

ÉCOLE NATIONALE DE LA STATISTIQUE  
ET DE L'ADMINISTRATION ÉCONOMIQUE



ANNÉE SCOLAIRE 2022-2023  
STAT'APP DE 2ÈME ANNÉE

---

**Deep learning pour la détection d'installations  
solaires individuelles à partir d'images  
aériennes**

---

*Groupe 12 :*

Florian BASSO

Théo FERRY

Dmitri LEBRUN

Yanis REMMACHE

*Sous la direction de :*  
Gabriel KASMI (RTE)



## **Remerciements**

Nos remerciements s'adressent tout particulièrement à Gabriel KASMI pour son encadrement et sa disponibilité que nous avons beaucoup appréciée. Ses remarques et conseils nous ont permis de mieux comprendre certains points clés du sujet ainsi que la quantité importante de notions à assimiler. Les interactions que nous avons pu avoir avec Gabriel ont largement contribué à la réussite du projet.

Avec notre reconnaissance, Florian, Théo, Dmitri et Yanis.

# Table des matières

<b>1 Déetecter les panneaux photovoltaïques : un enjeu public majeur</b>	<b>9</b>
1.1 Le réseau électrique : un système régi par la loi de l'offre et de la demande . . . . .	9
1.2 L'importance de la détection des panneaux photovoltaïque . . . . .	10
1.3 Du recensement à la détection des panneaux photovoltaïques : panorama des méthodes existantes . . . . .	11
1.3.1 Les limites des registres recensant les installations de panneaux photovoltaïques... . . . . .	11
1.3.2 ... Sont comblées par les méthodes algorithmiques, en particulier celles utilisant les réseaux de neurones convolutifs . . . . .	11
<b>2 Une méthode pour la détection automatique d'installations : l'apprentissage statistique et les réseaux de neurones convolutifs</b>	<b>12</b>
2.1 La théorie des réseaux de neurones appliquée à un cadre d'apprentissage supervisé	12
2.1.1 Le risque empirique : un estimateur du risque théorique que l'on souhaite optimiser . . . . .	12
2.1.2 La validation simple : une méthode utilisée pour garantir la consistance du risque empirique . . . . .	13
2.2 Du perceptron aux réseaux neuronaux convolutifs . . . . .	14
2.2.1 Le perceptron, unité de base du réseau de neurones . . . . .	14
2.2.2 Les réseaux de neurones convolutifs : un ensemble d'algorithmes plus adaptés au traitement d'images . . . . .	14
2.2.3 Architecture d'un réseau de neurones convolutif . . . . .	15
2.2.4 Invariance par translation et avantages des réseaux convolutif . . . . .	15
2.2.5 Optimisation des paramètres d'un réseau de neurones convolutifs par ré-tropagation du gradient et descente de gradient stochastique . . . . .	16
2.3 Évaluer et améliorer les performances d'un réseau de neurones convolutif . . . . .	17
2.3.1 Métriques d'évaluation des performances d'un réseau de neurones dans le cadre d'un problème de classification binaire . . . . .	17
2.3.2 Le rôle des data augmentations, du dropout et du weight decay dans la réduction du sur-apprentissage . . . . .	18

2.3.3	Le transfer learning : une méthode fréquemment utilisée pour limiter le sur-apprentissage et le coût computationnel . . . . .	19
<b>3</b>	<b>Un large éventail de données en amont de la construction du réseau de neurones convolutif</b>	<b>19</b>
<b>4</b>	<b>Application de la théorie des réseaux de neurones convolutifs pour la détection d'installations solaires</b>	<b>20</b>
4.1	Le LeNet5 : un réseau peu complexe procurant des résultats satisfaisants . . . . .	20
4.1.1	Architecture du LeNet5 . . . . .	20
4.1.2	Choix des data augmentations : le ColorJitter et les crops aléatoires ont été retenus sur le modèle final . . . . .	21
4.1.3	Le faible coût computationnel d'apprentissage permet d'hyper-paramétriser le LeNet5 en un temps fini . . . . .	21
4.1.4	En dépit d'une réduction significative de la résolution des images, le LeNet5 procure des résultats satisfaisants . . . . .	21
4.2	Le ResNet18 : un modèle plus complexe, adapté pour les images en grandes dimensions . . . . .	23
4.3	Utiliser des <i>crops</i> et actualiser les labels : la méthode de l' <i>auto-crop</i> . . . . .	23
4.4	Apprendre exclusivement les couches fully connected du ResNet18 pré-entraîné : une option computationnellement avantageuse produisant des résultats moyens .	24
4.5	Entraîner l'ensemble des couches du ResNet18 pré-entraîné s'avère être une stratégie efficace en dépit du coût computationnel élevé . . . . .	26
4.6	Le ResNet18 entraîné entièrement sur nos données d'apprentissage est le modèle qui sépare au mieux les données de validation . . . . .	28
<b>A</b>	<b>Rythme d'installation du photovoltaïque en France et en Europe</b>	<b>33</b>
<b>B</b>	<b>Détail des méthodes traditionnelles d'estimation de la production photovoltaïque</b>	<b>34</b>
<b>C</b>	<b>Du perceptron au réseau de neurones</b>	<b>34</b>
C.1	Le fondement des réseaux de neurones : le perceptron . . . . .	34
C.2	Définition d'un perceptron multicouches . . . . .	35
<b>D</b>	<b>Fonctions d'activation usuelles</b>	<b>35</b>

<b>E Définition formelle du produit de convolution, exemple et utilisation dans les réseaux de neurones convolutifs</b>	<b>36</b>
E.1 Définition du produit de convolution . . . . .	36
E.2 Exemple d'application . . . . .	36
E.3 Définition du max pooling et de l'average pooling . . . . .	37
E.4 Représentation graphique d'un neurone de convolution . . . . .	38
E.5 Représentation graphique d'une couche convulsive . . . . .	38
<b>F Descente de gradient stochastique et rétropropagation du gradient</b>	<b>38</b>
<b>G Définition et mise en contexte des métriques de classification</b>	<b>39</b>
<b>H Présentation des data augmentations utilisées</b>	<b>41</b>
<b>I Architecture du LeNet5</b>	<b>43</b>
<b>J Choix des data augmentations et fine tuning du LeNet5</b>	<b>43</b>
J.1 Les data augmentations : un outil contre le sur-apprentissage . . . . .	44
J.1.1 Analyse des performances du réseau sans data augmentations . . . . .	44
J.1.2 La normalisation des images et l'ajout de ColorJitter ne semble pas régler le problème du sur-apprentissage . . . . .	45
J.1.3 L'ajout de crops limite le sur-apprentissage . . . . .	47
J.2 Le choix de la métrique à optimiser dans l'hyper-paramétrisation du réseau est crucial . . . . .	49
<b>K Choix du learning rate pour le ResNet18</b>	<b>53</b>
<b>L T-SNE : une technique de réduction de dimension qui préserve les similarités locales</b>	<b>53</b>

## Table des figures

1	Courbes ROC et precision-recall du LeNet5, selon les différentes configurations testées . . . . .	22
2	Evolution de la log-loss au fil des epochs sur le ResNet18 pré-entraîné (entraînement des couches fully-connected) . . . . .	25
3	Evolution de la log-loss au fil des epochs sur le ResNet18 pré-entraîné (entraînement de toutes les couches) . . . . .	26
4	Courbes ROC et precision-recall du ResNet18, selon le nombre de couches entraînées . . . . .	27
5	Visualisation T-SNE des images contenues dans l'échantillon de validation (perplexity de 50). A gauche : LeNet5 ; au centre : ResNet18 : entraîné sur les couches fully connected ; à droite : ResNet18 entraîné sur toutes les couches. Une image encadrée en rouge contient une installation solaire. Une image encadrée en bleu n'en contient pas. . . . .	28
6	Rythme d'installation du photovoltaïque en France depuis 2010, tiré de l'ADEME	33
7	Rythme européen comparé d'installation du photovoltaïque, tiré de RTE 2022 .	33
8	Représentation schématique d'un perceptron linéaire . . . . .	34
9	Représentation schématique d'un réseau de neurones à trois couches . . . . .	35
10	Représentation schématique du produit de convolution en dimension 2 . . . . .	37
11	Average pooling de taille 2 . . . . .	37
12	Représentation schématique d'un neurone de convolution. Ici, le motif est de dimension 3 . . . . .	38
13	Couche de convolution prenant en entrée $k$ canaux et composée de $l$ filtres . . .	38
14	Mauvaise utilisation du ColorJitter : $brightness \in [1, 10]$ , $contrast \in [1, 10]$ , $saturation \in [1, 10]$ , $hue \in [-0.1, 0.1]$ . . . . .	42
15	Schéma présentant l'architecture du LeNet5 utilisé . . . . .	43
16	Evolution des métriques au fil des epochs pour le LeNet5, sans data augmentations	44
17	Evolution des métriques au fil des epochs pour le LeNet5 : normalisation des images et ColorJitter sur l'échantillon de validation . . . . .	46
18	Evolution des métriques au fil des epochs pour le LeNet5 : normalisation des images, ColorJitter et crops sur l'échantillon de validation . . . . .	48
19	Evolution des métriques au fil des epochs pour le LeNet5 : hyper-paramétrisation par rapport au F2-score . . . . .	50

20	Evolution des métriques au fil des epochs pour le LeNet5 : hyper-paramétrisation par rapport au F1-score . . . . .	52
21	Perte en fonction du learning rate . . . . .	53

# Notations

Les notations générales utilisées dans le texte sont les suivantes :

- $\mathbf{x}$  : vecteur  $\mathbf{x}$
- $\mathbf{x}^T$  : transposée de  $\mathbf{x}$
- $\mathcal{D}_n$  : n-échantillon
- $\mathcal{D}_A$  : échantillon d'apprentissage
- $\mathcal{D}_V$  : échantillon de validation
- TP : nombre de vrais positifs
- TN : nombre de vrais négatifs
- FP : nombre de faux positifs
- FN : nombre de faux négatifs

Nous présentons ici les notations relatives aux réseaux de neurones.

- $L$  : nombre de couches du réseau de neurones
- $q_l$  : nombre de neurones dans la couche  $l$  du réseau
- $w_{i,j}^l$  : coefficient qui relie le i-ème neurone de la  $l$ -ème couche au j-ème de la  $l - 1$  ème couche
- $W^l = (w_{i,j})_{1 \leq i \leq q_l, 1 \leq j \leq q_{l-1}}$  : matrice de dimension  $q_l \times q_{l-1}$  contenant les poids reliant la couche  $l$  à la couche  $l - 1$
- $b_i^l$  : biais du i-ème neurone de la  $l$ -ème couche
- $b^l = (b_1^l, \dots, b_{q_l}^l)^T$  : vecteur de taille  $q_l$  contenant les biais de la couche  $l$
- $a_i^l$  : valeur d'activation du i-ème neurone de la l-ième couche
- $a^L = (a_1^l, \dots, a_{q_l}^l)^T$  : vecteur de taille  $q_l$  contenant les sorties de la couche  $l$

# Introduction

Déjà multipliée par six entre 2011 et 2020 (RTE), la proportion de production électrique émanant de l'énergie solaire ne cesse de croître. Les estimations du Réseau de transport d'électricité (RTE), montrent que la capacité du parc solaire photovoltaïque<sup>1</sup> serait encore une fois multipliée par six entre 2030 et 2050 (RTE [2022]). La production photovoltaïque se scinde en deux parties : une production issue de centrales au sol et une production issue de petites installations individuelles (toits de particuliers, ombrières) dite "diffuse". Or la production diffuse ne fait l'objet d'aucune obligation de télémétrie. Avec la croissance attendue du photovoltaïque, et en l'absence d'obligation de télémétrie des installations, RTE doit améliorer ses techniques d'estimation de la production électrique issue d'installations diffuse.

Sans ces estimations, RTE risque de déroger à la loi fondamentale qui régit le fonctionnement du réseau électrique : comme il n'est pas possible de stocker de l'électricité telle quelle, il est nécessaire d'avoir un équilibre permanent entre offre (production) et demande (consommation).

Avec le développement de la théorie des réseaux de neurones convolutifs au cours des années 2010, de nombreux pays comme les États-Unis (Malof et al. [2016]) ou encore les Pays-Bas (Kausika et al. [2021a]) ont appliqué ces méthodes algorithmiques pour estimer la capacité de production électrique émanant de l'énergie solaire.

Or, détecter les installations photovoltaïques est un objectif qui s'inscrit dans un contexte métier bien particulier. Premièrement, le but, pour un gestionnaire de réseau de transport comme RTE, est de détecter le maximum d'installations photovoltaïques diffuses. Le risque, en pratique, est que la quasi-intégralité des bâtiments soient détectées avec une installation solaire.

Comment calibrer les réseaux de neurones convolutifs de sorte à détecter le maximum d'installations solaires, tout en ayant un nombre d'images classées comme faux positif acceptable ?

Par ailleurs, en comparaison de certaines bases de données d'entraînement usuelles comme ImageNet<sup>2</sup>(Russakovsky et al. [2015]), les images satellites utilisées pour la détection d'installations solaires sont par nature moins variées en termes de formes et de couleurs. De plus, elles contiennent certaines spécificités propres au pays et à la région dont elles sont issues. Notre base de données est en effet composée exclusivement d'images satellites prises en France. Ainsi, les panneaux sont toujours orientés au sud et occupent bien souvent une faible superficie sur l'image. Par ailleurs, certaines spécificités architecturales propres à chaque région peuvent être observées sur les images. Par exemple, dans le Nord, les tuiles sont principalement de couleur rouge, là où le gris est plutôt la couleur de référence en Bretagne.

Les modèles de détection d'installations solaires sont déployés à l'échelle nationale. Ainsi,

1. La capacité installée désigne la production maximale réalisable lorsque l'ensoleillement est optimal.

2. Base de données conçue à l'origine pour le challenge annuel *ImageNet Large Scale Visual Recognition Challenge*, challenge annuel de reconnaissance d'images qui a largement contribué à la popularité de l'utilisation des réseaux de neurones.

l'orientation des panneaux solaires est une information à conserver lors de l'apprentissage du modèle. Le choix de vouloir gommer ou non les spécificités architecturales relatives à chaque région est plus ambigu. D'un côté, ces dernières constituent une information utile pour apprendre un modèle de machine learning. De l'autre, les spécificités architecturales peuvent apparaître comme des facteurs de sur-apprentissage, pouvant restreindre le déploiement de tels modèles à l'échelle nationale. C'est le cas si la base d'apprentissage est constituée d'un nombre important d'images prises, par exemple, dans le nord de la France. Ajouté à cela le fait le que les installations solaires occupent une faible superficie de l'image, il apparaît que le choix des transformations appliquées au préalable sur les images est restreint.

Quelles techniques pouvons-nous employer pour conserver les spécificités propres aux images satellites françaises, tout en limitant le risque de sur-apprentissage ?

## 1 Déetecter les panneaux photovoltaïques : un enjeu public majeur

### 1.1 Le réseau électrique : un système régi par la loi de l'offre et de la demande

Le réseau électrique est un système complexe d'infrastructures assurant le transfert d'énergie entre des postes de production et de consommation via le transport d'électricité. Une fois l'énergie produite par le système électrique, son transport jusque dans les principaux centres de consommation est assuré par le réseau de transport (RTE pour la France). Se met alors en place une étape de distribution pour amener l'électricité jusqu'au consommateur (assurée par des entreprises comme Enedis par exemple).

L'utilisation de l'électricité comme vecteur de transport impose des contraintes physiques au réseau. En effet, il n'est pas possible de stocker de l'électricité en grande quantité. De plus, accumuler de l'électricité sur un point du réseau peut conduire à une surtension et des dégâts sur les matériels. De même, une sous-tension sur le réseau conduit, d'une part, à une non-satisfaction de la demande en électricité des consommateurs et d'autre part, à l'endommagement du réseau. C'est par une sous-tension liée à la rupture d'une ligne haute tension avec la Suisse que l'Italie a connu en 2003 un black-out total en quelques secondes. D'où la loi fondamentale à laquelle le réseau doit obéir : **l'équilibre permanent entre offre (production) et demande (consommation)**.

Cette contrainte d'équilibre entre l'offre et la demande est très forte, d'autant que la consommation est soumise à des **variabilités** et que les sources d'électricité ont des contraintes de production. La consommation d'électricité a deux échelles de variations : une **infra-journalière** liée à nos habitudes de vie (cuisine, chauffage, douche) et une **saisonnière** liée aux variations de températures. C'est en principe la production d'électricité qui suit les fluctuations

de la consommation<sup>3</sup>. C'est en partie en raison de cette contrainte d'équilibre que les sources d'énergie électrique sont séparées en deux catégories : les sources intermittentes et les non-intermittentes (ou pilotables). Les sources d'énergie intermittentes sont les sources de production d'énergie renouvelable correspondant à des flux naturels qui ne sont pas disponibles en permanence. Leur disponibilité varie fortement et n'est pas contrôlable (énergie solaire, éolien). Les sources pilotables sont au contraire celles pouvant être commandées dans un laps de temps restreint afin de s'ajuster à la demande (comme les barrages ou les centrales thermiques à gaz, au charbon ou nucléaire).

Pour assurer l'équilibre entre l'offre et la demande, il existe plusieurs leviers de contrôle dont **l'ajustement des moyens de productions** en fonction des besoins. Pour avoir une plus grande marge de manœuvre sur l'ajustement des productions, il convient de réfléchir de manière systémique aux sources de production exploitables par le réseau. Ainsi, augmenter les sources d'énergies intermittentes crée de la variabilité du côté de la production. Dans ce contexte, connaître l'emplacement des panneaux photovoltaïques, des sources de production d'électricité intermittentes dépendant du contexte d'ensoleillement local, est essentiel pour que le régulateur puisse moduler ses sources de production électrique pilotables afin d'équilibrer le réseau. Il pourra prédire via les prévisions météo les niveaux de production des panneaux et mieux planifier l'utilisation de ses sources d'énergie piloteable.

## 1.2 L'importance de la détection des panneaux photovoltaïque

Malgré un retard relatif, en comparaison des autres pays européens [7], le photovoltaïque a cru de manière importante en France où la proportion de production électrique solaire a été multipliée par 6 entre 2011 et 2020 (RTE). Cette tendance à la hausse résulte du développement du photovoltaïque en Europe. D'après Enedis, entre 2015 et 2021, le nombre de foyers français équipés en panneaux photovoltaïques a été multiplié par 37. Ce chiffre témoigne en particulier du rôle croissant des installations dites *diffuses* dans la production électrique française.

À l'inverse des centrales solaires, souvent possédées par des entreprises productrices d'énergie (c'est le cas par exemple de la Centrale solaire de Cestas possédée par Neonén), dont la surface s'étale sur plusieurs hectares, les installations diffuses sont de petites tailles et peuvent être détenues par des particuliers dans le but de fournir de l'énergie à une échelle modeste (alimenter une habitation, un centre commercial, etc.). Ces équipements sont ainsi décentralisés, au sens où ils ne sont pas possédés par les producteurs d'énergie. En raison de cet aspect et surtout du manque d'information contenu dans les registres nationaux<sup>4</sup> (Kasmi et al. [2022]), la comptabilisation, la caractérisation et la localisation de ces installations est devenue un enjeu majeur pour les acteurs publics (voir les travaux de Yu et al. [2018]).

3. Bien qu'il y ait de plus en plus d'initiatives afin d'ajuster et de lisser la demande d'électricité, voir par exemple les leviers de flexibilité recensés sur cette page du Ministère de la transition énergétique.

4. Aucune obligation de télémesure (contrairement aux centrales et aux autres modes de production), et les registres d'installations existants sont incomplets pour pouvoir faire de l'estimation de production d'après les auteurs de l'article qui suit.

Au total, les entreprises du service public ont besoin de données sur les équipements photovoltaïques pour, d'une part, planifier les capacités énergétiques des réseaux électriques à long terme (Gust et al. [2016]) et, d'autre part, pour mieux exploiter leur réseau (trading d'énergie et estimation de la production électrique en temps réel) (voir Mayer et al. [2022]).

## 1.3 Du recensement à la détection des panneaux photovoltaïques : panorama des méthodes existantes

### 1.3.1 Les limites des registres recensant les installations de panneaux photovoltaïques...

A l'heure actuelle, il existe dans certains pays des registres recensant les installations photovoltaïques diffuses. Par exemple, aux Pays-Bas, les propriétaires de petites installations (au maximum 15 kWp) doivent enregistrer leur équipement au registre national (comme exploité par Kausika et al. [2021b]). Or, d'après Mayer et al. [2022] la qualité de l'information contenue dans ces registres est discutable et même souvent incomplète. Les défauts de ces méthodes de collecte d'information sont présentées en détail en annexe B. Ainsi, d'après ces auteurs, ceci conduit à obtenir une vision erronée des capacités photovoltaïques présentes qui peut réduire l'efficacité des politiques énergétiques.

D'autres méthodes pour obtenir des informations sur la distribution des panneaux photovoltaïques, comme les sondages ou les autoquestionnaires, sont coûteuses et longues à mettre en place. Ces méthodes sont aussi plus limitées spatialement puisque aux États-Unis, par exemple, les données ne sont disponibles qu'au niveau fédéral ou au niveau des états (mentionné par Malof et al. [2016]).

### 1.3.2 ... Sont comblées par les méthodes algorithmiques, en particulier celles utilisant les réseaux de neurones convolutifs

Mayer et al. [2022] présentent une typologie des méthodes fréquemment utilisées pour détecter et caractériser les installations photovoltaïques à partir d'images aériennes. Certaines techniques sont plus adaptées que d'autres selon les objectifs visés.

Les méthodes appartenant au premier groupe ont pour point commun de s'appuyer sur l'utilisation d'un classifieur binaire pour détecter si une image contient ou non un panneau photovoltaïque. L'analyse peut être approfondie à grâce à l'ajout d'effets socio-démographiques et causaux. Ces techniques permettent exclusivement de détecter la présence d'un panneau. Aucune information sur la localisation précise ou les caractéristiques ne peut être obtenue.

Pour pallier ce manque, des méthodes visant à localiser les panneaux et à estimer leurs tailles ont été développées. Plutôt que d'utiliser un classifieur binaire, le but est de détecter les pixels dans une image qui correspondent à une installation photovoltaïque. Dès lors, il est possible d'obtenir la localisation exacte de l'équipement. Pour répondre à ce besoin, les méthodes nécessitent souvent l'usage du *deep learning*. Le projet DeepSolar est par exemple le premier projet utilisant ces méthodes pour détecter les équipements photovoltaïques et estimer leur surface (Kasmi et al. [2022]).

D'autres méthodes algorithmiques comme la forêt d'arbres décisionnels (mobilisée dans les travaux de Malof et al. [2016]) permettent de détecter la localisation des panneaux, mais ne permettent pas de délimiter la forme de l'objet détecté. Les réseaux de neurones convolutifs ont ainsi été utilisés dans le but de mieux estimer la forme des panneaux. Jong et al. [2020], par exemple, utilisent un réseau de neurones convolutifs pour faire de la détection d'objet des panneaux solaires à l'aide de données aériennes.

## 2 Une méthode pour la détection automatique d'installations : l'apprentissage statistique et les réseaux de neurones convolutifs

Dans cette partie, nous présentons la théorie de l'apprentissage supervisé appliquée aux réseaux de neurones. Ensuite, nous définissons formellement la notation de perceptron, avant de présenter les réseaux de neurones convolutifs. Enfin, nous expliquons les méthodes utilisées en pratique pour améliorer et évaluer les performances des réseaux convolutifs.

### 2.1 La théorie des réseaux de neurones appliquée à un cadre d'apprentissage supervisé

#### 2.1.1 Le risque empirique : un estimateur du risque théorique que l'on souhaite optimiser

Les réseaux de neurones constituent un ensemble d'algorithmes permettant de répondre à des problèmes d'apprentissage supervisé. Ainsi, le modèle s'écrit de façon très générale :

$$Y = h(\mathbf{X}) \quad (1)$$

Où  $\mathbf{X} = (X_1, \dots, X_p)^T \in \mathcal{X} = \mathbb{R}^p$ ,  $Y \in \mathcal{Y}$  et  $h : \mathcal{X} \rightarrow \mathcal{Y}$  est une fonction mesurable appartenant à  $\mathcal{F}$ , la classe de fonctions considérée.  $\mathbf{X}$  est le vecteur de covariables (*features*).  $Y$  est la variable cible (*target*).

Notre travail porte sur la détection d'installation solaire à partir d'images aériennes. On se place donc dans un cadre de *classification supervisée* :  $\mathcal{Y} = \{0, 1\}$ .  $Y$  vaut 1 si l'image comporte

une installation photovoltaïque, 0 sinon. De plus,  $\mathbf{X}$  représente l'intégralité des pixels de l'image ( $p = W \times H \times C$ ). Finalement,  $h$  est un classifieur.

Idéalement, on cherche le classifieur  $h^*$  qui minimise le risque théorique (l'oracle) :  $\mathbb{E}(l(Y, h(\mathbf{X})))$ , où  $l$  est la fonction de perte qui décrit l'écart entre la prédiction de notre modèle et sa vraie valeur. La loi du couple  $(\mathbf{X}, Y)$  n'étant pas connue en pratique, cette quantité n'est pas calculable. Ainsi, cela nous amène à trouver la fonction  $h_{Emp}$  qui minimise le risque empirique, défini à partir d'un n-échantillon  $\mathcal{D}_n = \{(\mathbf{X}_i, Y_i) | i = 1, \dots, n\}$  de loi  $P_{(\mathbf{X}, Y)}$ . Le risque empirique est défini comme  $\hat{R}(h) = \frac{1}{n} \sum_{i=1}^n l(Y_i, h(\mathbf{X}_i))$ .

### 2.1.2 La validation simple : une méthode utilisée pour garantir la consistance du risque empirique

Le classifieur  $h$  est construit à partir de  $\mathcal{D}_n$ . Ainsi, les variables  $(l(Y_i, h(\mathbf{X}_i)))_{1 \leq i \leq n}$  sont fortement corrélées. Dans ce contexte, la loi forte des grands nombres n'est pas applicable : rien ne nous assure que le risque empirique converge presque-sûrement vers le risque théorique. Pour palier à ce problème, nous partitionnons notre échantillon en un échantillon d'apprentissage ( $\mathcal{D}_A$ ) et un échantillon de validation<sup>5</sup> ( $\mathcal{D}_V$ ). Les deux échantillons sont supposés indépendants. Dans le cadre de ce projet, nous avons utilisé 80% de nos données pour l'entraînement et 20% pour la validation. Le classifieur  $h$ , i.e le réseau de neurones, est appris sur l'échantillon d'apprentissage. Les performances de ce classifieur sont évaluées sur l'échantillon de validation. Ainsi, sous réserve que  $P_{(\mathbf{X}, Y)}$  admette un moment d'ordre 1, le risque empirique calculé sur l'échantillon de validation converge presque-sûrement vers le risque théorique.

La validation simple permet aussi de détecter le sur-apprentissage (over-fitting) et le sous-apprentissage (under-fitting). Un classifieur sur-apprend si ses performances sont bonnes sur l'échantillon d'apprentissage, faibles sur d'autres données (son biais est faible mais sa variance est élevée). Typiquement, la loss sur l'échantillon d'apprentissage décroît avec la complexité du modèle alors que celle sur l'échantillon de validation augmente. A l'inverse, on parle de sous-apprentissage pour désigner une situation pour laquelle le classifieur est peu performant sur les données d'apprentissage. Il se généralise aussi peu sur d'autres données (son biais est élevé, sa variance est faible). Cette situation est plausible lorsque la loss sur l'échantillon d'apprentissage et de validation est élevée.

---

5. Un échantillon test peut être aussi utilisé pour comparer les performances de plusieurs prédicteurs différents. N'ayant pas utilisé un tel échantillon, nous utilisons aussi le terme "échantillon test" pour désigner l'échantillon de validation, ce qui est assez courant en pratique.

## 2.2 Du perceptron aux réseaux neuronaux convolutifs

### 2.2.1 Le perceptron, unité de base du réseau de neurones

La brique élémentaire du réseau de neurones est le perceptron. Cet élément transforme  $n$  entrées en une sortie. En particulier, le perceptron effectue une opération non-linéaire. Ainsi, en imbriquant plusieurs perceptrons de sorte à former un réseau de neurones, on peut approximer avec une précision arbitraire n'importe quelle fonction. Une description plus détaillée du perceptron et des réseaux de neurones multicouches se trouve en annexe C.

### 2.2.2 Les réseaux de neurones convolutifs : un ensemble d'algorithmes plus adaptés au traitement d'images

Avant de définir les réseaux de neurones convolutifs, nous commençons par expliquer le principe de la convolution et du pooling.

#### La convolution : une opération algébrique utilisable pour le traitement d'images

##### Principe de la convolution

La convolution est une opération mathématique qui, associe à un vecteur (ou une matrice) et un motif, un nouveau vecteur (ou matrice) dont chaque terme est issu du produit des nombres contenus dans le motif et de certains termes du vecteur (ou de la matrice). La figure 10 illustre cette opération pour une matrice de dimensions  $7 \times 7$  et un motif de dimensions  $3 \times 3$ . Une définition formelle de la convolution et un exemple simple sont présentés en annexes E.1 et E.2.

##### Définition du neurone de convolution

À partir des opérations décrites ci-dessus, on peut définir les neurones de convolution. Un neurone de convolution est construit de la même façon que les neurones décrits dans la section précédente. La principale différence est que les neurones d'une couche ne prennent en entrée qu'un nombre limité de valeurs de la couche précédente, de façon à réaliser le produit de convolution décrit plus haut.

Concrètement, un neurone de convolution réalise une étape de la convolution. Mis bout à bout, l'intégralité des neurones d'une couche réalisent la totalité de la convolution. Les poids du neurone correspondent aux valeurs du motif utilisé. Ainsi, le nombre de poids est égal à la dimension du motif. L'ensemble des poids forment le motif ou noyau. Une représentation schématique d'un neurone de convolution est présentée en annexe E.4.

#### Le pooling : une technique de réduction de dimensions

Le pooling de taille  $k$  est l'application qui à  $M$ , une matrice de taille  $n \times p$  associe  $M'$ , une matrice de taille  $\frac{n}{k} \times \frac{p}{k}$ . Le pooling est fréquemment utilisé dans le traitement des images pour réduire la taille de ces dernières. Les principaux types de pooling sont le max-pooling et l'average pooling. La définition de ces opérations est donnée en annexe E.3.

### 2.2.3 Architecture d'un réseau de neurones convolutif

Les réseaux de neurones convolutifs (CNN) désignent une sous-catégorie de réseaux de neurones spécialement conçus pour traiter des images en entrée. Une image de dimension  $n \times p$  en niveau de gris s'apparente à une matrice de taille  $n \times p$  où chaque coefficient représente l'intensité d'un pixel. Si l'image est en couleur, alors il convient d'ajouter une dimension. L'image s'apparente donc à un tenseur de dimensions  $n \times p \times 3$  (chaque pixel contient 3 composantes : rouge, vert, bleu). Dans le cadre du projet, les images traitées sont en couleur.

Les CNN permettent de traiter chaque pixel d'une image en prenant en compte dans le même temps les pixels qui l'entourent, afin de détecter certains motifs sur ces images, dans notre cas, des panneaux solaires. A la différence des réseaux de neurones basiques, les CNN contiennent des couches de *convolution* et de *pooling*.

Une couche de convolution est composée d'un ensemble de neurones de convolution. Le nombre de neurones composant la couche dépend de la dimension de l'entrée ainsi que du nombre de convolutions appliquées (filtres). Supposons qu'une couche convulsive ait pour entrée  $A = (A_1, \dots, A_k)$  où  $A_i$  est une matrice de dimension  $n \times p$ .  $A$  peut être perçu comme la superposition des différentes couches d'une image. Par exemple, si l'entrée est une image non traitée en couleur,  $A = (A_1, A_2, A_3)$ . Sur cette entrée, on applique  $l$  filtres  $M_1, \dots, M_l$ . Chaque filtre est un motif dont la largeur et la hauteur valent (arbitrairement)  $m$ . Comme on a  $k$  canaux,  $M_i$  est un tenseur de dimensions  $m \times m \times k$ . De façon imagée, chaque filtre  $M_i$  traite simultanément les  $k$  canaux pour retourner  $A * M_i$  de dimensions  $n \times p$ . La couche de convolution réalise donc la transformation suivante :

$$F : A = (A_1, \dots, A_k) \longrightarrow (A * M_1, \dots, A * M_l) \quad (2)$$

Une telle couche peut être représentée schématiquement en annexe E.5.

Finalement, les dernières couches d'un réseau de neurones CNN sont dites "*fully-connected*". Elles correspondent aux couches usuelles d'un réseau de neurones "classique", où chaque entrée est reliée à chaque neurone. On utilise, à minima dans la dernière couche, une fonction d'activation permettant de réaliser la classification comme la sigmoïde.

### 2.2.4 Invariance par translation et avantages des réseaux convolutif

L'invariance par translation est l'un des principaux biais inductifs des CNN. (Mallat [2016]) : dans une couche de convolution, chaque neurone partage les mêmes poids. Le nombre de poids

utilisé correspond à la dimension du noyau. Or, sur une tache de classification, la classe de l'objet ne dépend pas de sa localisation. Ainsi, en comparaison des réseaux de neurones plus usuels, le coût computationnel d'apprentissage d'un CNN est plus faible, sans pour autant sacrifier la performance.

### 2.2.5 Optimisation des paramètres d'un réseau de neurones convolutifs par rétropropagation du gradient et descente de gradient stochastique

Selon le paradigme de la minimisation du risque empirique (2.1.1), le but, en apprenant un réseau de neurones, est de trouver :

$$(W^{*l})_{1 \leq l \leq L}, (b^{*l})_{1 \leq l \leq L} \in \operatorname{argmin}_{(W^l)_{1 \leq l \leq L}, (b^l)_{1 \leq l \leq L}} \hat{R}(h) \quad (3)$$

Par définition, un réseau de neurones constitue une composition de fonctions non-linéaires sur les données. Ainsi, les conditions du premier ordre du programme de minimisation sont non-linéaires en  $(W^l)_{1 \leq l \leq L}$  et  $(b^l)_{1 \leq l \leq L}$ . De plus,  $\hat{R}(h)$  n'est pas nécessairement une fonction convexe en  $(W^l)_{1 \leq l \leq L}$  et  $(b^l)_{1 \leq l \leq L}$ . Ce faisant, la solution du programme peut être un minimum local. Des techniques d'optimisation comme la descente de gradient stochastique (SGD), couplées à la rétropropagation du gradient (Rumelhart et al. [1986]), permettent de contrer ce problème. L'algorithme est détaillé en annexe F. Nous présentons ici les grandes lignes de la méthode.

Le risque empirique peut se réécrire :  $\hat{R}(h) = \frac{1}{n} \sum_{j=1}^M \sum_{i \in B_j} l(h(\mathbf{X}_i), Y_i)$ . Où  $B_1, \dots, B_M$  sont des batchs de données : des sous-échantillons formant une partition de l'échantillon initial. Pour actualiser les poids et les biais d'une couche  $l$  au cours d'une itération de l'algorithme SGD, nous utilisons  $\nabla_l l(h(\mathbf{X}_i), Y_i)$  : le gradient de la loss par rapport aux poids de la couche  $l$ , calculé sur le batch passé. Or,  $\nabla_l l(h(\mathbf{X}_i), Y_i)$  dépend de  $\nabla_{l+1} l(h(\mathbf{X}_i), Y_i)$  qui dépend lui-même de  $\nabla_{l+2} l(h(\mathbf{X}_i), Y_i)$  etc. Ainsi, pour calculer  $\nabla_l l(h(\mathbf{X}_i), Y_i)$  sur un batch  $B_j$  donné, on calcule  $\nabla_{l+1} l(h(\mathbf{X}_i), Y_i), \dots, \nabla_L l(h(\mathbf{X}_i), Y_i)$ . Ceci fait, nous procédons à une nouvelle itération de l'algorithme SGD en passant un nouveau batch de données. A chaque batch, le classifieur  $h$  est entraîné. On calcule le gradient de la loss relativement aux poids actualisés lors de l'étape précédente, sur le nouveau batch considéré. La procédure peut être répliquée en utilisant des epochs. Une epoch correspond à l'utilisation de l'intégralité des batchs pour la descente de gradient stochastique.

Tout comme pour la descente de gradient usuelle, le *learning rate* est un hyperparamètre crucial qui contrôle la vitesse de la descente. D'un côté, en choisissant un learning rate trop petit, on prend le risque de converger trop lentement vers le minimum global et de ne pas l'atteindre. De l'autre, en prenant un learning rate trop grand, on peut diverger rapidement vers des points éloignés du minimum global.

Enfin, on peut aussi utiliser un *momentum*. A chaque actualisation  $t$  des poids, on ajoute une pénalisation  $\beta_t(W_{t-1} - W_{t-2})$  où  $W_{t-1} - W_{t-2}$  représente la "direction" obtenue à l'étape précédente.  $\beta$  est le facteur d'amortissement. En pratique,  $\beta \in \{0, 0.9, 0.99\}$ . Intuitivement, le momentum lisse les variations des poids au cours des itérations.

L'algorithme SGD présente plusieurs avantages, relativement à la descente de gradient usuelle. Premièrement, la descente de gradient stochastique engendre un coût computationnel inférieur puisque l'on passe un batch de données pour actualiser le paramètre d'intérêt lors d'une itération. De plus, l'estimateur du gradient utilisé est bruyant. L'usage de bruit limite les risques de converger vers un minimum local. Enfin, l'utilisation de batchs permet de faire un meilleur usage des capacités de calcul. En effet, il est possible de distribuer les batchs entre les différentes machines de calcul pour agréger les gradients résultants.

## 2.3 Évaluer et améliorer les performances d'un réseau de neurones convolutif

### 2.3.1 Métriques d'évaluation des performances d'un réseau de neurones dans le cadre d'un problème de classification binaire

Une multitude de métriques peuvent être utilisées pour évaluer les performances d'un réseau de neurones convolutif. Nous distinguons les métriques probabilistes des métriques de classification.

#### L'entropie croisée binaire : une métrique probabiliste limitée pour les échantillons déséquilibrés

Soient  $p_i = P(Y_i = 1)$  et  $\hat{p}_i = \hat{P}(Y_i = 1)$ . L'entropie croisée binaire (ou log-loss) est définie par :

$$BCE = -\frac{1}{n} \sum_{i=1}^n (y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i)) \quad (4)$$

La log-loss est la fonction de perte utilisée dans le cadre de ce projet. Elle mesure la différence entre deux distributions de probabilités. C'est la référence lorsque l'on construit un réseau de neurones pour répondre à un problème de classification. Notons que la métrique peut aussi être adaptée pour la classification multi-classes. Néanmoins, elle peut être fallacieuse sur des échantillons déséquilibrés.

La métrique a été utile au cours des phases d'apprentissage des réseaux afin de contrôler le sous-apprentissage et le sur-apprentissage.

#### Les métriques de classification comme mesures des performances globales

Les métriques usuelles de classification sont fréquemment employées pour évaluer le pouvoir prédictif des modèles de détection de panneaux photovoltaïques. Chacun d'elle répond à un

besoin spécifique. Par exemple, puisque nous nous intéressons, entre autre, à la manière dont les images comportant un panneau sont classées, le recall est une métrique pertinente. Comme mentionné par Kausika et al. [2021b], il n'est pas rare de travailler, comme dans notre cas, à partir d'échantillons déséquilibrés lors du déploiement de modèles de deep learning visant à faire de la classification d'installations photovoltaïques. Ainsi, les métriques présentées, en particulier la log-loss et l'accuracy, doivent être interprétées avec précaution. Une définition formelle des métriques utilisées est donnée en annexe G.

### 2.3.2 Le rôle des data augmentations, du dropout et du weight decay dans la réduction du sur-apprentissage

L'usage de *data augmentations* est une pratique courante lors de l'apprentissage d'un réseau de neurones convolutif. Intégrer des data augmentations consiste à ajouter de façon artificielle de nouvelles données dérivées de l'échantillon d'apprentissage. Les principales data augmentations sont la normalisation, le redimensionnement, les rotations, les translations, les crops, le color jitter, l'ajout de bruit. Premièrement, les data augmentations permettent de standardiser les images en termes de dimensions, moyenne et écart-type des pixels. Ce faisant, on peut facilement adapter le jeu de données à la structure des différents CNN utilisés. De plus, ces manipulations permettent d'avoir un échantillon d'apprentissage plus conséquent. Les data augmentations sont donc utiles lorsque l'échantillon d'apprentissage est composé de peu d'observations. Enfin, en transformant les images, on obtient un échantillon plus diversifié.

La finalité des data augmentations est donc de réduire le sur-apprentissage et de faciliter le déploiement du modèle utilisé sur des données différentes de celles utilisées pour l'apprentissage. Cependant, ces manipulations doivent être utilisées avec parcimonie si l'on souhaite conserver certaines informations relatives aux images considérées. Au cours de ce projet, nous avons utilisé les data augmentations suivantes : resize, conversion en tenseur, normalisation, colorjitter et random crop. Nous définissons ces dernières en annexe H et présentons leur usage. Sur les images à dispositions, les panneaux photovoltaïques pointent toujours vers le sud. Ainsi, nous avons fait le choix de ne pas intégrer de translation et de rotations pour préserver cette information.

Par ailleurs, il est aussi possible d'intégrer des couches dites de *dropout* dans un réseau de neurones. Le dropout est une méthode de régularisation introduite en 2014 (Srivastava et al. [2014]), dont le principe est le suivant. A chaque epoch, on désactive certains neurones du réseau aléatoirement, selon une certaine probabilité  $p$  définie. Ainsi, à chaque itération, le modèle apprend selon une configuration différente de neurones. On réduit donc le sur-ajustement sur les données d'entraînement, de sorte à limiter le sur-apprentissage. Notons que le dropout est actif uniquement durant l'entraînement. Lors de la phase de validation, les poids de chaque neurone sont multipliés par  $p$ .

Finalement, toujours dans une perspective de réduire le sur-apprentissage, il est possible d'ajouter un *weight decay* dans la fonction de perte, une pénalité dépendant des poids. Cette dernière est généralement de la forme  $\lambda \sum_i w_i^2$  où  $\lambda$  est petit.

### **2.3.3 Le transfer learning : une méthode fréquemment utilisée pour limiter le sur-apprentissage et le coût computationnel**

Le transfer learning est un ensemble de méthodes visant à transférer les connaissances acquises à partir de la résolution de problèmes donnés pour traiter un autre problème. En deep learning, le transfer learning consiste à utiliser des modèles pré-entraînés pour extraire des features. A partir de là, plusieurs stratégies sont possibles. Premièrement, il est possible d'utiliser directement le modèle pré-entraîné pour réaliser les tâches de classification ou de régression désirées. A l'inverse, il est possible d'entraîner l'intégralité du réseau importé, à partir des poids pré-entraînés. Finalement, on peut choisir d'entraîner exclusivement certaines couches du réseau.

L'utilisation des poids pré-entraînés dans le cadre du transfert learning présente plusieurs avantages. Dans un premier temps, cela permet de réduire considérablement les temps de calcul lors de la phase d'apprentissage. De plus, on limite les risques d'atteindre un minimum local lors de l'utilisation de la SGD. Par ailleurs, on peut utiliser des modèles de réseaux de neurones, même si notre échantillon d'entraînement est petit. Enfin, on réduit le sur-apprentissage dans la mesure où les poids utilisés sont partiellement entraînés sur une base de données qui n'est pas la nôtre. Pour ces différentes raisons, le transfert learning est une pratique usuelle. Reste à contrôler le nombre de couches à entraîner pour ne pas tendre vers le sous-apprentissage.

## **3 Un large éventail de données en amont de la construction du réseau de neurones convolutif**

La base de données sur laquelle nous avons déployé nos modèles d'apprentissage statistique est issue d'un article Kasmi et al. [2022] publié dans le journal *Scientific Data*. Elle a été constituée en plusieurs temps.

Les auteurs ont tout d'abord, avec l'aide de l'association à but non lucratif BDPV<sup>6</sup>, recensé les coordonnées et les caractéristiques de près de 28 000 installations photovoltaïques de particuliers en France.

Ensuite, grâce à ces coordonnées, les auteurs ont récupéré des *thumbnail*<sup>7</sup> via l'API de Google Earth Engine (GEE) et, pour leur seconde campagne d'annotation, Kasmi et al. ont extrait leurs *thumbnail* du géoportail de l'Institut Géographique National (IGN), où un pixel représente près de 0.2m au sol, contre 0.1m au sol par pixel pour les *thumbnail* Google. Point important : après réduction des *thumbnail* les images sont de résolution 400x400 pixels, ceci, comme on le verra plus tard, a conditionné notre manière d'appréhender le sujet.

---

6. Association citoyenne aidant des particuliers à évaluer et monitorer la production de leur installation photovoltaïque

7. Un *thumbnail* est une vignette aux dimensions réduites extraite d'un *raster*, c'est-à-dire une matrice de données géolocalisées.

Du fait d'approximations initiales dans les coordonnées des installations dans la base de données fournie par BDPV, certaines images ne comportaient pas de panneau photovoltaïque. Pour attribuer un label à ces quelque 42 000 images, les auteurs ont lancé une campagne d'attribution de label participative (ou *crowdsourcing*) qui se décomposait en deux phases : dans un premier temps, le participant indique si l'image contient ou non un panneau solaire et, dans un second temps, il détourne le panneau (pour obtenir des masques indispensables à la segmentation des panneaux).

Dans le cadre du projet, nous avons travaillé sur les images pour lesquelles le *thumbnail* a été généré à l'aide de l'API de GEE, soit 28 634 images. 80 % des images, soit 22 907, ont été utilisées pour l'apprentissage des CNN, les 20 % restant (5727 images) ont servi à la validation. Dans l'échantillon d'apprentissage, 53 % des images (soit 12 202) comportent une installation solaire. Ce nombre s'élève à 54 % dans l'échantillon test. Ainsi, nos données sont déséquilibrées, ce qui, comme nous le verrons par la suite, demande une vigilance particulière lors de l'interprétation des résultats.

## 4 Application de la théorie des réseaux de neurones convolutifs pour la détection d'installations solaires

Au cours de ce projet, nous avons utilisé deux architectures CNN : le LeNet5 et le ResNet18. Afin d'entraîner et de tester ces modèles, nous avons utilisé l'infrastructure *Cloud* de l'INSEE mise à la disposition des élèves de l'ENSAE. Ainsi, pour accélérer l'entraînement du ResNet18 nous avons pu utiliser deux GPU<sup>8</sup> NVIDIA et près de 50 giga-octets de mémoire vive. Nous avons, pour ce faire, indiqué la marche à suivre sur la plateforme de l'INSEE et créé plusieurs notebooks Jupyter sur la page Github du projet, afin qu'il soit entièrement reproductible.

### 4.1 Le LeNet5 : un réseau peu complexe procurant des résultats satisfaisants

#### 4.1.1 Architecture du LeNet5

Le LeNet5 est un CNN composé de cinq couches de neurones, dont trois convolutives (LeCun et al. [1998]). Ce réseau prend en entrée des images de dimension  $28 \times 28$ . Nous avons modifié la dernière couche du réseau original pour répondre à notre problème spécifique de classification binaire. En particulier, la dernière couche comporte une fonction sigmoïde afin de retourner des probabilités prédites. Le seuil de classification est de 0,5. De plus, nous avons ajouté quatre couches de dropout. L'architecture détaillée du réseau est présentée en annexe I.

---

8. De l'anglais *Graphic Processing Unit*, les processeurs graphiques sont couramment utilisés pour l'entraînement de modèles en apprentissage statistique, ils permettent, en parallélisant les calculs de plusieurs couches du réseau de neurones en même temps, de considérablement réduire le temps d'apprentissage et de validation.

#### **4.1.2 Choix des data augmentations : le ColorJitter et les crops aléatoires ont été retenus sur le modèle final**

Le coût computationnel associé au LeNet5 est moindre, en comparaison d'autres réseaux. Ce faisant, il a été possible de tester plusieurs configurations présentées en annexe J. Pour le modèle final, nous avons choisi de redimensionner nos images en dimensions  $28 \times 28$ , sur l'échantillon d'apprentissage et le test. Sur l'échantillon d'apprentissage, nous appliquons un ColorJitter où seule la luminosité est modifiée de façon aléatoire, dans un intervalle de valeurs compris entre 0,5 et 1,5. De cette manière, nous lissons les différences d'ensoleillement sur les images, sans pour autant omettre les installations. Nous utilisons aussi des crops aléatoires de taille 24. Nous avons fait en sorte de conserver environ 75 % de l'image afin de réduire les risques de disparition du panneau. Après avoir de nouveau redimensionné les images en  $28 \times 28$ , nous les convertissons en tenseurs, puis nous les normalisons. Les deux dernières opérations ont aussi été réalisées sur l'échantillon d'apprentissage.

#### **4.1.3 Le faible coût computationnel d'apprentissage permet d'hyper-paramétriser le LeNet5 en un temps fini**

Afin de limiter les variations des poids lors de l'apprentissage, un momentum a été utilisé lors de la SGD. Le modèle final a été hyper-paramétrisé selon la méthode *Randomised Grid Search* (pratique conventionnelle depuis les résultats de Bergstra and Bengio [2012]). Plutôt que de tester toutes les combinaisons possibles d'hyper-paramètres, nous testons 30 combinaisons aléatoires sur l'échantillon d'apprentissage, partitionné selon une validation croisée 5-blocs, afin de trouver les hyper-paramètres faisant partie des 10 % les plus performants, avec une confiance de 95 %. Au préalable, nous avons optimisé le modèle par rapport au F2-score, dans le but de détecter au mieux les installations existantes. De façon assez logique, le nombre de faux positifs résultant est très important. Les résultats obtenus n'étant pas concluants [J.2], la métrique optimisée a donc été le F1-score. Les valeurs des hyper-paramètres obtenus sont présentées en annexe J.2.

#### **4.1.4 En dépit d'une réduction significative de la résolution des images, le LeNet5 procure des résultats satisfaisants**

Comme en témoigne le graphique présentant l'évolution des métriques au fil des epochs [20], le LeNet5 ne semble pas sur-apprendre ou sous-apprendre. Premièrement, la loss diminue au fil des epochs sur l'échantillon d'apprentissage et sur l'échantillon de validation. A l'inverse, les valeurs prises par les métriques de classification augmentent au fil des epochs, peu importe l'échantillon considéré. Enfin, de façon générale, n'importe quelle métrique, considérée à n'importe qu'elle epoch, est meilleure sur l'échantillon de validation. Les résultats du LeNet5 sont synthétisés dans le tableau 4.5.

Métrique	Apprentissage	Validation
BCE	0,402	0,39
F1-score	0,839	0,852
Recall	0,849	0,896
F2-score	0,844	0,855
Accuracy	0,831	0,837
Precision	0,837	0,819

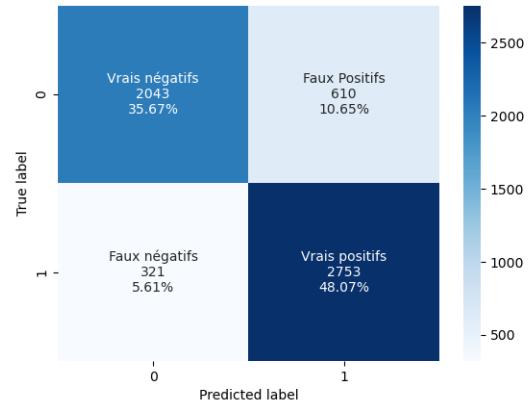


TABLE 1 – Métriques calculées sur l'échantillon d'apprentissage et de test, et matrice de confusion

L'intégralité des métriques de classification prennent une valeur supérieure à 0,8 sur l'échantillon de validation. Ainsi, de façon très globale, le modèle est performant pour la classification d'images aériennes. Premièrement, la valeur du F1-score (0,852) illustre le fait que, de façon générale, le modèle est bon pour détecter les installations solaires existantes. Avec un F2-score et un recall valant respectivement 0,855 et 0,896, il est clair que le modèle omet peu de panneaux solaires (89, 56 % des installations ont été détectées contre 87,2 % sans hyper-paramétrisation). Cependant, la precision, dont la valeur est inférieure (0,819) témoigne du fait qu'un nombre non négligeable de structures sont détectées à tort comme installations photovoltaïques. En effet, parmi l'ensemble des images ne comportant pas de panneau, 23 % ont été prédites comme en comportant un (ce chiffre s'élevait à 21,1 % lorsque réseau n'était pas hyper-paramétrisé).

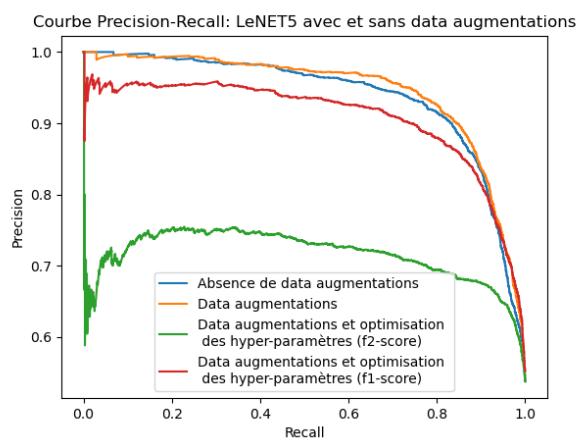
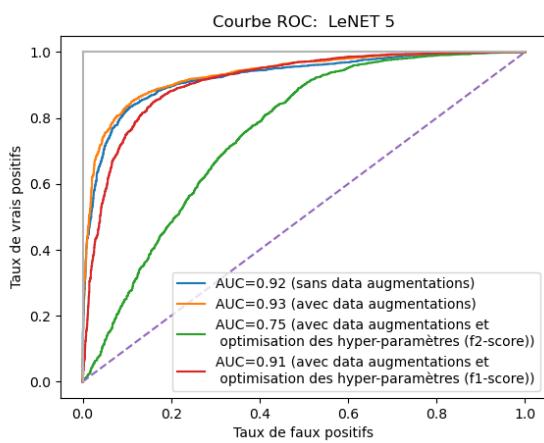


FIGURE 1 – Courbes ROC et precision-recall du LeNet5, selon les différentes configurations testées

Comme en témoigne la courbe precision-recall (figure 4.1.4), les gains de performance en

matière de détection des installations, apportés par l'hyper-paramétrisation du réseau, ne compensent pas entièrement la dégradation des performances en ce qui concerne l'absence de détection de panneau pour des images n'en comportant pas. Ainsi, le pouvoir prédictif global du modèle optimisé est moins bon (l'AUC diminue de deux points). En clair, bien que le LeNet5 soit performant pour détecter l'intégralité des installations solaires, des améliorations peuvent être envisagées pour limiter le nombre de faux positifs.

## 4.2 Le ResNet18 : un modèle plus complexe, adapté pour les images en grandes dimensions

Le ResNet18, second modèle que nous avons choisi de déployer, a une architecture plus complexe que le LeNet5. Il est composé de 72 couches de neurones dont 18 convolutives et prend en entrée des images de dimension  $224 \times 224$ .

L'approche des réseaux de neurones résiduels (ResNet) a été popularisée en 2015 par les travaux très influents de He et al. [2015]. Les modèles ResNet diffèrent de CNN classiques comme les LeNet car ils comportent un nombre de couches important et permettent de résoudre le problème de généralisation auxquels sont confrontés ces mêmes CNN lorsque le nombre de couches intermédiaires augmente (ou le problème d'évanescence des gradients ou *vanishing gradients*<sup>9</sup>) en ajoutant une liaison résiduelle entre chaque couche de convolution.

Du fait du nombre important de paramètres dans le réseau (11 689 512) et pour à nouveau réduire la probabilité de rencontrer le problème des gradients évanescents, nous avons importé les poids initiaux d'un modèle entraîné sur la base de données ImageNet et qui sont issus des travaux de He et al. [2015]. Nous avons également modifié la dernière couche du réseau pour faire de la classification binaire. Enfin, au vu de la complexité du modèle, nous avons ajouté un *weight decay* valant 0,0001 dans la fonction de perte, ainsi qu'un momentum de 0,9, afin de limiter un potentiel sur-apprentissage.

## 4.3 Utiliser des *crops* et actualiser les labels : la méthode de l'*auto-crop*

Lors de l'apprentissage de LeNet5, nous avons utilisé des *crops*. Bien que ces derniers soient parcimonieux, le risque de voir l'installation solaire disparaître de l'image subsiste. Afin de tirer avantage de cette data augmentation, sans prendre un tel risque, nous avons utilisé une variante des *crops* appelée *autocrop* pour entraîner le ResNet18.

L'*autocrop* est une data augmentation s'apparentant à un crop usuel, à la différence que les labels sont continuellement actualisés en fonction du résultat produit. En clair, si l'installation

---

9. C'est une problématique récurrente lorsque l'on entraîne un réseau de neurones classique : le gradient de la fonction de perte peut être trop faible pour que l'algorithme de la descente de gradient permette au réseau d'apprendre (via la rétropropagation), ceci avant même que la fonction de perte ait décrû. Voir par exemple les travaux de Bengio et al. [1994] sur le sujet.

photovoltaïque disparaît suite à l'usage de la transformation, le label de l'image est actualisé en 0. Dès lors, la perte d'information est considérablement réduite. Nous avons ainsi utilisé cette data augmentation, avec une taille de 224, sur le train comme le test, lors de l'apprentissage des deux ResNet18 présentés. Le choix d'utiliser une telle data augmentation sur l'échantillon de validation se justifie par la volonté de garder une cohérence dans les données en termes de résolution d'image (redimensionner, sans crop, les images de l'échantillon de validation, induirait une perte de résolution absente sur l'échantillon d'apprentissage, problème qui se posera pour la mise en production du modèle).

Ceci permet également d'avoir un échantillon d'entraînement le moins biaisé possible, où la différence entre images avec panneaux et les images sans panneaux est simplement due à la présence d'un panneau et non aux autres facteurs présents dans l'image comme l'environnement, à l'architecture, la végétation, etc.<sup>10</sup>

#### 4.4 Apprendre exclusivement les couches fully connected du ResNet18 pré-entraîné : une option computationnellement avantageuse produisant des résultats moyens

Dans un premier temps, nous testons les performances du ResNet18 entraîné sur la base de données ImageNet. Concrètement, au cours de la phase d'apprentissage, nous entraînons uniquement les couches fully connected du réseau sur nos données.

A la différence du LeNet5 et du fait de son architecture complexe, l'hyper-paramétrisation du ResNet18 demande beaucoup de ressources computationnelles. Ainsi, cette étape n'a pas été réalisée. Pour choisir le learning rate, nous avons utilisé la méthode *cyclical learning rates* (Smith [2015]). Cette méthode permet de trouver un learning rate optimal sans entraîner le modèle plusieurs fois. Graphiquement, on cherche le learning rate situé avant le minimum, pour lequel le taux d'accroissement de la loss est le plus élevé. D'après le graphique présenté en annexe K, on voit que ce point correspond à un learning rate de 0,001.

Enfin, nous fixons une taille de batch à 32 et un nombre d'epochs à 30. Comme en témoigne le graphique 3, cette configuration permet un apprentissage correct du réseau. Pour plus de clarté, nous présentons uniquement l'évolution de la log-loss sur l'échantillon train et test.

---

10. *i.e.* le réseau apprend exactement l'effet de l'ajout d'un panneau sur une image.

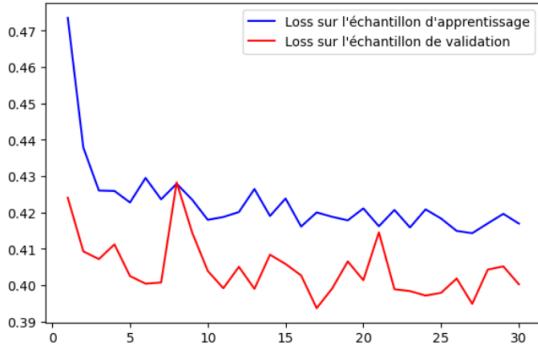


FIGURE 2 – Evolution de la log-loss au fil des epochs sur le ResNet18 pré-entraîné (entraînement des couches fully-connected)

Métrique	Apprentissage	Validation
BCE	0,4	0,41
F1-score	0,83	0,83
Recall	0,86	0,86
F2-score	0,85	0,85
Accuracy	0,82	0,81
Precision	0,8	0,79

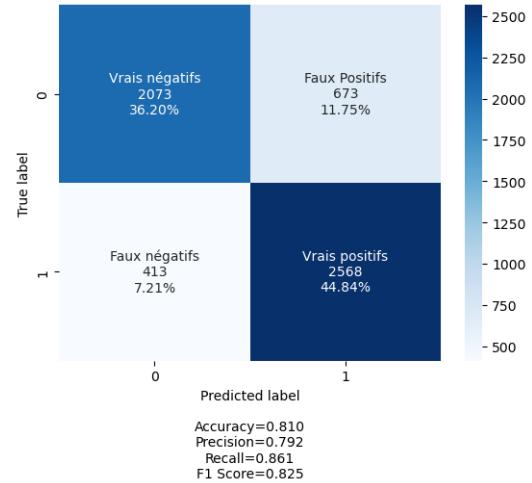


TABLE 2 – Métriques calculées sur l'échantillon d'apprentissage et de test, et matrice de confusion

En règle générale, les résultats obtenus avec le ResNet18, entraîné exclusivement à partir de nos données sur les couches fully connected, sont moins bons en comparaison de ceux obtenus avec le LeNet5 hyper-paramétré. On remarque par exemple, que le F1-score est plus faible de 0,022 points. Le recall et la precision ont aussi diminué. Au global, le réseau est moins bon pour classifier correctement les images. En effet, le nombre de vrais positifs a diminué de 3,23 points, le nombre de faux positifs et de faux négatifs ont augmenté respectivement de 1,1 points et 1,6 points. Seul le nombre de vrais négatifs a légèrement augmenté, mais cette hausse ne compense pas la dégradation des performances générales. En clair, il n'est pas pertinent d'utiliser un CNN, pour lequel les couches convolutives ont été simplement entraînées sur une base aussi diversifiée qu'ImageNet, dans le but de détecter des installations solaires. Face à ce constat, nous décidons d'apprendre l'intégralité des couches du modèle.

## 4.5 Entraîner l'ensemble des couches du ResNet18 pré-entraîné s'avère être une stratégie efficace en dépit du coût computationnel élevé

Du fait du nombre important de données à notre disposition et des résultats peu probants de la sous-section précédente nous entraînons l'intégralité des couches du ResNet18 après avoir chargé les poids pré-entraînés de la base ImageNet. Nous ajoutons une couche de dropout entre chaque bloc de convolution. Les probabilités associées sont celles utilisées sur le LeNet5 hyper-paramétrisé [J.2]. Comme pour l'entraînement du modèle précédent, nous utilisons des batchs de 32 et choisissons un learning rate de 0,001, avec la méthode de Smith [2015]. Après plusieurs essais, nous avons constaté qu'après 30 epochs d'entraînement qu'il n'y avait pas d'amélioration significative de la fonction de perte pour l'échantillon de validation et que la variabilité de la fonction de perte augmentait, signe de surapprentissage. Nous avons donc réduit l'entraînement du modèle à 30 epochs.

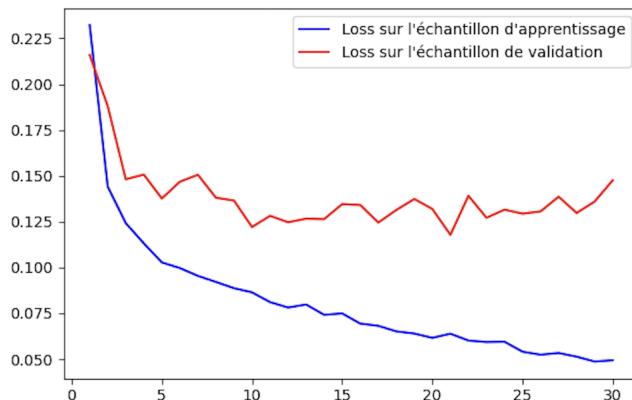


FIGURE 3 – Evolution de la log-loss au fil des epochs sur le ResNet18 pré-entraîné (entraînement de toutes les couches)

Métrique	Apprentissage	Validation
BCE	0,06	0,14
F1-score	0,98	0,96
Recall	0,99	0,98
F2-score	0,99	0,97
Accuracy	0,98	0,96
Precision	0,96	0,94

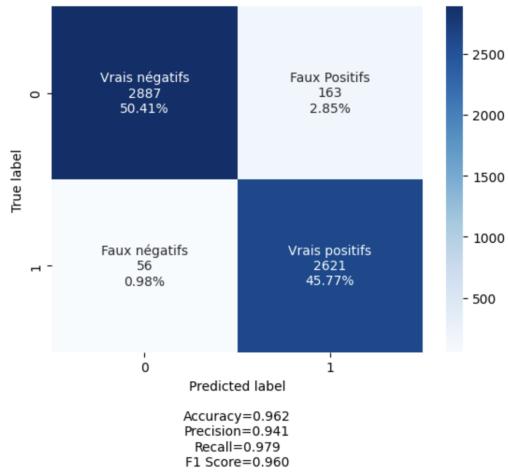


TABLE 3 – Métriques calculées sur l'échantillon d'apprentissage et de test, et matrice de confusion

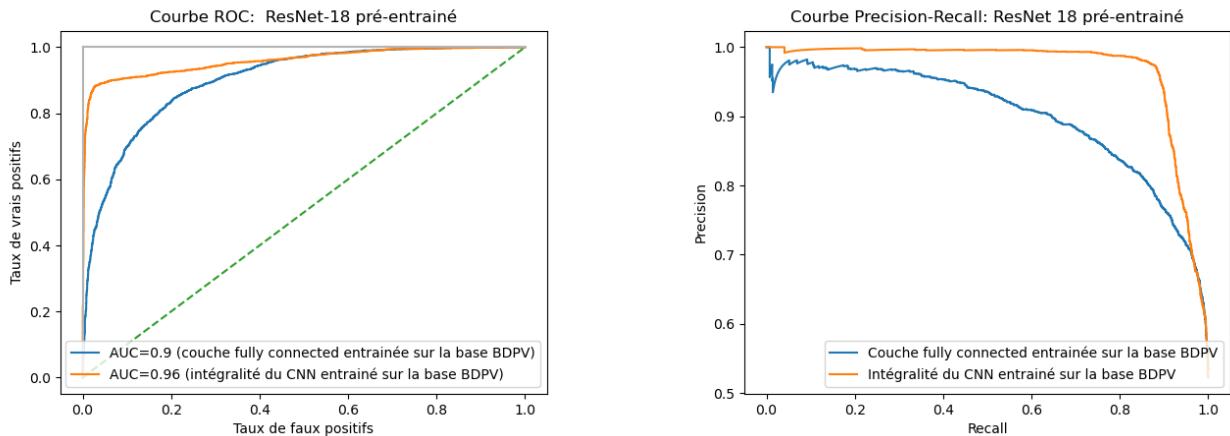


FIGURE 4 – Courbes ROC et precision-recall du ResNet18, selon le nombre de couches entraînées

On note d'emblée que toutes les métriques ont cru comparées à celles du modèle précédent, ce qui nous permet de dire que ce modèle est le meilleur classifieur que nous ayons entraîné. Le nombre de faux positifs reste plus important que le nombre de faux négatifs (2,85% contre 0,98%) mais a presque été divisé par trois, comparé au nombre de faux positifs du ResNet fully connected. On constate également que le pouvoir prédictif, représenté par l'AUC, du modèle entraîné dans sa totalité est meilleur que lorsqu'on entraîne uniquement les dernières couches du ResNet, ce qui est plutôt rassurant. On retrouve à nouveau le problème de faux-positifs qui s'était présenté également pour le LeNet5 (et récurrent dans la littérature) avec la precision plus faible que le recall. Les performances du modèle sont donc très encourageantes et le léger problème de faux positifs cohérent avec l'état de l'art.

## 4.6 Le ResNet18 entraîné entièrement sur nos données d'apprentissage est le modèle qui sépare au mieux les données de validation

Finalement, afin de comparer visuellement le pouvoir discriminant des modèles utilisés, nous procérons à un T-SNE (der Maaten and Hinton [2008]). Contrairement à l'ACP, le T-SNE est une technique de réduction de dimension stochastique qui permet d'identifier les groupes d'observations proches (là où l'ACP permet de visualiser la structure générale des données). Une explication formelle de l'algorithme est présentée en annexe L. Voici la démarche adoptée.

Afin de comprendre comment les modèles ont classifié les images, nous injectons ces dernières dans les couches entraînées de chaque modèle, sans aller jusqu'à la dernière couche de classification. Pour le premier ResNet 18 présenté, nous avons injecté les images uniquement dans les couches fully connected. Ainsi, nous pouvons voir visuellement si les transformations appliquées par le réseau sont efficaces pour séparer les images avec panneau de celles sans.

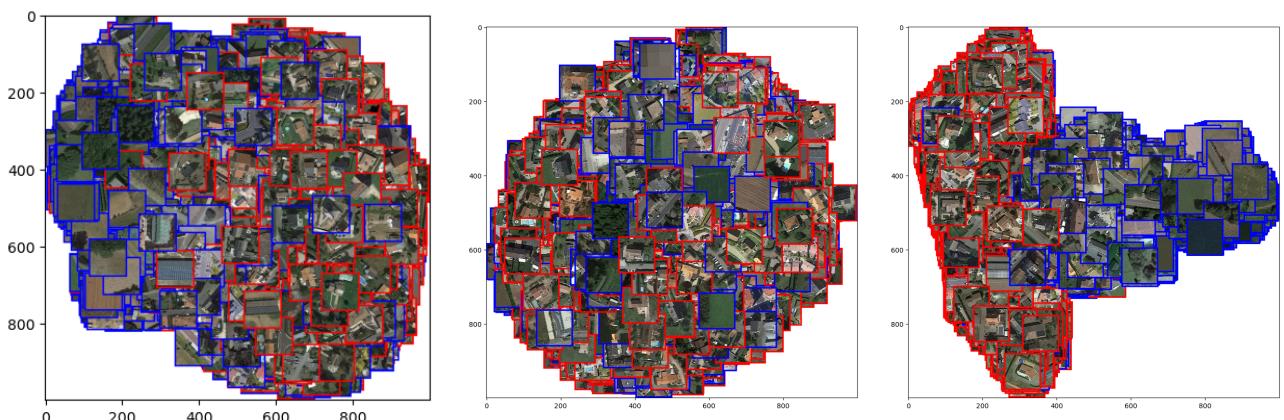


FIGURE 5 – Visualisation T-SNE des images contenues dans l'échantillon de validation (perplexity de 50). A gauche : LeNet5 ; au centre : ResNet18 : entraîné sur les couches fully connectées ; à droite : ResNet18 entraîné sur toutes les couches. Une image encadrée en rouge contient une installation solaire. Une image encadrée en bleu n'en contient pas.

Visuellement, on constate que le ResNet18, pour lequel les couches convolutives ont été entraînées sur nos données, sépare clairement les images sans installation (à droite), de celles avec installations (à gauche). De plus, parmi les images ne comportant pas de panneaux photovoltaïques, on peut clairement distinguer celles de prairies et forêts (à l'extrême-droite) des images d'habitations (plus au centre, car similaires aux images avec installation).

Le LeNet5 distingue nettement les photos d'habitats (à droite) des images rurales (à l'extrême-gauche). Cependant, il peine à identifier clairement les différences entre images d'infrastructures comportant ou non un panneau.

Finalement, aucune structure particulière ne semble se dégager du nuage d'images associé au ResNet 18, pour lequel les couches fully connected ont été entraînées sur nos données. Ce constat visuel coïncide donc avec les résultats obtenus.

# Conclusion

Construire un modèle de réseaux de neurones convolutifs pour la détection d'installations solaires ne peut se faire sans une définition rigoureuse des concepts utilisés. Ainsi, un travail important de recherche a été réalisé avant l'implémentation du LeNet5 et du ResNet18. Les outils exposés dans les travaux de recherche durent souvent être adaptés en fonction de notre sujet. Par exemple, nous n'avons pas appliqué de rotations d'images pour préserver l'orientation des panneaux alors que ces dernières font partie des data augmentations largement utilisées.

La connaissance approfondie des outils utilisés est d'autant plus importante dans le cadre de notre travail, dans la mesure où les réseaux de neurones sont des algorithmes coûteux computationnellement. Cette contrainte nous a conduit à être vigilant sur la paramétrisation des modèles et sur le choix des data augmentations.

La découverte des auto-crops s'est faite tardivement, après l'hyper-paramétrisation du LeNet5. C'est la raison pour laquelle, cette data augmentation a été utilisée exclusivement sur le ResNet18. Sans conteste, le fait d'actualiser les labels, sur le train comme sur le test, en fonction des résultats du crop participe à l'obtention des très bons résultats que nous avons eus. Ce choix a été fait pour garantir une cohérence dans la résolution des images d'apprentissage et de validation. Cependant, il convient de noter que les résultats seraient potentiellement moins bons sur un modèle déployé à grande échelle. En effet, en pratique, les images pour lesquelles on souhaiterait obtenir des prédictions sont redimensionnées sans crop<sup>11</sup> selon les pré-requis du CNN. D'où une perte de résolution (relativement aux images d'apprentissage) pouvant être à l'origine de performances moins importantes. A grande échelle, cela pourrait conduire à une sous estimation de la production électrique diffuse.

Dans le cadre de ce projet, nous avons exclusivement travaillé sur un problème de classification binaire. Or, il existe des CNN plus complexes permettant de faire de la segmentation d'images<sup>12</sup> (on attribue à chaque pixel un label particulier) ou de la localisation d'objets, en l'occurrence d'installations photovoltaïques. De façon encore plus poussée, des techniques pour estimer l'orientation 3D des panneaux (i.e leur orientation Nord-Sud-Est-Ouest et leur inclinaison) ont été développées (Mayer et al. [2022]). En plus des images aériennes, ces méthodes s'appuient sur des modèles "3D city" : des modèles numériques de zones urbaines représentant à l'échelle 3D les surfaces, les sites, les bâtiments, la végétation, les infrastructures, qui composent la zone modélisée.

Il aurait également pu être intéressant de tester notre modèle sur des images aériennes de pays frontaliers, provenant d'une autre base de données, pour observer le potentiel de généralisation de notre ResNet18 entraîné et de ses performances, généralisation souvent ardue en pratique comme reporté dans la littérature (Jong et al. [2020]).

---

11. Etant donné que nous n'aurons pas leur label.

12. C'est le cas de U-Net (cf. Ronneberger et al. [2015]) par exemple

## Références

- F. R. Arnaud Bodin. *Deepmath : mathématiques des réseaux de neurones.* 2021. ISBN 979-8692672872.
- Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2) :157–166, Mar. 1994. ISSN 1941-0093. doi : 10.1109/72.279181. Conference Name : IEEE Transactions on Neural Networks.
- J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(null) :281–305, Feb. 2012. ISSN 1532-4435.
- G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4) :303–314, 1989.
- L. V. der Maaten and G. E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9 :2579–2605, 2008.
- G. Gust, C. M. Flath, T. Brandt, P. Ströhle, and D. Neumann. Bringing analytics into practice : Evidence from the power sector. In P. Ågerfalk, N. Levina, and S. S. Kien, editors, *Proceedings of the International Conference on Information Systems - Digital Innovation at the Crossroads, ICIS 2016, Dublin, Ireland, December 11-14, 2016*. Association for Information Systems, 2016. ISBN 978-0-9966831-3-5. URL <http://aisel.aisnet.org/icis2016/Practice-OrientedResearch/Presentations/8>.
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arxiv 2015. *arXiv preprint arXiv :1512.03385*, 14, 2015.
- T. D. Jong, S. Bromuri, X. Chang, M. Debusschere, N. Rosenski, C. Schartner, K. Strauch, M. Boehmer, and L. Curier. Monitoring spatial sustainable development : semi-automated analysis of satellite and aerial images for energy transition and sustainability indicators, 2020.
- G. Kasmi, L. Dubus, P. Blanc, and Y.-M. Saint-Drenan. Deepsolar tracker : towards unsupervised assessment with open-source data of the accuracy of deep learning-based distributed pv mapping. *arXiv preprint arXiv :2207.07466*, 2022.
- B. B. Kausika, D. Nijmeijer, I. Reimerink, P. Brouwer, and V. Liem. Geoai for detection of solar photovoltaic installations in the netherlands. *Energy and AI*, 6 :100111, 2021a. ISSN 2666-5468. doi : <https://doi.org/10.1016/j.egyai.2021.100111>. URL <https://www.sciencedirect.com/science/article/pii/S2666546821000604>.
- B. B. Kausika, D. Nijmeijer, I. Reimerink, P. Brouwer, and V. Liem. Geoai for detection of solar photovoltaic installations in the netherlands. *Energy and AI*, 6 :100111, 2021b. ISSN 2666-5468. doi : <https://doi.org/10.1016/j.egyai.2021.100111>. URL <https://www.sciencedirect.com/science/article/pii/S2666546821000604>.

- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.
- S. Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A : Mathematical, Physical and Engineering Sciences*, 374(2065) :20150203, 2016.
- J. M. Malof, K. Bradbury, L. M. Collins, and R. G. Newell. Automatic detection of solar photovoltaic arrays in high resolution aerial imagery. *Applied Energy*, 183 :229–240, 2016. ISSN 0306-2619. doi : <https://doi.org/10.1016/j.apenergy.2016.08.191>. URL <https://www.sciencedirect.com/science/article/pii/S0306261916313009>.
- K. Mayer, B. Rausch, M.-L. Arlt, G. Gust, Z. Wang, D. Neumann, and R. Rajagopal. 3d-pv-locator : Large-scale detection of rooftop-mounted photovoltaic systems in 3d. *Applied Energy*, 310 :118469, 2022. ISSN 0306-2619. doi : <https://doi.org/10.1016/j.apenergy.2021.118469>. URL <https://www.sciencedirect.com/science/article/pii/S0306261921016937>.
- B. Rausch, K. Mayer, M. Arlt, G. Gust, P. Staudt, C. Weinhardt, D. Neumann, and R. Rajagopal. An enriched automated PV registry : Combining image recognition and 3d building data. *CoRR*, abs/2012.03690, 2020. URL <https://arxiv.org/abs/2012.03690>.
- O. Ronneberger, P. Fischer, and T. Brox. U-net : Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015 : 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III* 18, pages 234–241. Springer, 2015.
- RTE. Rte bilan électrique 2020. <https://bilan-electrique-2020.rte-france.com/production-production-totale/>.
- RTE. Futurs énergétiques 2050 : rapport complet. <https://www.rte-france.com/analyses-tendances-et-prospectives/bilan-previsionnel-2050-futurs-energetiques>, 2022.
- D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088) :533–536, 1986.
- O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115 :211–252, 2015.
- L. N. Smith. Cyclical Learning Rates for Training Neural Networks. *arXiv e-prints*, art. arXiv :1506.01186, June 2015. doi : 10.48550/arXiv.1506.01186.
- N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15 :1929–1958, 2014.

J. Yu, Z. Wang, A. Majumdar, and R. Rajagopal. DeepSolar : A machine learning framework to efficiently construct a solar deployment database in the united states. *Joule*, 2(12) : 2605–2617, 2018. ISSN 2542-4351. doi : <https://doi.org/10.1016/j.joule.2018.11.021>. URL <https://www.sciencedirect.com/science/article/pii/S2542435118305701>.

## Annexes

### A Rythme d'installation du photovoltaïque en France et en Europe

**Parc total photovoltaïque et production d'électricité annuelle en France**

Sources : Sdes pour les capacités installées, Eurostat pour la production.

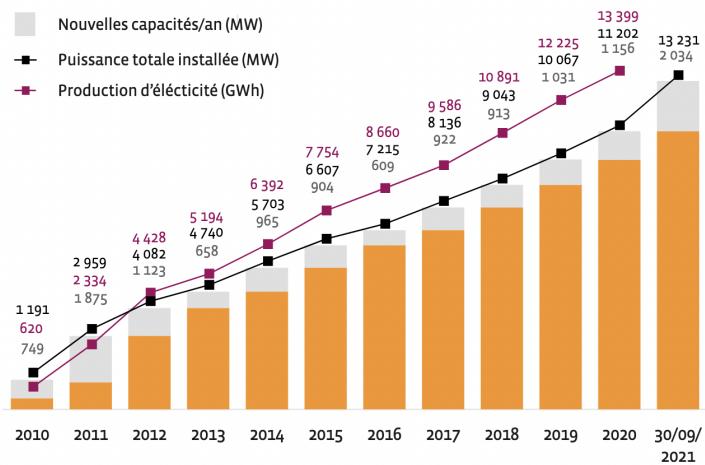


FIGURE 6 – Rythme d'installation du photovoltaïque en France depuis 2010, tiré de l'ADEME

**Figure 4.20** Rythmes moyens de développement historiques et projetés (scénario de référence) du solaire

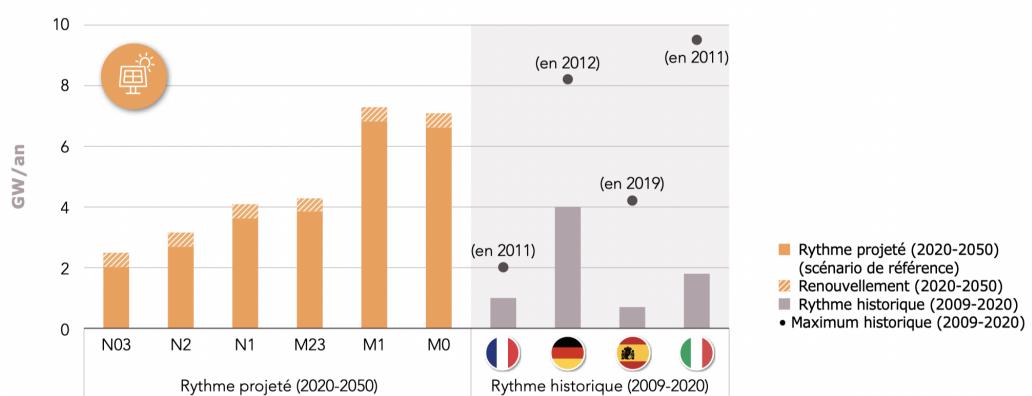


FIGURE 7 – Rythme européen comparé d'installation du photovoltaïque, tiré de RTE 2022

## B Détail des méthodes traditionnelles d'estimation de la production photovoltaïque

D'après Mayer et al. [2022], des erreurs humaines peuvent conduire à la falsification des informations. De plus, la plupart des installations sont enregistrées avec l'adresse du propriétaire. Or, cette adresse ne correspond pas toujours à l'emplacement de l'installation. Finalement, certaines applications nécessitent l'usage de caractéristiques qui ne sont pas renseignées dans les registres. Rausch et al. [2020] mentionnent par exemple l'orientation. Kausika et al. [2021b] évoquent la capacité, la localisation et la production électrique.

## C Du perceptron au réseau de neurones

### C.1 Le fondement des réseaux de neurones : le perceptron

Les réseaux de neurones présentent l'avantage d'être utilisables pour résoudre à la fois des problèmes de classification (cas où  $\mathcal{Y}$  est discret) et de régression (cas où  $\mathcal{Y} \subset \mathbb{R}$ ). Un réseau de neurones est un algorithme censé imiter le fonctionnement du cerveau humain. Les neurones correspondent à des *perceptrons*.

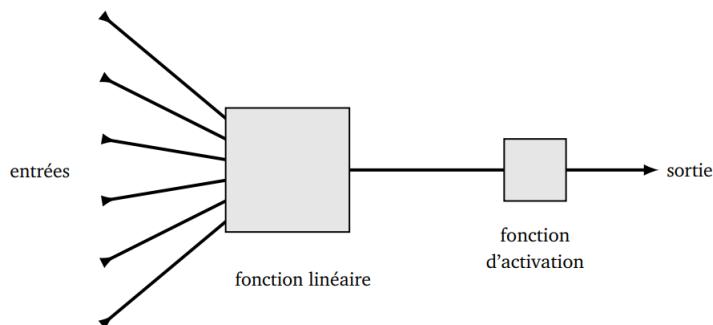


FIGURE 8 – Représentation schématique d'un perceptron linéaire  
Arnaud Bodin [2021]

Un perceptron est composé de deux fonctions, une fonction linéaire ( $f$ ) suivie d'une fonction d'activation ( $\phi$ ). Une collection de fonctions d'activation usuelles est présentée en annexe D. Si on est dans la première couche, la fonction d'activation prend en entrée le vecteur de features auquel elle associe des poids, à savoir des valeurs réelles. Ces poids décrivent la force de la connexion entre un neurone et le neurone correspondant de la couche précédente. Dans les couches suivantes, la fonction d'activation prend en entrée la sortie des neurones composant la

couche précédente. Formellement,  $\forall l \in \{1, \dots, L\}, \forall i \in \{1, \dots, q_l\}$  :

$$f : a^{l-1} = (a_1^{l-1}, \dots, a_{q_{l-1}}^{l-1})^T \longrightarrow z_i^l = b_i^l + \sum_{j=1}^{q_{l-1}} w_{i,j}^l a_j^{l-1} \quad (5)$$

Le biais  $b_i^l$  et les coefficients  $w_{i,j}^l$  forment les poids du neurone  $i$  appartenant à la couche  $l$ . La valeur de sortie  $z_i^l$  de cette fonction linéaire devient ensuite la valeur d'entrée de la fonction d'activation. Cette fonction d'activation renverra ensuite une valeur qui sera la seule sortie du neurone. Formellement :

$$\phi : z_i^l \longrightarrow a_i^l \quad (6)$$

Un neurone peut donc prendre plusieurs valeurs en entrée mais ne renvoie qu'une seule valeur en sortie. Un perceptron définit donc une composition  $\phi \circ f$ . Si  $b_i^l \in \mathbb{R}$ , on parle de perceptron affine. Si  $b_i^l = 0$ , on parle de perceptron linéaire ou de neurone artificiel.

## C.2 Définition d'un perceptron multicouches

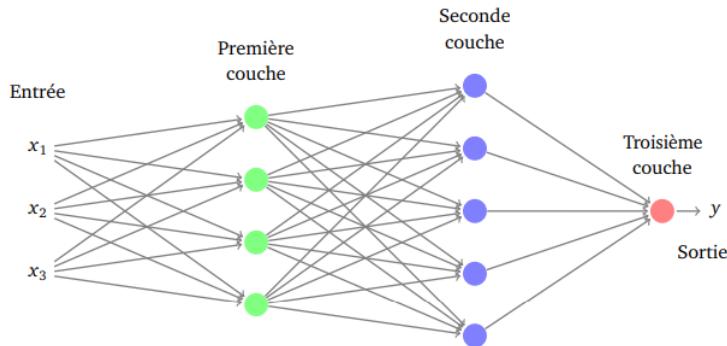


FIGURE 9 – Représentation schématique d'un réseau de neurones à trois couches  
Arnaud Bodin [2021]

Pour constituer un réseau de neurones à partir des neurones décrits ci-dessus, on construit donc plusieurs couches successives de neurones. Avec ce système, les sorties des neurones de la couche  $l$ , deviennent ensuite les entrées des neurones de la couche  $l + 1$  (voir figure 9). Ces réseaux sont très puissants car ils permettent, par exemple, d'après le théorème d'approximation universelle (Cybenko [1989]), d'approcher n'importe quelle fonction continue sur des sous-ensembles compacts de  $\mathbb{R}^n$ .

## D Fonctions d'activation usuelles

Il existe de nombreux types de fonctions d'activation. Les principales sont les suivantes :

- Linéaire :  $\phi(x) = Id(x)$
- Seuil :  $\phi(x) = \mathbb{1}_{[0,\infty[}(x)$
- Sigmoïde :  $\phi(x) = \frac{1}{1+\exp(-x)}$
- ReLU :  $\phi(x) = \max(0, x)$
- Softmax :  $\phi_j(x) = \frac{\exp x_j}{\sum_{k=1}^K \exp x_k}, \forall j \in \{1, \dots, K\}$
- Radiale :  $\phi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$

## E Définition formelle du produit de convolution, exemple et utilisation dans les réseaux de neurones convolutifs

### E.1 Définition du produit de convolution

Soient  $(f(n))_{n \in \mathbb{Z}}$  et  $(g(n))_{n \in \mathbb{Z}}$ , deux suites à valeurs réelles. Le produit de convolution  $f * g$  est la suite  $((f * g)(n))_{n \in \mathbb{Z}}$  de terme général :

$$f * g(n) = \sum_{k=-\infty}^{+\infty} f(n-k)g(k) \quad (7)$$

Si  $f$  et  $g$  sont à valeurs dans un intervalle fermé  $[-K, K]$ , le produit de convolution est alors défini par :

$$f * g(n) = \sum_{k=-K}^{K} f(n-k)g(k) \quad (8)$$

Une propriété importante du produit de convolution est la symétrie :  $\forall n \in \mathbb{Z}, f * g(n) = g * f(n)$ .

### E.2 Exemple d'application

Soient  $A \in \mathcal{M}_{n,p}(\mathbb{R})$ , une matrice, et  $M \in \mathcal{M}_k(\mathbb{R})$ , un motif. La démarche générale pour calculer le produit de convolution  $A * M$  est la suivante :

1. Si  $k$  est pair, on rajoute une colonne de zéro artificiels à gauche et une ligne de zéro artificiels en haut
2. On retourne  $M$  pour obtenir  $M'$  : le motif retourné.
3. Pour chaque composante  $(i, j)$  de  $A$ , on centre  $M'$  sur cette même composante. On multiplie chaque composante de la matrice avec la composante superposée du motif qui lui est associé. On somme les résultats obtenus et on itère pour les autres coordonnées de  $A$ .

Pour notre exemple, on prend  $A = \begin{bmatrix} 2 & -1 & 7 & 3 & 0 \\ 2 & 0 & 0 & -2 & 1 \\ -5 & 0 & -1 & -1 & 4 \end{bmatrix}$  et  $M = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$ . Le motif retourné est donc  $M' = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 1 & -1 \\ 0 & -1 & -2 \end{bmatrix}$ . Dans un premier temps, on centre le motif retourné sur  $A[1, 1] = 2$ . La convolution obtenue est égale à  $1 \times 2 + (-1) \times (-1) + (-1) \times 2 + 0 \times (-2) = 1$ . Puis, on centre ce même motif sur  $A[1, 2]$  pour obtenir  $-6$ . En itérant l'opération sur les coordonnées restantes de  $A$ , on trouve  $A * M = \begin{bmatrix} 1 & -6 & 7 & 10 & 2 \\ 9 & 7 & 10 & 7 & 1 \\ -3 & 0 & 0 & -8 & 0 \end{bmatrix}$

Schématiquement, le produit de convolution peut être représenté de la façon suivante.

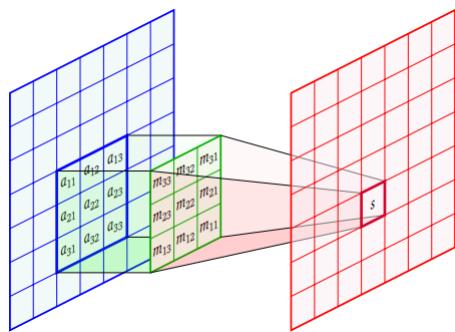


FIGURE 10 – Représentation schématique du produit de convolution en dimension 2  
Arnaud Bodin [2021]

### E.3 Définition du max pooling et de l'average pooling

Le max-pooling de taille  $k$  consiste à prendre la maximum de chaque sous-matrice de taille  $k \times k$ . L'average pooling de taille  $k$  consiste à faire la moyenne de chaque sous-matrice de taille  $k \times k$ .

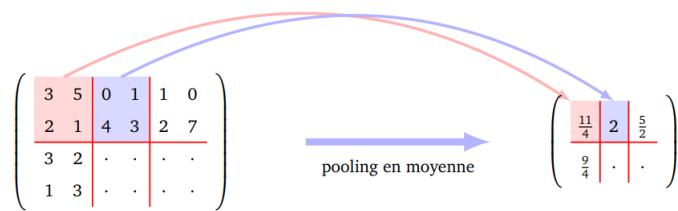


FIGURE 11 – Average pooling de taille 2  
Arnaud Bodin [2021]

## E.4 Représentation graphique d'un neurone de convolution

Dans un CNN, la convolution est réalisée par un neurone de convolution.

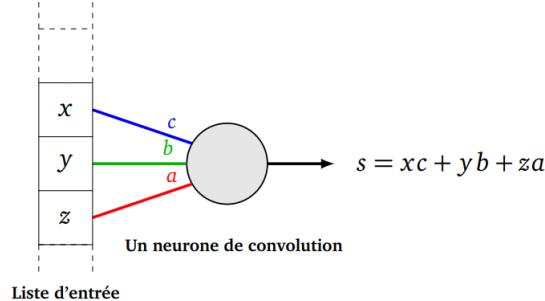


FIGURE 12 – Représentation schématique d'un neurone de convolution. Ici, le motif est de dimension 3

Arnaud Bodin [2021]

## E.5 Représentation graphique d'une couche convulsive

Une couche convulsive réalise plusieurs opérations de convolution sur les différents canaux d'une image afin de séparer les différents motifs.

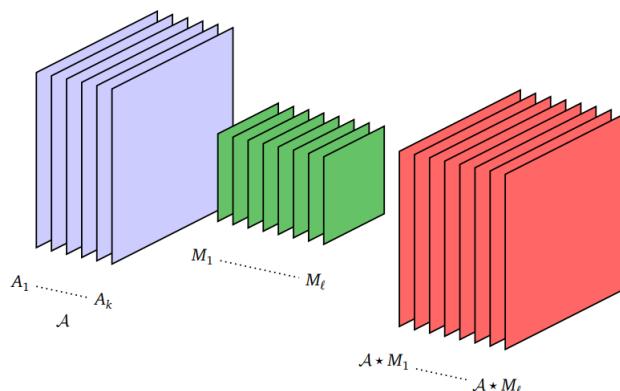


FIGURE 13 – Couche de convolution prenant en entrée  $k$  canaux et composée de  $l$  filtres  
*Note : en réalité les motifs  $M_i$  devraient être représentés par des cubes de dimensions  $m \times m \times k$*   
 Arnaud Bodin [2021]

## F Descente de gradient stochastique et rétropropagation du gradient

Pour plus de simplicité, nous omettons la distinction entre biais et coefficients.

---

**Algorithm 1** Algorithme SGD + rétropropagation du gradient

---

**Require:**  $\eta$  : learning rate,  $\beta$  : facteur d'amortissement, n\_epoch : nombre d'epochs,  $M$  : nombre de batchs,  $L$  : nombre de couches du réseau

- 1: Initialiser les poids  $(W_l^0)_{1 \leq l \leq L}$  aléatoirement
- 2: **for**  $k = 1, \dots, n\_epoch$  **do**
- 3:     Mélanger les données
- 4:     **for**  $j = 1, \dots, M$  **do**
- 5:         Estimer  $h$  sur le batch  $j$  :  $\hat{h}_j$
- 6:         **for**  $l = 1, \dots, L$  **do**
- 7:             **for**  $i \in B_j$  **do**
- 8:                 Calculer  $\nabla_l l(\hat{h}_j(\mathbf{X}_i), Y_i)$  à l'aide de  $\nabla_{l+1} l(\hat{h}_j(\mathbf{X}_i), Y_i), \dots, \nabla_L l(\hat{h}_j(\mathbf{X}_i), Y_i)$
- 9:             **end for**
- 10:              $W_l^j = W_l^{j-1} - \eta \frac{1}{|B_j|} \sum_{i \in B_j} \nabla_l l(\hat{h}_j(\mathbf{X}_i), Y_i) + \beta(W_l^{j-1} - W_l^{j-2})$
- 11:         **end for**
- 12:     **end for**
- 13: **end for**
- 14: **return**  $(W_l^*)_{1 \leq l \leq L}$

---

## G Définition et mise en contexte des métriques de classification

### Accuracy

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (9)$$

C'est le ratio entre le nombre total de prédictions correctes et le nombre total de prédictions. L'accuracy mesure la performance du modèle sur toutes les classes. La métrique est utile lorsque les classes sont équilibrées. A l'inverse, elle n'est pas robuste lorsque les classes sont déséquilibrées.

### Precision

$$Precision = \frac{TP}{TP + FP} \quad (10)$$

C'est la part de vrais positifs parmi l'ensemble des individus prédits comme positifs. Dis autrement, c'est le pourcentage d'individus prédits comme positifs, correctement classés comme positifs. La precision mesure la fiabilité du modèle pour classer uniquement les individus positifs comme positifs. La precision est élevée si, d'une part, beaucoup d'individus positifs sont prédits comme tel ( $TP$

est élevé), d'autre part, peu d'individus négatifs sont classés positifs ( $FP$  est faible).

## Recall

$$Recall = \frac{TP}{TP + FN} \quad (11)$$

C'est la part de vrais positifs parmi l'ensemble des individus positifs. Dis autrement, c'est le pourcentage d'individus positifs correctement classés comme positifs. Avec le recall, on s'intéresse uniquement à la manière dont les individus positifs sont classés. Le recall est élevé si, d'une part, beaucoup d'individus positifs sont prédits comme tel ( $TP$  est élevé), d'autre part, peu d'individus positifs sont prédits comme négatifs ( $FN$  est faible). Ainsi, le taux de faux positifs ( $FP$ ) n'intervient pas. Par exemple, si tous les individus (positifs ou négatifs) sont prédits comme positifs,  $TP = 1$ ,  $FN = 0$ . Donc,  $Recall = 1$ .

## F- $\beta$ score

$$F_\beta = \frac{(1 + \beta^2)precision \times recall}{\beta^2 \times precision + recall} \quad (12)$$

C'est une moyenne harmonique pondérée de la precision et du recall. Si :

- $\beta < 1$  : on pénalise le recall au profit de la precision (généralement  $\beta = 0,5$ )
- $\beta = 1$  : on accorde le même poids à la precision et au recall
- $\beta > 1$  : on pénalise la precision au profit du recall (généralement  $\beta = 2$ )

## Courbe ROC

La courbe ROC (*Receiver Operating Characteristic*) permet de visualiser la performance d'un classificateur binaire. Elle prend en abscisse le taux de faux positifs, et en ordonnée le taux de vrai positifs, on y trace souvent la performance d'un classificateur dit naïf (fonction identité).

## AUC

L'AUC (*Area Under the Curve*) est l'aire située sous la courbe ROC et mesure la qualité discriminatoire d'un classificateur binaire. C'est donc une mesure de la capacité d'un modèle à faire la distinction entre les deux catégories qu'identifie le modèle. Plus précisément, l'AUC donne la probabilité que le modèle renvoie une probabilité de contenir un panneau plus élevée pour une image quelconque de l'échantillon contenant réellement un panneau photovoltaïque, que la probabilité

qu'il renverra pour une image quelconque de l'échantillon ne contenant pas de panneau. L'AUC et la courbe ROC sont des mesures de performances sensibles lorsque l'échantillon est déséquilibré. On privilégiera ainsi l'usage de la courbe precision-recall.

### Courbe precision-recall

La courbe precision-recall mesure la capacité d'un modèle à prédire la catégorie d'une image lorsque la répartition de cette classe dans les données disponibles est très déséquilibrée. Cette courbe montre le compromis entre la précision et le recall pour différentes valeurs de ces métriques.

## H Présentation des data augmentations utilisées

### Resize

Le resize est une data augmentation permettant de redimensionner les images selon une valeur spécifiée par l'utilisateur. Certains CNN requièrent l'usage d'images d'une dimension spécifique. Ainsi, grâce à cette augmentation, il est possible d'utiliser nos images pour entraîner les modèles désirés.

### Conversion en tenseur

En convertissant les images en tenseurs, on peut facilement réaliser des opérations mathématiques sur ces dernières. La fonction *ToTensor()* du module *torchvision* applique une transformation supplémentaire de sorte que chaque pixel de l'image soit compris dans l'intervalle [0, 1].

### Normalisation

Pour normaliser les images, on calcule au préalable la moyenne et l'écart-type des pixels par couche. Pour chaque couche d'une image, on soustrait la moyenne des pixels calculée et on divise par l'écart-type.

### ColorJitter

Le ColorJitter consiste à changer de façon aléatoire la luminosité, le contraste, la saturation et le ton d'une image. Pour chaque paramètre, l'utilisateur renseigne un intervalle de valeurs. Ce faisant, chaque aspect de l'image est modifié selon une valeur tirée aléatoirement (loi uniforme) dans chaque intervalle. Dans la pratique, il est très important de se questionner sur le choix des paramètres à modifier. Utiliser un intervalle de valeurs trop large peut omettre toutes les nuances de l'image, voire rendre la présence d'un panneau indétectable, comme en témoigne l'exemple ci-dessous.

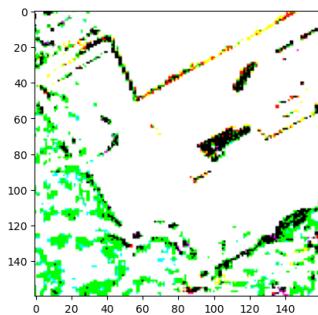


FIGURE 14 – Mauvaise utilisation du ColorJitter :  $brightness \in [1, 10]$ ,  $contrast \in [1, 10]$ ,  $saturation \in [1, 10]$ ,  $hue \in [-0.1, 0.1]$

## Random Crop

Lorsque l'on applique un random crop, on centre l'image sur un point aléatoire. En pratique, l'utilisateur peut paramétriser la proportion de pixels qu'il souhaite conserver dans l'image transformée. Dans notre cas, l'usage de crops peut aboutir à la disparition du panneau solaire dans une image qui en contenait un au préalable. Ainsi, les crops doivent être utilisés avec parcimonie.

## I Architecture du LeNet5

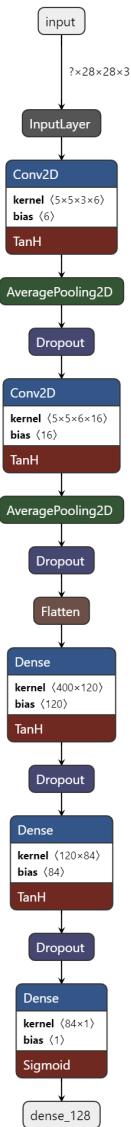


FIGURE 15 – Schéma présentant l'architecture du LeNet5 utilisé

## J Choix des data augmentations et fine tuning du LeNet5

## J.1 Les data augmentations : un outil contre le sur-apprentissage

### J.1.1 Analyse des performances du réseau sans data augmentations

Mis à part la convention en tenseurs et le redimensionnement des images, nous n'appliquons pas de data augmentations sur les images, dans un premier temps. Nous fixons les hyper-paramètres de façon arbitraire. Le learning rate vaut 0,01, le momentum 0,9. Dans chaque couche de dropout, nous fixons la probabilité de désactiver un neurone à 0,2. Enfin, le nombre d'epochs est de 10, le batch size est de 32.

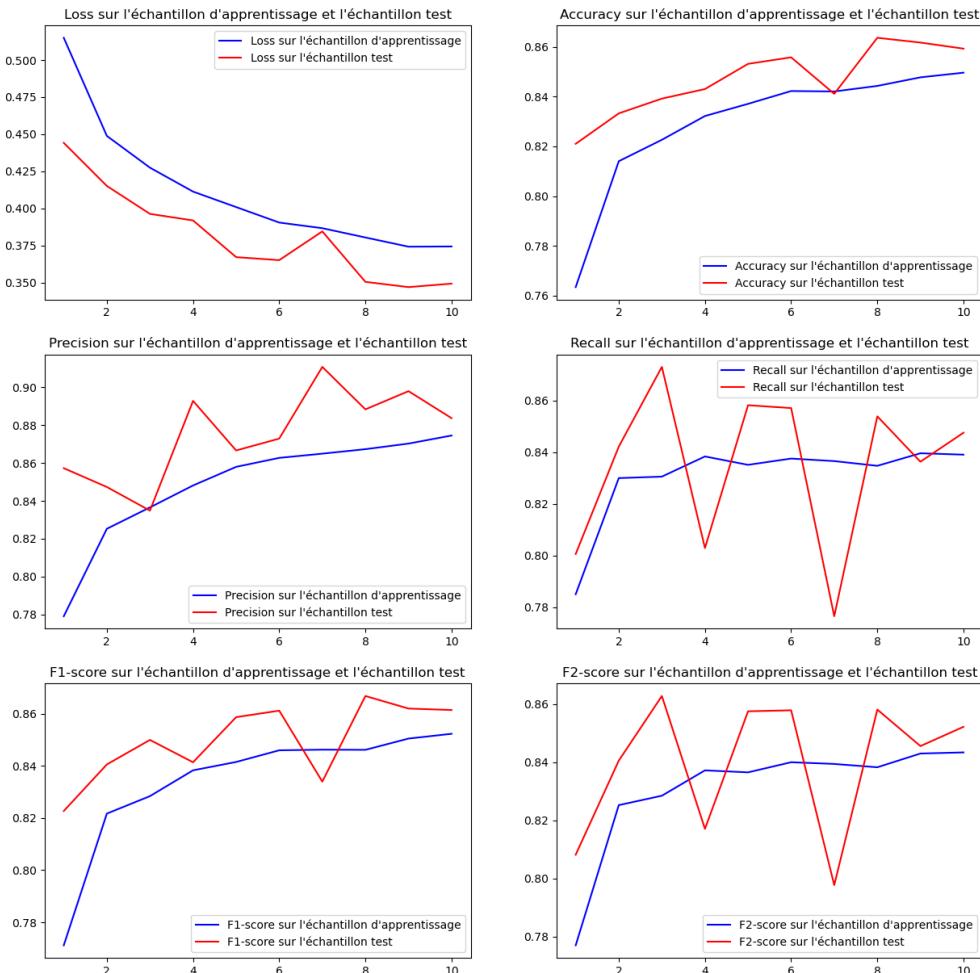


FIGURE 16 – Evolution des métriques au fil des epochs pour le LeNet5, sans data augmentations

Comme le montre la figure 16, la log-loss diminue au fil des epochs, sur l'échantillon train et le test. De plus, la valeur prise par la métrique est toujours plus faible sur l'échantillon de validation. Cependant, on remarque que le recall et le F2-score varient beaucoup sur l'échantillon test, au cours de l'apprentissage du réseau. De plus, le F1-score, la precision ainsi que l'accuracy diminuent après 8 epochs, sur ce même échantillon. On peut donc soupçonner la présence d'overfitting. Les résultats sont présentés ci-dessous.

Métrique	Apprentissage	Validation
BCE	0,346	0,349
F1-score	0,865	0,866
Recall	0,854	0,848
F2-score	0,857	0,852
Accuracy	0,863	0,859
Precision	0,883	0,885

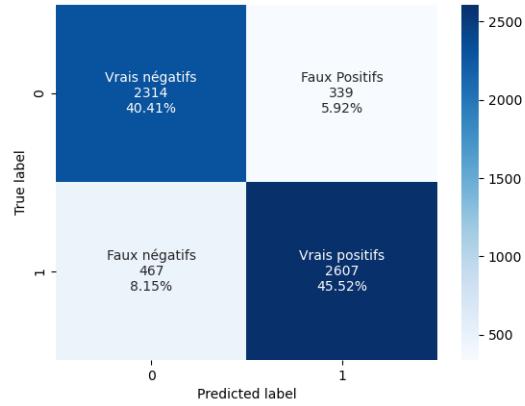


TABLE 4 – Métriques calculées sur l'échantillon d'apprentissage et de test, et matrice de confusion (absence de data augmentations)

Les résultats obtenus sur l'échantillon de validation sont satisfaisants pour un premier modèle. Nous noterons en particulier que le nombre de faux positifs et de faux négatifs est faible. Au global, 14,07 % des images n'ont pas été bien classées.

### J.1.2 La normalisation des images et l'ajout de ColorJitter ne semble pas régler le problème du sur-apprentissage

Au vu des résultats obtenus, l'ajout de data augmentations supplémentaires semble être une bonne option pour réduire le sur-apprentissage. Ainsi, nous normalisons les données d'apprentissage et de validation. Nous ajoutons du ColorJitter sur les images de l'échantillon test. Seule la luminosité des images est modifiée aléatoirement avec une valeur comprise entre 0,5 et 1,5, de sorte à ne pas rendre les panneaux indétectables.

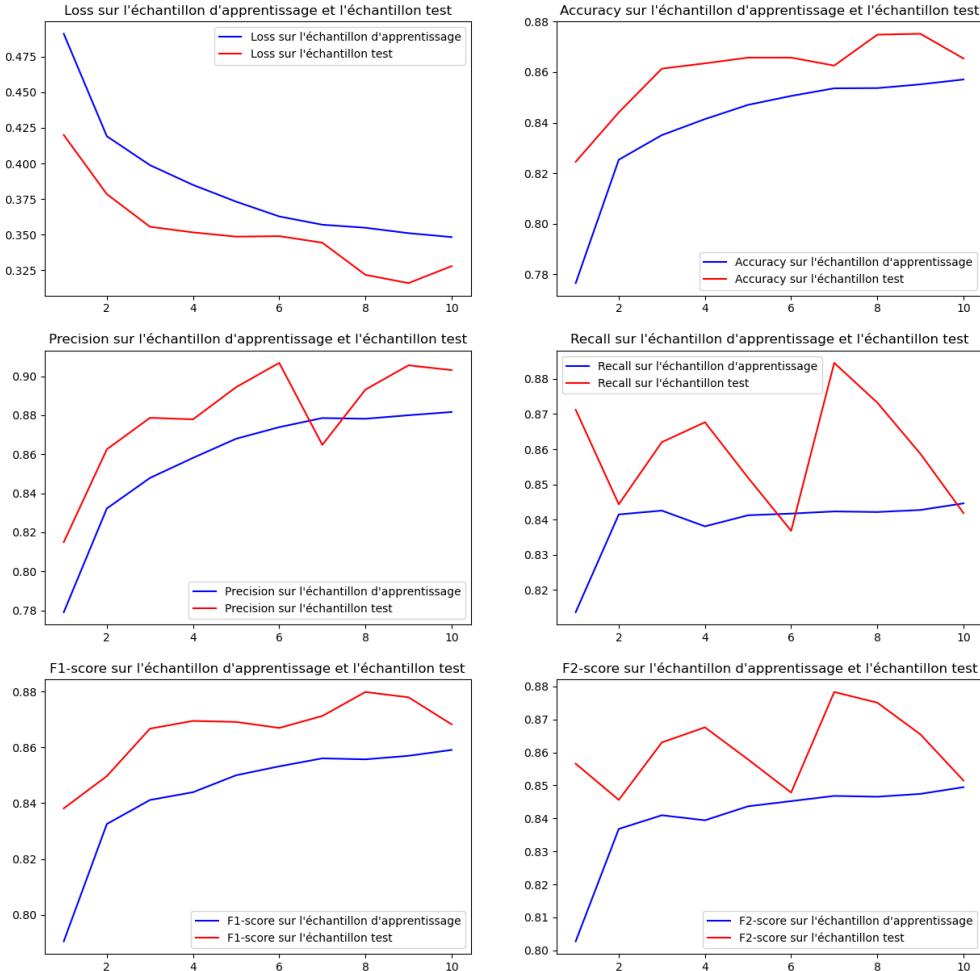


FIGURE 17 – Evolution des métriques au fil des epochs pour le LeNet5 : normalisation des images et ColorJitter sur l'échantillon de validation

Comme le montre la figure 17, le F1-score, le recall et le F2-score continuent de décroître après la huitième epoch. De plus, la log-loss augmente légèrement passé la neuvième itération. Ainsi, le problème de sur-apprentissage semble persister.

Métrique	Apprentissage	Validation
BCE	0,333	0,328
F1-score	0,861	0,87
Recall	0,831	0,841
F2-score	0,842	0,851
Accuracy	0,862	0,865
Precision	0,901	0,902

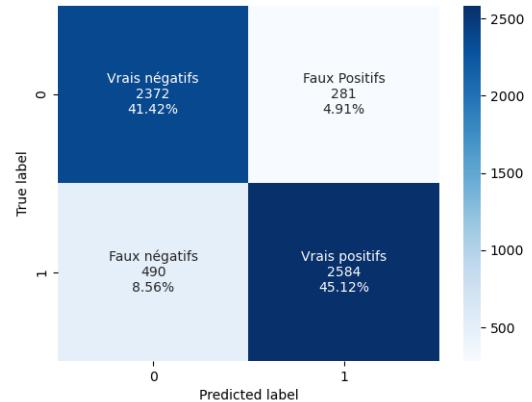


TABLE 5 – Métriques calculées sur l'échantillon d'apprentissage et de test, et matrice de confusion (normalisation des données, ajout de ColorJitter sur l'échantillon test)

Désormais, l'entropie croisée binaire est plus faible sur l'échantillon de validation. Cependant, les valeurs prises par les différentes métriques de classification ne varient pas significativement, en comparaison des résultats précédents. Le réseau détecte un peu mieux les images sans installation (le nombre de vrais négatifs augmente d'un point). A l'inverse, le nombre de vrais positifs diminue de 0,4 points. Cette différence de résultat non significative peut d'autant plus être liée au hasard.

### J.1.3 L'ajout de crops limite le sur-apprentissage

Nous conservons la configuration précédente en ajoutant des crops de taille 24 sur l'échantillon de validation. Nous redimensionnons ensuite les images en  $28 \times 28$  afin de rester cohérent à l'égard de l'architecture du réseau.

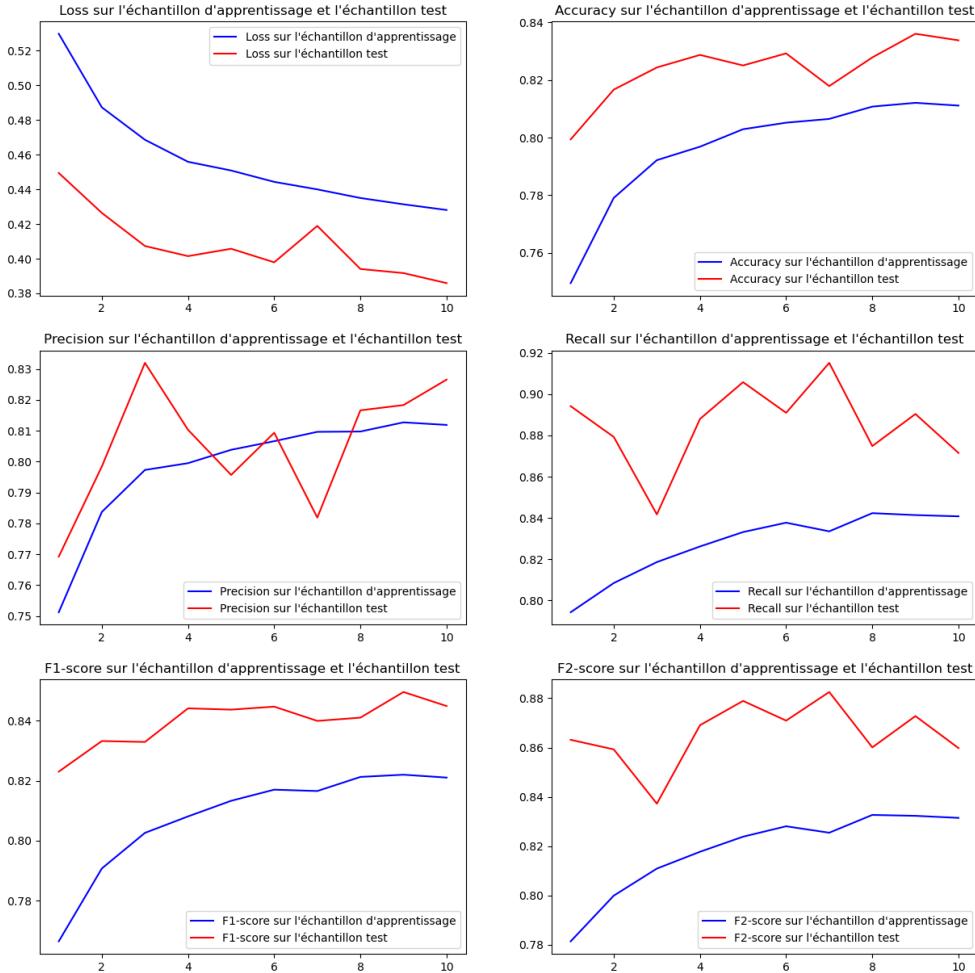


FIGURE 18 – Evolution des métriques au fil des epochs pour le LeNet5 : normalisation des images, ColorJitter et crops sur l'échantillon de validation

Désormais, la log-loss atteint son minimum sur l'échantillon de validation au cours de la dernière epoch. Sur ce même échantillon, le F1-score continue de croître jusqu'à la neuvième epoch. Seuls le recall et le F2-score diminuent en fin d'apprentissage. Ainsi, l'ajout de crops semble avoir réduit l'over-fitting, même si le résultat n'est pas optimal.

Métrique	Apprentissage	Validation
BCE	0,403	0,386
F1-score	0,834	0,849
Recall	0,839	0,872
F2-score	0,836	0,86
Accuracy	0,827	0,834
Precision	0,837	0,827

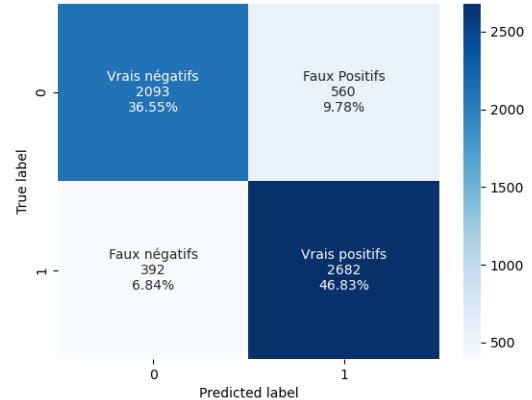


TABLE 6 – Métriques calculées sur l'échantillon d'apprentissage et de test, et matrice de confusion (normalisation des données, ajout de ColorJitter et de crops sur l'échantillon test)

Désormais, le modèle est plus performant pour détecter les installations existantes (le recall et le F2-score augmentent respectivement de 3 et 1 points sur l'échantillon test, le nombre de vrais positifs augmente de 1,7 points). Pour autant, le nombre de faux positifs a augmenté 4,87 points ce qui se traduit par une baisse de la precision.

## J.2 Le choix de la métrique à optimiser dans l'hyper-paramétrisation du réseau est crucial

Le ColorJitter, combiné aux crops semble avoir limité le sur-apprentissage. Ainsi, nous retenons cette configuration pour le LeNet5. Afin d'améliorer les performances du modèle, nous hyper-paramétrisons le modèle selon la méthode *Randomised Grid Search CV*. Les hyper-paramètres optimisés sont le learning rate, le momentum, les probabilités de désactiver un neurone dans les couches de dropout, la taille de batch et le nombre d'epochs. Nous utilisons une validation croisée 5-blocs sur l'échantillon d'apprentissage. Reste à déterminer le nombre de combinaisons aléatoires d'hyper-paramètres à tester.

Soit  $M$ , le nombre de combinaisons. Après  $M$  itérations, la probabilité qu'aucun paramètre parmi les  $p\%$  meilleurs n'ait été tiré est de  $(1 - p)^M$ . Donc, la probabilité de tirer au moins un paramètre optimal est de  $1 - (1 - p)^M$ . Donc, si on veut tirer les 10% meilleurs avec une confiance de 95%, on résoud :  $1 - (1 - 0,1)^M = 0,95 \iff M = \frac{\log 0,05}{\log 0,9} \approx 30$ . On teste donc 30 combinaisons de paramètres.

Dans une perspective de détection optimale des installations solaires, nous avons dans un premier temps choisi le F2-score comme métrique à optimiser.

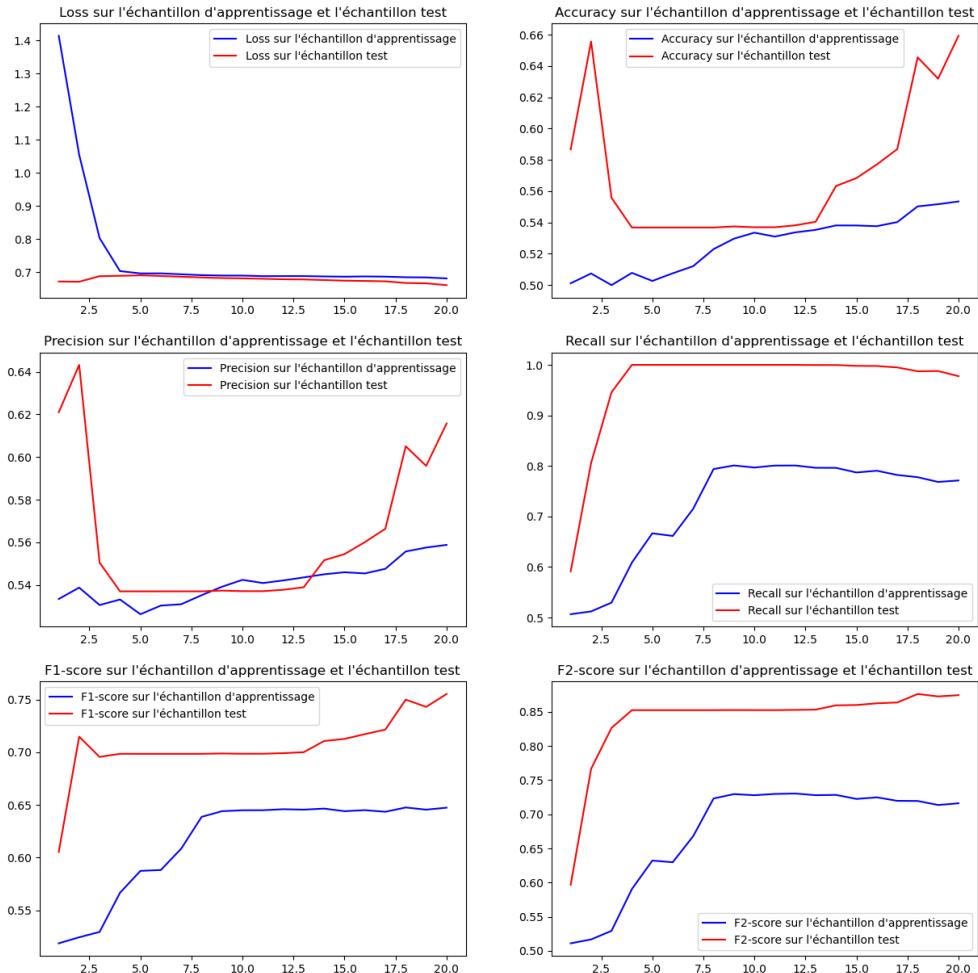


FIGURE 19 – Evolution des métriques au fil des epochs pour le LeNet5 : hyper-paramétrisation par rapport au F2-score

Comme en témoigne le graphique 18, ce choix s'avère être mauvais. Que ce soit sur l'échantillon train ou test, la loss se stabilise très vite autour d'une valeur élevée de 0,7. De plus, les métriques de classification évoluent très peu lors de l'apprentissage. De façon claire, le modèle sous-apprend.

Métrique	Apprentissage	Validation
BCE	0,661	0,661
F1-score	0,744	0,755
Recall	0,97	0,978
F2-score	0,863	0,871
Accuracy	0,653	0,659
Precision	0,609	0,614

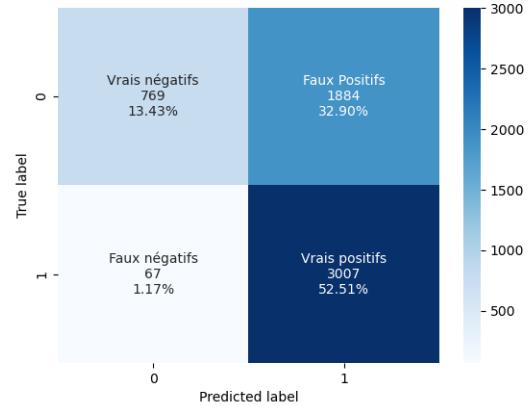


TABLE 7 – Métriques calculées sur l'échantillon d'apprentissage et de test, et matrice de confusion (hyper-paramétrisation par rapport au F2-score)

La quasi-intégralité des images ont été classées comme contenant une installation solaire. Ainsi, le recall est très proche de 1, on compte un tiers de faux positifs. Hyper-paramétriser le modèle selon le F2-score n'est pas une option adéquate. Afin de réduire significativement la part de faux positifs, nous choisissons d'optimiser le modèle relativement au F1-score, une métrique plus équilibrée. Le tableau J.2 récapitule le choix des hyper-paramètres selon la métrique considérée. Notons que le sous-apprentissage du LeNet5, optimisé relativement au F2-score, provient sûrement du faible learning rate. Les résultats finaux du LeNet5 hyper-paramétrisé selon le F1-score sont présentés dans la section 4.1.4.

Hyper-paramètre	Valeurs possibles	Valeur optimale (F2-score)	Valeur optimale (F1-score)
Learning rate	{0, 01, 0, 001, 0, 0001}	0,0001	0,01
Momentum	{0, 9, 0, 99}	0,99	0,9
Dropout 1	{0, 1; ...; 0, 9}	0,3	0,1
Dropout 2	{0, 1; ...; 0, 9}	0,8	0,6
Dropout 3	{0, 1; ...; 0, 9}	0,8	0,3
Dropout 4	{0, 1; ...; 0, 9}	0,9	0,8
Batch size	{4, 8, 16, 32, 64, 128, 256}	256	32
Epochs	{10, 20, 30, 40, 50}	20	40

TABLE 8 – Valeurs optimales des hyper-paramètres selon la métrique considérée

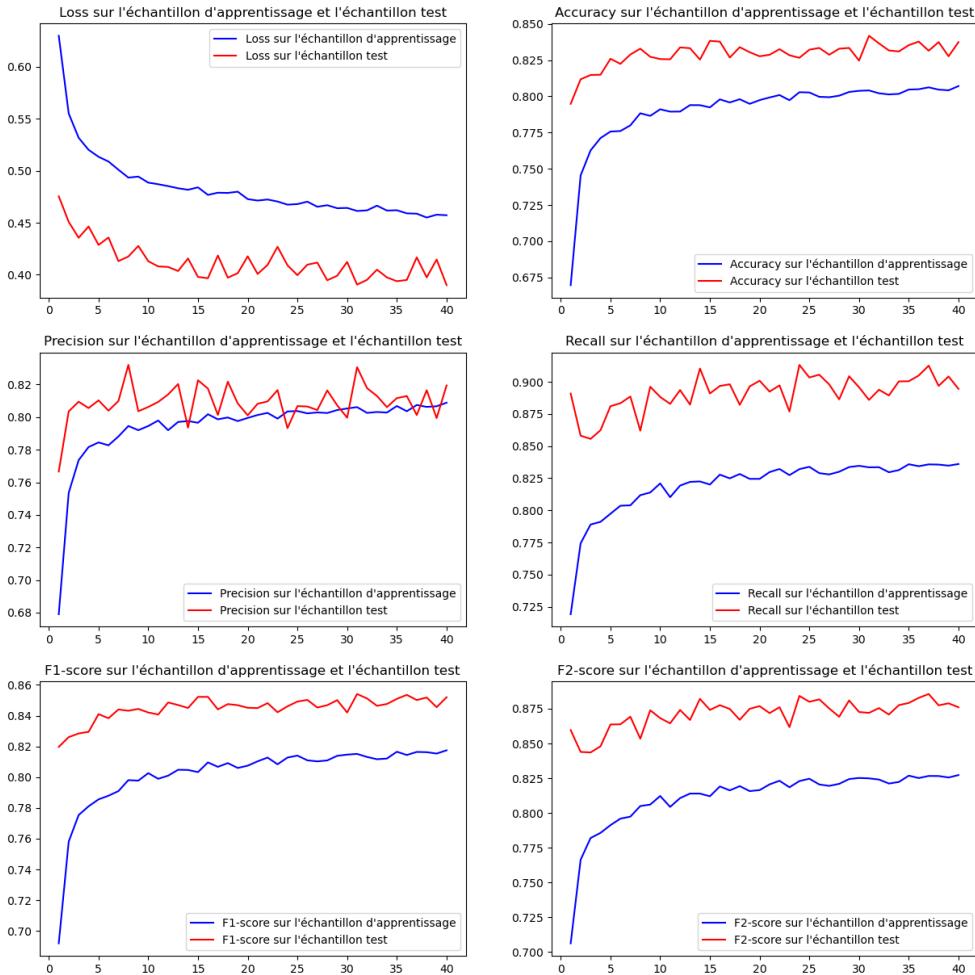


FIGURE 20 – Evolution des métriques au fil des epochs pour le LeNet5 : hyper-paramétrisation par rapport au F1-score

## K Choix du learning rate pour le ResNet18

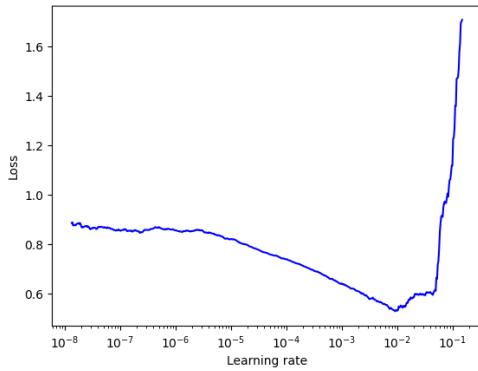


FIGURE 21 – Perte en fonction du learning rate

## L T-SNE : une technique de réduction de dimension qui préserve les similarités locales

Le T-SNE pour *T-distributed Stochastic Neighbor Embedding* est une méthode de réduction de dimension conservant les relations de voisinages entre les points : les objets similaires deviennent voisins dans l'espace de faible dimension et ceux dissemblables deviennent distants. La technique est idéale pour représenter des images dans un espace compréhensible humainement (deux ou trois dimensions).

L'idée est d'introduire une probabilité  $p_{i|j}$  définissant la probabilité de choisir le point  $x_j$  sachant que nous sommes au point  $x_i$  :

$$p_{i|j} = \frac{\exp(-\|x_j - x_i\|^2/\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_k - x_i\|^2/\sigma_i^2)}$$

Ici, la probabilité est modulée par l'écart-type  $\sigma_i$  qui dépend du point  $x_i$  afin de prendre en compte la densité de points autour de  $x_i$ .

De cette façon, on peut définir

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}$$

caractérisant la distance entre deux points.

De même, nous pouvons définir la distance entre deux points  $y_i, y_j$  en dimension 2 ou 3 :

$$q_{ij} = \frac{(1 + \|y_j - y_i\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|y_j - y_l\|^2)^{-1}}$$

L'objectif est désormais de trouver les  $(y_i)$  de sorte que les distances  $q_{ij}$  ressemblent aux distances  $p_{ij}$ . Dans ce but, nous minimisons au sens de la divergence de Kullback-Leibler la différence entre ces deux distributions :

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Intuitivement, cette divergence KL représente l'espérance de surprise (au sens de l'information de Shannon) entre les distributions  $P$  et  $Q$ . Afin de trouver les meilleures positions  $y_i$ , il suffit d'appliquer un algorithme de descente de gradient pour minimiser la divergence KL.

Enfin, notons que la perplexity est un paramètre à calibrer. La perplexity correspond à l'entropie de Shannon, associée à la loi conditionnelle induite par un point, mise au carré. Plus la perplexity est élevée, moins les informations locales sont retenues lors de la réduction de dimension. Ainsi, les clusters sont davantage visibles.