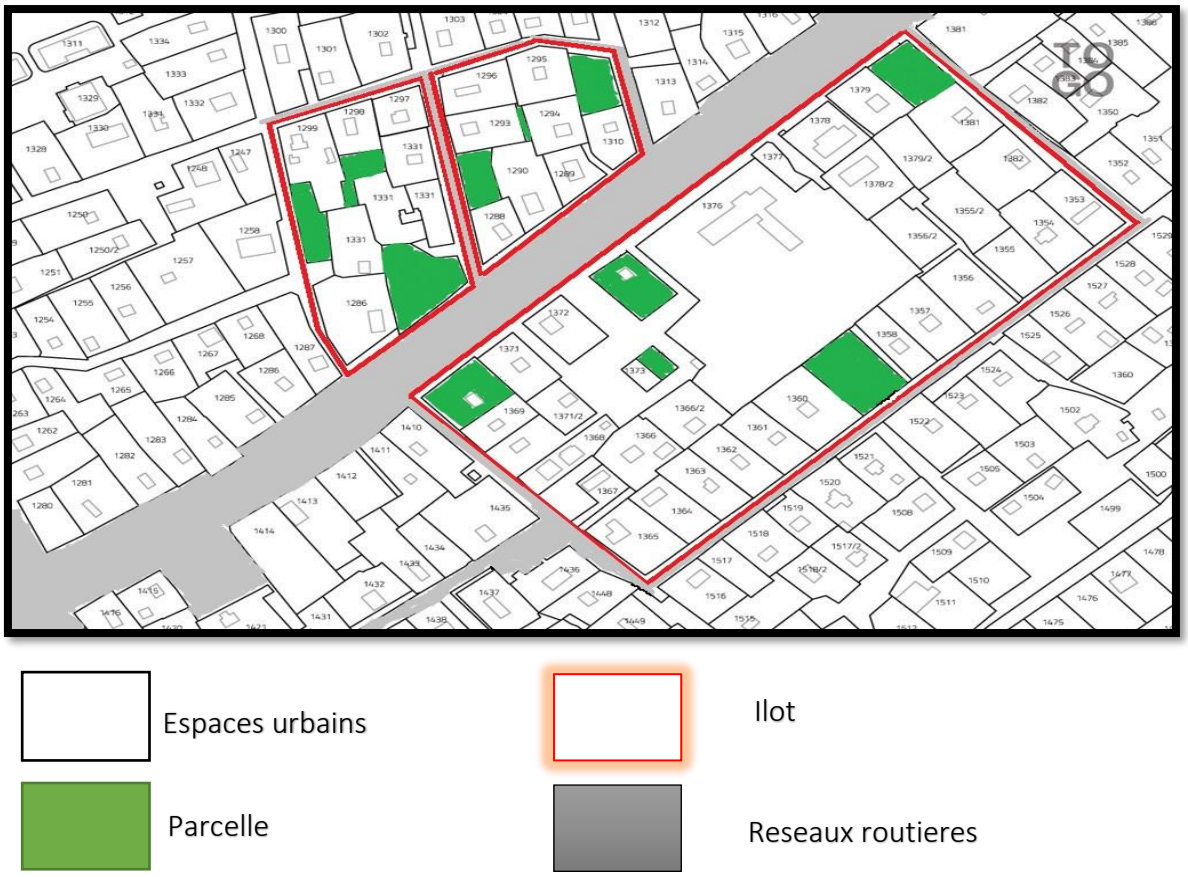


Figure I.10 : *Représentation d'un carte urbanisme.*

La figure (I.11) est aussi une carte satellitaire qui fait l'objet de notre étude, en effet, on essayera de d'extraire des ilots en utilisant le Deep Learning.

Figure I.11: *Image satellitaire d'une zone urbaine.*

I.7 Conclusion :

Après toutes les études faites sur la morphologie urbaine et les critères de connaissance des ilots dans une zone urbanisme ainsi que leur placement sur la carte urbain (topographique ou satellitaire), notre étude représente seulement un aspect de lecture d'un tissu urbaine afin de crée un system supervisée à l'aide de Deep Learning qui reconnais les ilots urbains.

Chapitre II :

Généralités sur le Deep Learning : les outils et le modèle.

II.1 Introduction :

Le but de ce travail est l'utilisation du Deep Learning pour entrainer des données, dans notre cas les données utilisées sont des images de cartes urbaines et d'images satellitaires, et les objets d'intérêt à reconnaitre sont des ilots sous leurs différentes formes et tailles. Ce chapitre est consacré à la présentation des généralités sur le Deep Learning,

II.2 L'intelligence humaine (IH) :[6]

Le Deep Learning est inclus dans le domaine l'intelligence artificielle et ce dernier inspiré par l'intelligence humaine, ceci nous pousse à définir d'une manière générale l'intelligence humaine.

L'IH est constitué d'un Ensemble de fonctions mentales ayant pour objectif la connaissance conceptuelle et rationnelle, qui permettent à l'être humain de résoudre un problème ou s'adapter à son environnement.

On peut la définir d'une façon plus profonde : L'intelligence est une propriété émergente, résultant de la cascade de facteurs environnementaux, génétiques, cérébraux et cognitifs qui influent sur la performance aux différents tests (Tableau II.1).

Facteurs Génétiques	DRD4	COMT	BDNF	DTNB1	CHERM2
Propriété Cérébrale	Neurones	Cortex Préfrontale	Régions Per sylviennes	Hippocampe	Sillon interpariétal
Fonctions Cognitives	Vitesse	Attention	Mémoire De travail	Langage	Visuospatiale

Tableau II.1 : Fonctionnalités du cerveau humain.

Il existe différents types d'intelligences, en effet, l'IH peut subdivisée en différentes composantes :

- Intelligence pratique : permet à l'individu de s'adapter à son environnement unique.

Ex : pour un chasseur après des années d'expériences, il peut chasser facilement plus rapide qu'une personne simple.

- Intelligence collective : désigne la capacité d'une communauté à faire résoudre des problèmes très complexes.

Ex : un groupe de fourmi font construire un pont pour que les autres puisse traverser.

- Intelligence émotionnelle : la capacité à comprendre et à analyser ses propres émotions mais aussi celles des autres.

Ex : capacité à gérer face à diverses situations qui conduises à la colère pour certaines personnes.

- Intelligence des affaires : représente la collecte, l'exploitation et l'analyse de données servant à la prise de décisions. Ces données peuvent être liées aux ventes

Par extension, l'IH a été adaptée aux machines : on parle alors d'intelligence artificielle.

II.3 L'intelligence artificielle (IA) :[7]

L'IA est un ensemble de techniques et théories qui cherchent à développer des modèles capables de **simuler** le comportement humain.

Dans le domaine de l'IA, le but des scientifiques est de pouvoir simuler l'intelligence humaine et le comportement d'un être humain doté de conscience et de sentiments.

Parmi les techniques de l'IA, on trouve le **Machine Learning (**L'apprentissage machine**),**

II.4 Machine Learning (ML) :

Le ML désigne la science de la création et de l'étude d'algorithmes qui améliorent leur propre comportement de manière itérative par la conception (fig II.1).

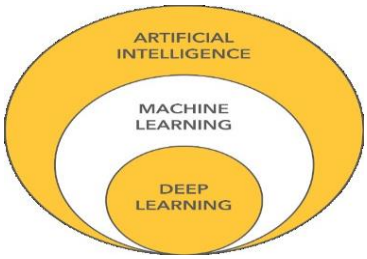


Figure II.1 : *Relation entre AI et ML et DL.*

À l'origine, le domaine était consacré au développement de l'intelligence artificielle, mais en raison des limites de la théorie et de la technologie qui existaient, il est devenu plus logique de concentrer ces algorithmes sur des tâches spécifiques. La plupart des algorithmes (ML) tels qu'ils existent aujourd'hui se concentrent sur l'optimisation des fonctions.

Ainsi, l'utilisation d'algorithmes (ML) devient souvent un processus répétitif d'essais et d'erreurs, dans lequel le choix de l'algorithme parmi les problèmes donne des résultats de performance différents. C'est très bien dans certains contextes, mais dans le cas de la modélisation du langage et de la vision par ordinateur, cela devient problématique.

Après les progrès des capacités théoriques et technologiques dont nous disposons aujourd'hui, l'apprentissage approfondi (Deep Learning) est apparu et se développe rapidement comme l'un des domaines scientifiques les plus passionnants. Il est utilisé dans des technologies telles que les voitures à conduite autonome, la reconnaissance d'images sur les plateformes de médias sociaux et la traduction de textes d'une langue à l'autre.

II.5 Deep Learning (DL) : [8]

Le Deep Learning est basé sur l'idée des réseaux de neurones artificiels, ces derniers sont des modèles bien plus complexes que tous les autres modèles de Machine Learning dans le sens où ils représentent des fonctions mathématiques avec des millions de coefficients (les paramètres).

Avec une telle puissance, il est possible d'entraîner la machine sur des tâches bien plus avancées :

- La reconnaissance d'objets et reconnaissance faciale.
- L'analyse de sentiments.
- L'analyse du langage naturel.

Cependant, développer une fonction aussi complexe à un coût. Pour y parvenir, il faut souvent fournir :

- Un Dataset beaucoup plus **grand** (des millions de données)
- Un temps d'apprentissage plus **long** (parfois plusieurs jours)

- Une **grande** puissance de calcul.

Le DL est une technologie actuellement utilisée dans tous les domaines (médicale, militaire…), elle est basée sur les neurones, dans ce qui va suivre, nous allons introduire deux notions, le neurone biologique, et le neurone l'artificiel.

II.5.1 Neurone biologique :

Un neurone comprend un corps cellulaire ou cellule somatique ou soma, centre de contrôle de celui-ci, qui fait la somme des informations qui lui parviennent. Il traite ensuite l'information et renvoie le résultat sous forme de signaux électriques du corps cellulaire à l'entrée des autres neurones. Il est constitué de :

a. Dendrites : ce sont des prolongements du corps cellulaire des neurones. Elles se divisent créant une arborescence du neurone sous forme de filaments courts et ramifiés, selon leur longueur, ils affectent la quantité d'influx nerveux qui se rend au noyau, ils représentent les entrées du neurone.

b. Axones : ils relient les neurones entre eux jouent donc un rôle important dans le comportement logique de l'ensemble. Les synapses du neurone quant à eux reçoivent les informations des autres neurones via l'axone et permettent donc aux neurones de communiquer entre eux.

c. Noyau : Le noyau est le centre des réactions électrochimiques. Si les stimulations externes sont suffisantes, le noyau provoque l'envoi d'un influx nerveux électrique à travers l'axone.

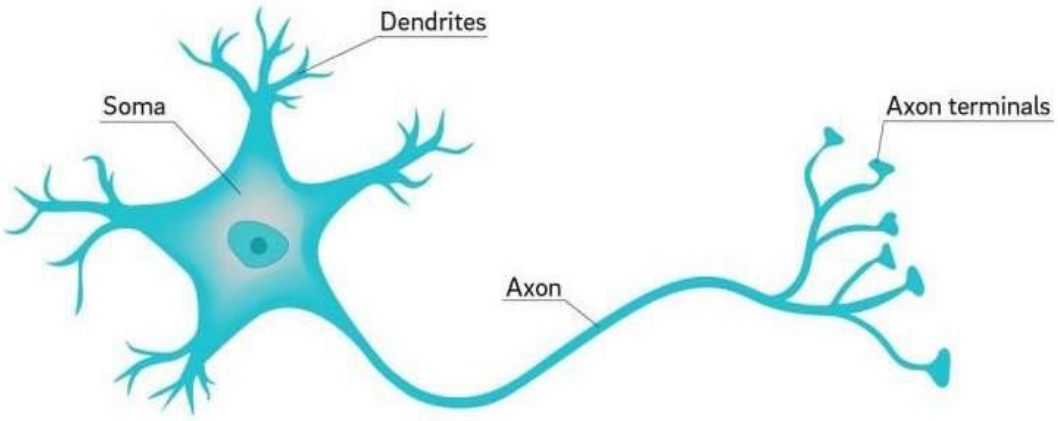


Figure II.2 : *Neurone naturelle.*

II.5.2 Neurone artificiel (Le perceptron) :

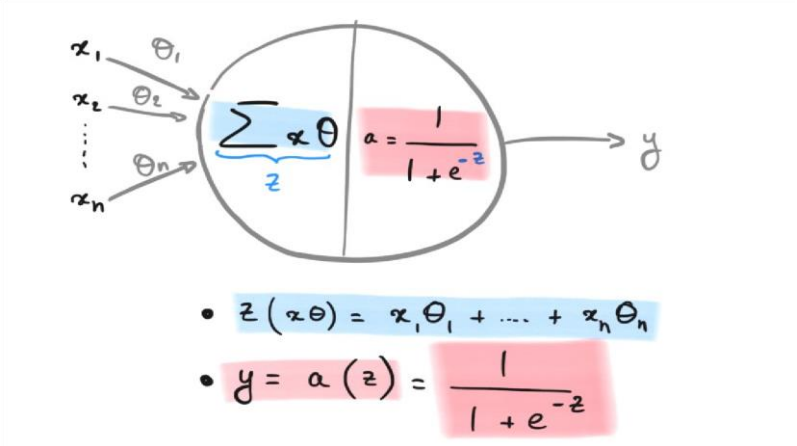
Le réseau de Neurones le plus simple qui existe porte le nom de perceptron.

Les entrées du neurone sont les ***entrées*** $\boldsymbol{x}$ multipliées par des **paramètres appelés poids** $\boldsymbol{\theta}$ modifiable (apprentissage). Le calcul effectué par un neurone est divisé en deux étapes :

1. Le neurone calcule la ***somme*** $\boldsymbol{z}$ de toutes les entrées $\boldsymbol{z} = \sum \boldsymbol{x\theta}$. C'est un calcul linéaire.

2. Le neurone introduit z dans la fonction d'activation qui est la fonction sigmoïde. On remarque qu'il ya un calcul non-linéaire qui génère la sortie y (fig II.3).

Figure II.3 : *Neurone formel.*



On peut utiliser d'autres fonctions d'activation que la fonction sigmoïde pour simplifier le calcul du gradient et ainsi obtenir des cycles d'apprentissage **plus rapides** :

- La fonction tangente hyperbolique tanh (z)
- La fonction RELU (z)

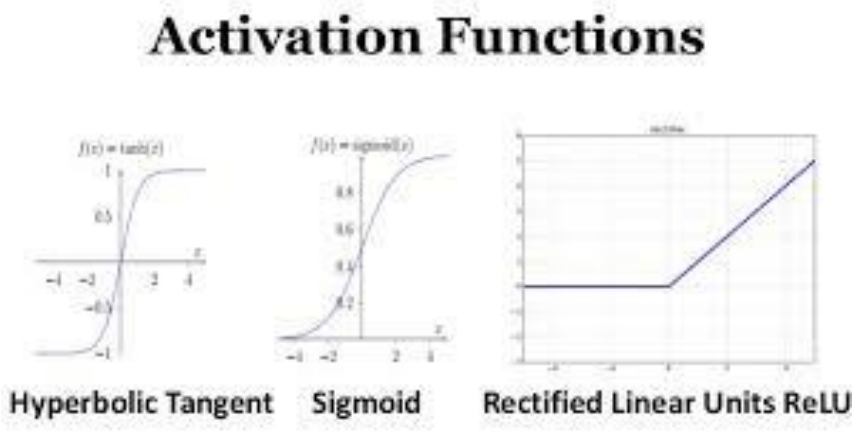


***Figure II.4** : Les fonctions d activations.*

II.5.3 Réseaux à plusieurs neurones (MLP) :

Pour créer un Réseaux de Neurones, il suffit de développer plusieurs de ces perceptrons et de les **connecter** les uns aux autres d'une façon particulière :

- On réunit les neurones en **colonne** (on dit qu'on les réunit en **couche**, en *layer*). Au sein de leur colonne, les neurones ne sont pas connectés entre eux.
- On connecte toutes les **sorties** des neurones d'une colonne à gauche aux **entrées** de tous les neurones de la colonne de droite qui suit.

La figure (2.5) représente un réseau de neurones. Tous les *layers* entre la couche d'entrée et la couche de sortie sont dits **cachés** car nous n'avons pas accès à leur entrées/sorties, qui sont utilisées par les *layers* suivants.

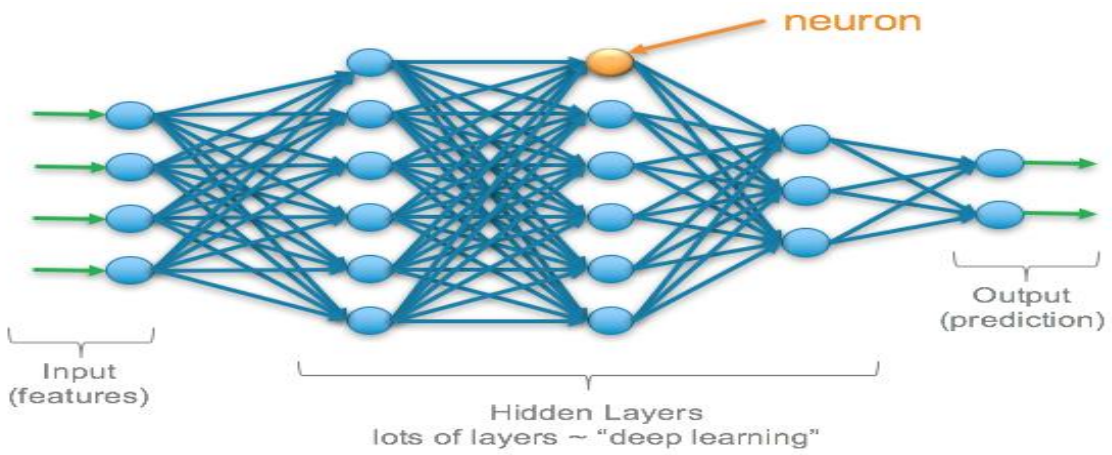


***Figure II.5** : Réseau neurone (MLP).*

On peut ainsi construire un réseau avec autant de couches et de neurones que l'on veut. Plus il y a de couches, plus on dit que le réseau est **profond** (*deep*) et plus le modèle devient **riche**, mais aussi **difficile** à entraîner. C'est ça, le **Deep Learning**.

II.6 L'Apprentissage :[9]

Nous venons de voir comment un neurone reçoit les informations d'entrées, comment il les utilise pour calculer la sortie, et comment sont les interconnections entre eux.

Comme le cerveau humain, les réseaux de neurones artificiels peuvent apprendre par expérience. Dans cette section, nous allons voir comment les réseaux de neurones peuvent apprendre est minimiser les erreurs à travers les techniques d'apprentissage.

Les algorithmes d'apprentissage modifient la valeur des poids entre les neurones ainsi que la valeur des biais de façon à améliorer la performance. Trois grandes classes d'apprentissage existent :

II.6.1 Apprentissage non-supervisé :

Il n'y pas de connaissances à priori des sorties désirées pour des entrées données. En fait, c'est de l'apprentissage par exploration où l'algorithme d'apprentissage ajuste les poids des liens entre neurones de façon à maximiser la qualité de classification des entrées (fig II.6).

***Figure II.6** : Apprentissage non supervisé.*

II.6.2 Apprentissage par renforcement :

Dans ce cas, bien que les sorties idéales ne soient pas connues directement, il y a un moyen quelconque de connaître si les sorties s'approchent ou s'éloignent du but visé. Ainsi, les poids sont ajustés de façons plus ou moins aléatoire et la modification est conservée si l'impact est positif ou rejetée sinon.

II.6.3 Apprentissage supervisé (back propagation) :

Cet algorithme d'apprentissage ne peut être utilisé que lorsque les combinaisons d'entrées-sorties désirés sont connues. L'apprentissage est alors facilité et par là, beaucoup plus rapide que pour les deux autres algorithmes puisque l'ajustement des poids est fait directement à partir de l'erreur, soit la différence entre la sortie obtenue et la sortie désirée.

L'algorithme le plus connue pour l'apprentissage supervisé BP est le gradient

Par définition, le gradient pour un neurone est l'erreur relative à ce neurone. Il peut en quelque sorte être vu comme la contribution du neurone à l'erreur globale.

A chaque **back propagation**, on calcule le gradient de chaque neurone, en commençant par ceux de la couche de sortie (car les gradients des couches inférieures se calculent se calculent à partir des gradients des couches supérieures) (fig II.7)**.**

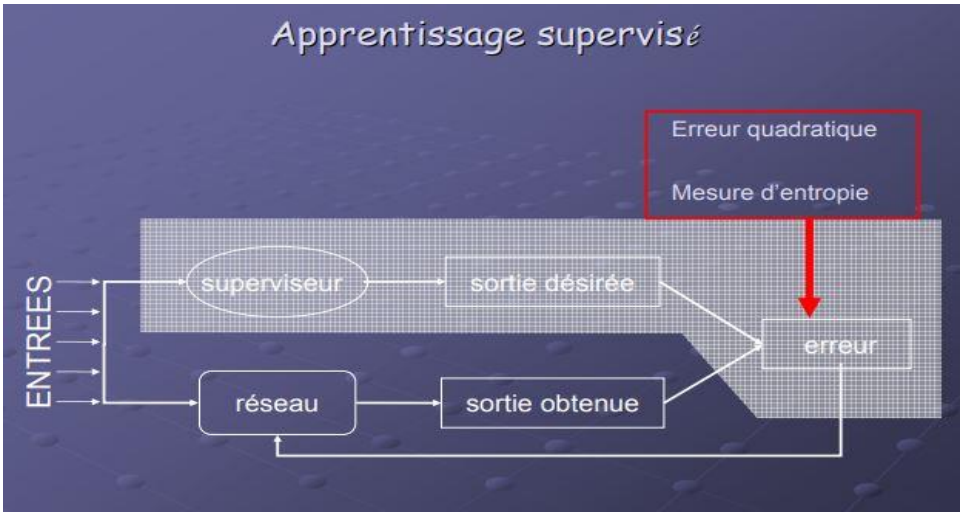


***Figure II.7** : Apprentissage supervisé.*

Comme nous devions identifier des objets et classer des images, nous avons besoin d'un modèle d'apprentissage supervisé.

Parce que l'apprentissage supervisé est une méthode qui consiste à utiliser des données d'entrainement étiquetées pour former un modèle qui peut ensuite être généralisée pour de nouveaux exemples.

II.7 Les différentes de Structure d'interconnexion des neurones :[10]

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle. Elle peut être quelconque, mais le plus souvent il est possible de distinguer une certaine régularité.

On a cité précédemment les deux topologies les plus simples (MLP, perceptron), on va présenter les autres types.

II.7.1 Réseau à connexions récurrentes (RNN) :

Les connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation défini dans un réseau multicouche. Ces connexions sont le plus souvent locales (fig II.8).

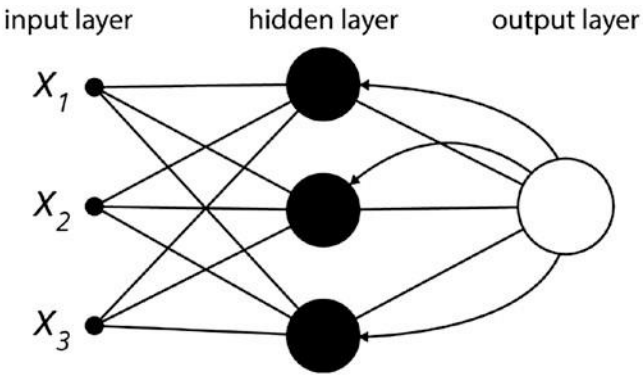


***Figure II.8** : Réseaux à connexions récurrente.*

II.7.2 Réseau à connexion complète :

C'est la structure d'interconnexion la plus générale. Chaque neurone est connecté à tous les neurones du réseau à double sens (fig II.9).

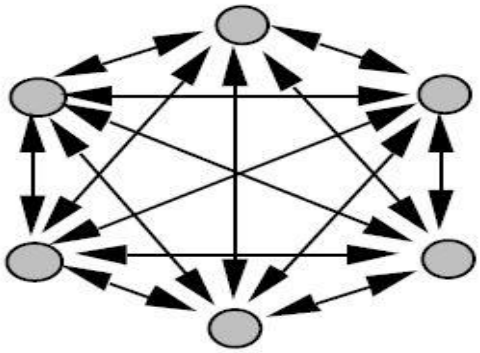


***Figure II.9** : Réseau à connexion complète.*

Il existe plusieurs types de réseaux avec des connections très spécifiques comme Restricted Boltzmann Machines (RBMs), et Deep Belief Networks (DBNs).

II.8 Réseau de neurone convolutif (CNN) :[11]

Le type de réseau qui nous concerne (intéresse) est un type très connu, il utilise comme données d'entrées des images, c'est le réseau de neurone convolutif CNND'une manière générale les couches qui constituent un CNN sont données en deux phases :

- Apprentissage des caractéristiques : contient deux couches (convolution et pooling)
- Classification : contient une couche entièrement connectée

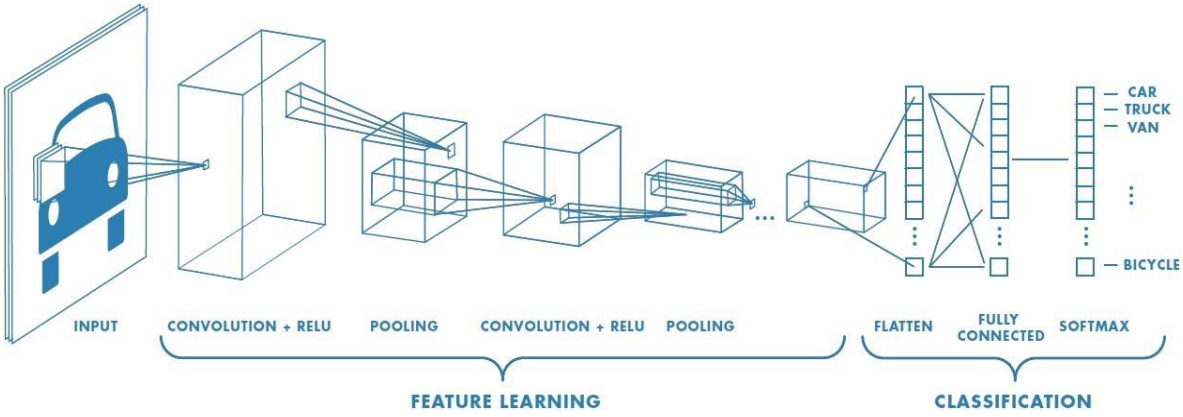


***Figure II.10** : Différentes couches CNN.*

L'objectif est de permettre aux machines de voir le monde comme le font les humains, de le percevoir de manière similaire et même d'utiliser les connaissances pour une multitude de tâches telles que la reconnaissance d'images et de vidéos, l'analyse et la classification d'images, la recréation de médias, les systèmes de recommandation, le traitement du langage naturel, etc. Les progrès de la vision par ordinateur avec apprentissage approfondi ont été construits et perfectionnés avec le temps, principalement au moyen d'un algorithme particulier - un réseau neuronal convolutif.

II.8.1 Définitions de CNN :

Un réseau neuronal convolutif (CNN) est un algorithme d'apprentissage approfondi qui peut prendre en compte une image d'entrée, attribuer de l'importance (poids et biais appelables) à divers aspects ou objets de l'image et être capable de les différencier les uns des autres. Le prétraitement requis dans un CNN est beaucoup moins important que pour les autres algorithmes de classification. Alors que dans les méthodes primitives, les filtres sont conçus à la main, les

CNN ont la capacité de choisir les filtres d'une manière systématique (i.e. il n'y a aucune conception manuelle).

L'architecture d'un CNN est analogue à celle du schéma de connectivité des neurones dans le cerveau humain et s'inspire de l'organisation du cortex visuel.

II.8.2 l'importance de CNN :

Dans la figure 2.11, nous avons une image RVB 4x4 qui a été séparée par ses trois plans de couleur - Rouge, Vert et Bleu (fig II.11).

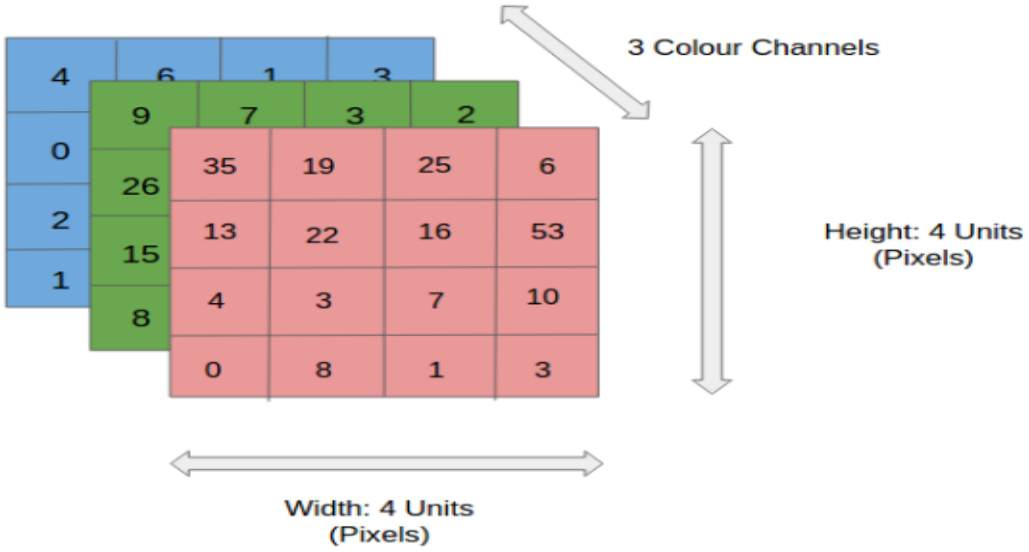


***Figure II.11** : Image RGB 4*4.*

Il est clair que l'intensité des calculs est très grande pour des images de tailles importantes (à partir de 8K). Le rôle du CNN est de réduire les images en une forme plus facile à traiter, sans perdre les caractéristiques qui sont essentielles pour obtenir une bonne prédiction. Ceci est important lorsque nous devons concevoir une architecture qui est non seulement bonne pour apprendre les caractéristiques mais qui est également extensible à des ensembles de données massifs.

II.8.3 les couches utilisées dans un CNN :

a) La couche de convolution (Convolution Layer) :

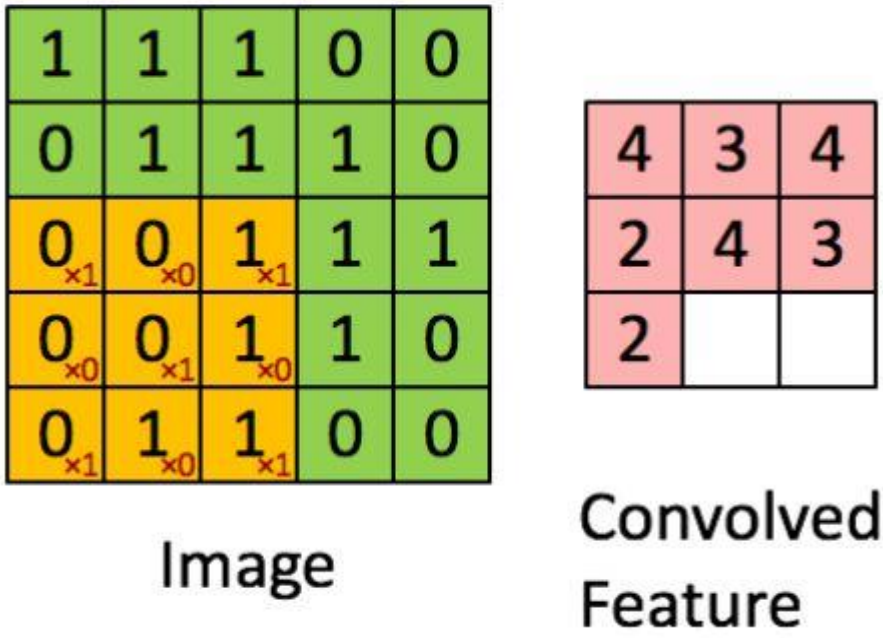


***Figure II.12** : Image de 5*5 avec le filtre de convolution.*

Dimensions de l'image = 5 (hauteur) x 5 (largeur) x 1 (nombre de canaux, par exemple RGB)

Dans la démonstration ci-dessus, la section verte ressemble à notre image d'entrée 5x5x1, I. L'élément impliqué dans la réalisation de l'opération de convolution dans la première partie d'une couche convolution elle est appelé le filtre, représenté dans la couleur jaune. Nous avons choisi le filtre comme matrice 3x3x1.

Filtre=

1 0 1

0 1 0

1 0 1

Le noyau se décale 9 fois, chaque fois qu'il effectue une opération de multiplication matricielle entre le filtre et la partie de l'image.

*Mouvement du filtre :

Le filtre se déplace vers la droite avec une certaine valeur de déplacement jusqu'à ce qu'il analyse la largeur complète. Ensuite, il descend jusqu'au début (à gauche) de l'image avec la même valeur de déplacement et répète le processus jusqu'à ce que l'image soit entièrement parcourue.

*Opération de convolution sur une matrice d'image MxNx3 avec un filtre 3x3x3 :

Dans le cas d'images à canaux multiples (par exemple RVB), le noyau a la même profondeur que celle de l'image d'entrée. La multiplication matricielle est effectuée entre le filtre K_n et l'image I_n du plan n(n =1,2,3) ([K1, I1] ; [K2, I2] ; [K3, I3]) et tous les résultats sont additionnés avec le biais pour nous donner une sortie de caractéristiques convolutives , La figure II-13 introduit toutes les étapes de convolution d'une image a canaux multiples.

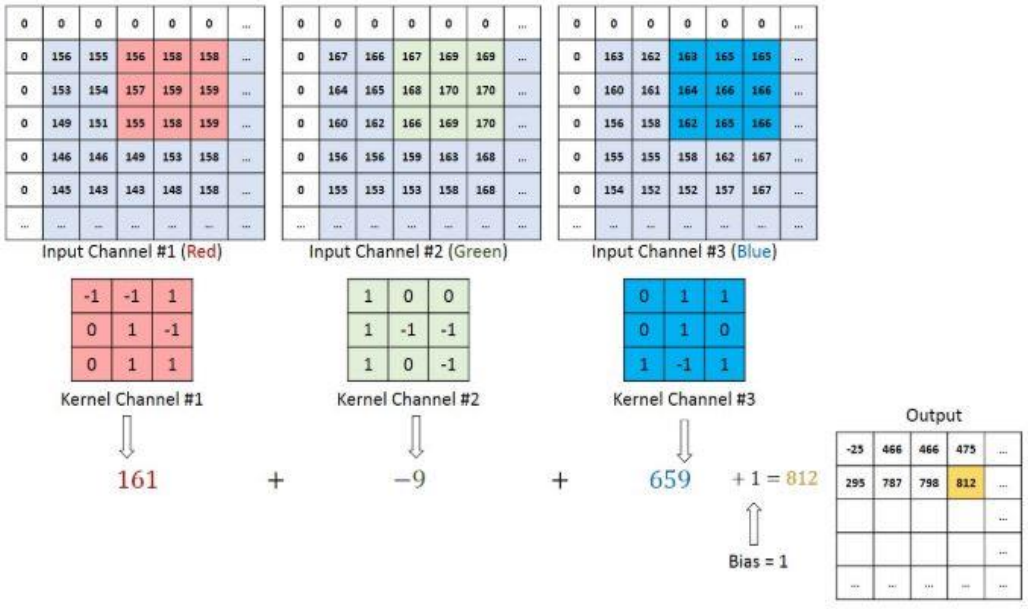


***Figure II.13** : Opération de convolution sur une matrice d'image MxNx3 avec un filtre 3x3x3.*

b) la couche de mise en commun (Pooling layer) :

Tout comme la couche convolution, la couche de mise en commun est responsable de la réduction de la taille spatiale de la caractéristique convolutive. Cela permet de réduire la puissance de calcul requise pour traiter les données par la réduction de la dimension. De plus, elle est utile pour extraire les caractéristiques dominantes qui sont invariantes en rotation et en position, maintenant ainsi le processus du l'entrainement efficace du modèle.

Il existe deux types de mise en commun :

- Mise en commun maximale (Max Pooling) : renvoie la valeur maximale de la partie de l'image couverte par le filtre.
- Mise en commun moyenne (average Pooling) : renvoie la moyenne de toutes les valeurs de la partie de l'image couverte par le filtre.
  (Figure II-14).

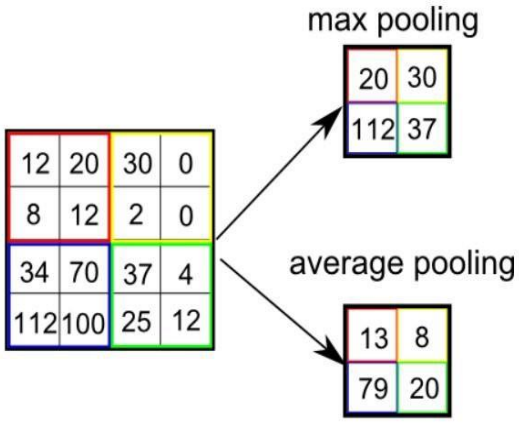


***Figure II.14**: Les types de pooling*

Dans un CNN on trouve une multitude de paires de couches **convolutions** et **de mise en commun** pour extraire les caractéristiques d'un objet d'intérêt, dans notre cas les objets d'intérêt sont les ilots**.** En fonction de la complexité des images, le nombre de ces paires de couches peut être augmenté pour capturer encore plus de détails de bas niveau, mais au prix d'une plus grande puissance de calcul.

c) La couche entièrement connectée (Fully-connected)

L'ajout d'une couche entièrement connectée est un moyen (généralement) peu coûteux d'apprendre des combinaisons non linéaires des caractéristiques de haut niveau telles que représentées par la sortie de la couche convolution. La couche entièrement connectée apprend une fonction éventuellement non linéaire dans cet espace.

Maintenant image d'entrée converti en une forme appropriée pour le Perceptron multi-niveaux, l'image va être aplatie en un vecteur colonne. La sortie aplatie est transmise à un réseau de neurones de feed-forward et une rétropropagation est appliquée à chaque itération de l'apprentissage. Sur une série d'époques, le modèle est capable de distinguer les éléments dominants et certains éléments de bas niveau dans les images et de les classer en utilisant la technique de classification Softmax.

II.9 Définition du modèle YOLO : [11] [12]

YOLO (**Y**ou **O**nly **L**ook **O**nce) est un réseau qui utilise des algorithmes de (DL) pour la détection d'objets. YOLO effectue la détection d'objets en classant certains objets dans l'image et en déterminant où ils se trouvent sur celle-ci.

Par exemple, si nous entrons une image d'un troupeau de moutons dans un réseau YOLO, celui-ci générera un vecteur de boîtes englobantes pour chaque mouton et le classifiera comme tel.

II.9.1 l'importance de YOLO :

Les autres méthodes de détection d'objets, comme les réseaux neuronaux régionalisés (R-CNN), y compris d'autres variantes comme les R-CNN rapides, se concentre sur une région spécifique de l'image et entraîne chaque composant séparément.

Ce processus nécessite que le R-CNN classifie 2000 régions par image, ce qui le rend très long (47 secondes par image test individuelle). Il ne peut donc pas être mis en œuvre en temps réel.

Cela rend les réseaux de détection d'objets tels que R-CNN plus difficiles à optimiser et plus lents que YOLO.

Le modèle YOLO est beaucoup plus rapide (45 images par seconde) et plus facile à optimiser que les algorithmes précédents, car il est basé sur un algorithme qui n'utilise qu'un seul réseau neuronal pour exécuter tous les éléments de la tâche.

Le tableau suivant (Tableau II.2) représente une comparaison entre R-CNN et YOLO.

Le modèle	FPS	Vitesse en temps real
Fast YOLO	155	OUI
YOLO	45	OUI
YOLO VGG-16	21	NON
FAST R-CNN	0.5	NON
FAST R-CNN VGG-16	7	NON
FAST R-CNN ZF	18	NON

Tableau II.2 : Comparaison entre R-CNN et YOLO.

II.9.2 L'architecture YOLO :

Un réseau YOLO se compose les parties principales :

a) Un réseau de neurones entièrement convolutif :

YOLO n'utilise que des couches convolutives, ce qui en fait un réseau entièrement convolutif (FCN). Dans le document YOLO v3, les auteurs présentent une nouvelle architecture plus profonde de l'extracteur de caractéristiques appelée Darknet-53. Comme son nom l'indique, il contient 53 couches convolutionnelles, chacune suivie d'une couche de normalisation par lots et d'une activation du ReLU de fuite. Aucune forme de mise en commun n'est utilisée et une couche convolutionnelle avec la phase 2 est utilisée pour le sous-échantillonnage des cartes de caractéristiques. Cela permet d'éviter la perte de caractéristiques de bas niveau souvent attribuée à la mise en commun.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

***Figure II.15** : Modèle Darknet-53.*

YOLO est invariant par rapport à la taille de l'image d'entrée. Cependant, dans la pratique, nous pourrions vouloir nous en tenir à une taille d'entrée constante en raison de divers problèmes qui ne se manifestent que lorsque nous mettons en œuvre l'algorithme.

L'un de ces problèmes est que si nous voulons traiter nos images par lots (les images par lots peuvent être traitées en parallèle par le GPU, ce qui permet d'augmenter la vitesse), nous devons avoir toutes les images de hauteur et de largeur fixes. Cela est nécessaire pour concaténer plusieurs images en un grand lot.

Le réseau sous-échantillonne l'image par un facteur appelé la foulée du réseau. Par exemple, si le pas du réseau est de 32, alors une image d'entrée de taille 416 x 416 donnera une sortie de taille 13 x 13. En général, la longueur de pas d'une couche du réseau est égale au facteur par lequel la sortie de la couche est inférieure à l'image d'entrée du réseau.

b) Interprétation de la prédiction :

L'entrée est un lot d'images de la forme (m, 416, 416, 3).

Le résultat est une liste de boîtes englobantes avec les classes reconnues. Chaque boîte englobante est représentée par 6 chiffres (pc,bx,by,bh,bw,c)(pc,bx,by,bh,bw,c). Si nous

développons *c*c en un vecteur à 80 dimensions, chaque boîte englobante est alors représentée par 85 chiffres.

Comme dans tous les détecteurs d'objets, les caractéristiques apprises par les couches convolutionnelles sont transmises à un classificateur/régresseur qui effectue la prédiction de détection (coordonnées des boîtes englobantes, étiquette de classe, etc.).

Dans YOLO, la prédiction est faite en utilisant une couche convolutionnelle qui utilise des convolutions 1 x 1. Ainsi, la première chose à remarquer est que notre sortie est une carte des caractéristiques. Comme nous avons utilisé des convolutions 1 x 1, la taille de la carte de prédiction est exactement la taille de la carte des caractéristiques qui la précède. Dans YOLO v3, la façon dont vous interprétez cette carte de prédiction est que chaque cellule peut prédire un nombre fixe de cases limites.

Par exemple, si nous avons des entrées (B x (5 + C)) dans la carte des caractéristiques. B représente le nombre de cases limites que chaque cellule peut prédire. Selon l'article, chacune de ces boîtes englobantes B peut se spécialiser dans la détection d'un certain type d'objet. Chacune des boîtes englobantes a des attributs 5 + C, qui décrivent les coordonnées du centre, les dimensions, le score d'objectivité et les confidences de classe C pour chaque boîte englobante. YOLO v3 prévoit 3 boîtes de délimitation pour chaque cellule.

Nous attendons de chaque cellule de la carte des caractéristiques qu'elle prévoie un objet à travers une de ses boîtes englobantes si le centre de l'objet tombe dans le champ de réception de cette cellule.

Ceci est lié à la façon dont YOLO est formé, où une seule boîte de délimitation est responsable de la détection d'un objet donné. Tout d'abord, nous devons déterminer à laquelle des cellules appartient cette boîte englobante.

Pour ce faire, nous divisons l'image d'entrée en une grille de dimensions égales à celles de la carte finale des caractéristiques. Regardons l'exemple ci-dessous, où l'image d'entrée est de 416 x 416, et la longueur du réseau de 32 pas. Comme indiqué précédemment, les dimensions de la carte des caractéristiques seront de 13 x 13. Nous divisons ensuite l'image d'entrée en cellules de 13 x 13.

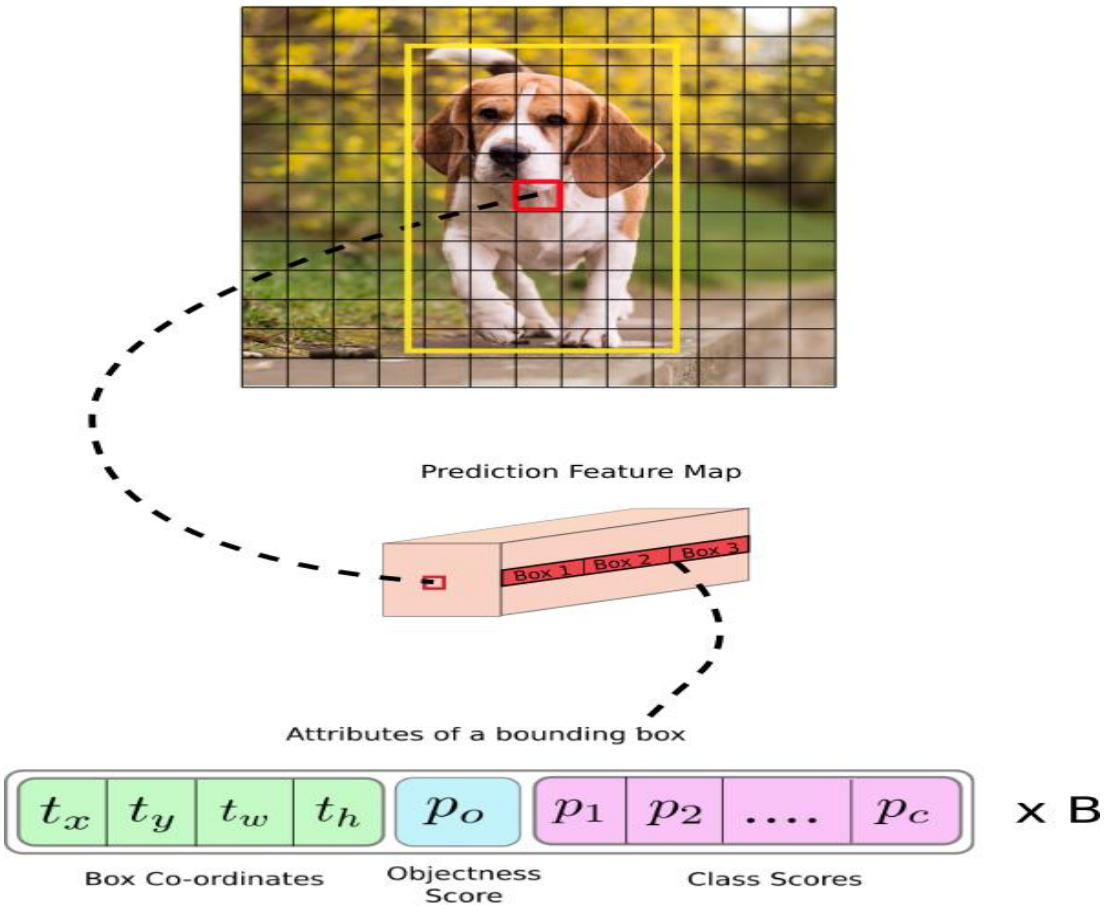


Figure II.16 : *Les attribues des cadres.*

Ensuite, la cellule (sur l'image d'entrée) contenant le centre de la boîte de vérité de sol d'un objet est choisie pour être celle responsable de la prédiction de l'objet. Dans l'image, c'est la cellule marquée en rouge, qui contient le centre de la boîte de vérité de sol (marquée en jaune).

Maintenant, la cellule rouge est la 7ème cellule de la 7ème ligne de la grille. Nous attribuons maintenant la 7e cellule de la 7e rangée de la carte des caractéristiques (cellule correspondante sur la carte des caractéristiques) comme étant celle qui est responsable de la détection du chien.

Cette cellule peut maintenant prédire trois cases de délimitation. Laquelle sera assignée à l'étiquette de chien . Pour comprendre cela, nous devons nous pencher sur le concept des ancres. (Notez que la cellule dont nous parlons ici est une cellule de la carte des caractéristiques de prédiction. Nous divisons l'image d'entrée en une grille afin de déterminer quelle cellule de la carte des caractéristiques de prédiction est responsable de la prédiction).

c) Boîtes d'ancrage et prévision :

Il peut être judicieux de prévoir la largeur et la hauteur de la boîte de délimitation, mais dans la pratique, cela conduit à des pentes instables pendant l'entraînement. Au lieu de cela, la plupart des détecteurs d'objets modernes prédisent des transformations log-espace, ou simplement des décalages vers des boîtes de délimitation prédéfinies par défaut appelées ancres.

Ensuite, ces transformations sont appliquées aux boîtes d'ancrage pour obtenir la prédiction. YOLO v3 possède trois ancres, ce qui permet de prédire trois boîtes de délimitation par cellule.

Les ancres sont des sortes de boîtes englobantes préalables, qui ont été calculées sur l'ensemble de données COCO. Nous allons prédire la largeur et la hauteur de la boîte en tant que décalage par rapport aux centroïdes de la grappe. Les coordonnées du centre de la boîte par rapport à l'emplacement de l'application du filtre sont prédites à l'aide d'une fonction sigmoïde.

La formule suivante décrit comment la sortie du réseau est transformée pour obtenir des prédictions de boîte englobante :

*Image pour le poste :

Ici bx, by, bw, bh sont les coordonnées du centre x,y, la largeur et la hauteur de notre prédiction. tx, ty, tw, th est ce que le réseau produit. cx et cy sont les coordonnées en haut à gauche de la grille. pw et ph sont les dimensions d'ancrage de la boîte.

*Coordonnées du centre :

Nous utilisons une fonction sigmoïde pour prédire les coordonnées du centre. Normalement, YOLO ne prédit pas les coordonnées absolues du centre de la boîte englobante. Il prédit les décalages qui sont :

Par rapport au coin supérieur gauche de la cellule de la grille qui prédit l'objet.

Normalisés par les dimensions de la cellule de la carte des caractéristiques, c'est-à-dire 1.

Prenons par exemple le cas de notre image de chien ci-dessus. Si les coordonnées de prédiction pour le centre sont (0,4, 0,7), cela signifie que le centre se trouve à (6,4, 6,7) sur la carte des caractéristiques 13 x 13. (Puisque les coordonnées en haut à gauche du carré rouge sont (6,6)).

Mais attendez, que se passe-t-il si les coordonnées x et y prédites sont supérieures à un, par exemple (1,2, 0,7). Cela signifie que le centre se trouve à (7,2, 6,7). Le centre se trouve maintenant dans la cellule juste à droite de notre cellule rouge, ou dans la 8e cellule de la 7e rangée. Cela rompt avec la théorie de YOLO car si nous postulons que la case rouge est responsable de la prédiction du chien, le centre du chien doit se trouver dans la case rouge, et non dans celle qui se trouve à côté. Pour résoudre ce problème, la sortie est donc passée par une fonction sigmoïde, qui écrase la sortie dans une plage de 0 à 1, maintenant effectivement le centre dans la grille qui est prédite.

d) Dimensions de la boîte de délimitation :

Les dimensions de la boîte englobante sont prédites en appliquant une transformation log-espace à la sortie, puis en la multipliant par une ancre.

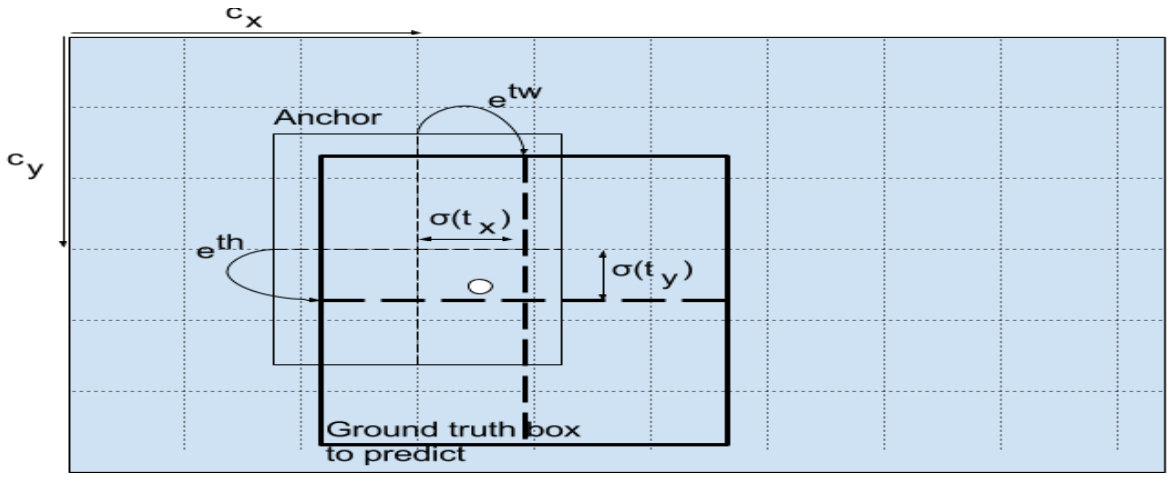


Figure II.17 : *Les paramètres de prédiction.*

Ici, les prédictions, *bw* et *bh*, sont normalisées par la hauteur et la largeur de l'image. (Les étiquettes de formation sont choisies de cette façon). Ainsi, si les prédictions bxbx et $by$by pour la boîte contenant le chien sont (0,3, 0,8), alors la largeur et la hauteur réelles sur la carte des caractéristiques 13 x 13 sont (13*0,3, 13*0,8).

e) Score d'objectivité et confiance en classe :

Le score "sapin de tous les objets" représente la probabilité qu'un objet soit contenu dans une boîte englobante. Il devrait être proche de 1 pour la grille rouge et les grilles voisines, alors qu'il est proche de 0 pour, disons, la grille aux coins.

Le score d'objectivité est également passé par un sigmoïde, car il doit être interprété comme une probabilité.

Parlant de Confidences de classe, elles représentent les probabilités que l'objet détecté appartienne à une classe particulière (chien, chat, personne, voiture, vélo, etc.). Dans les anciennes versions de YOLO, il était utilisé pour adoucir les scores de classe.

Dans les versions YOLO, les auteurs ont décidé d'utiliser plutôt le sigmoïde. La raison en est que le softmaxing des scores de classe suppose que les classes sont mutuellement exclusives. En d'autres termes, si un objet appartient à une classe, il ne peut pas appartenir à une autre classe. Cela est vrai pour la base de données COCO, que nous allons d'abord mettre en œuvre.

f) Prévision à différentes échelles :

YOLO v3 fait des prédictions sur 3 échelles différentes. La couche de détection est utilisée pour détecter des cartes de caractéristiques de trois tailles différentes, ayant des pas de 32, 16, 8 respectivement. Cela signifie qu'avec une entrée de 416 x 416, nous effectuons des détections sur des échelles de 13 x 13, 26 x 26 et 52 x 52.

Le réseau sous-échantillonne l'image d'entrée jusqu'à la première couche de détection, où une détection est effectuée à l'aide de cartes de caractéristiques d'une couche à pas de 32. De plus, les couches sont suréchantillonnées par un facteur de 2 et concaténées avec les cartes de caractéristiques d'une couche précédente ayant des tailles de cartes de caractéristiques identiques. Une autre détection est maintenant effectuée sur la couche avec la foulée 16. La même procédure de suréchantillonnage est répétée, et une détection finale est effectuée sur la couche de la foulée.

À chaque échelle, chaque cellule prévoit 3 boîtes de délimitation utilisant 3 ancres, ce qui fait que le nombre total d'ancres utilisées est de 9 (les ancres sont différentes selon les échelles).

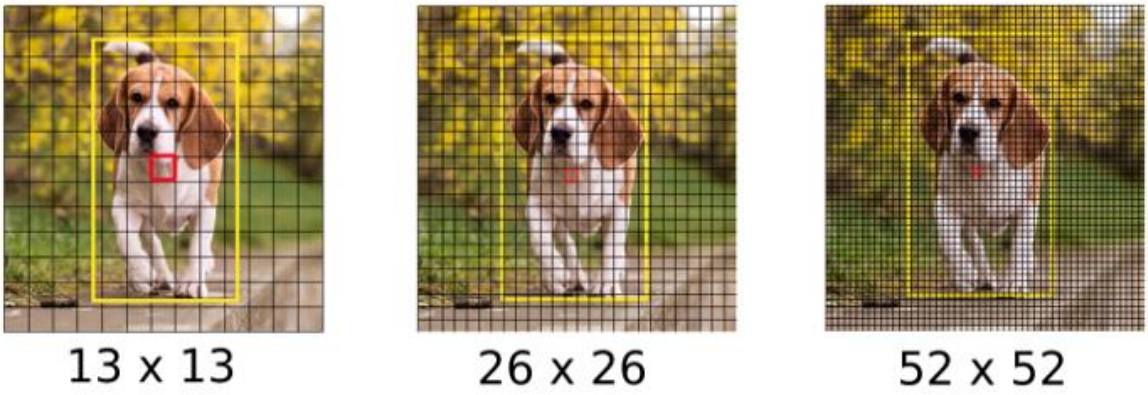


Figure II.18 : *Trois échelles de d'édiction.*

Les auteurs de YOLO signalent que cela permet de mieux détecter les petits objets, une plainte fréquente avec les versions précédentes de YOLO. Le suréchantillonnage peut aider le réseau à apprendre des caractéristiques à grain fin qui sont essentielles pour la détection de petits objets.

g) Traitement des sorties (filtrage avec un seuil sur les notes de classe) :

Pour une image de taille 416 x 416, YOLO prédit ((52 x 52) + (26 x 26) + 13 x 13)) x 3 = 10647 boîtes englobantes. Cependant, dans le cas de notre image, il n'y a qu'un seul objet, un chien. Alors, vous pouvez vous demander comment réduire les détections de 10647 à 1 ?

Tout d'abord, nous filtrons les boîtes en fonction de leur score d'objectivité. En général, les cases ayant un score inférieur à un seuil (par exemple inférieur à 0,5) sont ignorées. Ensuite, la suppression non maximale (NMS) vise à remédier au problème des détections multiples d'une même image. Par exemple, les 3 cases limitrophes de la grille rouge peuvent détecter une case ou les cases adjacentes peuvent détecter le même objet, après supprimer les détections multiples.

Plus précisément, nous allons effectuer ces étapes :

- Se débarrasser des cases ayant un faible score (c'est-à-dire que la case n'est pas très sûre de détecter une classe).
- Sélectionnez une seule case lorsque plusieurs cases se chevauchent et détectent le même objet (suppression non maximale).

Dans un premier temps, nous allons appliquer un premier filtre par seuillage. Nous voudrions nous débarrasser de toute boîte pour laquelle la classe "score" est inférieure à un seuil choisi.

Le modèle nous donne un total de 19x19x5x85 chiffres, chaque case étant décrite par 85 chiffres. Il sera pratique de réarranger le tenseur dimensionnel (19,19,5,85) (ou (19,19,425)) en les variables suivantes :

box_confidence : tenseur de forme (19×19,5,1) contenant pc (probabilité de confiance qu'il y ait un objet) pour chacune des 5 cases prédites dans chacune des cellules 19x19.

boxes : tenseur de forme (19×19,5,4) contenant (bx,by,bh,bw) pour chacune des 5 cases par cellule.

box_class_probs : tenseur de forme (19×19,5,80) contenant les probabilités de détection ($c1$,$c2$,...$c80$) pour chacune des 80 classes pour chacune des 5 cases par cellule.

Même après un filtrage par seuil sur les scores des classes, nous finissons par avoir beaucoup de boîtes qui se chevauchent. Un deuxième filtre pour sélectionner les bonnes cases est appelé NMS. NMS utilise la fonction très importante appelée "Intersection over Union", ou IoU.

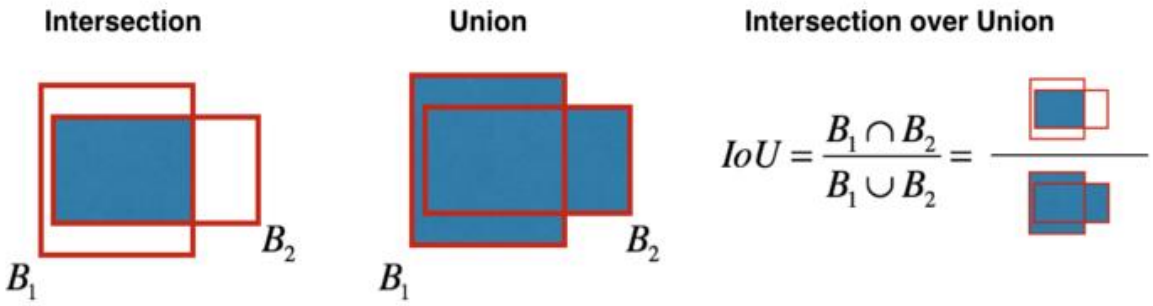


Figure II.19 : *Fonction "Intersection over Union".*

Mise en œuvre du protocole d'accord :

Nous définissons donc une boîte en utilisant ses deux coins (en haut à gauche et en bas à droite) : (x1, y1, x2, y2) plutôt que le point médian et la hauteur/largeur.

Pour calculer la surface d'un rectangle, nous devons multiplier sa hauteur (y2 - y1) par sa largeur (x2 - x1).

Nous devons également trouver les coordonnées (xi1, yi1, xi2, yi2) de l'intersection de deux cases :

3.1. xi1 = maximum des coordonnées x1 des deux cases

3.2. yi1 = maximum des coordonnées y1 des deux cases

3.3. xi2 = minimum des coordonnées x2 des deux cases

3.4. yi2 = minimum des coordonnées y2 des deux cases

Arguments :

case1 - première case, objet de la liste avec coordonnées (x1, y1, x2, y2)

case2 - deuxième case, objet de la liste avec coordonnées (x1, y1, x2, y2)

xi1 = max(case1[0], case2[0])

yi1 = max(case1[1], case2[1])

xi2 = min(case1[2], case2[2])

yi2 = min(case1[3], case2[3])

inter_area = (xi2 - xi1)*(yi2 - yi1)

Formule : Union(A,B) = A + B - Inter(A,B)

box1_area = (box1[3] - box1[1])*(box1[2]- box1[0])

box2_area = (box2[3] - box2[1])*(box2[2]- box2[0])

union_area = (case1_area + case2_area) - inter_area

calculer l'IoU

IoU = inter_area / union_area

Maintenant, pour mettre en œuvre la suppression non maximale, les étapes clés sont les suivantes :

- Sélectionnons la case qui a le score le plus élevé.
- Calculer son chevauchement avec tous les autres champs, et supprimer les champs qui le chevauchent plus que le seuil iou_threshold.
- Revenez à l'étape 1 et répétez jusqu'à ce qu'il n'y ait plus de cases ayant un score inférieur à celui de la case actuellement sélectionnée.

Ces étapes permettront de supprimer toutes les cases qui se chevauchent largement avec les cases sélectionnées. Seules les "meilleures" cases restent :

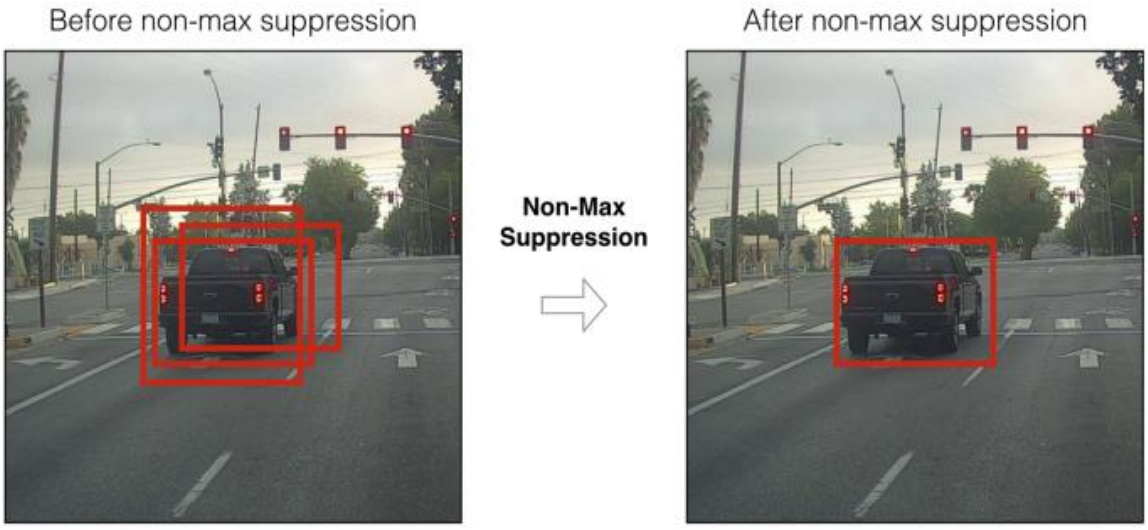


Figure II.20 : *Filtre non-max suppression.*

II.9.3 Comparaison entre les versions de YOLO :

La version la plus récente de YOLO est la troisième itération du réseau de détection d'objets. Les créateurs de YOLO ont conçu de nouvelles versions afin d'apporter des améliorations par rapport aux versions précédentes, en se concentrant principalement sur l'amélioration de la précision de la détection.

a) YOLO V1 :

La première version de YOLO a été introduite en 2015, qui s'entraînait sur le jeu de données ImageNet-1000. En effet, YOLO V1 avait du mal à identifier les petits objets qui apparaissaient comme un groupe et était inefficace pour généraliser les objets dans les images qui avaient des dimensions différentes de l'image formée. Il en résultait une mauvaise localisation des objets dans l'image d'entrée.

b) YOLO V2 :

YOLO V2 est sorti en 2016 sous le nom de YOLO9000. YOLO V2 utilisait un réseau à 19 couches avec 11 autres couches chargées de la détection d'objets. YOLO V2 est conçu pour prendre en charge le R-CNN plus rapide et le Single Shot multi-box Detector (SSD) qui a montré de meilleurs scores de détection d'objets.

c) YOLO V3 :

YOLO V3 est une mise à niveau progressive par rapport à YOLO V2. Cette architecture est composée de 53 couches formées sur Imagenet et de 53 autres chargées de la détection d'objets, soit 106 couches. Bien que cela ait considérablement amélioré la précision du réseau, cela a également permis de réduire la vitesse de 45 à 30 images par seconde.

II.9.4 Avantages et inconvénients de YOLO :

YOLO est plus rapide d'un ordre de grandeur (45 images par seconde) que les autres algorithmes de détection d'objets.

La limitation de l'algorithme YOLO est qu'il se bat avec de petits objets dans l'image, par exemple, il peut avoir des difficultés à détecter un groupe d'oiseaux.

II.10 Conclusion :

Dans ce chapitre, on a cité les différentes connaissances de base sur le Deep Learning, et aussi on a défini le modèle choisi, tous les outils indispensables ont été définis, afin d’être appliqués sous python dans le chapitre suivant pour avoir un modèle qui puisse reconnaitre les ilots.

Chapitre III :

Extraction des ilots à partir des cartes urbaines et satellitaires.

III.1 Introduction :

Dans les chapitres précédents, on a donné les notions de bases, qui sont des connaissances nécessaires pour entamer le but de ce chapitre qui sert à réalisation du projet.

Notre objectif est basé sur l'utilisation des outils Google Drive, Google Colab et le modèle YOLO qui nous permettons de gérer et traiter les données et les Dataset jusqu'aux résultats obtenus.

III.2 Les outils utilisés :

Il est nécessaire de présenter les outils informatiques utilisés avant d'entamer la présentation des différentes parties qui constituent notre système de détection d'ilots.

III.2.1 Langage de programmation python :

Python est un langage programmation open source le plus employé par les informaticiens.

Ce langage s'est propulsé en tête de la gestion d'infrastructure, d'analyse de données ou dans le domaine du développement de logiciels.

Malgré qi' il ne soit pas le plus puissant du point de vue calcule et rapidité d'exécution, mais dans le domaine AI est le plus utilisé car sa syntaxe et facile et pratique, il permet d'y progresser très certainement plus aisément que dans d'autres langages.

Enfin, concernant le domaine de (DL) Python se distingue tout particulièrement en offrant une pléthore de librairies de très grande qualité, couvrant tous les types d'apprentissages disponibles sur le marché.

III.2.2 Jupiter Notebook :

Jupiter Notebook est une application Web Open Source permettant de créer et de partager des documents contenant du code (écrit par langage de programmation python), des équations, des images et du texte. Avec cette application il est possible de faire du traitement de données de la visualisation de données, du Deep Learning.

III.2.3 Google Colab :

Google Colab ou Colaboratory est un service Cloud, offert par Google (gratuit), basé sur Jupiter Notebook et destiné à l'entrainement et à la recherche dans l'apprentissage automatique. Cette plateforme permet d'importer un ensemble de données d'images, d'entraîner un classificateur d'images sur cet ensemble et d'évaluer le modèle de Deep Learning directement dans le Cloud (fig III.1).

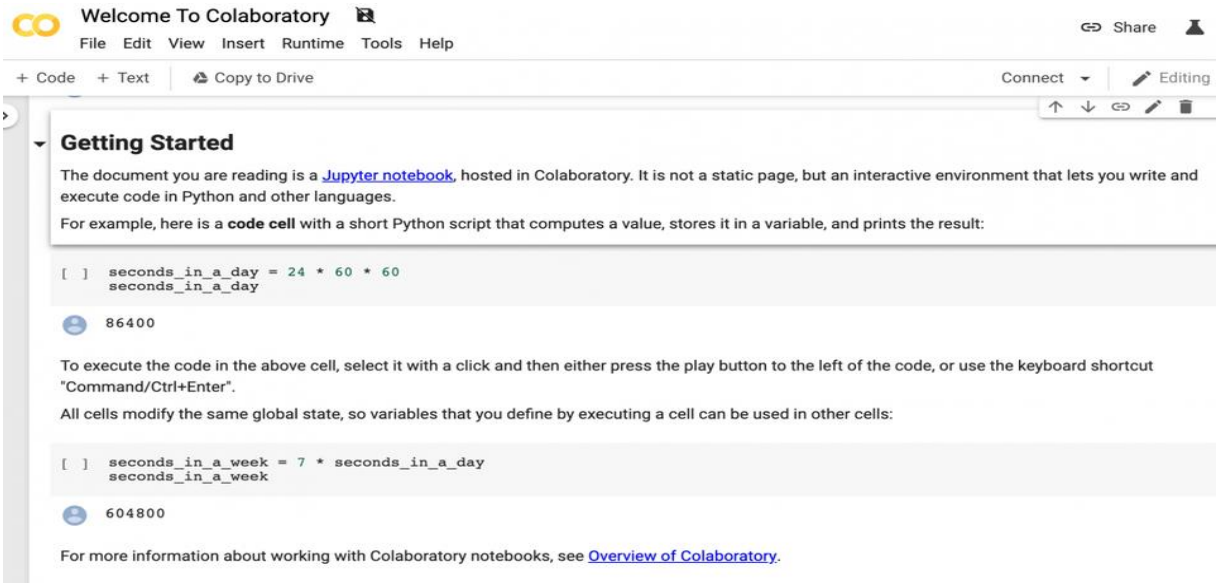


Figure III.1 : *Interface de Google Colab.*

On a utilisé Google Colab, car il présente les avantages suivants :

- Aucune configuration requise

  On n'a pas besoin d'installer quoi que ce soit sur notre ordinateur à l'exception d'un navigateur (Google chrome).

- Accès gratuit aux GPU

  Les Notebooks Colab exécutent les codes sur les serveurs Cloud de Google, nous avons donc à notre disposition toute la puissance du matériel Google.

III.2.4 Google Drive :

Google Drive est un service stockage et de partage de fichiers en ligne lancé par Google en avril 2012., il permet aux utilisateurs de stocker, partager, modifier et visualiser différents types de fichiers, et de les synchroniser à distance avec des terminaux fixes (PC, Mac) ou mobiles.

Google Drive est utilisé car :

- Il Permet de réserver un espace de stockage qui communique avec les autres services de Google (Google Docs, Google Colab).
- Il permet de Transfer des données entre Colab et Drive et de les synchroniser plus vite qu'avec une machine locale (PC, téléphone).
- Il permet une sauvegarde rapide et automatique de l'état d'avancement dans le Drive.

III.2.5 Logiciel Labelimg :

Labelimg est un outil graphique d'annotation d'images qui fournit des images avec des cadres après l'étiquetage. Il est écrit en Python et utilise Qt (une des interfaces graphiques les plus courantes pour python) pour son interface graphique.

Les données étiquetées en sortie sont enregistrées sous forme de fichiers XML au format PASCAL VOC. Le format YOLO est également pris en charge par cet outil (fig III.2).

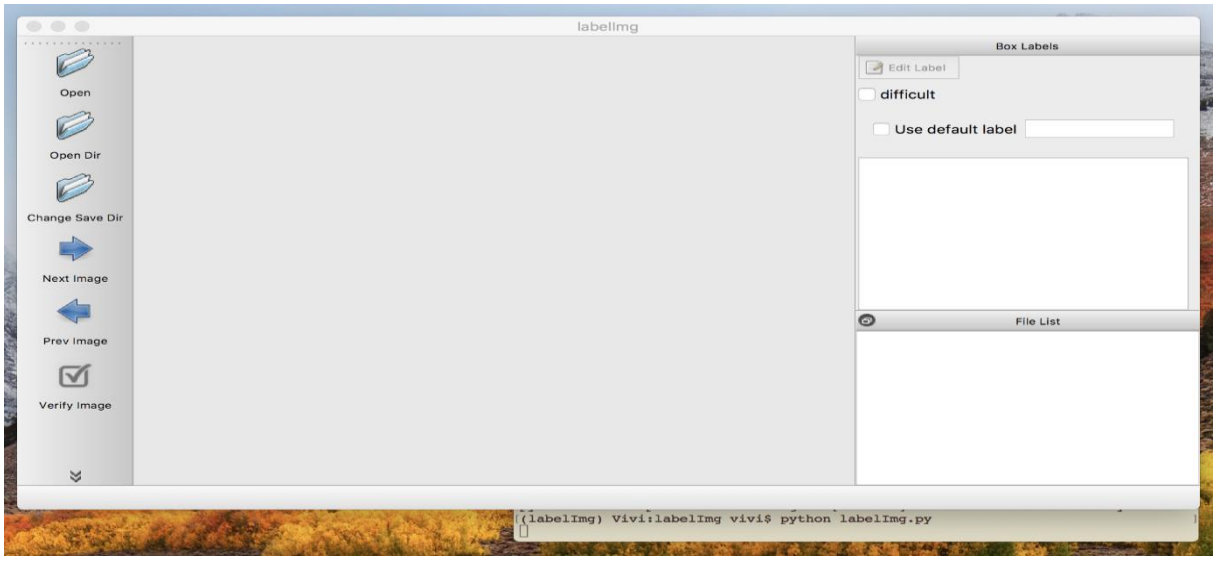


***Figure III.2** : Interface Labelimg.*

III.3 Schématisations des étapes du DL :

La schématisation employée par la majorité des projets de vision utilisant le DL est donnée dans le synoptique ci-dessous :

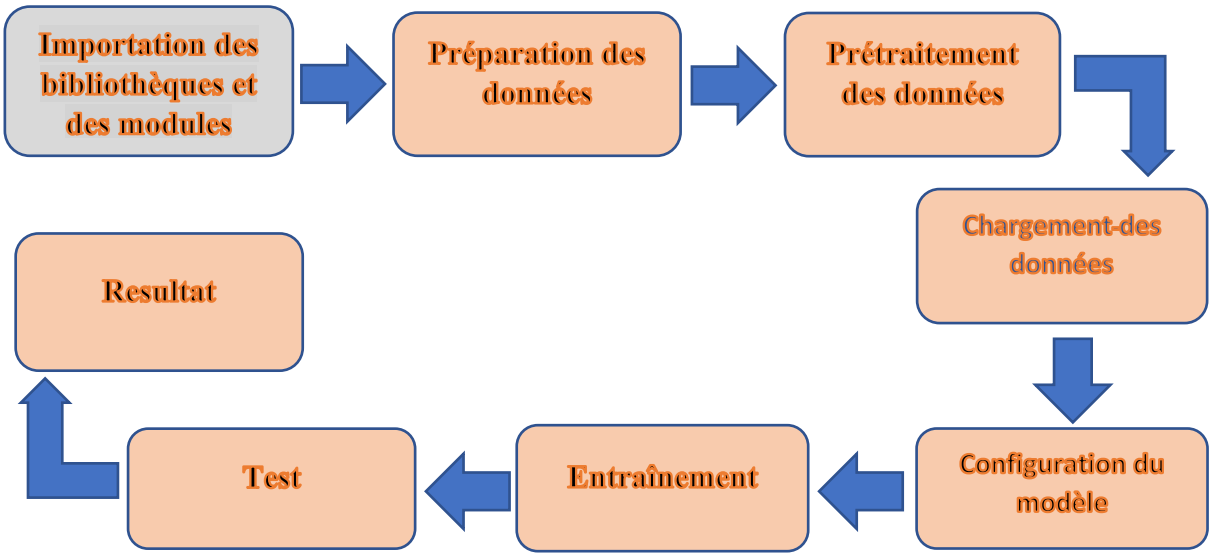


Figure III.3 : *Les étapes de détection par DL.*

- Importation des bibliothèques et modules : cette phase consiste installer des bibliothèques comme (Tensorflow,open cv…), elles contiennent des fonctions de (DL), pour notre cas toutes les fonctions préinstaller et inclue dans Google Colab.
- Préparation des donnes : cette phase a but de régler des paramètres qui nous permettre de préparer les donnes.
- Prétraitement des donnes : avec utilisation de logiciel Labelimg, on peut clairement montrer au machine les objets voulus (ilots), pour qu'elle puisse les employés dans la phase de l'entrainement.
- Chargement des données : dans cette phase on va envoyer les donnes étiqueter vers Cloud.
- Configuration de modèle : pour que le modèle choisi puisse être exécuté il lui faut des fichiers comme (train.txt,obj.data…) dans l'entré de modèle .
- Entrainement : le modèle va être entrainer pour qu'il puisse détecter les ilots.

- Test : après l'entrainement, le modèle on le change au mode test.
- Prédiction : on lui donne des images contiennes des ilots, il va les prédicter l'emplacement des ilots par des cadres.

III.4 Préparation des données :

Il est impératif de préparer l'étape de chargement des données, pour cela on doit Faire :

III.4.1 Une Activation du GPU :

Nous devons activer l'accélération GPU dans notre Google Colab afin que le système YOLO.v3 puisse traiter les détections plus de 100 fois plus vite que le CPU (fig III.4).

- Le GPU utiliser par Cloud : TESLA P4

Pour cela il y a des étapes à faire :

- Cliquons sur Modifier (Edit) en haut à gauche de notre carnet (Notebook).
- Cliquons sur Paramètres (Notebook settings) dans la liste.
- Sélectionnons GPU et appuyez sur "Save".

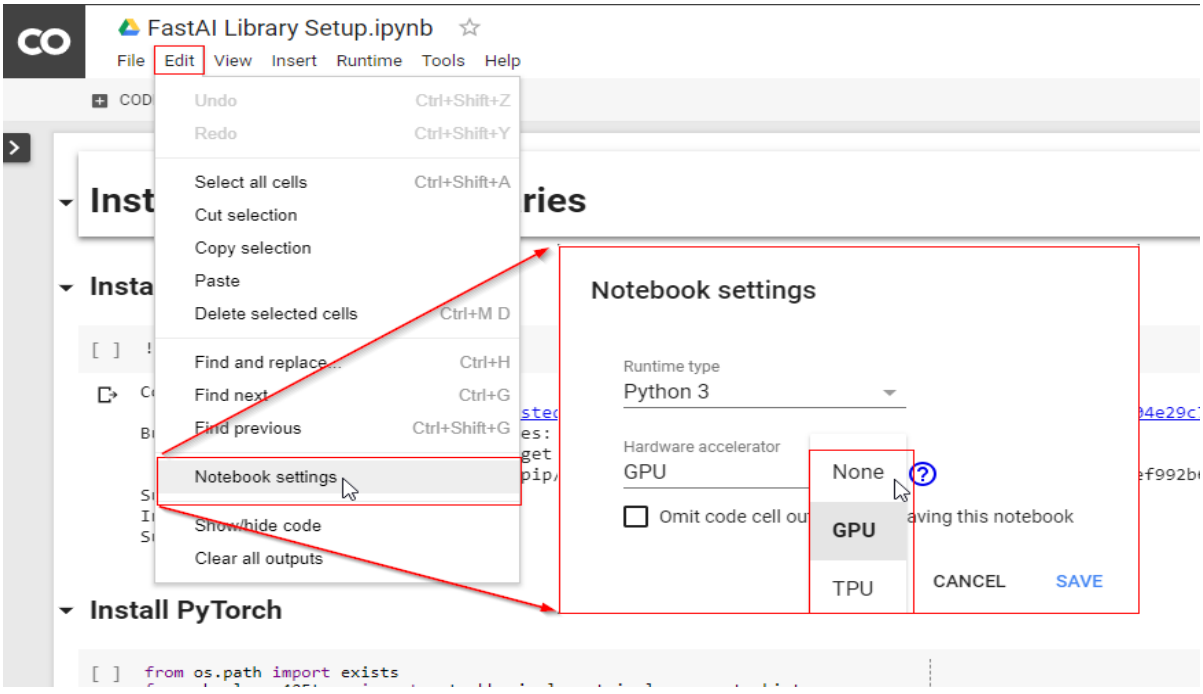


Figure III.4 *: Les étapes d'activation GPU.*

Pour connaitre la version (CUDA) de GPU on exécute la commande (1)

- !/usr/local/cuda/bin/nvcc –version (1)

En sortie on aura le code de la confirmation de la version utilisée (fig III.5)

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2019 NVIDIA Corporation
Built on Sun_Jul_28_19:07:16_PDT_2019
Cuda compilation tools, release 10.1, V10.1.243
```

Figure III.5 : Mise en évidence de la version de GPU.

III.4.2 Une Création de répertoire :

Il est important de créer un répertoire au niveau du PC, ce dernier a pour rôle de contenir tous les dossiers ainsi que les scripts utilisés, on distingue :

- Le Dossier qui contient la base de la donnée employée.
- Le Module YOLOv3.
- Le Fichier de la configuration du module.
- Les poids YOLOv3 pré-entraînés.

Et d'autre fichiers non cités et qui sont indispensables (fichier.cfg, fichier obj.data et obj.name, fichier train.txt).

Le répertoire est ensuite envoyé vers le Cloud.

III.4.3 Une utilisation de fonctions pour gérer les images :

Dans la gestion des images, nous avons utilisé deux fonctions :

- La fonction « ImShow » pour visualiser les résultats obtenus après l'entrainement (image de sortie), est définie comme suit :

```
def imShow(path):
  import cv2
  import matplotlib.pyplot as plt
  %matplotlib inline

  image = cv2.imread(path)
  height, width = image.shape[:2]
  resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)

  fig = plt.gcf()
  fig.set_size_inches(18, 10)
  plt.axis("off")
  plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
  plt.show()
```

Figure III.6 *: La fonction Imshow.*

- La fonction « Download » pour télécharger les images de sorties sur notre machine locale, elle est définie par le code suivant :

```
def download(path):
  from google.colab import files
  files.download(path)
```

Figure III.7 *: La fonction download.*

III.4.4 Une Connexion de Google Drive avec Google Cloud :

Il existe un code Pour relier le Google Drive avec le Cloud (fig III.8).

```
%cd ..
from google.colab import drive
drive.mount('/content/gdrive')
```

Figure III.8 *: Code pour relier Google Drive et Colab.*

Après la mise en connexion de ces deux derniers, on peut alors afficher, modifier, créer, supprimer et envoyer des fichiers entre eux.

III.5 Synoptique des entrées et sorties du modèle de détection des ilots :

Jusqu'à maintenant on a préparé un répertoire qui contient toutes nos données, et on a activé le GPU, défini les fonctions, Il est temps de créer un détecteur d'objets YOLOv3 personnalisé, c'est notre modèle, il possède des entrées, et des entrées de configuration et procure des sorties (fig III.9).

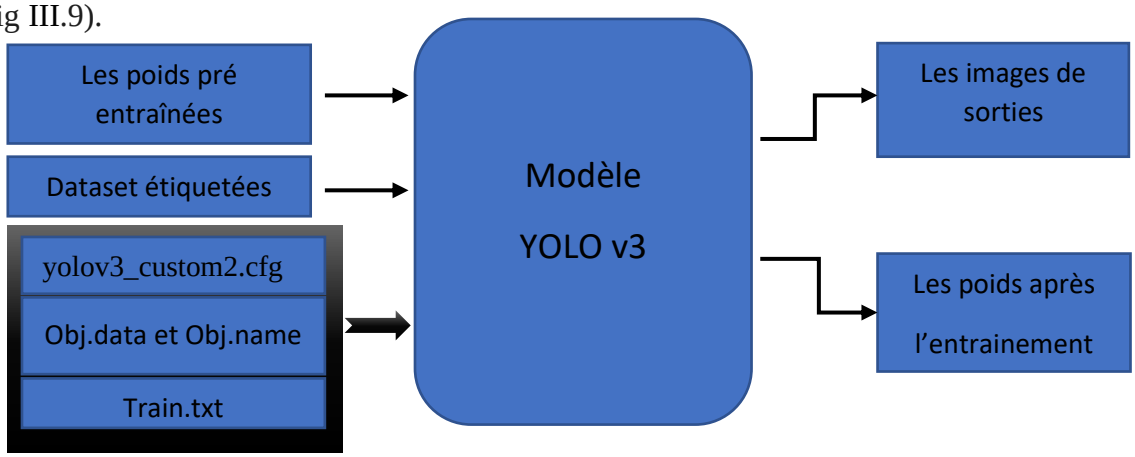


Figure III.9 : *Schéma synoptique des E/S du modèle YOLO.*

Les entrées sont :

- Ensemble de données personnalisées étiquetées.
- Ensemble des poids pré-entrainés.

Les entrées de configuration sont :

- Fichier .cfg personnalisé.
- Fichiers obj.data et obj.name.
- Fichier train.txt.

Les sorties sont :

- Les images de sorties.
- Les poids après l'entrainement.

Comme nous l'avons vu précédemment (chapitre 2), le modèle YOLO effectue la détection d'objets par génération d'un vecteur de cadres englobant les objets d'intérêt dans l'image puis en classant et déterminant où ils se trouvent.

III.6 Phase de prétraitement des données :

Cette phase consiste à rassembler et étiqueter un ensemble de données personnalisé en réalisant les étapes suivantes :

III.6.1 Une collection des données :

« Les ordures entrent et les mêmes qui sortent. De bonnes données entraînent de bons modèles. » **[10]**.

Pour créer un détecteur d'objets personnalisé, nous devons disposer d'un ensemble d'images et d'étiquettes afin que le détecteur puisse être formé efficacement à la détection d'objets.

Pour notre cas, on a utilisé un nombre d'images suffisant pour former les données d'entrées (fig III.10).

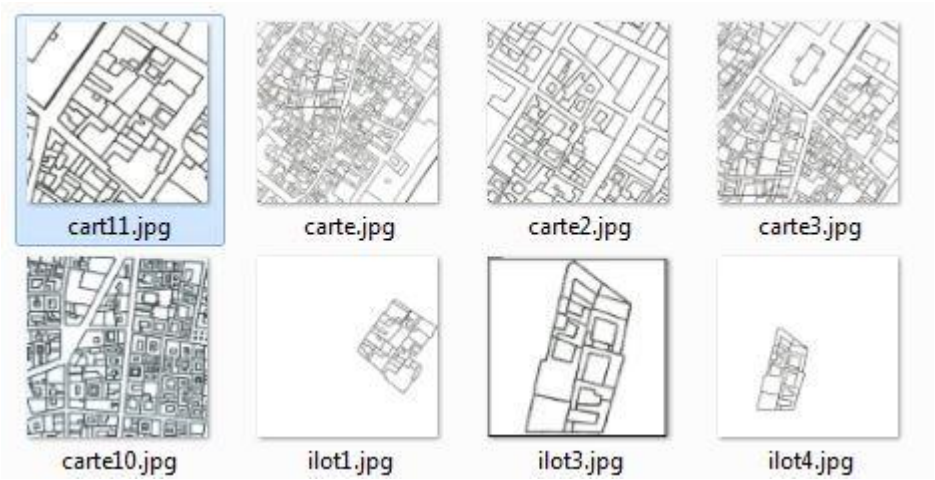


***Figure III.10** : Exemple des données utilisées.*

III.6.2 Un étiquetage et une labellisation par le logiciel Labelimg :

Cette étape consiste à détecter et à étiqueter l'échantillon de données afin que les données étiquetées puissent être utilisées dans le processus d'apprentissage machine.

Le processus d'étiquetage est normalement manuel, mais il peut être assisté par certaines applications pour réduire le temps d'exécution.

L'utilisation de Labelimg consiste à dessiner des cadres autour des objets que nous souhaitons détecter dans les images, et à les associer aux classes ou bien catégories d'objets correspondantes afin de pouvoir les montrer clairement à la machine, pour notre cas les images sont les cartes et les objets sont les ilots.

L'avantage de Labelimg est qu'il nous permet de sauvegarder les annotations directement au format YOLO.

Labelimg peut être facilement installé avec pip depuis le terminal (fig III.11).

```
pip install labelimg
```

(2)

***Figure III.11 :** Commande d'installation de Labelimg.*

Une fois que Labelimg installé avec succès, son lancement nécessite les paramètres [path to image] et [Classes file] (fig III.12).

```
labelImg [path to image] [classes file]
```
(3)

Figure III.12 : *Commande d'activation Labelimg.*

Avec :

- [path to image] est le chemin d'accès à une image dans le répertoire contenant les images que nous allons étiqueter.
- [Classes file] est le fichier où nous listons les classes d'objets que nous allons étiqueter. Nous ne l'avons pas encore créé.

Nous allons Créer un fichier texte nommé « classes.txt », dans lequel nous allons ajouter le mot "ilot",ceci pour designer notre type d'objet.

Une fois les commandes (2) et (3) exécutées, nous avons accès à l'interface de labelimg dans lequel nous sélectionnons le répertoire où toutes les images sont enregistrées. Une fois que toutes les images sont chargées, nous pouvons commencer à les étiqueter avec le format d'annotation YOLO.

III.6.3 Création des fichiers texte correspondant à chaque image :

Lorsque nous enregistrons les étiquettes après chaque image, Labelimg crée un fichier texte pour chaque image avec le même nom que l'image. Tous les détails d'annotation pour cette image particulière sont enregistrés dans ce fichier. Par exemple, "ilot1.jpg" aura un "ilot1.txt" correspondant dans le même répertoire.

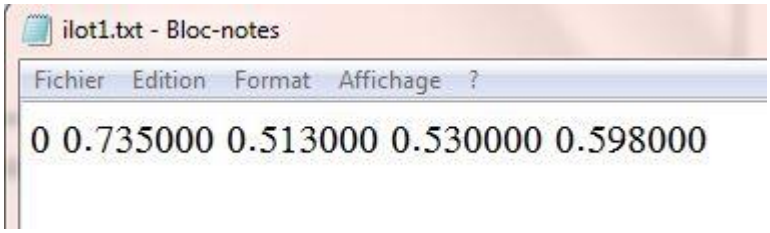


Figure III.13 *: Exemple de fichier texte.*

On peut reconnaitre après l'ouverture de fichier « ilot1.txt », il contient 5 nombres :

[objet_id] [centre_x] [centre_y] [largeur] [hauteur]

Ou,

- [objet_id] : représente le nombre correspondant à la catégorie d'objet que nous avons énuméré dans "classes.txt" plus tôt.
- [centre_x] et [centre_y] : sont les coordonnées du point central de cadre.

Par exemple, (0,25,0,75) est le point situé à 25% de la largeur et 75% de la hauteur. Nous pouvons multiplier ce nombre (0,25,0,75) par la largeur et la hauteur originales de l'image pour obtenir le point réel.

- [Largeur] et [hauteur] : représente la largeur et la hauteur de cadre, normalisé à la gamme 0 à 1 en divisant par la largeur et la hauteur originales de l'image.

Le fichier texte d'annotation généré contiendra chaque ligne comme ci-dessus pour chaque cadre dans l'image et un fichier texte pour chaque image.

A la fin on aura une base de données comme l'indique la figure III.14 :

carte2.jpg	25/06/2010 21:45	Image JPEG	135 Ko
carte2.txt	30/05/2020 12:39	Document texte	1 Ko
carte4.jpg	04/06/2010 23:06	Image JPEG	46 Ko
carte4.txt	30/05/2020 12:39	Document texte	1 Ko
carte5.jpg	25/06/2010 11:19	Image JPEG	553 Ko
carte5.txt	30/05/2020 12:39	Document texte	1 Ko
carte7.jpg	25/06/2010 21:11	Image JPEG	30 Ko
carte7.txt	30/05/2020 12:40	Document texte	1 Ko
carte8..jpg	26/06/2010 08:23	Image JPEG	11 Ko
carte8..txt	30/05/2020 12:40	Document texte	1 Ko
carte9.jpg	28/06/2010 17:50	Image JPEG	84 Ko
carte9.txt	30/05/2020 12:41	Document texte	1 Ko
carte10.jpg	28/06/2010 17:53	Image JPEG	252 Ko
carte10.txt	30/05/2020 12:41	Document texte	1 Ko
carte11.jpg	29/06/2010 00:10	Image JPEG	591 Ko
carte11.txt	30/05/2020 12:44	Document texte	2 Ko
ilot1.jpg	22/05/2010 10:25	Image JPEG	47 Ko
ilot1.txt	30/05/2020 12:45	Document texte	1 Ko
ilot3.jpg	03/06/2010 14:30	Image JPEG	35 Ko
ilot3.txt	30/05/2020 12:45	Document texte	1 Ko
ilot4.jpg	22/05/2010 10:03	Image JPEG	36 Ko
ilot4.txt	30/05/2020 12:45	Document texte	1 Ko
scan0001.jpg	28/06/2010 16:30	Image JPEG	3 535 Ko
scan0001.txt	30/05/2020 12:49	Document texte	3 Ko

***Figure III.14** : Exemple de Dataset utilise.*

III.7 Chargement des données :

Maintenant que nous avons notre base de données avec des objets correctement étiquetés, cette dernière va être transférée dans le Cloud pour être utilisée dans la phase d'entrainement.

Pour le transfert des données, on met dans un fichier ''obj.zip'' le dossier contenant nos fichiers images et textes sur notre machine locale, Ensuite, on télécharge ce dernier sur Google Drive. Puis on le copie dans le Cloud et on le décompresse.

Les commandes utilisées sont donnes dans la table ci-dessous :

Commande	Signification
!ls /myDrive/yolov3 (4)	C'est ici que ''obj. Zip'' est stocké
!cp /myDrive/yolov3/obj.zip ../ (5)	Pour copier le fichier .zip dans le répertoire racine du Cloud
!unzip ../obj.zip -d data/ (6)	Dézipper sur Cloud.

Table III.1 : Commandes de chargement des données.

III.8 Configuration du modèle YOLO pour l'entrainement :

Le modèle YOLO possède deux modes, l'un pour l'entrainement et l'autre pour le test.
Dans un premier temps, on se met dans le mode d'entrainement, et on procède au paramétrage des fichiers de configurations, il s'agit de configurer le modèle YOLO à travers les fichiers, ''obj.data'' et ''obj.names'' , ''train.txt'' et le fichier personnalisé ''nom_fichier.cfg'' (fig III.9).

III.8.1 Fichier personnalisé : ''yolov3_custom2.cfg''

Le fichier ''yolov3_custom2'' est chargé dans le Google Drive par la commande (7) afin de changer certains paramètres qui nous intéressent, ceci pour l'adapter à nos besoins en fonction de notre détecteur d'objets

- cp cfg/yolov3.cfg /myDrive/yolov3/yolov3_custom2.cfg (7)

Le fichier de la figure III.15 présente une partie du fichier de configuration du modèle (le fichier '' yolov3_custom2.cfg'' au complet est donné en annexe 1).

Figure III.15 *: Une partie du fichier de la configuration ‘’yolov3_custom2.cfg’’.*

```
[net]
# Testing
batch=1
subdivisions=1
# Training
# batch=64
# subdivisions=16
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1
```

Le fichier ‘’ yolov3_custom2.cfg ‘’ contient toutes les informations concernant les couches de convolutions et les pooling et le nombre de filtre pour chaque couche, ainsi que les paramètres que nous avons modifié selon nos besoins, ces derniers sont :

- **le paramètre [batch]** : représente La taille du lot est le nombre d'images que nous traitons avant de mettre à jour les poids du réseau.
- **le paramètre [subdivisions]** : représente La taille du lot est divisée par subdivisions lorsque le réseau est chargé, c'est le nombre d'images qui peuvent tenir dans le GPU à la fois.
- **le paramètre [Le nombre d'époques]** : représente un hyper paramètre qui définit le nombre de fois que l'algorithme d'apprentissage fonctionnera sur l'ensemble des données de l’entrainement.

Le nombre d'époques est traditionnellement important, souvent des centaines ou des milliers, ce qui permet à l'algorithme d'apprentissage de fonctionner jusqu'à ce que l'erreur du modèle ait été suffisamment minimisée.

Pour notre cas, les paramètres du fichier '' yolov3_custom2.cfg '' sont:

*batch=64 ; subdivisions=16 ; epoque=2000 ; classes=1.

Après avoir configurer le fichier '' yolov3_custom2.cfg '' selon nos besoins, il va être envoyer vers le répertoire dans le Cloud avec la commande(8).

- !cp /myDrive/yolov3/yolov3_custom.cfg ./cfg (8)

III.8.2 Les fichiers Obj.names et obj.data :

Le contenu de ces fichiers est comme suit

- Pour ''obj.names'' : il doit contenir le même nom que celui du fichier ''classe.txt'' à l'étape de génération du jeu de données, soit pour notre cas le nom ''ilot'' (fig III.16).

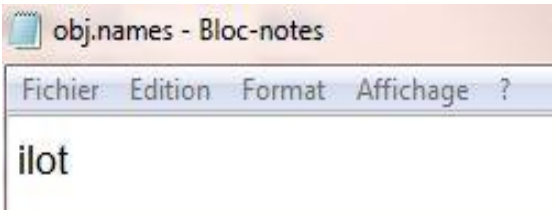


Figure III.16 *: Fichier ''obj.names''.*

- Pour ''obj.data'' : il doit contenir les informations citées dans la figure III.17.

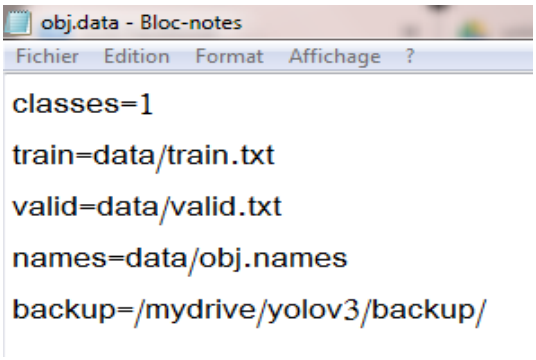


Figure III.17 *: Fichier obj.data.*

Durant la phase d'entrainement il y a une mise à niveau et une sauvegarde des poids du modèle tout au long de l'entraînement. La sauvegarde vase faire au niveau du fichier « backup » dans Google Drive, le chemin d'accès doit être correctement spécifié (fichier ''obj.data'').

III.8.3 Le fichier ‘’ train.txt’’ :

Ce fichier de configuration est nécessaire avant de commencer à former notre détecteur personnalisé car il contient les chemins relatifs de toutes nos images de l’entrainement.

Le script de la figure III.18 se charge de générer ces chemins automatiquement et de les mettre dans le fichier ‘’train.txt’’.

```
import os

image_files = []
os.chdir(os.path.join("data", "obj"))
for filename in os.listdir(os.getcwd()):
    if filename.endswith(".jpg"):
        image_files.append("data/obj/" + filename)
os.chdir("..")
with open("train.txt", "w") as outfile:
    for image in image_files:
        outfile.write(image)
        outfile.write("\n")
    outfile.close()
os.chdir("..")
```

Figure III.18 *: Génération fichier train.txt automatique.*

Une fois généré, il sera envoyé vers le Cloud avec la commande (9) :

- !cp /myDrive/yolov3/generate_train.py ./ (9)

La commande (10) permet de lancer la génération des chemins des images qui vont être exploitées dans la phase d’entrainement

- !python generate_train.py (10)

La figure III.19 montre le contenu du fichier ‘’train.txt’’:

Figure III.19 *: Fichier train.txt.*

III.8.4 Utilisation de poids pré entraînés :

Pour permettre au détecteur d'objets personnalisé d'être plus précis, de s'entraîner durant un temps optimum et de diminuer le nombre d'étapes pour que la sortie converge, nous avons utilisé des poids pré entrainés du réseau YOLOv3, ces derniers sont téléchargés à partir du site ''https://pjreddie.com/media/files/darknet53.conv.74''.

Généralement, pour la tâche de détection, les caractéristiques extraites seront similaires. Au lieu d'initialiser le modèle avec des poids aléatoires, il est initialisé avec des poids pré entraînées, ceci réduit le temps d'entrainement.

III.9 Phase d'entraînement :

La phase d'entrainement revient à lancer le modèle YOLO à base de CNN dans l'environnement Google Colab, dans cette dernière on détecte l'objet d'intérêt et on calcul les poids entrainés (fig III.20)

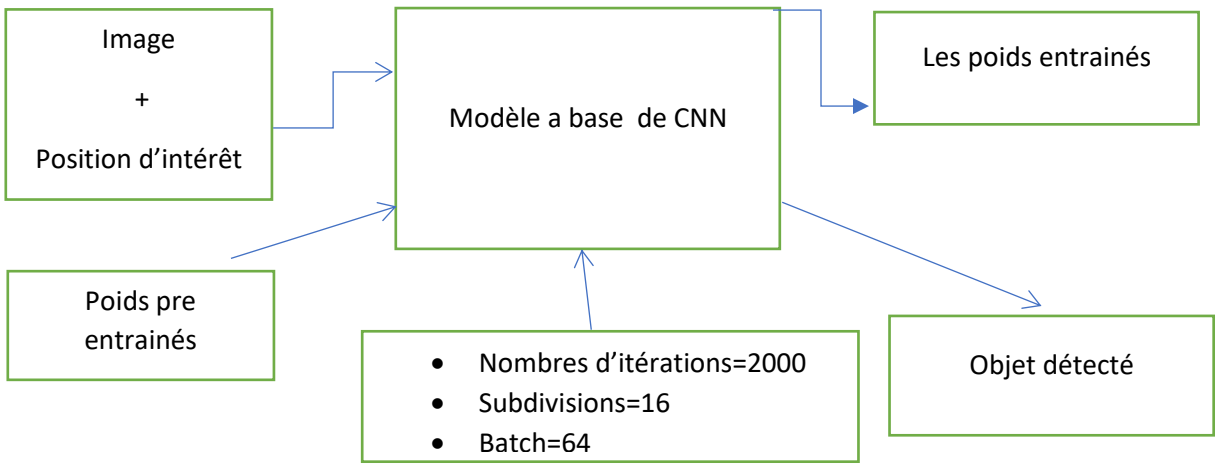


Figure III.20 : *Phase d'entrainement.*

La figure III.21 donne une sortie du modèle YOLO, en indiquant entre autres la valeur de l'erreur moyenne et le temps qui reste pour exécuter 2000 itérations.

```
 2: 2008.389771, 2016.493774 avg loss, 0.000000 rate, 12.589574 seconds, 128 images, 6.897764 hours left
Loaded: 0.000034 seconds
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.365721, GIOU: 0.266872), Class: 0.617444, (
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.269302, GIOU: 0.013373), Class: 0.295844, (
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000,
 total_bbox = 3078, rewritten_bbox = 0.097466 %
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.334720, GIOU: 0.264322), Class: 0.590482, (
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.161301, GIOU: -0.149597), Class: 0.353353,
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000,
 total_bbox = 3107, rewritten_bbox = 0.096556 %
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 82 Avg (IOU: 0.388966, GIOU: 0.313626), Class: 0.616914, (
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 94 Avg (IOU: 0.266361, GIOU: 0.266361), Class: 0.188736, (
v3 (mse loss, Normalizer: (iou: 0.75, cls: 1.00) Region 106 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000,
```

***Figure III.21**: Début de l'entrainement.*

Il est a remarquer que ce dernier est d'environ 7 heures, c'est le temps mis pour entrainer le modèle. Or ce temps ne permet pas le maintien du GPU étant donné le nombre important d'utilisateurs du Cloud.

Cependant pour ne pas être exclu de cet environnement, on exécute une procédure, cette dernière est donnée en annexe 2.

A la fin de l'entrainement on aura les résultats donnés dans la figure III.22.

```
2000: 0.834873, 0.951242 avg loss, 0.000010 rate, 12.698812 seconds,
Saving weights to /mydrive/yolov3/backup//yolov3_custom_2000.weights
Saving weights to /mydrive/yolov3/backup//yolov3_custom_last.weights
Saving weights to /mydrive/yolov3/backup//yolov3_custom_final.weights
```

***Figure III.22 :** La fin de l'entrainement.*

Après l'itération 2000, on a obtenu un modèle précis étant donné que la perte obtenue est inférieure à 1 et tous nos poids ont été sauvegardés dans le backup dans le chemin choisi.

Le graphe de la performance de notre modèle tout au long du processus de l'entrainement est donné dans la figure III.23.

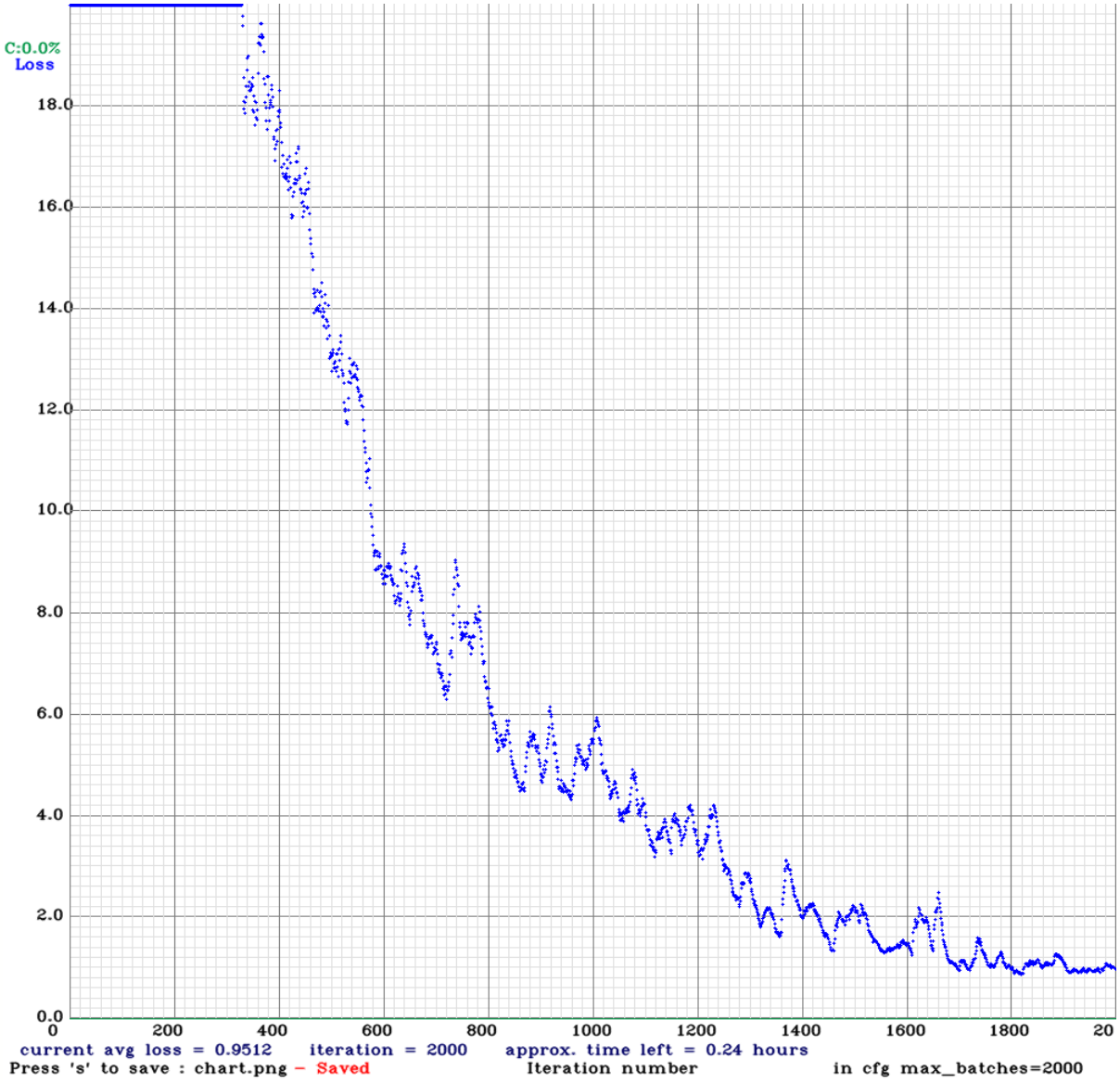


Figure III.23 : *Graphique de la perte en fonction du nombre d'itération.*

D'après la figure 3.23 On peut extraire des points essentiels qui sont :

- Entre 0 et 340 : une grande diminution de perte car on a vu dans le démarrage de l'entrainement, la valeur mesurer 2016 après la 340ieme itérations on a une chute importante jusqu'à 20.
- Entre 340 et 800 : on a une chute considérable jusqu'à 6.
- Entre 800 et 1600 : une chute est plus lente jusqu'à 2.
- Entre 1600 et 2000 : on a une progression constante malgré les pics jusqu'à 0.95.

D’après la courbe de la figure 3.23, on peut voir que le nombre d’itérations doit être autour de 2000.

III.10 Phase de test :

Dans cette phase, il est impératif de modifier le mode vers test par le changement de [batch] et [subdivisons]= 1 du fichier ‘’ yolov3_custom2.cfg’’ en exécutant le script de la figure III.24.

```
%cd cfg
!sed -i 's/batch=64/batch=1/' yolov3_custom.cfg
!sed -i 's/subdivisions=16/subdivisions=1/' yolov3_custom.cfg
%cd ..
```

***Figure III.24** : Code de changement vers test mode.*

Nous disposons maintenant d'un détecteur d'objets personnalisé pour faire nos propres détections. Il est temps de le tester avec la commande.

```
!./darknet detector test data/obj.data cfg/yolov3_custom.cfg
/mydrive/yolov3/backup/yolov3_custom_last.weights /mydrive/images/scan0006.jpg
imShow('predictions.jpg')
download('predictions.jpg')
```

***Figure III.25 :** Code de lancement de test.*

Pour lancer un teste il faut déterminer 4 choses :

- Le détecteur au mode test.
- Le nom de fichier de configuration personnalisé.
- La voie où se trouve les poids de notre entraînement.
- La voie où se trouve l’image choisi pour le test.

III.11 résultats obtenus :

Le modèle YOLOv$_3$ a été entrainé et personnalisé pour qui il puisse reconnaitre les ilots sous différentes formes et tailles, dans ce que suit nous allons donner les résultats de l’application du modèle suivant un critère d’étude, et basé sur la performance de détection du modèle.

Nous avons travaillé sur la carte de centre de ville de Blida, et nous avons fait l’étude sur deux types de cartes, urbaines et satellitaires.

- Pour les cartes urbaines, nous avons procédée comme suit, on a considéré dans un premier temps ,une image possédant un seul ilot. dans un second temps , une image qui comporte un certain nombres d'ilots de formes simples. Puis, nous avons pris une image contenant des ilots de formes quelconques (complexes).
- Pour la carte satellitaire nous avons entrainé notre modèle pour extraire les ilots du centre de ville de Blida.

III.11.1 Application sur un ilot seul :

On applique notre modèle sur deux images ci-dessous (figure III.26 et figure III.27) qui contient un ilot seul.

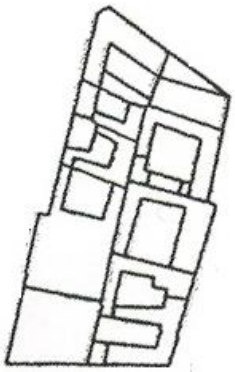


Figure III.26.a *: ilot avant la détection.*

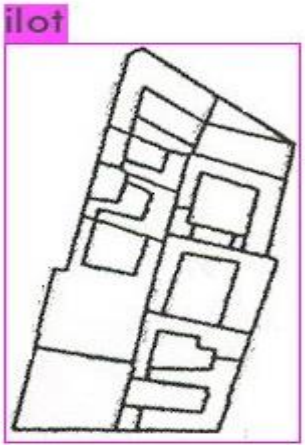


Figure III.26.b *: ilot après la détection.*

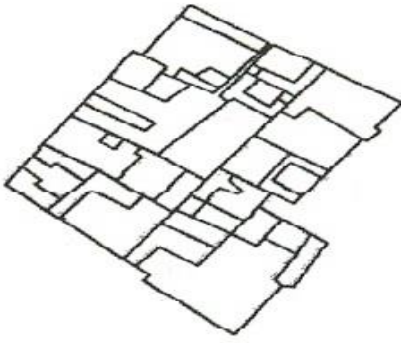


Figure III.27.a *: ilot avant la détection.*

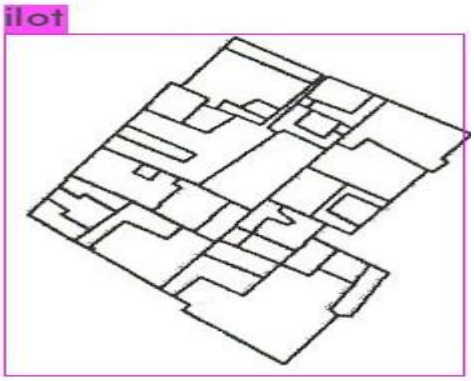


Figure III.27.b *: ilot après la détection.*

❖ COMMENTAIRE :

Pour la détection d'un seul ilot, on remarque que l'extraction s'est faite parfaitement.

III.11.2 Application sur des ilots de forme simple :

On applique notre modèle sur la carte ci-dessous qui contient des ilots avec formes simples.

a) ***(b)***

Figure III.28 : *a) carte avec ilots avant la détection. b) carte avec ilots simples après la détection. c) une partie des les ilots détectés séparément (le reste se trouvent en Annexe).*

- COMMENTAIRE :

On remarque que le modèle fournit les ilots de la carte (fig III.28.c).

III.11.3 Application sur des ilots de formes quelconques complexes) :

Dans cette partie de l'expérience, nous avons voulu compliquer la détection en considérant deux cartes urbaines qui présentaient une certaine complexité dans les formes et les tailles des ilots, la première est celle du centre-ville de Blida prise au complet (fig III.29) et la seconde est celle d'une ville Italienne (choisie aléatoirement) (fig III.30).

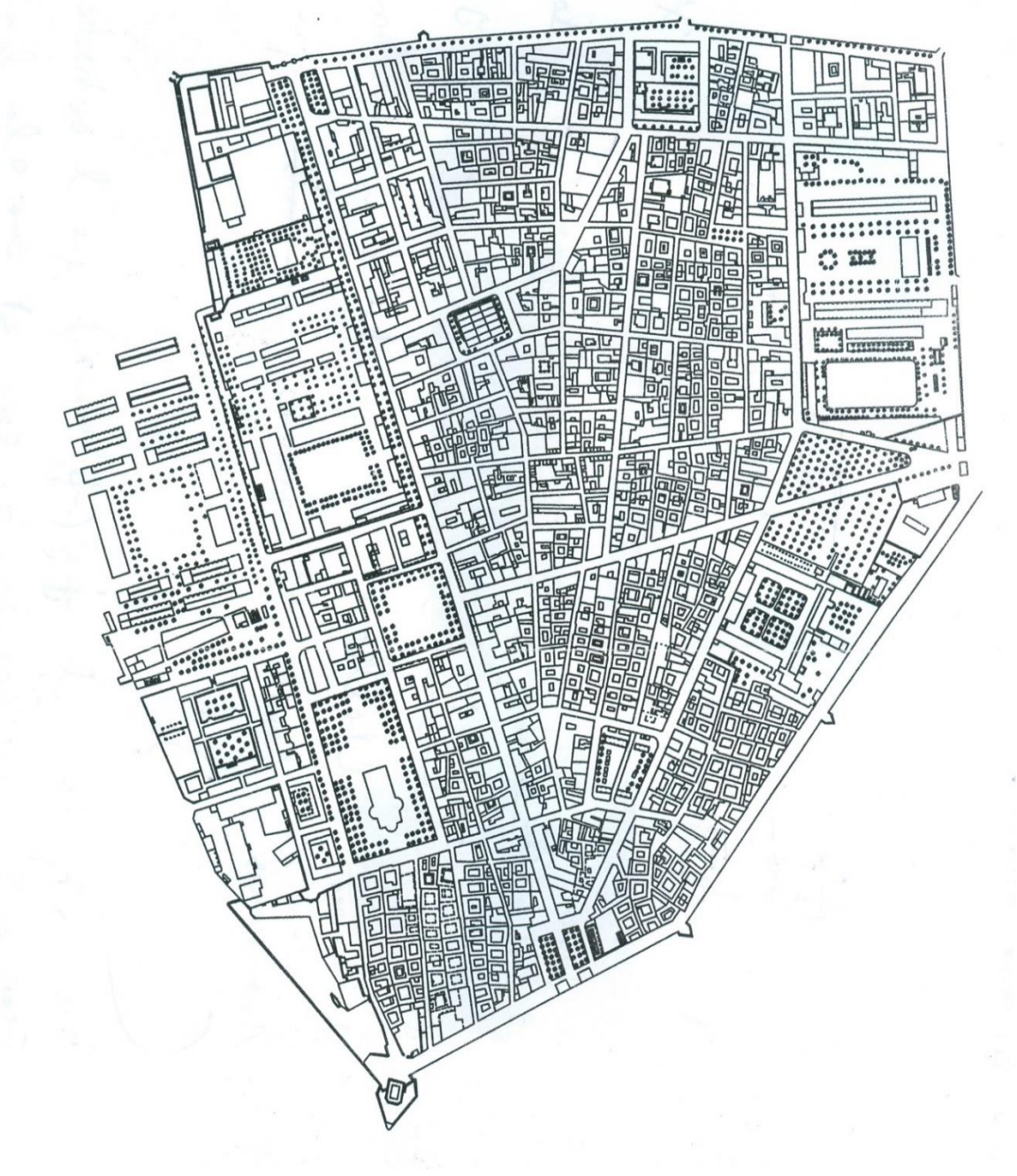


Figure III.29.a) : *Carte de centre de Blida avant la détection.*

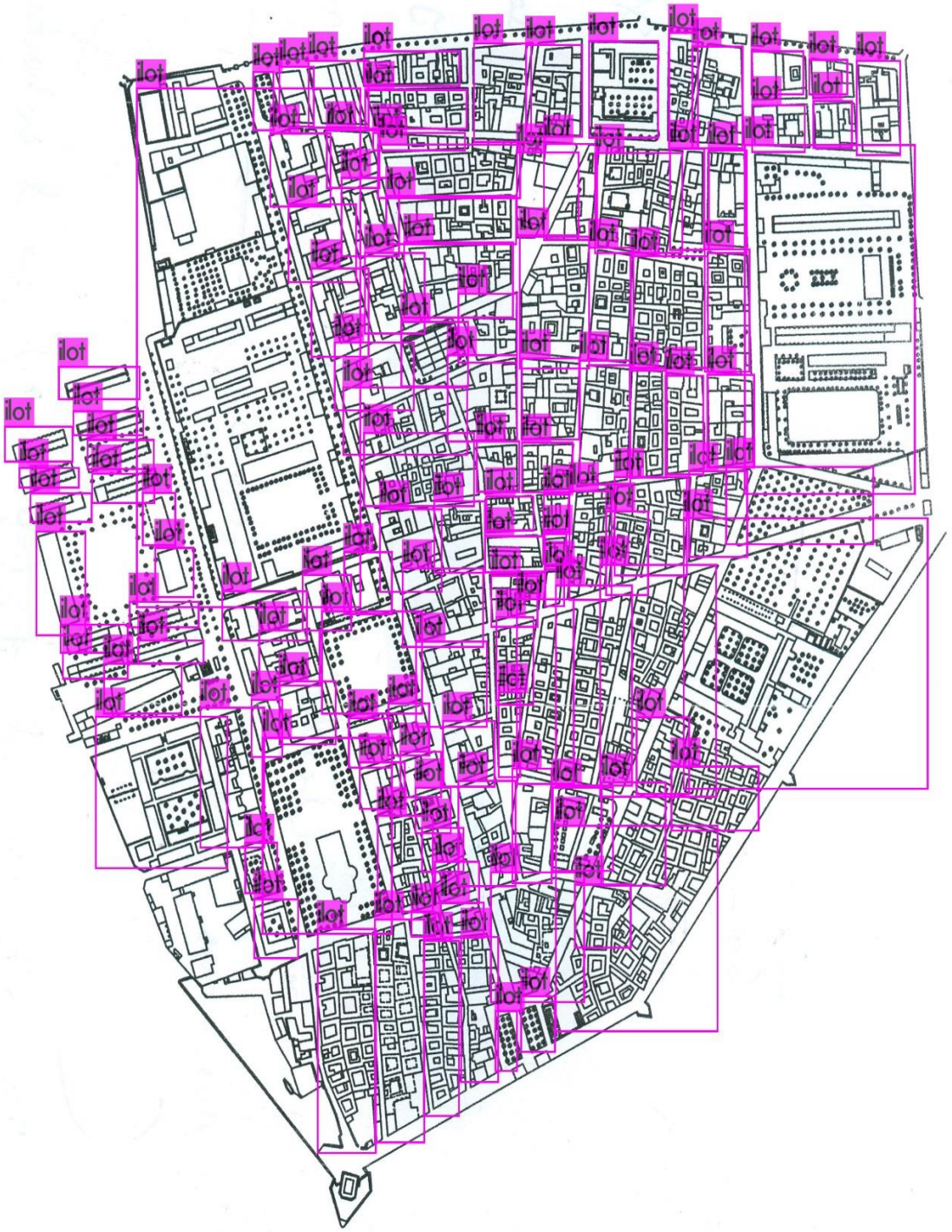


Figure III.29.b : *Carte de centre de Blida après la détection.*

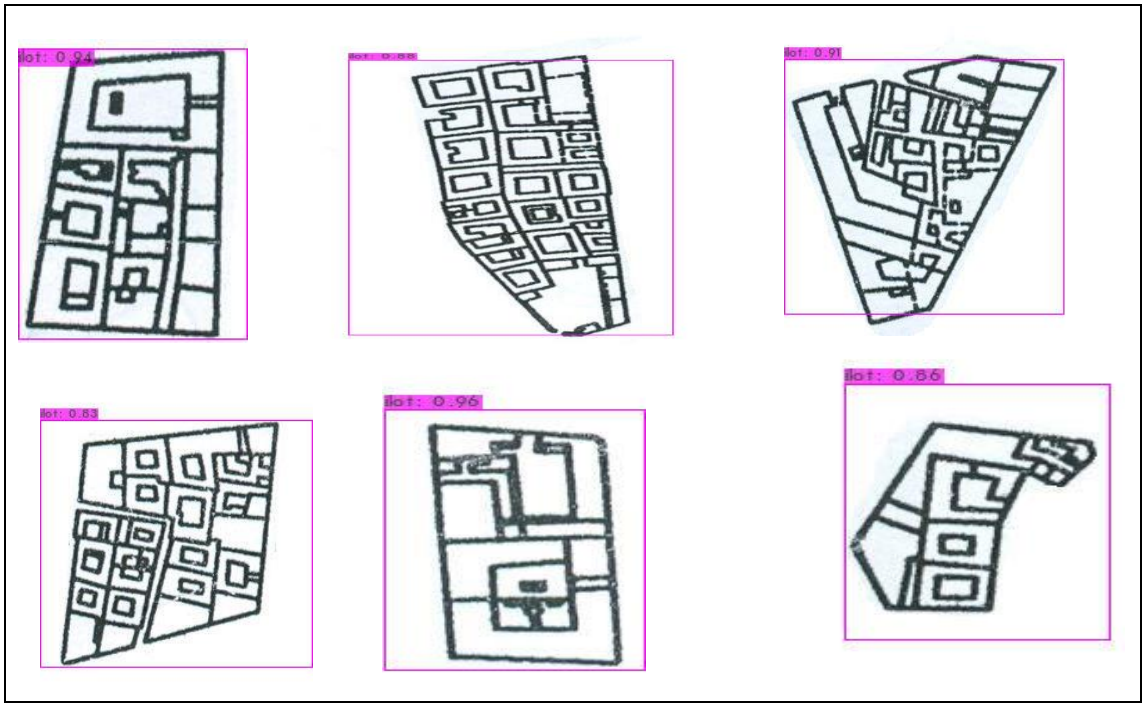


Figure III.29.c) *Une partie des les ilots détectés séparément (le reste se trouvent en Annexe).*

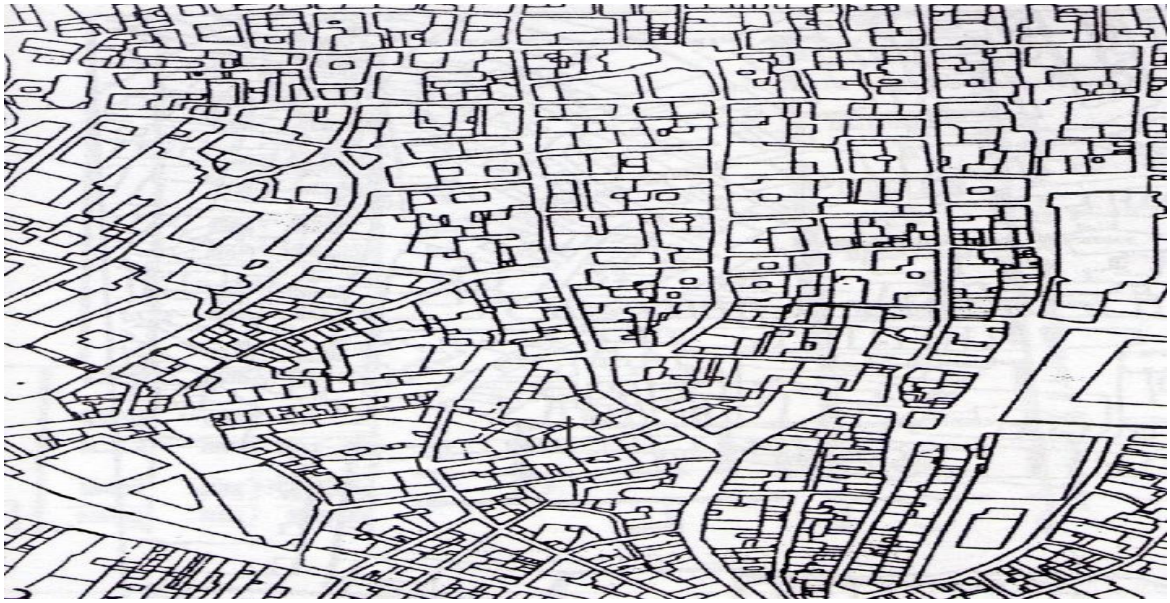
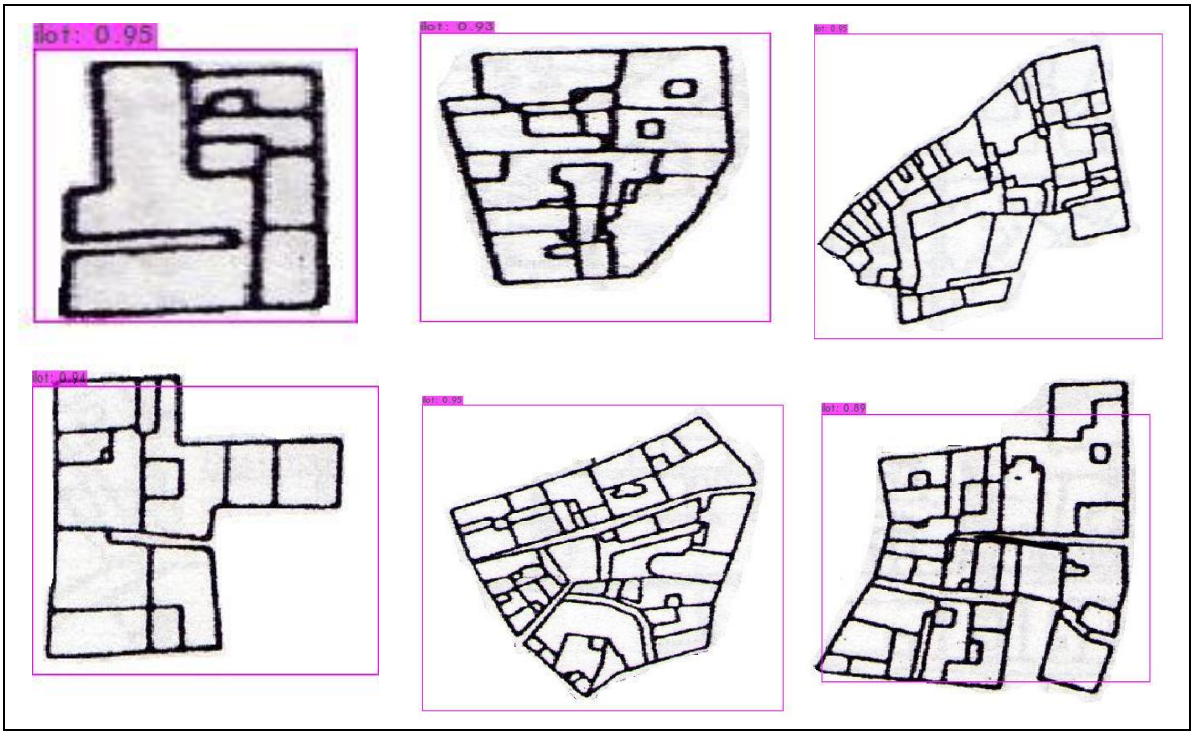


Figure III.30.a : *Carte avec ilots complexes avant la détection.*

Figure III.30.b : *carte avec ilots complexes après la détection.*

Figure III.30.c) *une partie des les ilots détectés séparément (le reste se trouvent en Annexe).*



❖ COMMENTAIRE :

L'application du modèle YOLO sur ces deux cartes c'est avérée très efficace car tous les ilots ont bien été détectés même si on ne peut pas les apercevoir à l'œil nu, nous avons pour cela archivé tous ces derniers un par un en Annexe 3.1.

III.11.4 Application sur une carte satellitaire de centre de blida :

On applique notre modèle sur la carte ci-dessous qui contient des ilots du centre de blida :

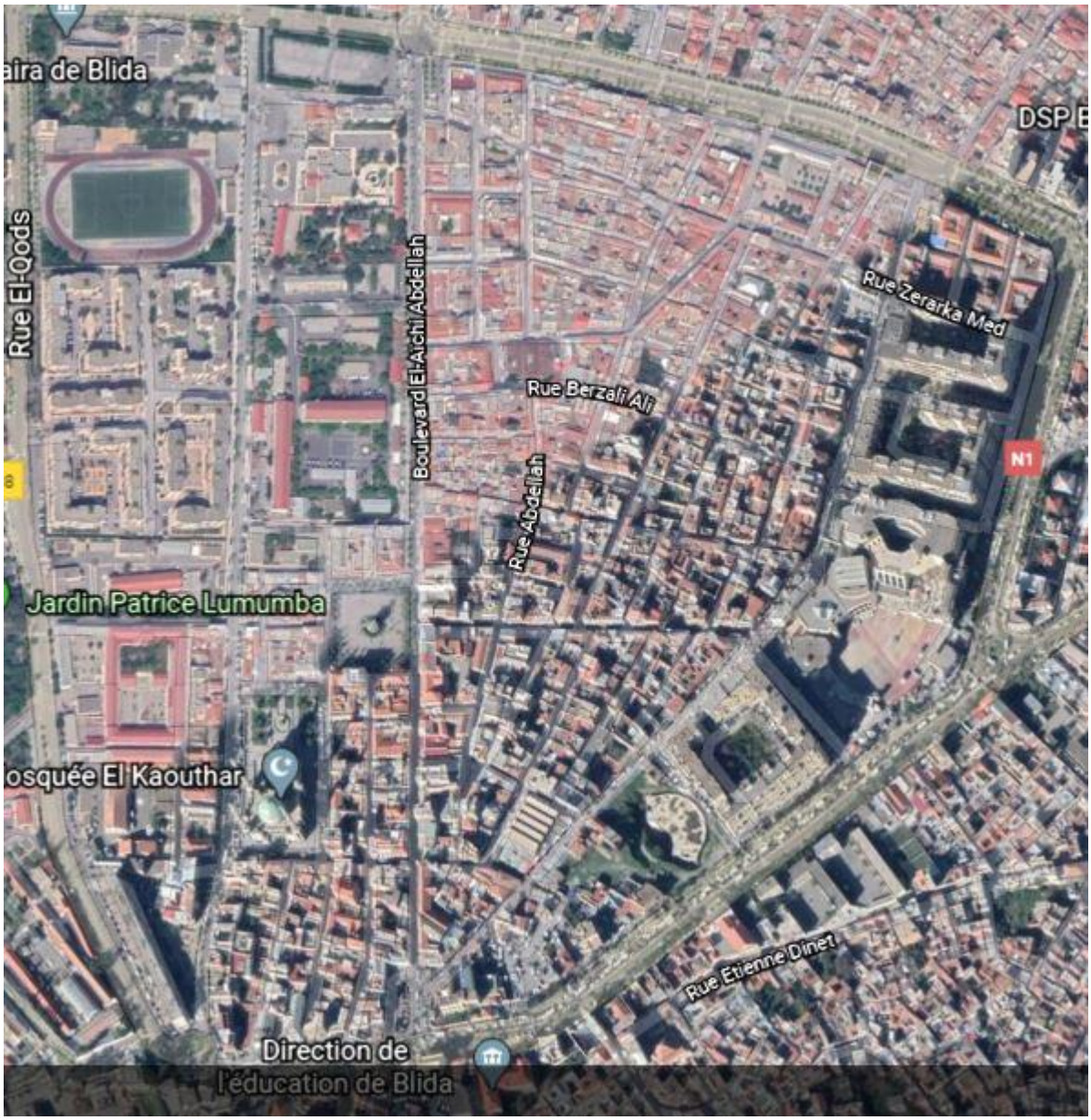


Figure III.31.a : *Carte de centre de blida avant la détection.*

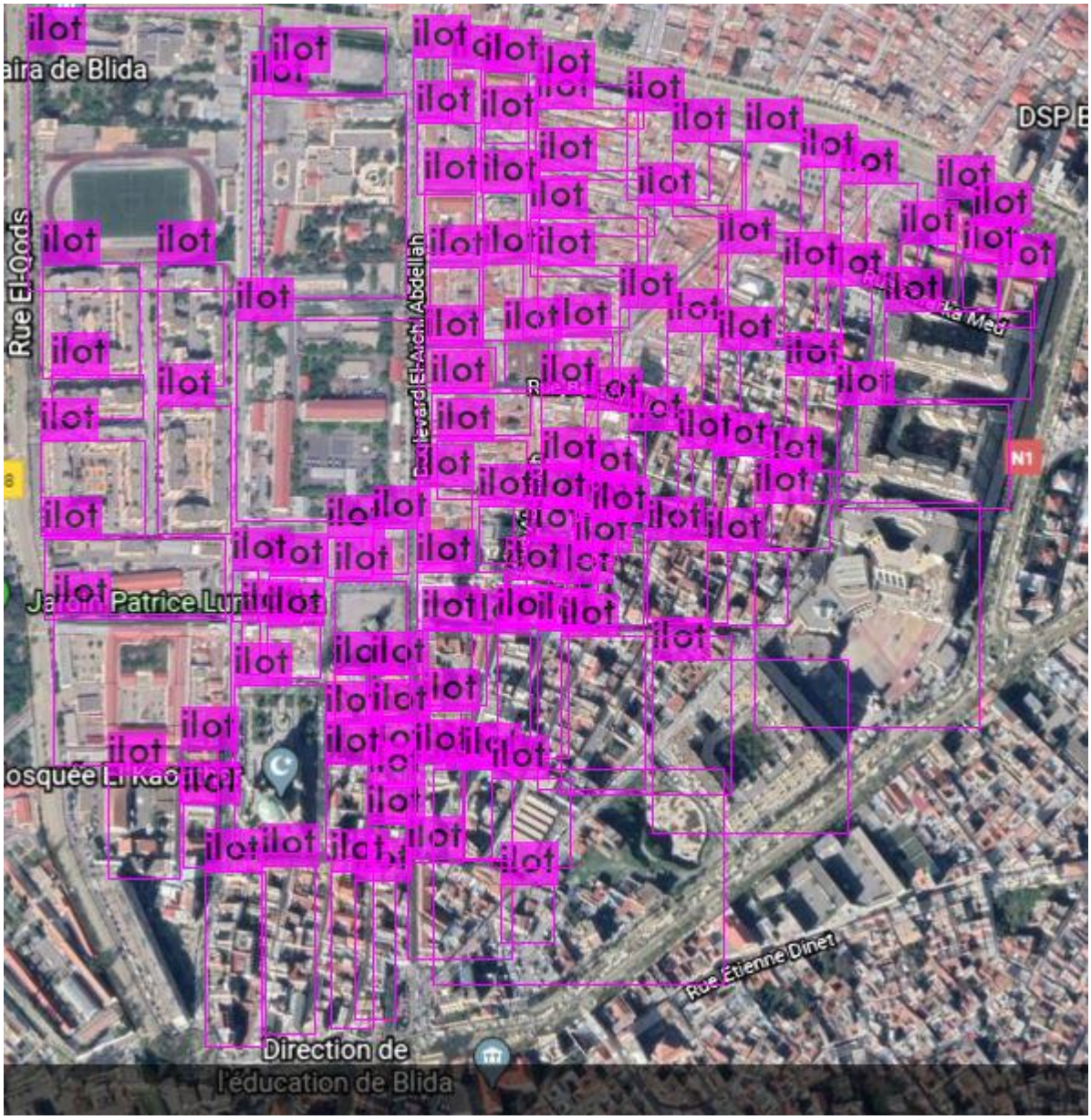


Figure III.31.b : *Carte de centre de blida après la détection.*

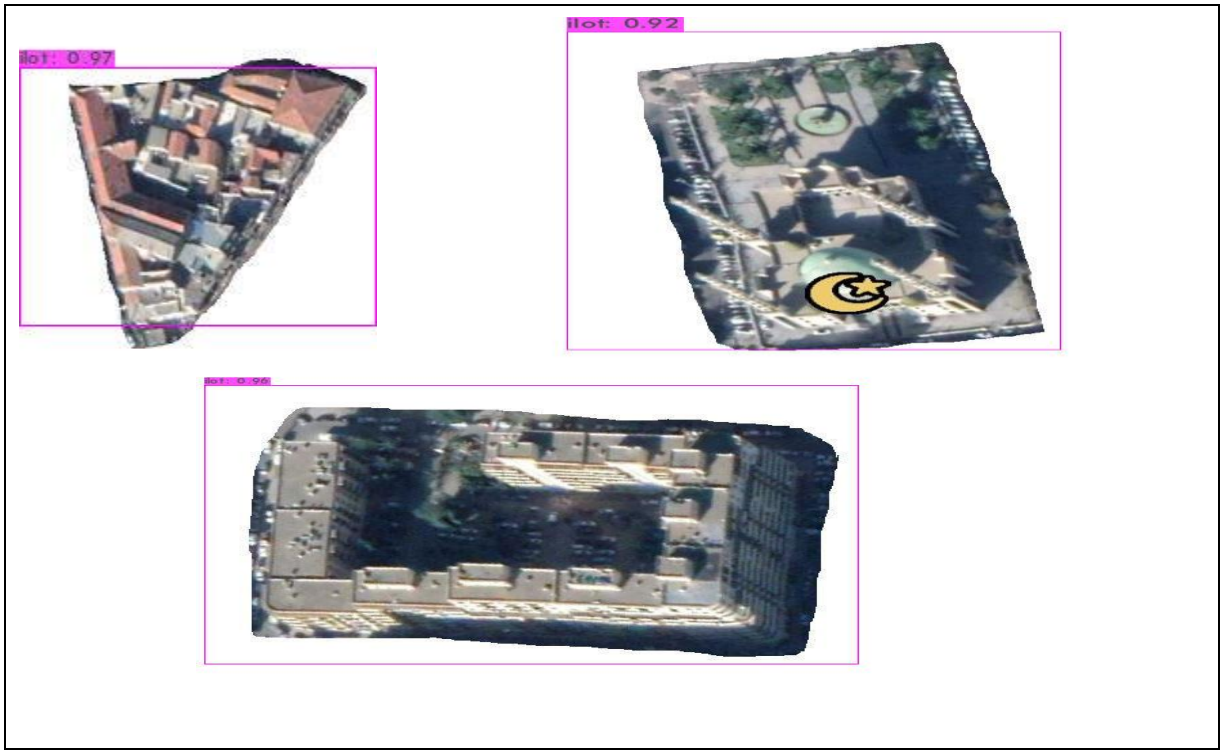


***Figure III.31.c** : une partie des les ilots détectés séparément (le reste se trouvent en Annexe).*

- Commentaire :

Pour l'extraction des objets d'intérêt (ilots) à partir des cartes satellitaires, nous avons dû ajouter des données satellitaires étiquetées dans le dataset, puis entrainer à nouveau le modèle avec deux types de données, nous remarquons que l'extraction des ilots est faite avec succès également.

III .12 Conclusion :

Dans ce chapitre, nous avons suivi la schématisation générale employée par la majorité des projets de vision utilisant le DL, nous avons par le biais de cette dernière configurer notre modèle YOLO et procurer à ce dernier des donnés en entrée afin qu'il puisse nous faire des détections après entrainement de celui-ci.

Les résultats obtenus sont très satisfaisants pour des itérations à partir de 2000.

L'avantage que nous procure YOLO est qu'il puisse faire des extractions d'objets à partir des cartes urbaines et satellitaires.

Conclusion générale

Notre projet, a été proposé dans le but de participer à l'élaboration d'une partie d'un système d'aide à la reconnaissance des parcours structurants dans les centres urbains anciens selon le modèle interprétatif de G.Caniggia issu de la méthode Muratorienne. En utilisant le Deep Learning et en développant un modèle d'extraction d'objets d'intérêt dans une scène graphique, dans notre cas nous devions préparer une reconnaissance des parcours structurants en se basant sur une méthode d'extraction d'ilots à partir d'une carte urbaine qui repose sur l'application du détecteur YOLO.

Durant ce travail, on a essayé de suivre le cahier des charges qui nous a été confié, à savoir. La tâche demandée est d'extraire les ilots à partir de carte urbaine et satellitaire, particulièrement les ilots qui se situent au centre de Blida, par la méthode (DL).

Pour atteindre notre but, nous avons créé une base de données, que nous avons étiqueté, et l utilisée dans la phase d'entrainement, une fois le modèle YOLO ayant été configuré, a été bien entrainé pour compléter sa tâche de détection.

Les résultats obtenus, nous permettent d'afficher une grande satisfaction car le modèle employé a pu reconnaitre les ilots formes et en grandeurs que ce soit pour les cartes urbaines ou satellitaires, ce qui nous permet de dire que les objectifs visés par le cahier des charges ont été correctement exécutés.

Il est à noter que le réseau de neurones obtenu peut être utilisé sur un CPU, implémenté sur Raspberry PI par exemple, ou implémenté sur circuit FPGA pour gagner en puissance.

Nous avons rencontré quelques problèmes dans le début de notre étude, car les bibliothèques voulus, affichaient des erreurs, après l'installation (Il n'y a pas une compatibilité des versions entre elles), mais on a pu contourner cet obstacle par l'utilisation Google Colab qui utilise des bibliothèques préinstallées. En utilisant le GPU qu'il fournit, la durée d'entrainement a été écourtée. (Quelques heures au lieu des jours par CPU).

Nous espérons que ce projet va être exploité pour compléter la réalisation du système d'aide à la reconnaissance des parcours structurants dans les centres urbains anciens, en réalisant l'extraction des réseaux de voiries et en élaborant des algorithmes d'optimisation pour déterminer les parcours structurants afin qu'ils puissent être utilisé par les spécialistes en urbanisme pour la lecture automatique des cartes.

Enfin ce projet nous a permis de développer nos connaissances en touchants à plusieurs domaines tels que la typomorphologie des villes, le Deep Learning, surtout le langage de programmation python, en espérant qu'il va nous permettre d'ouvrir des opportunités dans le future.

Bibliographie

[1] Jean cartex, ` lecteur d'une ville : Versailles '.

[2] Traduction de l'ouvrage de canggia, ` lecteur de bâti de base '.

[3] Andi, 'wilaya de blida',2016.

[4] Ouadah Sofiane, 'requalification du cartier ramoul a blida et conception d'une gare multimodale', mémoire de master, université de Blida département d'architecture, 2014.

[5] Benchabane leila, Zouggari Zakaria et Benkara omar, 'recomposition d'un ilot du 19eme siècle dans l'hyper-centre d'alger, conception d'un immeuble d'habitat et d'une résidence pour chercheurs', université de Blida département d'architecture, 2018.

[6] Franck Ramus, `L'intelligence humaine, dans tous ses états ',2011.

[7] T. Beysolow II, `Introduction to Deep Learning Using R',2017.

[8] Guillaume Saint-Cirgue, `Apprendre le machine learning en une semaine',2019.

[9] http://conf.laas.fr/ignotus/archives/Doncescu_reseaux_neuronnes.pdf.

[10] Claude Touzet,' les reseaux de neurones artificiels, introduction au connexionnisme, Editions la Machotte,2016.

[11]https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[12] Joseph Redmon, Ali Farhadi,' YOLOv3: An Incremental Improvement'2015.

[13] consulter sur le https://pjreddie.com/yolo/.

[14] https://missinglink.ai/guides/computer-vision/yolo-deep-learning-dont-think-twice/.

[15] https://pluralsight.com//

[16] Golubev, Alexey , Садовникова, Наталья , Parygin, Danila , Glinyanova, Irina , Finogeev, Alexey , Shcherbakov,, Woody Plants Area Estimation Using Ordinary Satellite Images and

Deep Learning: Third International Conference, DTGS 2018, St. Petersburg, Russia, May 30 – June 2, 2018, Revised Selected Papers, Part I. 10.1007/978-3-030-02843-5_24

[17] K. Simonyan and A. Zisserman, ’Very deep convolutional networks for large-scale image recognition‘ CoRR, vol. abs/1409.1556, 2014.

[18] BERRI Nacira, ’ Utilisation des techniques de Deep Learning pour l'extraction des concepts à partir des documents textuels ‘mémoire de master, université de Biskra département d’informatique, 2019.

[19] Ouzeri Abderrezak ,Kettab Imad Eddine’ Deep Learning pour la détection de la rétinopathie ‘mémoire de master, université de Blida département d’électronique, 2018.

[20] MALKI Yacine, TIACHACHAT Nassim,’Deep Learning pour la détection des plaques d’immatriculation‘, mémoire de master, université de Blida département d’électronique, 2019.

[21] GHEZAL AHMED, HADJAM ADEL, ’conception et réalisation d’une partie dun système d’aide à la reconnaissance des tissue anciens : automatisation de calcul du nombre de parcelles et des longueurs des cotes dans un ilots, mémoire d’ingéniorat, université de Blida département d’électronique, 2010.

[22] Chourar Mehdi, ’ Développement d’un système d’aide à l’étude morphologique des villes à partir de plans cadastraux ’mémoire de master, université de Blida département d’électronique, 2013.

Annexes

ANNEXE 1 :

(FICHIER DE CONFIGURATION (yolocustum.cfg)

(1)

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
 batch=64
 subdivisions=16
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 2000
policy=steps
steps=1600,1800
scales=.1,.1

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

# Downsample

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=32
size=1
stride=1
pad=1
activation=leaky
```

(2)

```
[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

# Downsample

[convolutional]
batch_normalize=1
filters=128
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=leaky
```

(3)

```
[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

# Downsample

[convolutional]
batch_normalize=1
filters=256
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky
```

[convolutional] batch_normalize=1 filters=256 size=3 stride=1 pad=1 activation=leaky [shortcut] from=-3 activation=linear [convolutional] batch_normalize=1 filters=128 size=1 stride=1 pad=1 activation=leaky [convolutional] batch_normalize=1 filters=256 size=3 stride=1 pad=1 activation=leaky [shortcut] from=-3 activation=linear [convolutional] batch_normalize=1 filters=128 size=1 stride=1 pad=1 activation=leaky [convolutional] batch_normalize=1 filters=256 size=3 stride=1 pad=1 activation=leaky [shortcut] from=-3 activation=linear	[convolutional] batch_normalize=1 filters=256 size=3 stride=1 pad=1 activation=leaky [shortcut] from=-3 activation=linear [convolutional] batch_normalize=1 filters=128 size=1 stride=1 pad=1 activation=leaky [convolutional] batch_normalize=1 filters=256 size=3 stride=1 pad=1 activation=leaky [shortcut] from=-3 activation=linear [convolutional] batch_normalize=1 filters=128 size=1 stride=1 pad=1 activation=leaky [convolutional] batch_normalize=1 filters=256 size=3 stride=1 pad=1 activation=leaky [shortcut] from=-3 activation=linear	[convolutional] batch_normalize=1 filters=256 size=3 stride=1 pad=1 activation=leaky [shortcut] from=-3 activation=linear [convolutional] batch_normalize=1 filters=128 size=1 stride=1 pad=1 activation=leaky [convolutional] batch_normalize=1 filters=256 size=3 stride=1 pad=1 activation=leaky [shortcut] from=-3 activation=linear [convolutional] batch_normalize=1 filters=128 size=1 stride=1 pad=1 activation=leaky [convolutional] batch_normalize=1 filters=256 size=3 stride=1 pad=1 activation=leaky [shortcut] from=-3 activation=linear
(4)	(5)	(6)

```
[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

# Downsample

[convolutional]
batch_normalize=1
filters=512
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky
```

(7)

```
[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
```

(8)

```
[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear
```

(9)

ANNEXE 2 :

Colab Cloud Service nous renvoi dans ses machines virtuelles si nous restons trop longtemps sans rien faire (30 à 90 minutes).

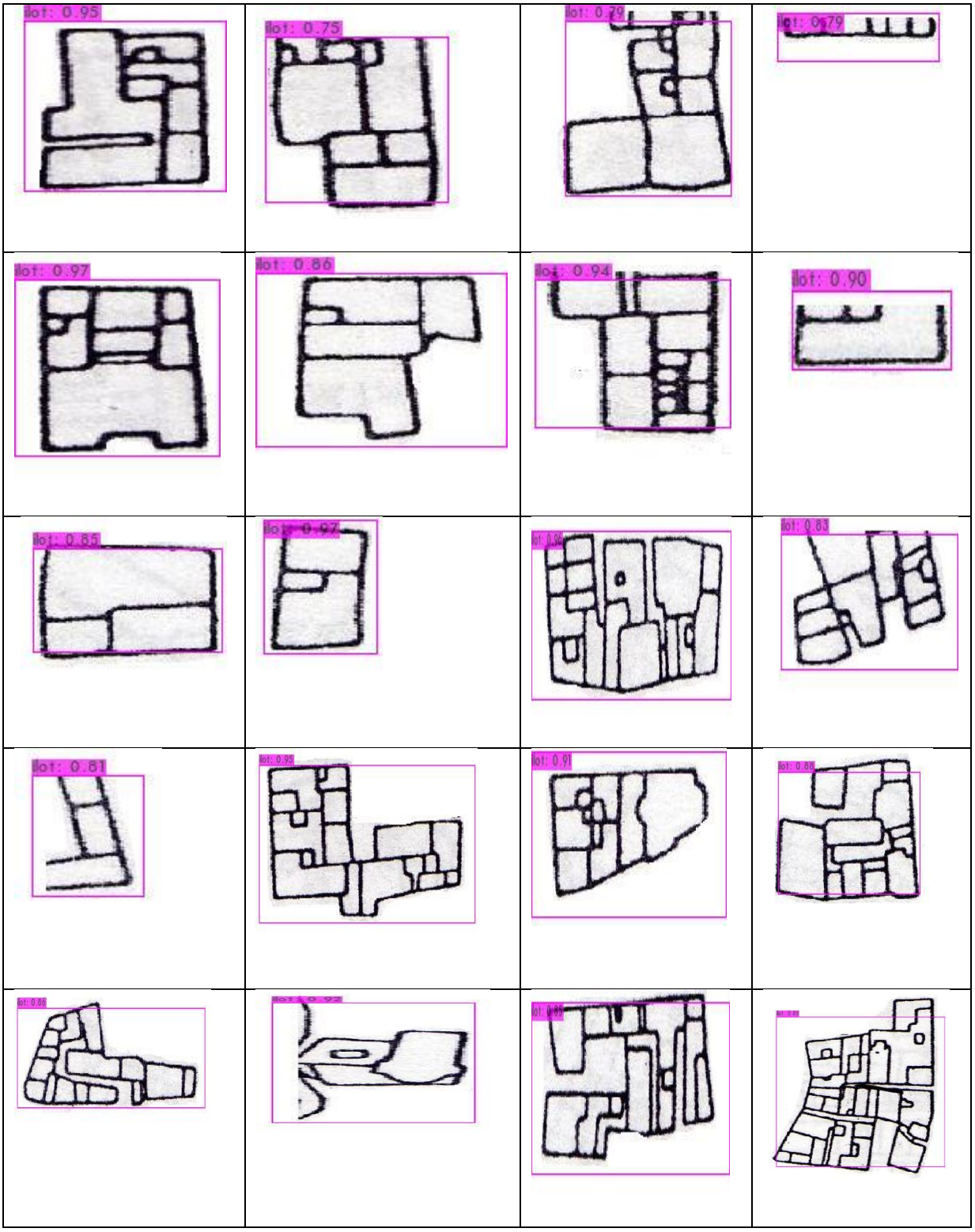
Pour éviter cette attente (CTRL + MAJ + i), nous devons en même temps ouvrir la vue de l'inspecteur sur notre navigateur.

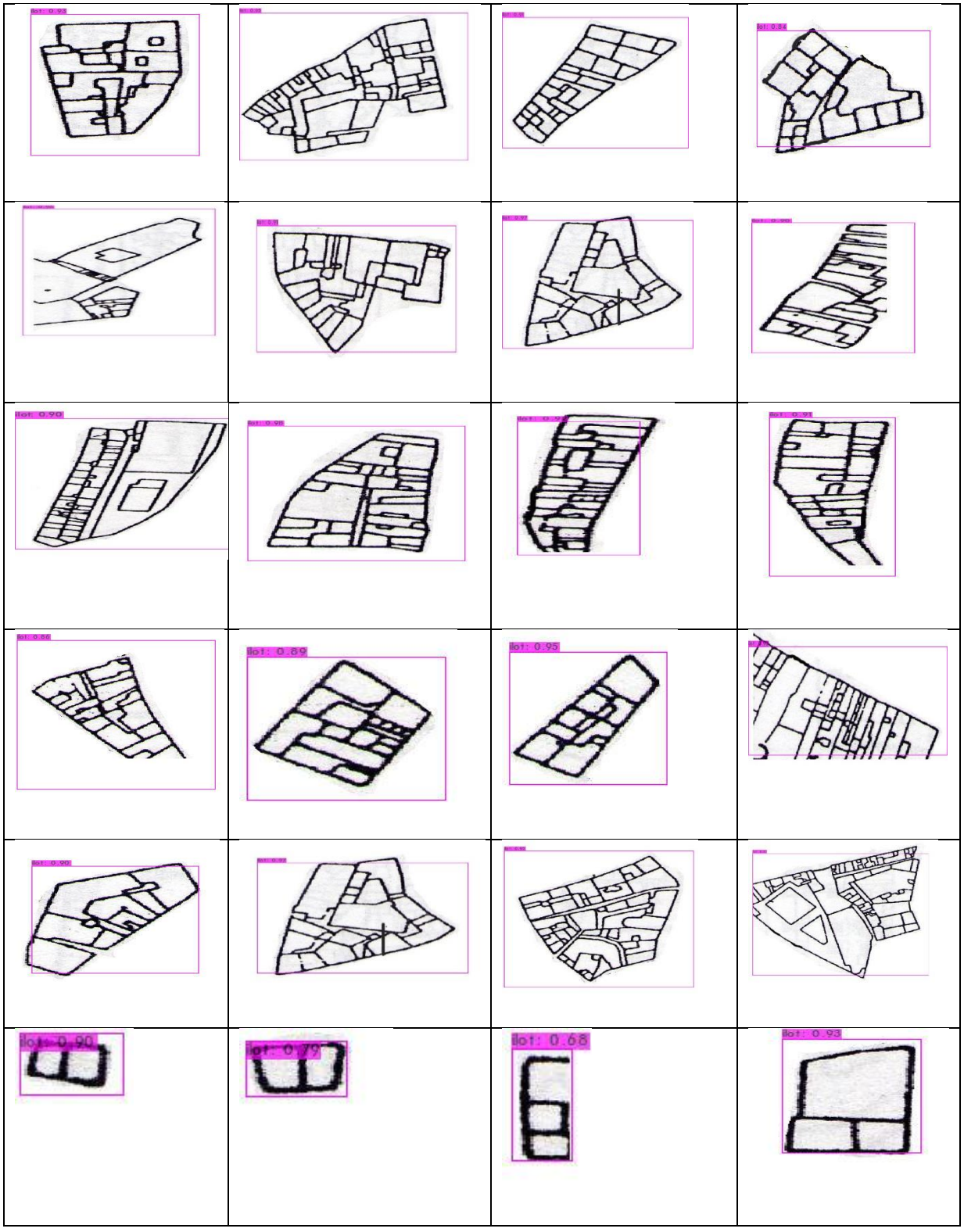
Collons le code suivant dans la fenêtre de notre console et appuyez sur Entrée.

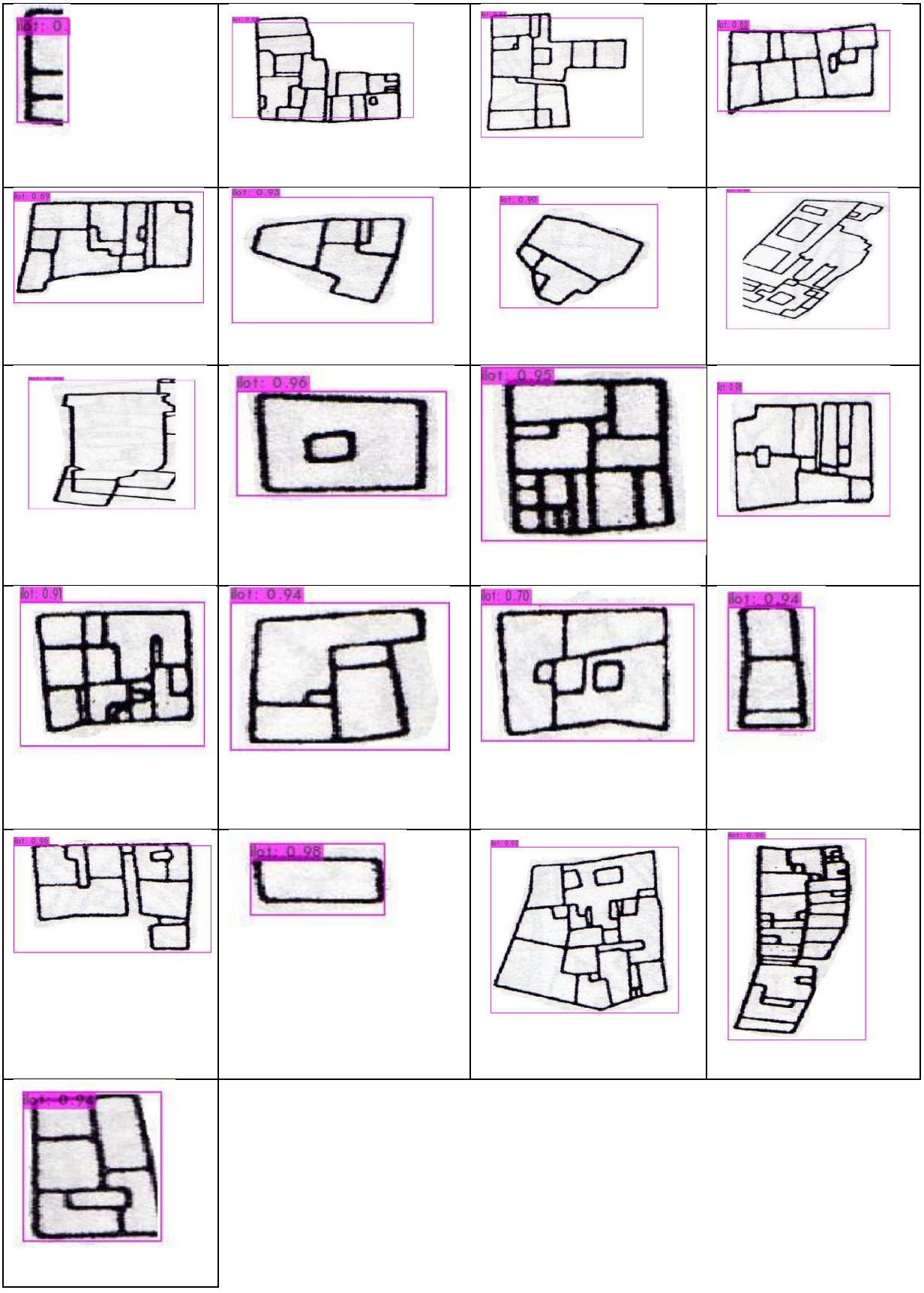
```
function ClickConnect(){
console.log("Working");
document.querySelector("colab-toolbar-button#connect").click()
}
setInterval(ClickConnect,60000)
```

ANNEXE 3:

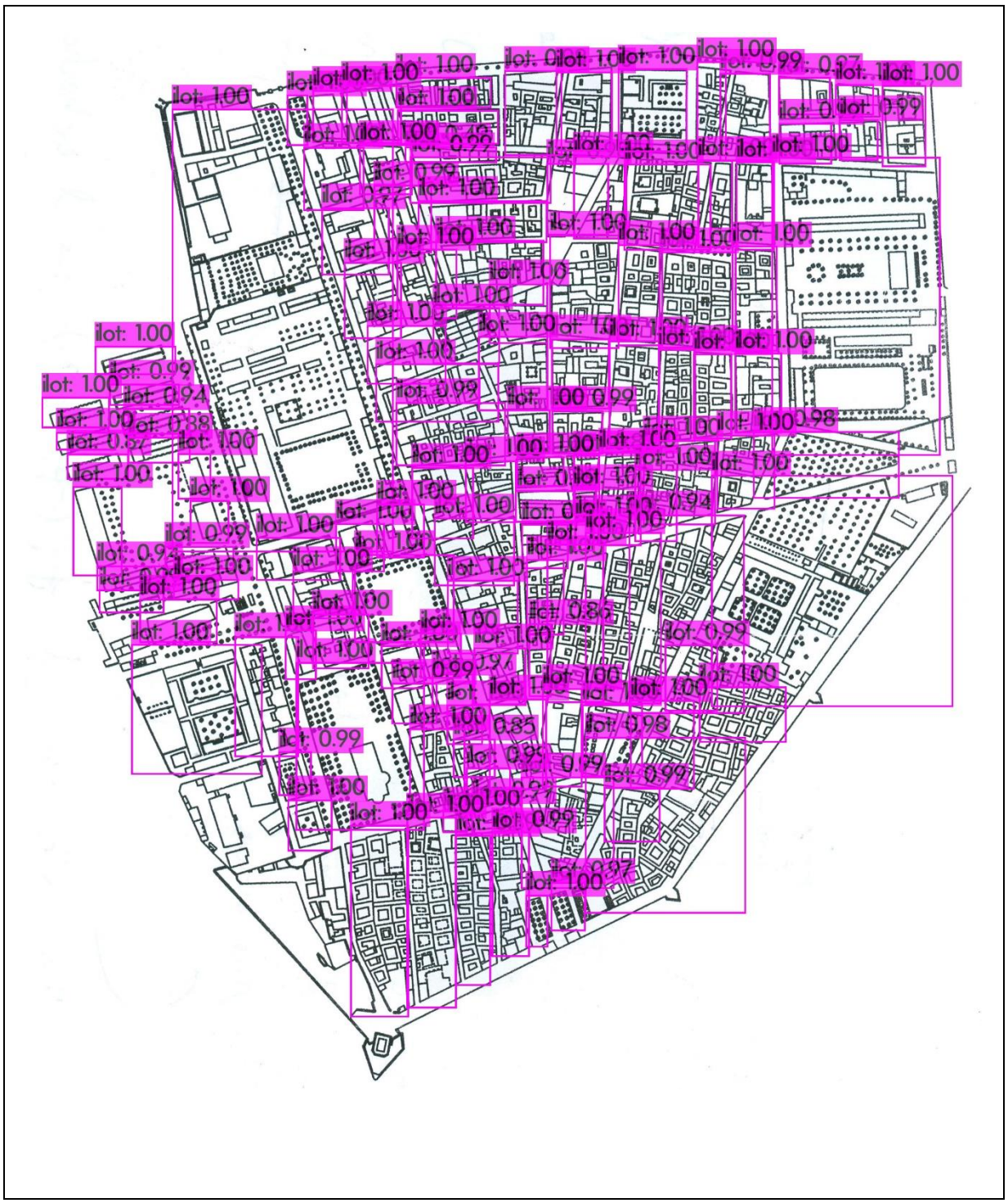
3.1 Carte complexe (quelconque):



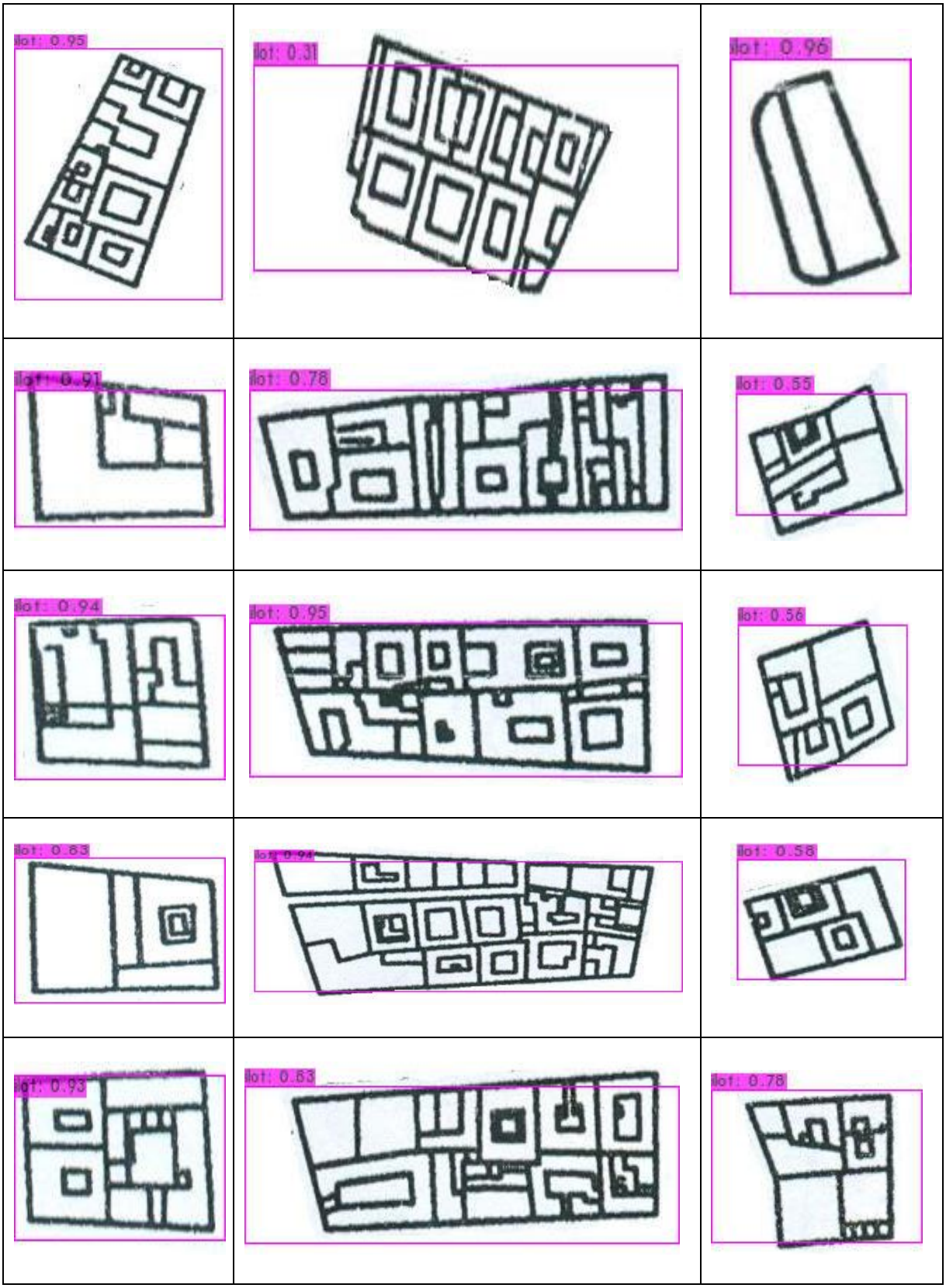

lot: 0.93
lot: 0.84
lot: 0.90
lot: 0.89
lot: 0.95
lot: 0.90
lot: 0.79
lot: 0.68
lot: 0.93

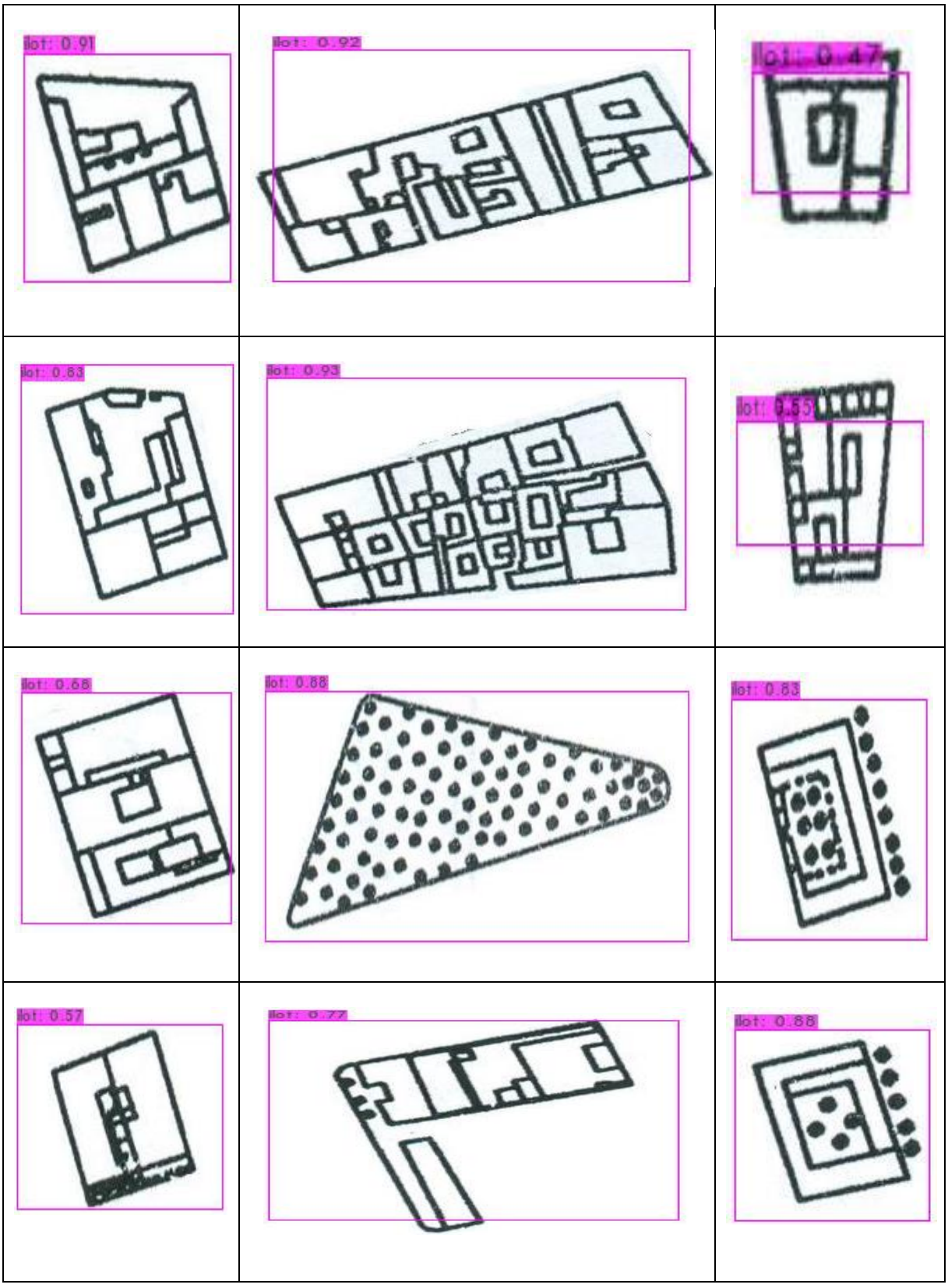


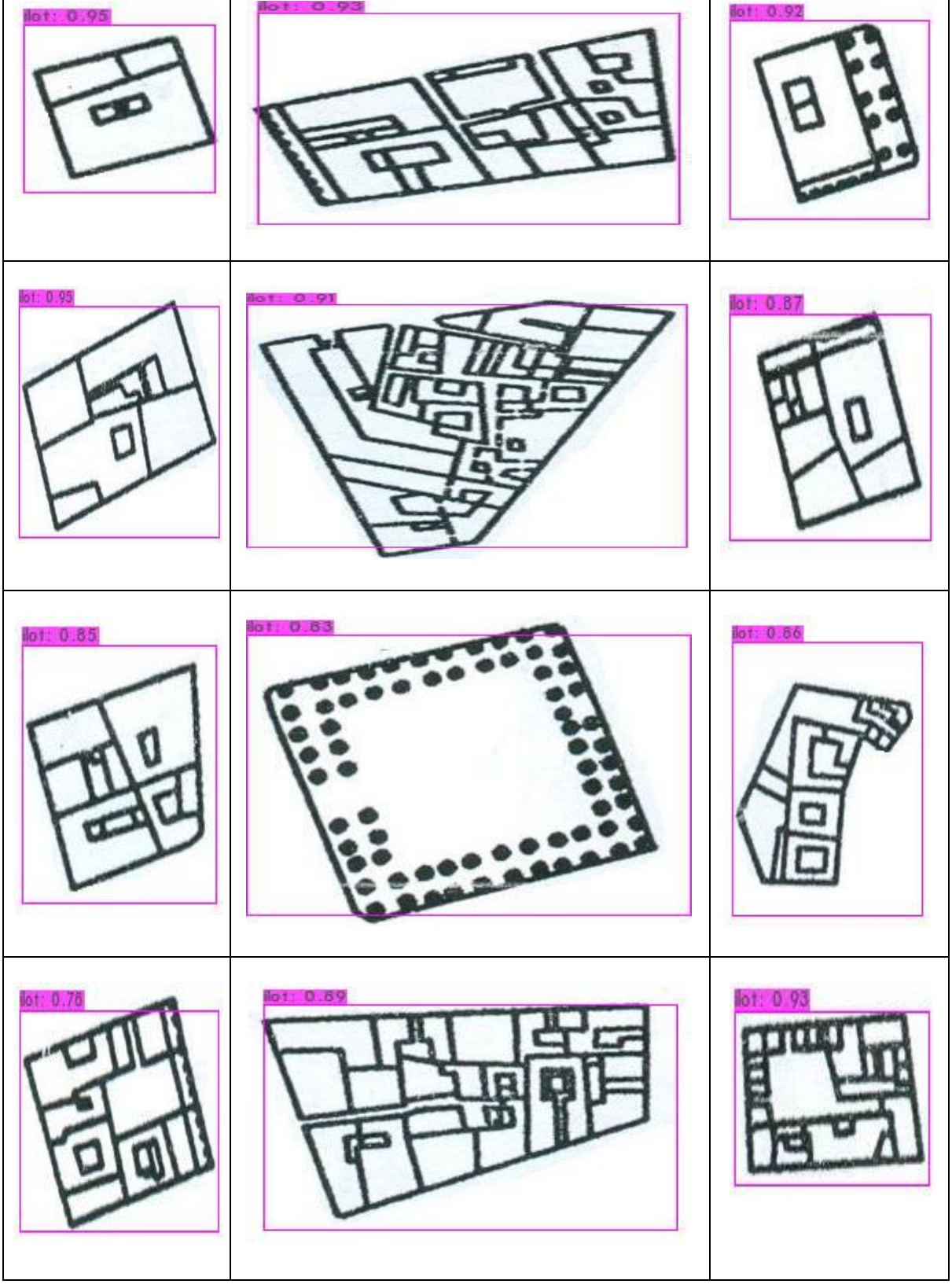
3.2 La carte complexe (centre de blida) :

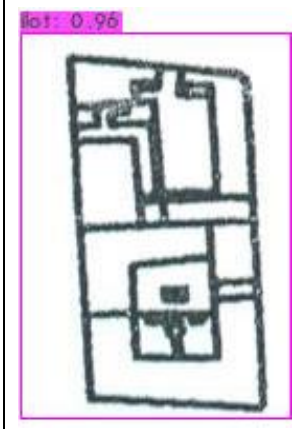


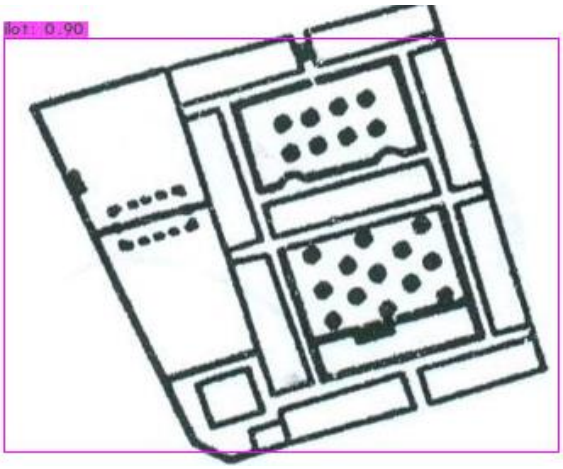
Les ilots:

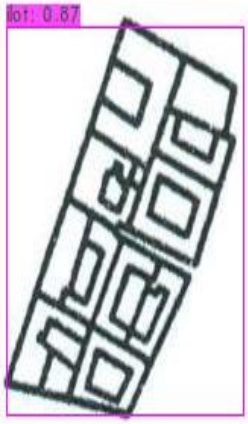

ilot: 0.95
ilot: 0.31
ilot: 0.96
ilot: 0.91
ilot: 0.78
ilot: 0.55
ilot: 0.94
ilot: 0.95
ilot: 0.56
ilot: 0.83
ilot: 0.94
ilot: 0.58
ilot: 0.93
ilot: 0.83
ilot: 0.78

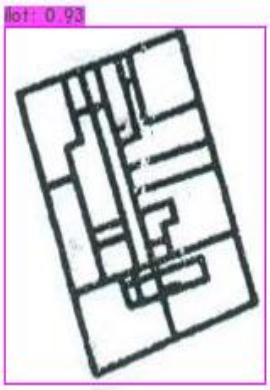

llot: 0.91
llot: 0.92
llot: 0.47
llot: 0.63
llot: 0.93
llot: 0.55
llot: 0.68
llot: 0.88
llot: 0.83
llot: 0.57
llot: 0.88

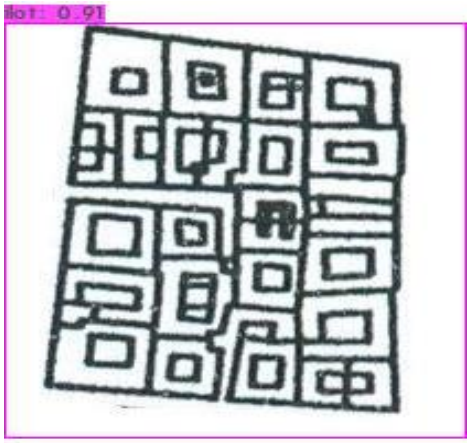

Ilot: 0.95
Ilot: 0.93
Ilot: 0.92
Ilot: 0.93
Ilot: 0.91
Ilot: 0.87
Ilot: 0.85
Ilot: 0.63
Ilot: 0.66
Ilot: 0.76
Ilot: 0.69
Ilot: 0.93

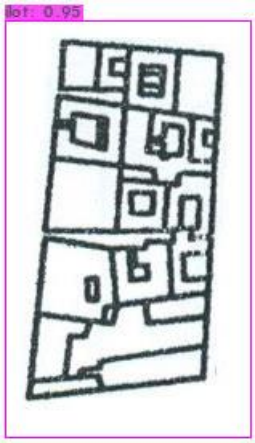

lot: 0.96

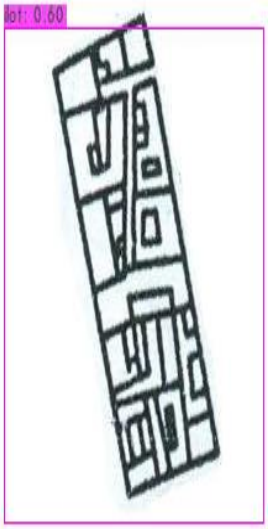

lot: 0.90

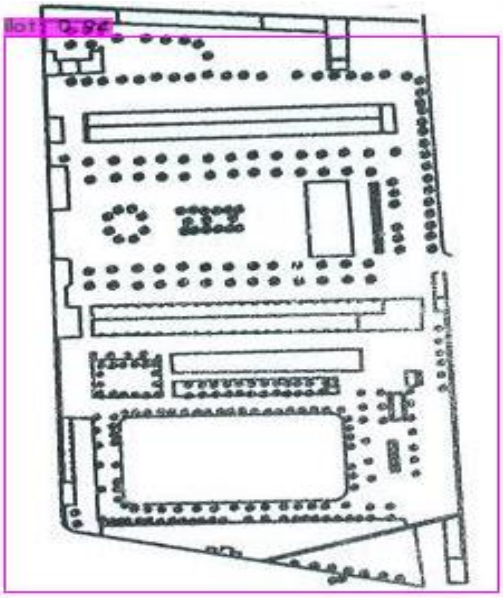

lot: 0.87

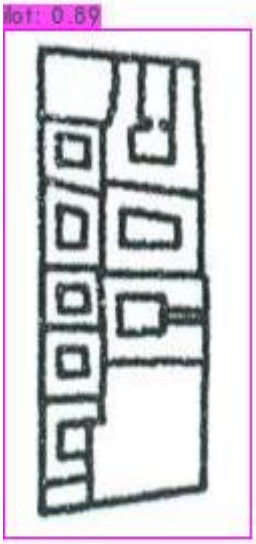

lot: 0.93

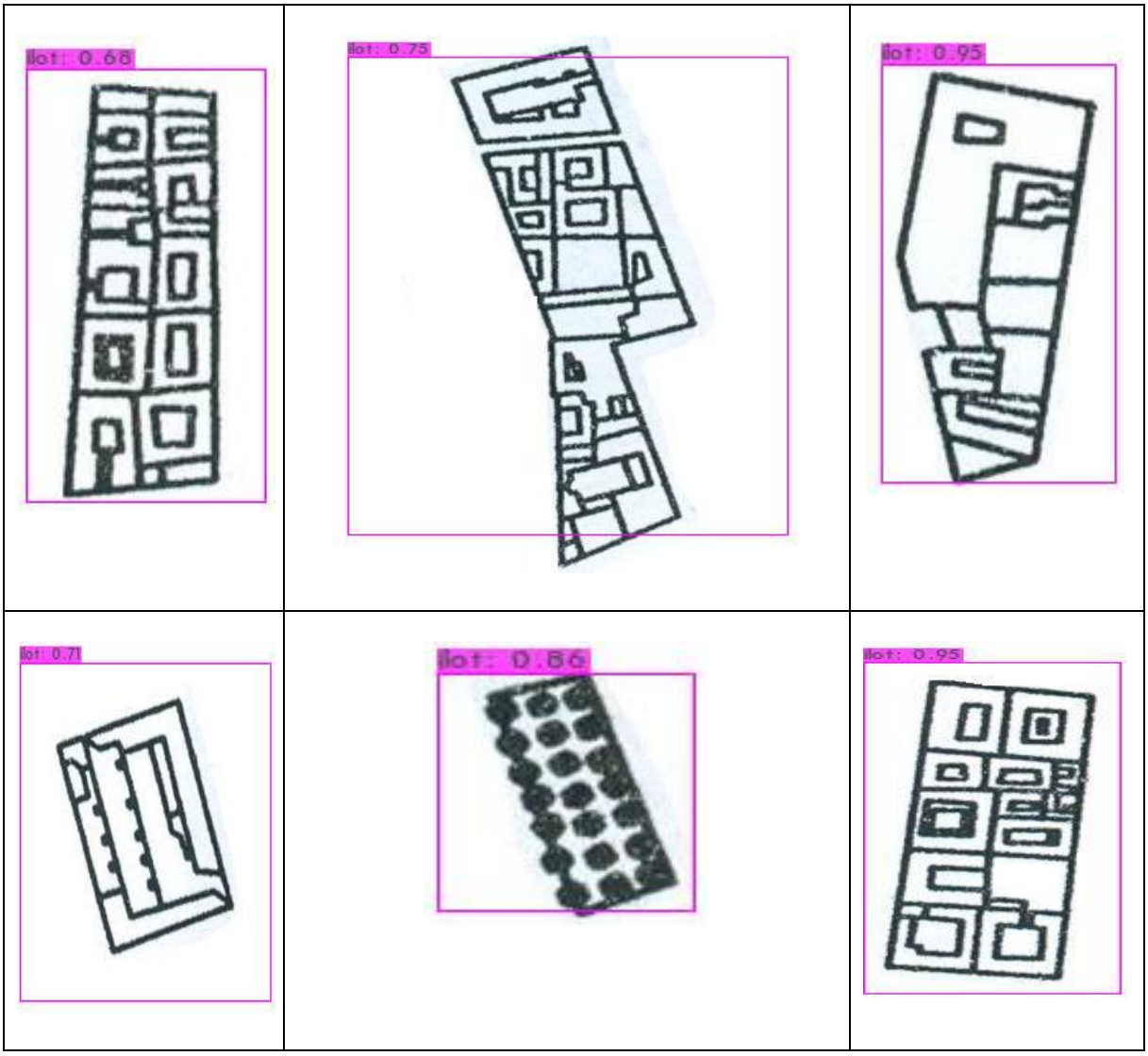

lot: 0.91

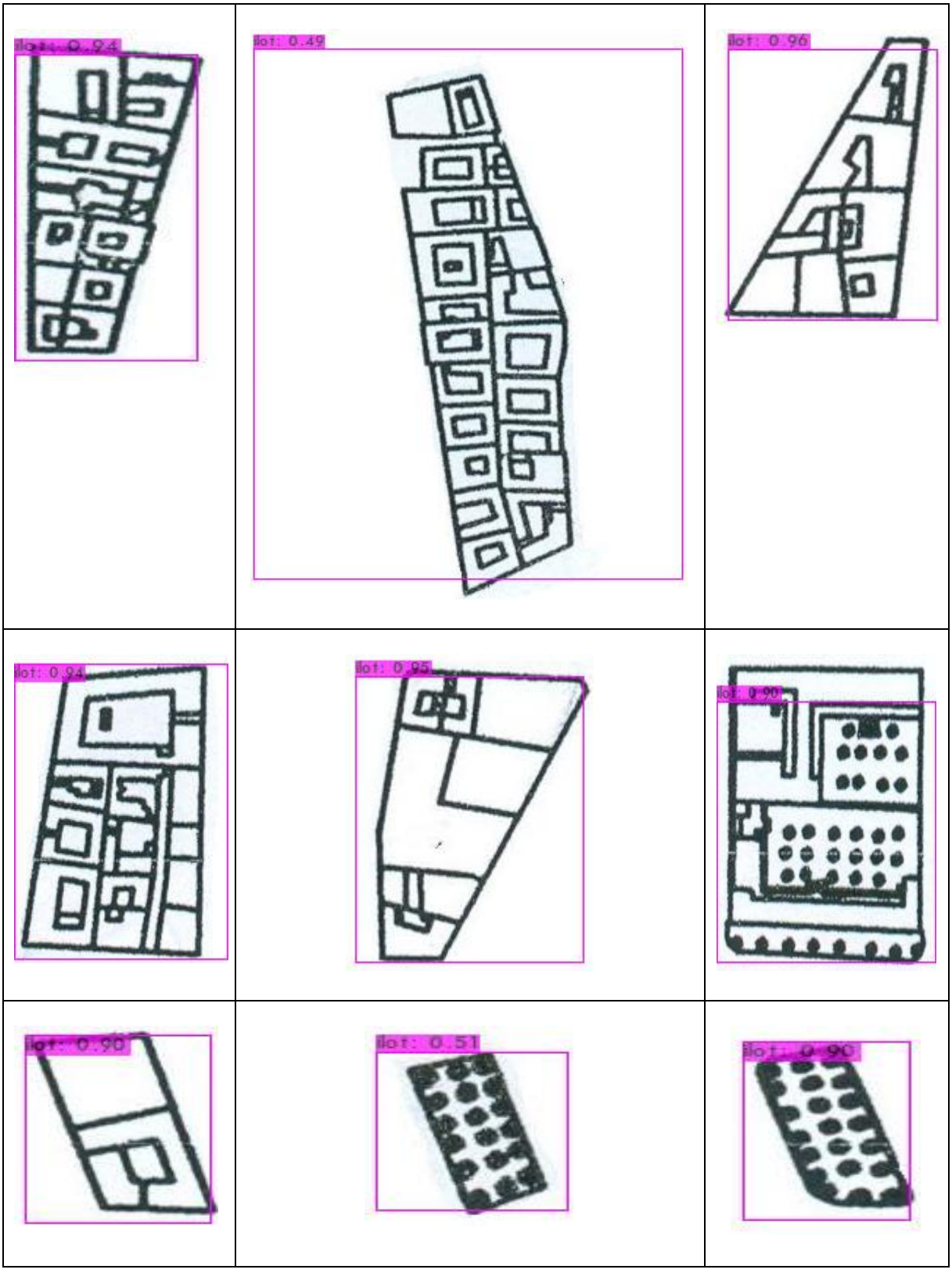

lot: 0.95

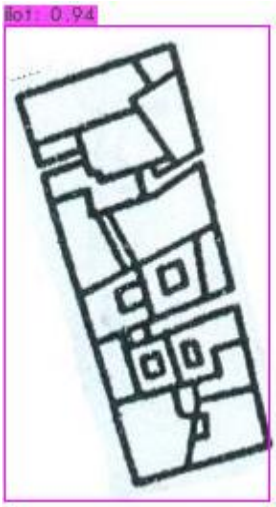
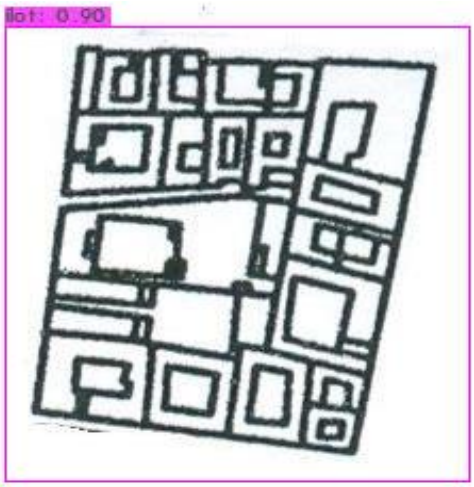
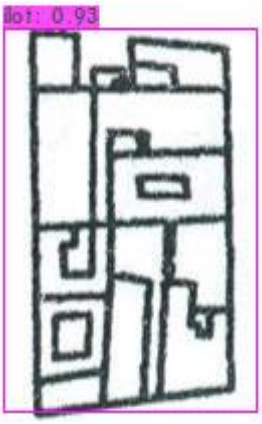
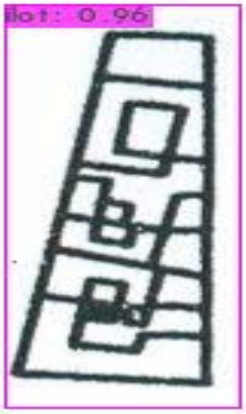
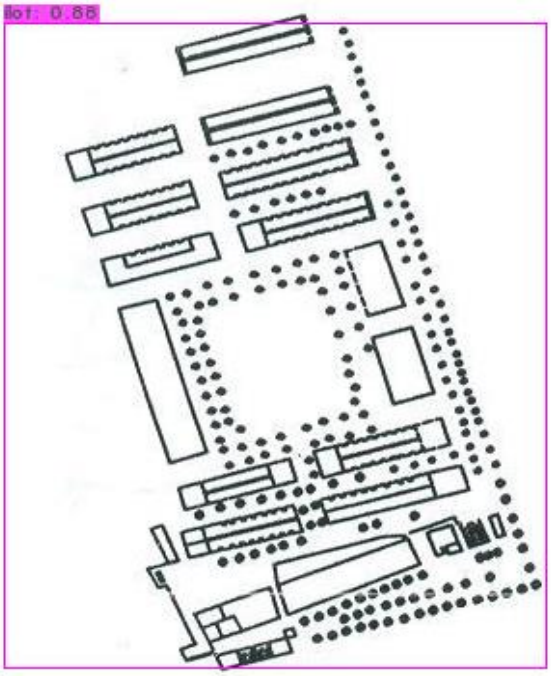
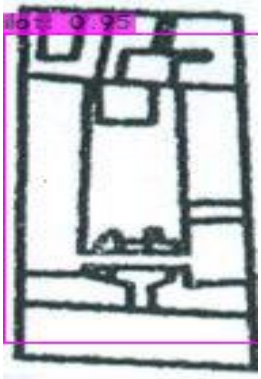

lot: 0.60

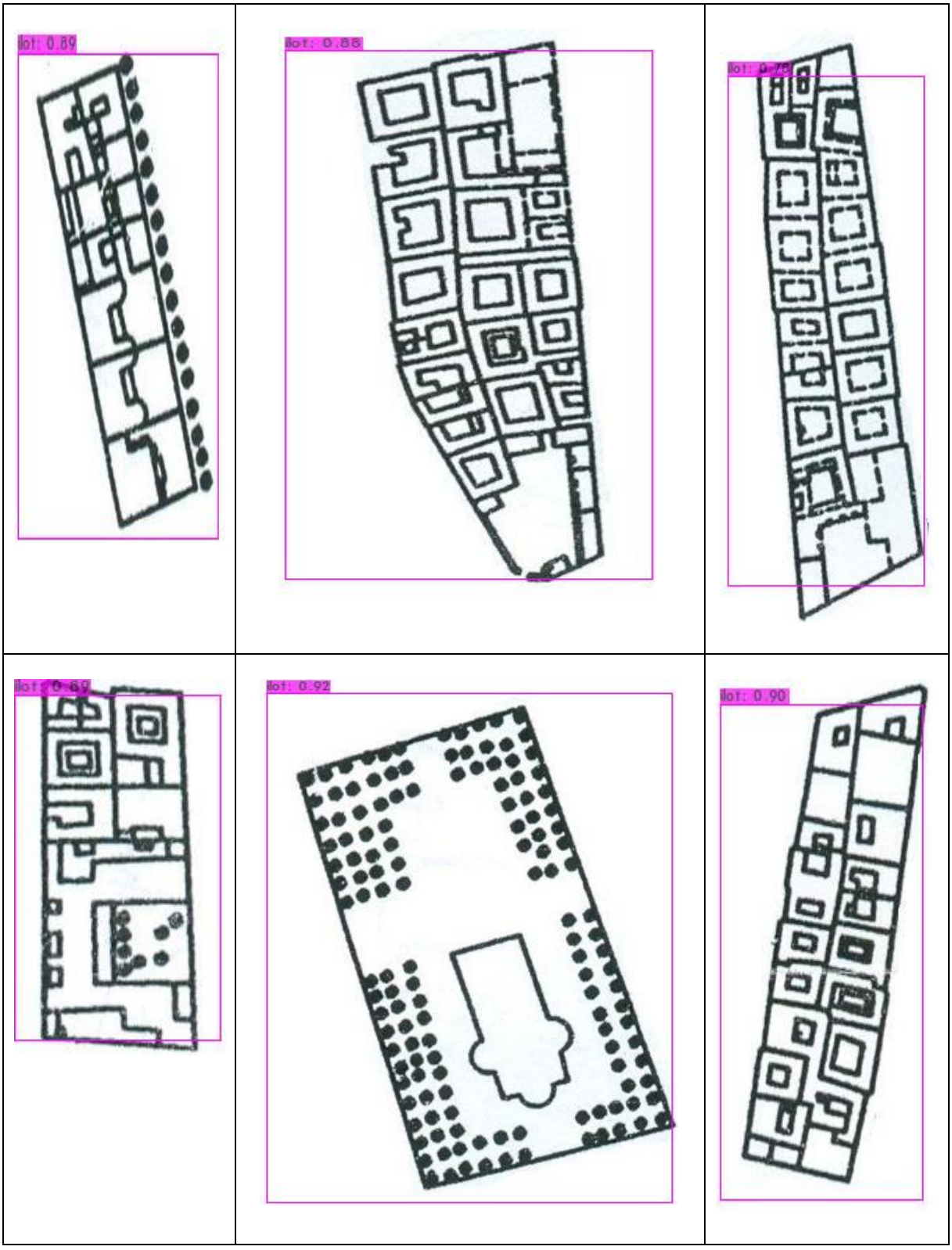

lot: 0.96

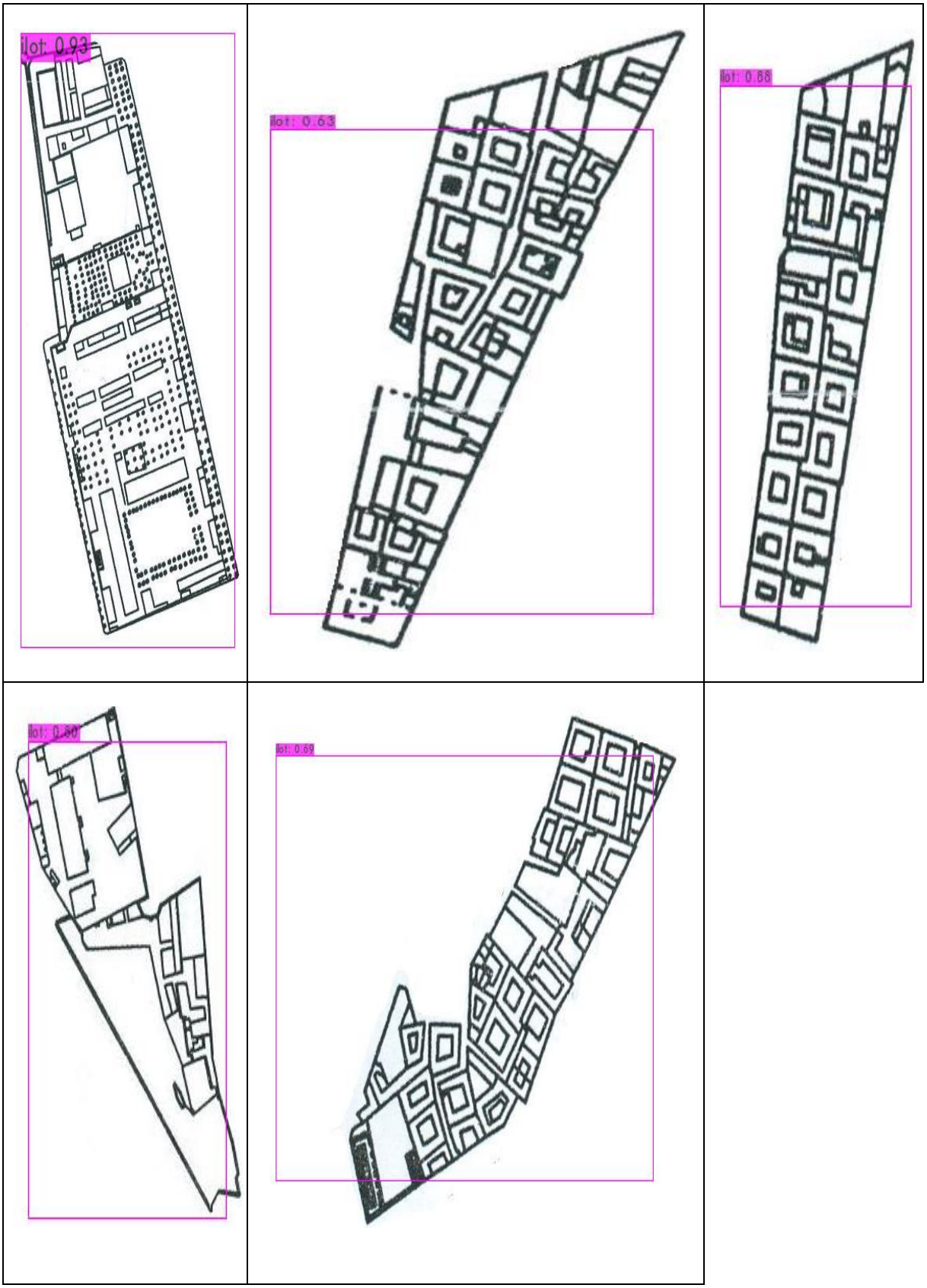

lot: 0.89


llot: 0.88
llot: 0.75
llot: 0.95
llot: 0.71
llot: 0.86
llot: 0.95


dot: 0.94
dot: 0.49
dot: 0.96
dot: 0.94
dot: 0.95
dot: 0.90
dot: 0.90
dot: 0.51
dot: 0.90

 dot: 0.94	 dot: 0.90	 dot: 0.93
 dot: 0.96	 dot: 0.88	 dot: 0.95


lot: 0.89
lot: 0.88
lot: 0.78
lot: 0.89
lot: 0.92
lot: 0.90


Ilot: 0.93
Ilot: 0.63
Ilot: 0.88
Ilot: 0.80
Ilot: 0.69

3.3 Carte satellitaire:


ilot: 0.95
ilot: 0.93
ilot: 0.92
ilot: 0.86
ilot: 0.94
ogts Houari
ilot: 0.86