



IMT Lille Douai
École Mines-Télécom
IMT-Université de Lille

IMT Lille-Douai
2020-2021

Rapport de Projet

Anna PASQUIER
Nicolas GAUFFIN
Claire DELGOVE

élèves ingénieurs, deuxième année

Détection du port du masque facial – COVID 19

Réalisation d'algorithmes de Deep Learning en Python

Projet réalisé en période 5 de l'année de M1
Professeur référent : José Mennesson

Merci à Mr Mennesson pour nous avoir accompagnés durant la réalisation de ce projet.

Sommaire

<u>Résumé</u>	4
<u>Introduction</u>	5
 <u>Partie 1 - Présentation des données</u>	6
1) <u>Choix du dataset</u>	6
2) <u>Préparation des données</u>	6
<u>Partie 2 - Etudes préliminaires</u>	7
1) <u>Démarche</u>	7
2) <u>Résultats</u>	7
<u>Partie 3 - Réseau de neurones convolutif (CNN)</u>	8
1) <u>Principe</u>	8
2) <u>Fonctionnement</u>	9
3) <u>Mise en place</u>	13
4) <u>Modèles de CNN étudiés</u>	15
1. <u>Modèle pré-entraîné InceptionV3</u>	15
2. <u>Modèle pré-entraîné VGG16</u>	16
3. <u>Adaptation de l'algorithme</u>	17
4. <u>Entraînement du modèle VGG16</u>	19
<u>Partie 4 - Implémentation sur flux vidéo</u>	22
1) <u>Démarche</u>	22
2) <u>Détecteur de visages utilisé</u>	22
3) <u>Lancement du flux vidéo</u>	23
<u>Partie 5 - Utilisation d'un classifieur Viola et Jones</u>	24
1) <u>Principe de la méthode</u>	24
2) <u>Création d'un Haarcascade</u>	25
1. <u>Entraînements de classifieurs Haarcascade</u>	25
2. <u>Tests et utilisations des trois classifieurs</u>	27
<u>Conclusion</u>	30
<u>Annexe</u>	31
<u>Bibliographie</u>	32

Résumé

Le masque facial, autrefois pratiquement réservé au personnel médical, est devenu incontournable depuis le début de la pandémie de COVID19 pour l'ensemble de la population.

L'objectif de ce projet est de réaliser des algorithmes de détection du port du masque sur les individus. Réalisés en Python, ces algorithmes pourront analyser des photos ou des vidéos selon différentes méthodes de traitement d'image et de Deep Learning.

Mots clés	Reconnaissance faciale – Deep Learning – Data Science – Programmation – Analyse de données – Traitement d'images – Méthodes de vision
Pré-requis	Python- Notions de science des données

Introduction

L'analyse d'objets dans l'image est un sujet majeur de l'apprentissage automatique. Les progrès continuels dans ce domaine conduisent aujourd'hui à des modèles si performants qu'ils sont considérés comme fiables. Par exemple, la médecine s'appuie sur la détection automatique de caractéristiques dans une cellule pour déceler certains cancers, avec des résultats meilleurs que par un diagnostic humain (source). Nous nous proposons d'entreprendre un projet de développement d'un tel algorithme. Pour se faire, nous nous appuierons sur l'actualité, en essayant de déterminer le port du masque sur des individus.

Pour se faire, il est possible d'entraîner des algorithmes de Deep Learning capables de détecter le port du masque. La mise en place de ce système de contrôle reste très large et peut être appliquée selon diverses exigences et cela pour différents supports (photos, vidéos). En effet, au-delà de la détection du masque, un algorithme entraîné de manière spécifique peut déterminer si un masque est porté correctement ou non, et cibler les individus sujets à de mauvaises pratiques (masque sous le nez, masque sous le menton,...). Plus le dataset utilisé pour l'apprentissage est conséquent et diversifié, plus l'algorithme sera performant. L'objectif de ce projet est donc d'explorer et d'illustrer différents algorithmes permettant de détecter le port du masque selon différentes exigences sur des photos et des vidéos d'individus.

Afin de recenser de manière organisée nos scripts python, les datasets utilisés ainsi que nos démonstrations, nous avons créé un Github accessible via le lien suivant : <https://github.com/ClaireDel/Projet-P5>. Les codes python relatifs à une partie donnée seront directement accessibles au sein du rapport en cliquant sur le lien indiqué par le logo Github.

Exemple :




[Projet-P5/](https://github.com/ClaireDel/Projet-P5)

Si vous souhaitez lancer un code, pensez bien à vérifier si les modèles et datasets sont correctement importés, et si les chemins d'accès sont corrects. La liste des bibliothèques Python nécessaires à ce projet est en fin de rapport.

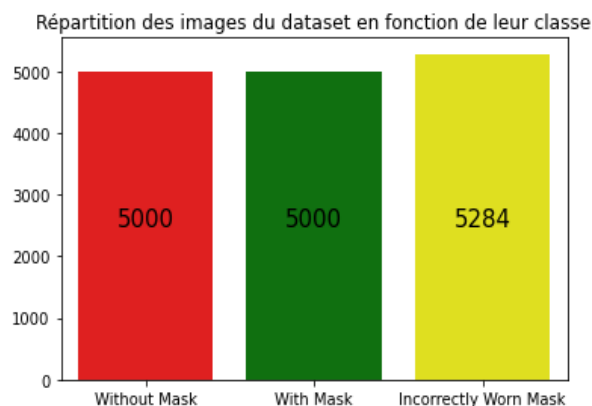
Partie 1 – Présentation des données

1. Choix du dataset

 [Ressource : Dataset utilisé](#)



Le dataset utilisé pour réaliser l'entraînement des différents modèles qui vont suivre contient 15 284 images en couleur appartenant à 3 classes différentes : **With Mask / Without Mask / Mask Incorrectly Worn**. Le diagramme à bandes illustre la répartition plus ou moins équitable entre les classes :



Codage des labels :

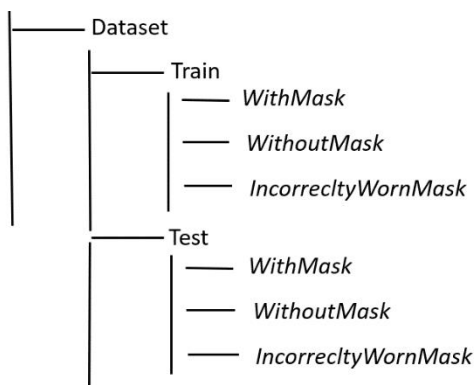
0 : No Mask

1 : Mask

2 : Mask Incorrectly Worn

2. Préparation des données

 [Projet-P5/Partie1/Data Processing.py](#)



Le dataset choisi est un jeu de données organisé en dossiers d'images non labellisées. Pour exploiter ce jeu de données, il a fallu associer un label à toutes les images de chacun des sous-dossiers Train et Test. Cette association nous a permis d'obtenir des données exploitables pour l'entraînement et pour le test du modèle utilisé.

Enfin, il suffit de transformer les données en vecteurs NumPy, d'adapter la taille de l'image, et d'enregistrer le tout sous forme de pickles prêts à l'emploi.

Partie 2 – Etudes préliminaires

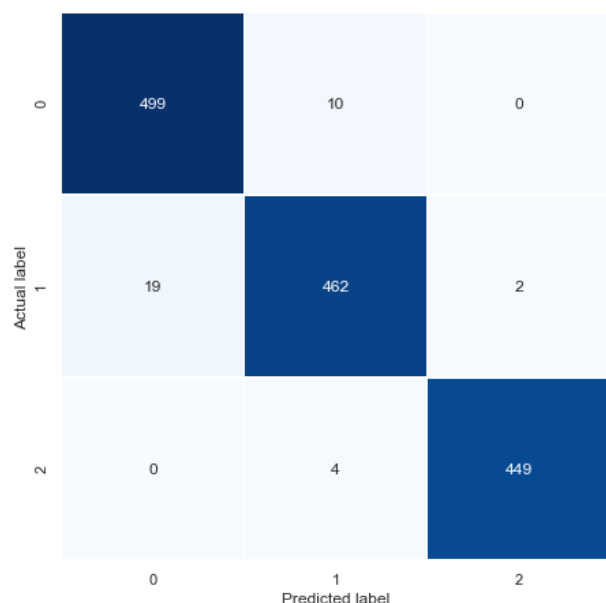
1. Démarche

L'enjeu de cette partie est d'illustrer les performances de différents modèles de classifieurs d'images de visages masqués par apprentissage supervisé. Les modèles sont en mesure de déterminer si un masque est porté ou non, et si oui, s'il est correctement porté. Les modèles étudiés sont les suivants : KNN, Decision Tree, Naive Bayes, SVM, et CNN.

2. Résultats

 [Projet-P5/ Partie2/](#)

Méthode	Précision
<i>KNN with Tuning</i>	0,85
<i>KNN without Tuning</i>	0,94
<i>Decision Tree</i>	0,90
<i>Naive Bayes : GaussianNB</i>	0,86
<i>Naive Bayes : MultinomialNB</i>	0,82
<i>SVM</i>	0,98



Accuracy 0.975779					
	precision	recall	f1-score	support	
0	0.96	0.98	0.97	509	
1	0.97	0.96	0.96	483	
2	1.00	0.99	0.99	453	
accuracy			0.98	1445	
macro avg	0.98	0.98	0.98	1445	
weighted avg	0.98	0.98	0.98	1445	

Résultats de précision pour SVM

Matrice de confusion pour SVM

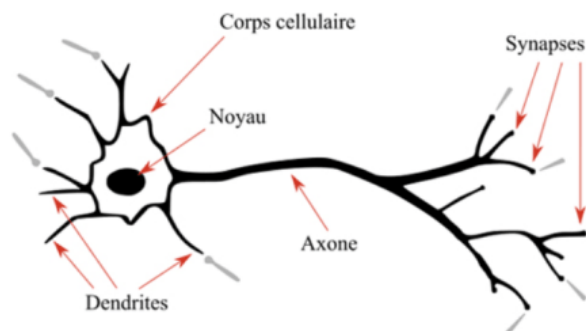
Le tableau comparatif nous montre que les différents modèles utilisés sont assez performants, notamment le SVM avec une précision de 0.98. Par la suite, nous avons décidé de nous intéresser davantage aux réseaux de neurones convolutifs, réputés pour leur performance en termes de classification. Néanmoins, les autres modèles ne sont pas à rejeter pour la détection de visages masqués.

Partie 3 – Réseau de neurones convolutif (CNN)

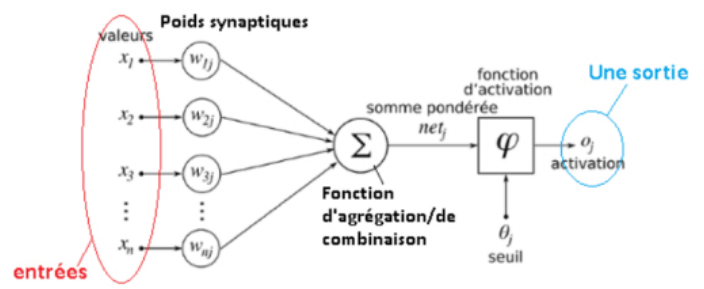
1. Principe

Les réseaux de neurones convolutifs ont une méthodologie similaire à celle des modèles traditionnels d'apprentissage supervisé : ils reçoivent des images en entrée, détectent les features de chacune d'entre elles, puis entraînent un classifieur dessus.

Il s'agit d'un type de réseau de neurones dont le motif est inspiré du fonctionnement des neurones du cerveau. Le CNN reproduit plus spécifiquement le motif du cortex visuel. Il permet de larges applications dans la reconnaissance d'images.

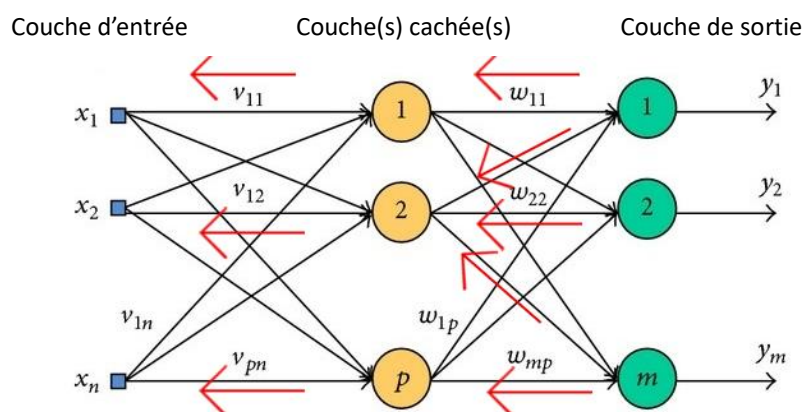


Neurone Biologique



Neurone Artificiel

Lorsque le CNN renvoie une prédiction, on peut vérifier le résultat et ajuster les poids des différentes couches par rétropropagation du gradient de l'erreur. Cet algorithme consiste à calculer le gradient de l'erreur pour chaque neurone du réseau, de la dernière couche vers la première. Les erreurs sont corrigées selon un ordre d'importance, afin d'ajuster les poids permettant de minimiser une fonction de coût.



Principe de la rétropropagation du gradient de l'erreur

2. Fonctionnement

Un CNN applique généralement 4 types d'opérations différentes à une image afin d'en extraire les informations pertinentes : la **convolution**, le **pooling**, l'**activation**, le **flattening**.

Ensuite, ces informations pertinentes sont étudiées ensemble et l'analyse de chacune d'entre elles permet à l'algorithme de **prédire** la classe de l'image en entrée.

La convolution

La convolution est l'étape primaire de processing de l'image. Celle-ci est découpée en zones appelées tuiles. A l'intérieur de ces tuiles, à chaque pixel est associé un poids par le travail d'un neurone artificiel. Chaque tuile possède son propre neurone artificiel, mais ceux-ci ont tous les mêmes paramètres de base. De ce fait, toutes les tuiles sont analysées de la même manière. En pratique, les tuiles se chevauchent selon un pas défini au préalable. Un calcul de convolution est ensuite effectué au sein de chaque tuile pour chaque caractéristique recherchée. On empile ensuite les différentes convolutions, permettant d'obtenir en sortie une carte d'activation (feature map) qui traduit "de manière simplifiée" la présence et la localisation des caractéristiques dans l'image.



Image d'entrée

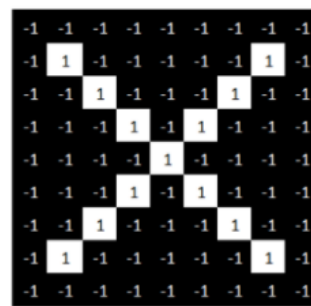
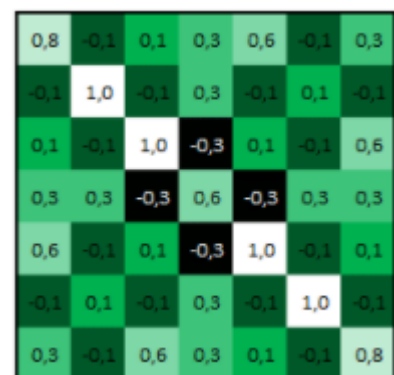
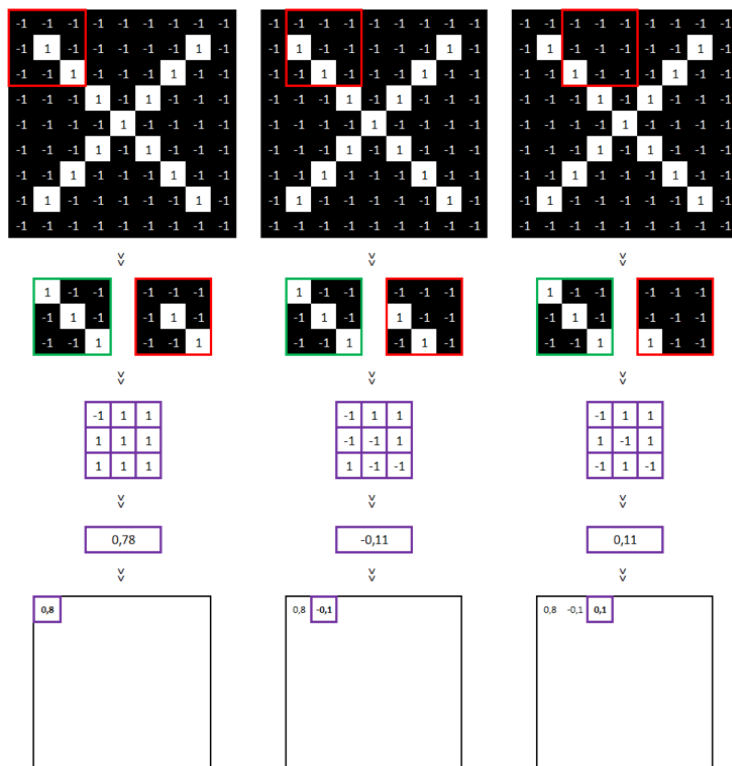


Image encodée équivalente



Caractéristique recherchée

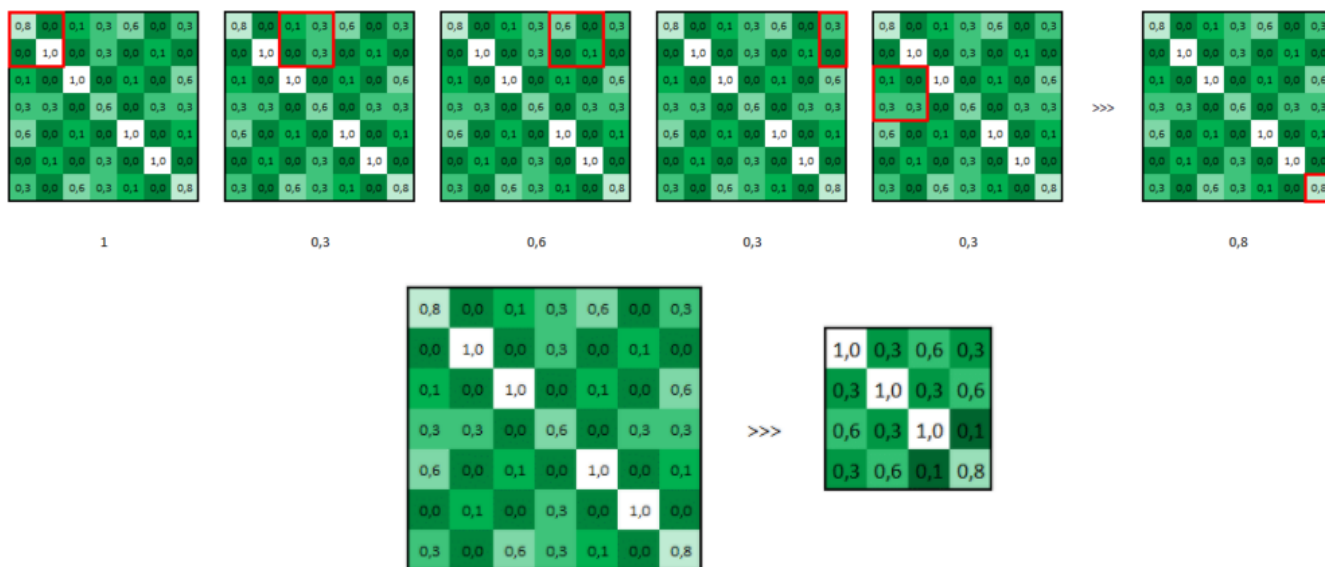
Recherche de la caractéristique dans les tuiles (zones rouges) :



Couche de convolution
obtenue

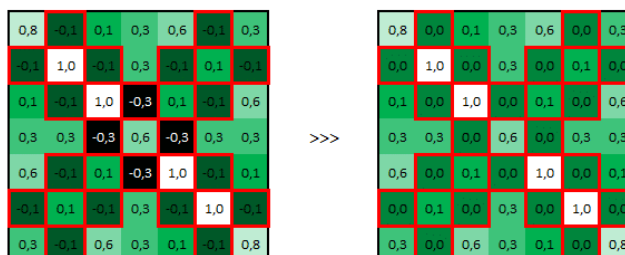
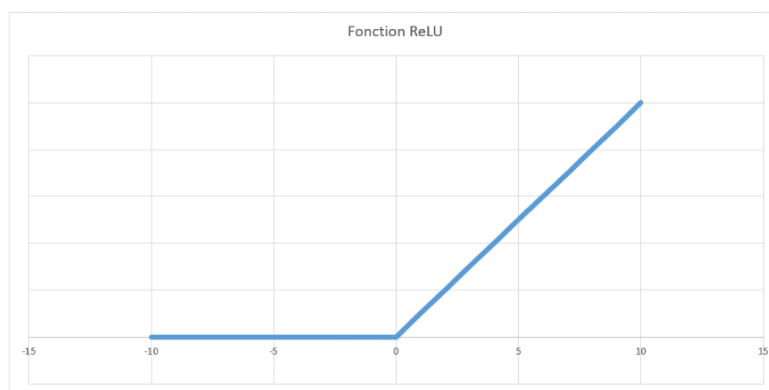
Le pooling (Max-pooling)

Après plusieurs étapes de convolution, on effectue une étape de pooling, qui correspond à prendre la valeur maximale de chaque portion de 4 pixels (carré 2x2) des couches (Max-pooling). Cette étape permet de réduire les dimensions de la donnée traitée d'un facteur 2.



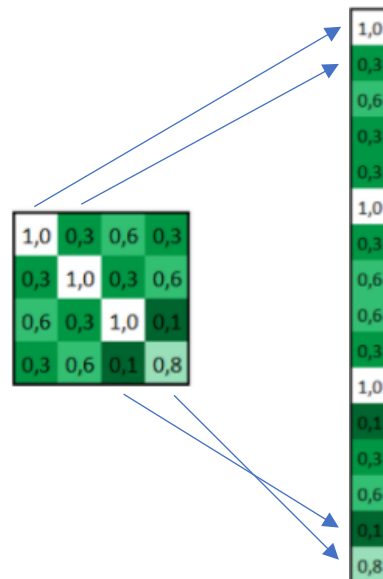
La fonction d'activation ReLU (Unité de rectification linéaire)

Cette étape permet de se séparer des valeurs négatives en sortie de convolution. Toutes les valeurs négatives sont transformées en 0.



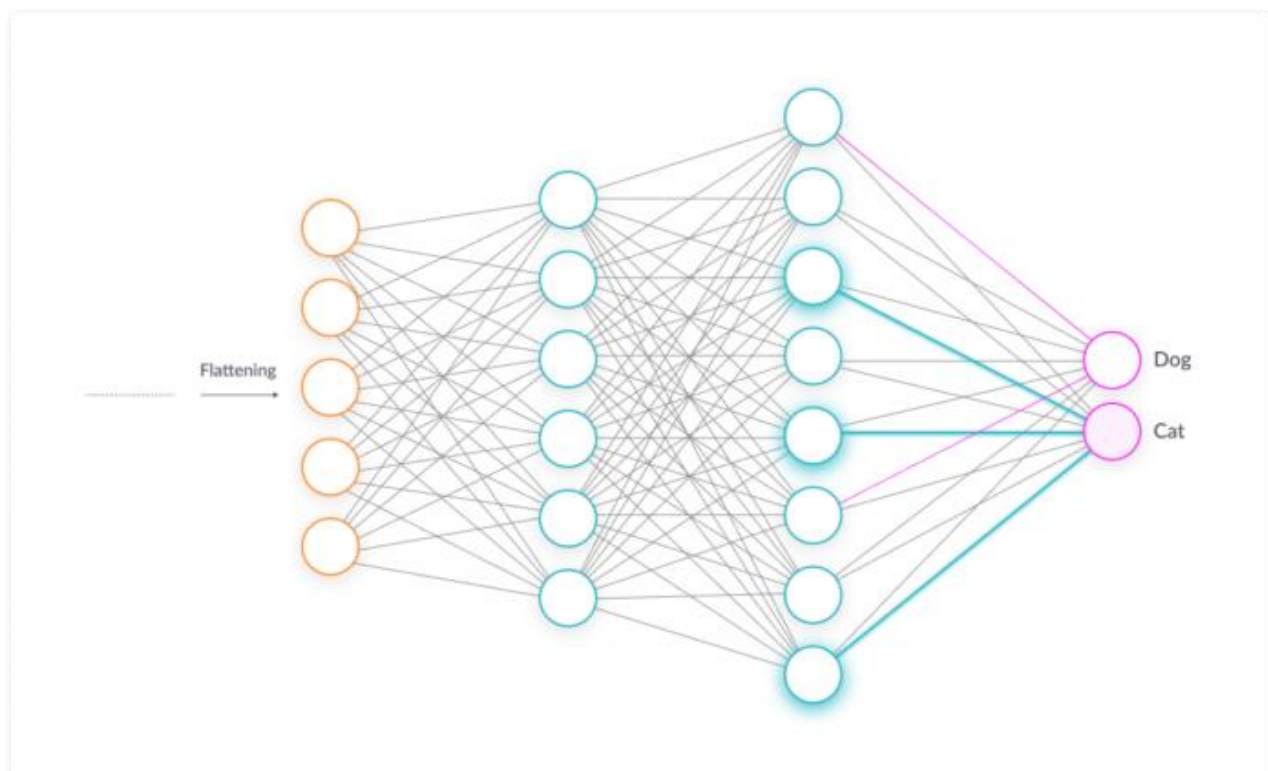
Le flattening

Le flattening est la dernière étape de traitement de l'image, qui consiste à transformer l'empilement des couches de convolution en un seul vecteur. Elle intervient après plusieurs cycles de convolution.



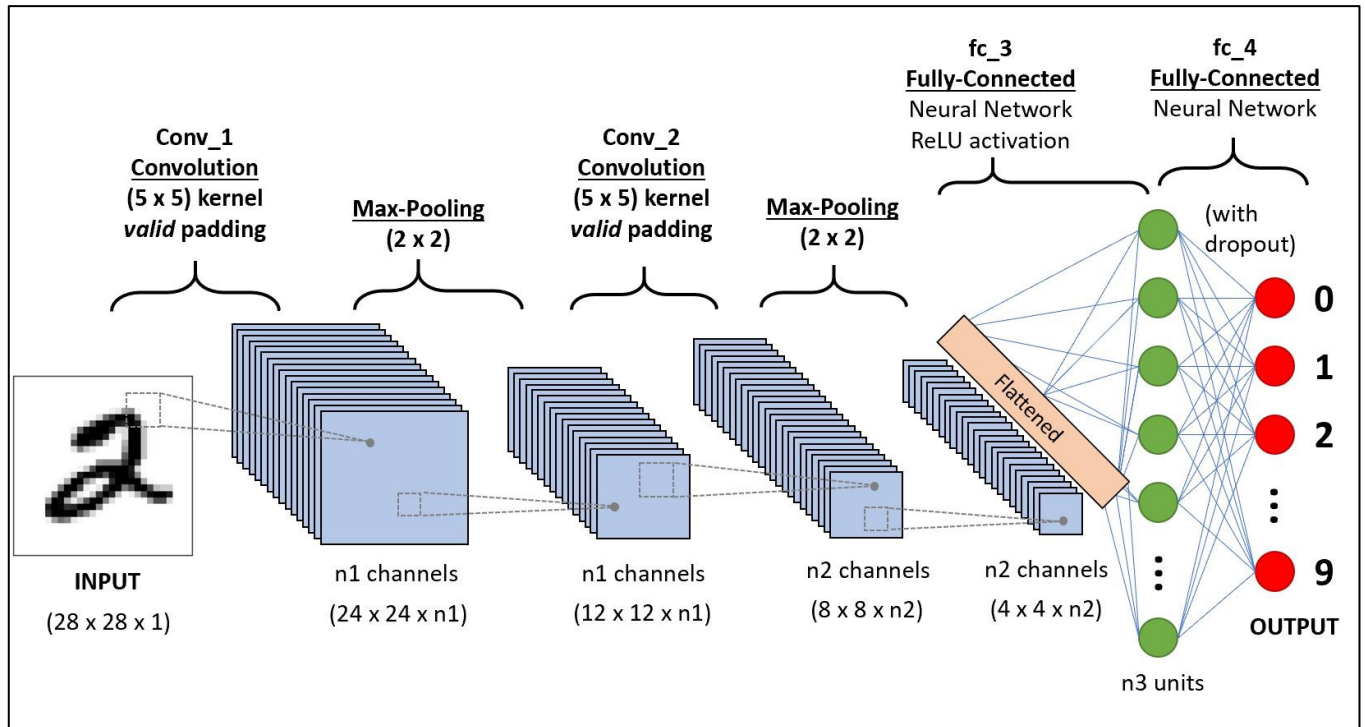
La prédiction

Le vecteur mis à plat, résultante des étapes successives de convolution, est finalement introduit dans plusieurs couches neuronales interconnectées (hidden layers). Les caractéristiques sont évaluées globalement et une prédiction finale de la classe à laquelle appartient l'image est donnée. Le terme *Fully Connected* traduit le fait que chaque neurone d'une couche est relié à l'intégralité des neurones des deux couches adjacentes.



Synthèse

Le réseau de neurones est donc construit de telle sorte que les couches des différentes étapes soient superposées et répétées selon la profondeur du réseau. Le modèle est alors capable, après entraînement, de repérer des caractéristiques clés pour la prédiction, et de réaliser une classification en fonction de son analyse de l'image envoyée en entrée.

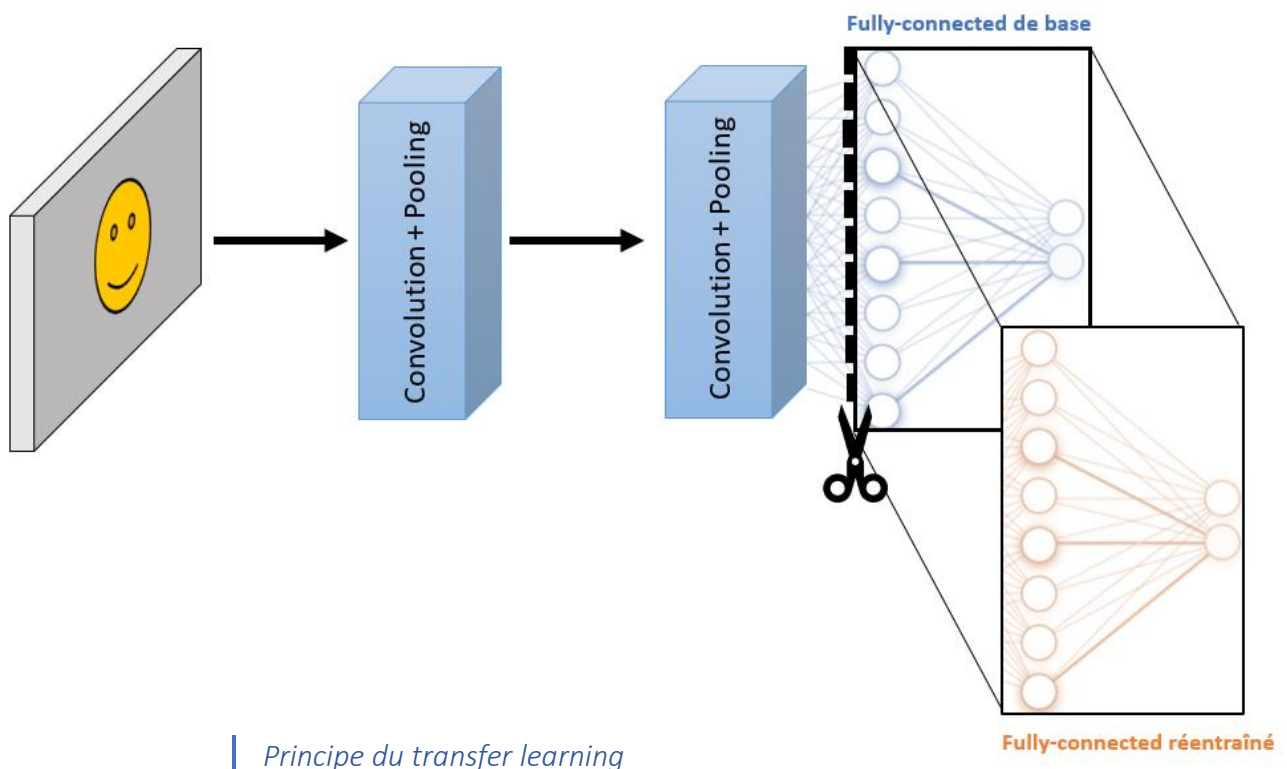


Architecture d'un CNN

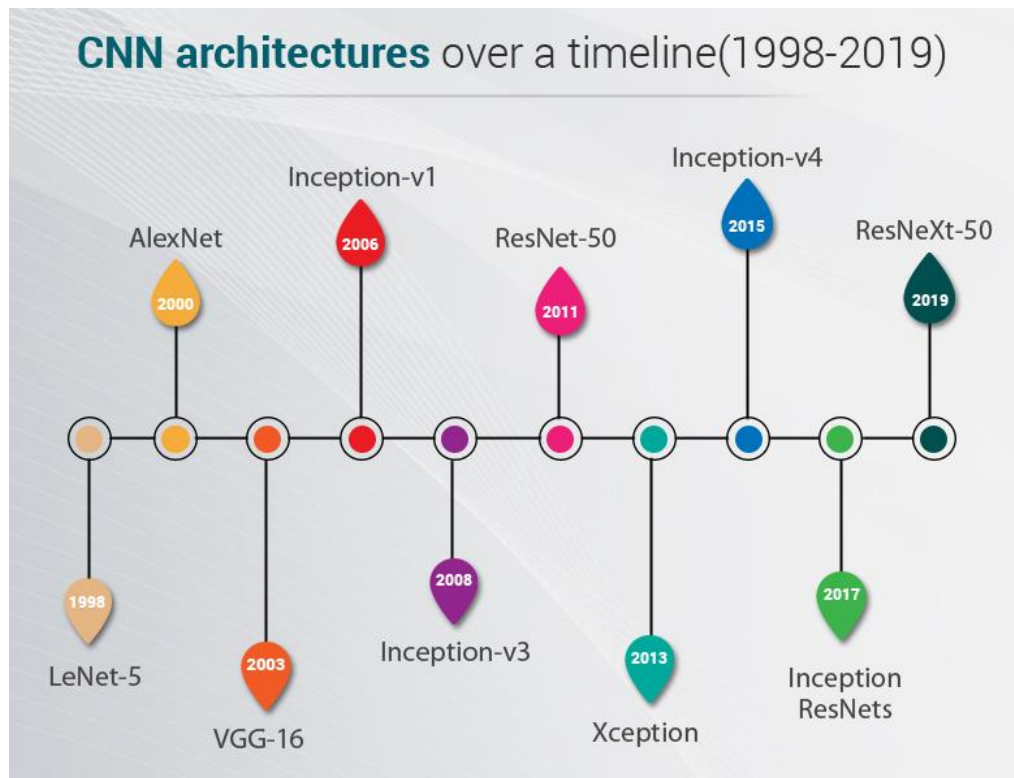
3. Mise en place

Le modèle de réseau neuronal convolutif nécessite un entraînement, c'est-à-dire une modification des poids des paramètres pour apprendre les caractéristiques clés des images. Ceux-ci sont initialisés aléatoirement, puis ajustés de manière empirique au fil du traitement d'un dataset dans le but d'améliorer la précision. Le nombre de paramètres étant très grand (de l'ordre de la dizaine de millions), l'entraînement est très coûteux et demande beaucoup de temps. En effet, tout l'enjeu d'une bonne prédiction réside dans la capacité qu'à le modèle à définir les caractéristiques utiles à détecter dans l'image.

Cependant, il n'est pas nécessaire de construire un CNN dans son intégralité pour développer un modèle de prédiction souhaité. En effet, l'avantage de ce type d'algorithmes est qu'il est modulable et adaptable pour d'autres prédictions. Le *Transfer Learning*, qui correspond à cette transposition de l'algorithme pour une nouvelle prédiction, consiste à conserver les couches de convolution entraînées et de remplacer les couches de prédiction fully-connected en sortie du réseau.



On utilise de ce fait des architectures très performantes qui servent de base à la construction d'algorithmes personnalisés. Pour la suite de ce projet, nous nous sommes appuyés sur les modèles [InceptionV3](#) et [VGG16](#), qui sont largement utilisés et reconnus pour leurs bonnes performances.

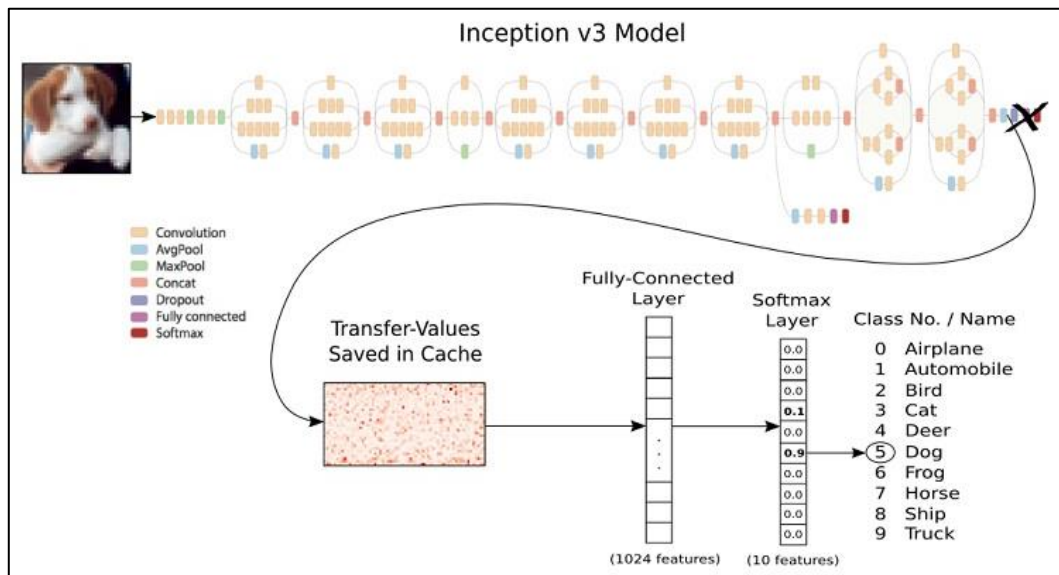


4. Modèles de CNN étudiés

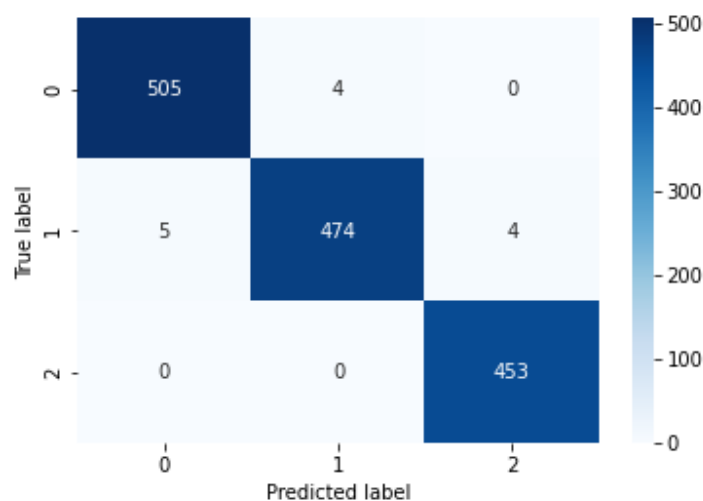
4.1 Modèle pré-entraîné InceptionV3

 [Projet-P5/ Partie3 /InceptionV3/](#)

[InceptionV3](#) est un réseau de neuronal convolutif (CNN) permettant l'analyse d'images et la détection d'objets. Il est profond de 48 couches et peut classer jusqu'à 1000 catégories d'objets. Le modèle requiert une taille minimale des images de 75 par 75 pixels.



Afin d'exploiter ce modèle pré-entraîné, nous avons réalisé un Transfer Learning et obtenu les résultats suivants (sur le jeu de données Test, plus de détails sur celui-ci p21) :

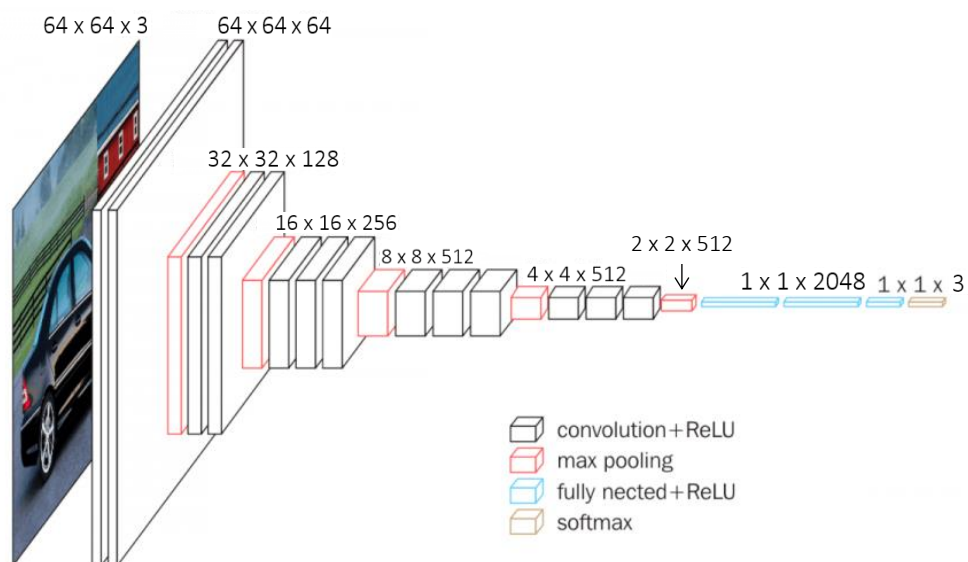


Matrice de confusion

Ainsi, le modèle se trouve être plutôt performant. Néanmoins, nous avons étudié un autre modèle pré-entraîné encore plus adapté pour la détection sur flux vidéo : le modèle [VGG16](#). La partie suivante rentre donc plus en détail sur les démarches entreprises avec ce modèle.

4.2 Modèle pré-entraîné VGG16

[VGG16](#) est une version du réseau de neurones convolutif VGG-Net. Cette version est constituée de plusieurs couches, dont 13 de convolution et 3 fully-connected, et doit donc apprendre les poids de 16 couches. Il prend en entrée une image en couleurs de taille 224x224 pixels (ou 64x64, ce que nous avons choisi) et la classifie dans une des 1000 classes. Il renvoie donc un vecteur de taille 1000, qui contient les probabilités d'appartenance à chacune des classes.

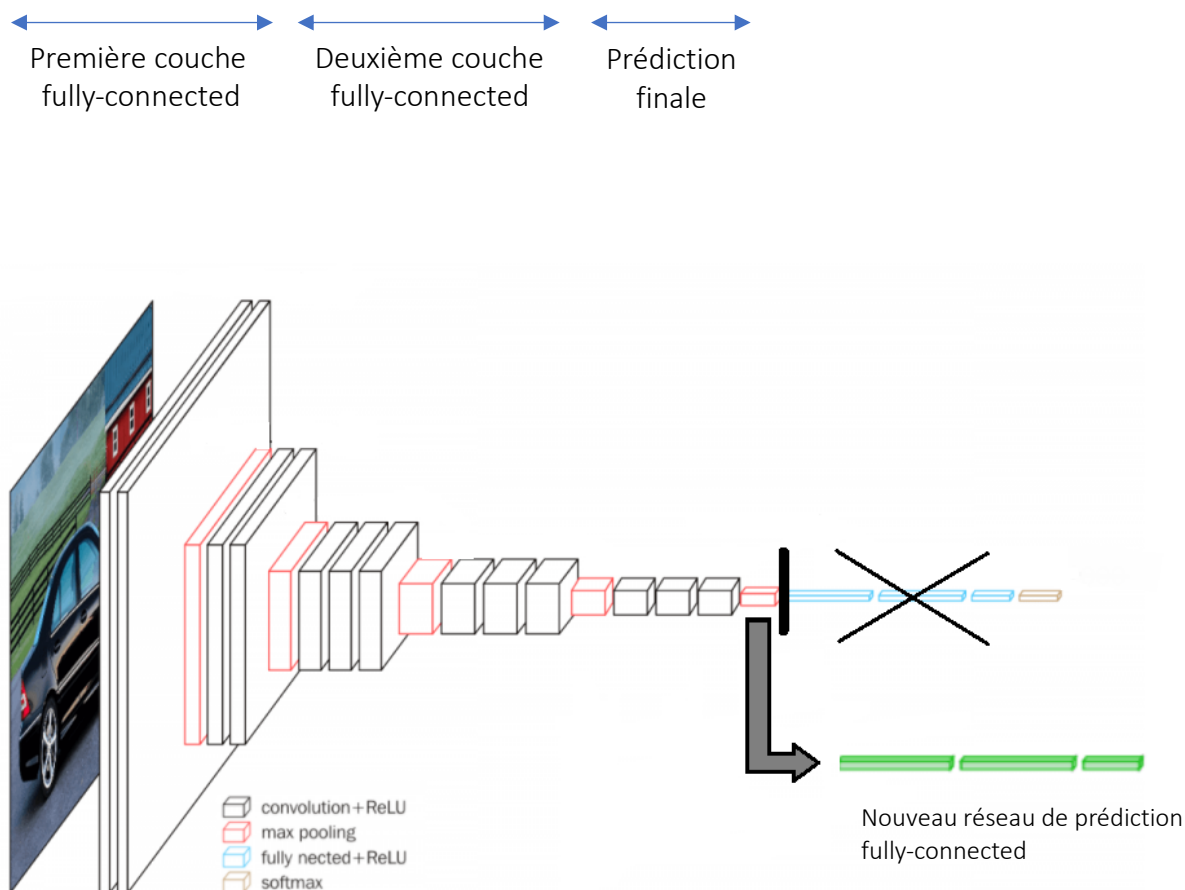


Architecture de base du modèle VGG16

4.3 Adaptation de l'algorithme

Pour notre projet, nous avons conservé les 13 couches de convolution du modèle et remplacé les 3 couches fully-connected par 3 nouvelles, de sorte à ce qu'il renvoie un vecteur de taille 3, contenant les probabilités d'appartenance aux trois classes *WithoutMask*, *WithMask*, *MaskIncorrectlyWorn*.

Sur ce nouveau réseau, nous réalisons uniquement l'entraînement sur les 3 couches ajoutées. Cela correspond à $(2048*2048 + 2048) + (2048*1024 + 1024) + (1024*3 + 3) = 6\,297\,603$ poids à entraîner (cf. p18).



| Méthode du transfer learning appliqué au modèle VGG16

Couches de convolution
bloquées
(pas d'entraînement effectué)

Couches fully-connected
entraînées avec le dataset de
masques

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 64, 64, 3)]	0
block1_conv1 (Conv2D)	(None, 64, 64, 64)	1792
block1_conv2 (Conv2D)	(None, 64, 64, 64)	36928
block1_pool (MaxPooling2D)	(None, 32, 32, 64)	0
block2_conv1 (Conv2D)	(None, 32, 32, 128)	73856
block2_conv2 (Conv2D)	(None, 32, 32, 128)	147584
block2_pool (MaxPooling2D)	(None, 16, 16, 128)	0
block3_conv1 (Conv2D)	(None, 16, 16, 256)	295168
block3_conv2 (Conv2D)	(None, 16, 16, 256)	590080
block3_conv3 (Conv2D)	(None, 16, 16, 256)	590080
block3_pool (MaxPooling2D)	(None, 8, 8, 256)	0
block4_conv1 (Conv2D)	(None, 8, 8, 512)	1180160
block4_conv2 (Conv2D)	(None, 8, 8, 512)	2359808
block4_conv3 (Conv2D)	(None, 8, 8, 512)	2359808
block4_pool (MaxPooling2D)	(None, 4, 4, 512)	0
block5_conv1 (Conv2D)	(None, 4, 4, 512)	2359808
block5_conv2 (Conv2D)	(None, 4, 4, 512)	2359808
block5_conv3 (Conv2D)	(None, 4, 4, 512)	2359808
block5_pool (MaxPooling2D)	(None, 2, 2, 512)	0
flatten (Flatten)	(None, 2048)	0
fc1 (Dense)	(None, 2048)	4196352
dropout_2 (Dropout)	(None, 2048)	0
fc2 (Dense)	(None, 1024)	2098176
dropout_3 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 3)	3075
Total params: 21,012,291		
Trainable params: 6,297,603		

Architecture de l'algorithme

4.4 Entraînement du modèle VGG16

 [Projet-P5/ Partie3 /VGG16/](#)

L'architecture construite doit être entraînée correctement, c'est-à-dire ni trop peu, ni trop conséquemment.

En effet, on utilise un entraînement répétitif par itérations successives (epochs). A chaque epoch, le jeu de données est entièrement exploité pour apprendre. Cette méthode permet d'ajuster au fur et à mesure les caractéristiques du modèle. Il est cependant important de choisir un nombre d'epochs approprié.

Si le nombre d'epochs n'est pas assez grand, le modèle n'aura pas assez eu le temps d'apprendre (underfitting), et la prédiction ne sera pas optimale, car basée sur des critères encore trop vagues.

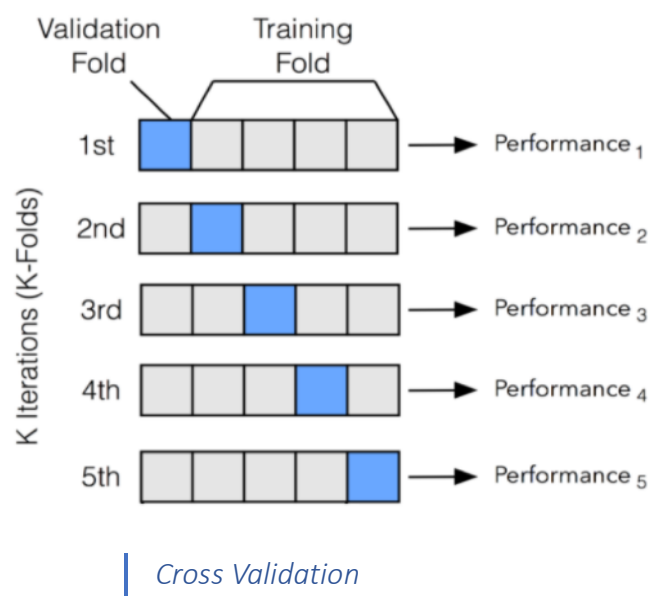
A contrario, lorsque le nombre d'epochs est trop conséquent, le modèle aura "trop" appris (overfitting), ce qui signifie que sa prédiction sera trop spécifique au jeu de données utilisé.

Toute la difficulté d'obtenir une qualité de prédiction maximale réside ainsi dans la détermination du taux d'apprentissage optimal. Le nombre d'epochs étant un paramètre dépendant d'un grand nombre de facteurs, il n'est pas possible de le prédire théoriquement. Pour le trouver, on utilise la méthode de Cross Validation.

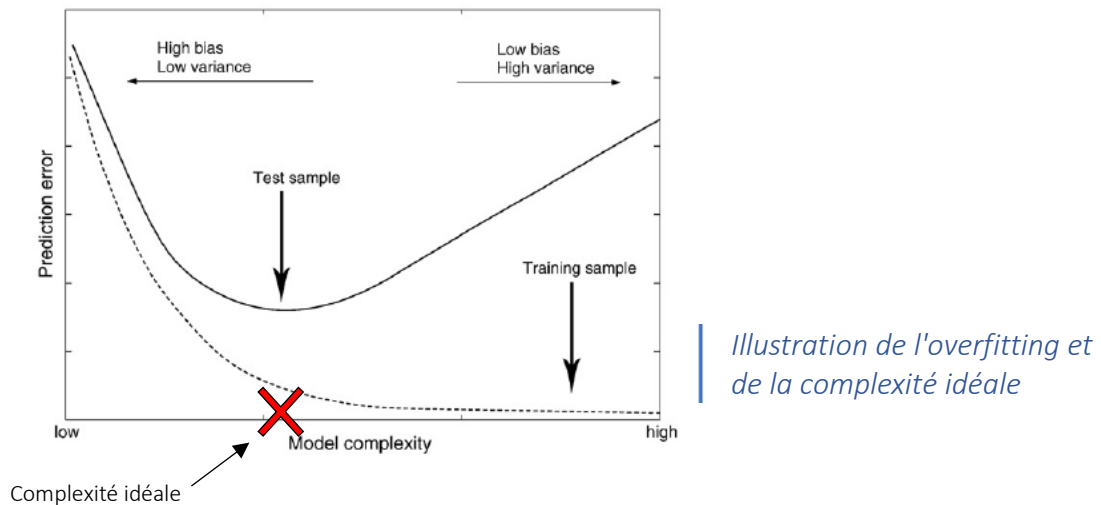
Principe de la Cross Validation

La Cross Validation est une méthode qui repose sur les techniques d'échantillonnage du dataset qui permet de définir la fiabilité d'une prédiction.

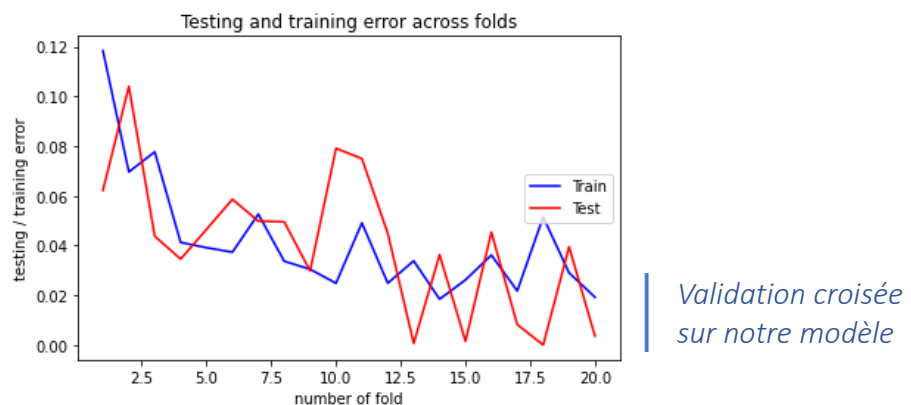
On divise le jeu de données en k folds : le premier va servir à tester la prédiction sur l'apprentissage avec les k-1 autres. Cette action est réalisée k fois, de sorte que chaque fold serve à la validation une fois.



On peut déterminer le comportement du modèle en observant certains résultats de la Cross Validation, notamment les courbes d'erreur sur les folds d'entraînement et de test. Si la courbe d'erreur du test se décolle et augmente par rapport à celle de l'entraînement, cela traduit de l'overfitting. La complexité idéale de l'algorithme définit ainsi un apprentissage maximal sans overfitting.



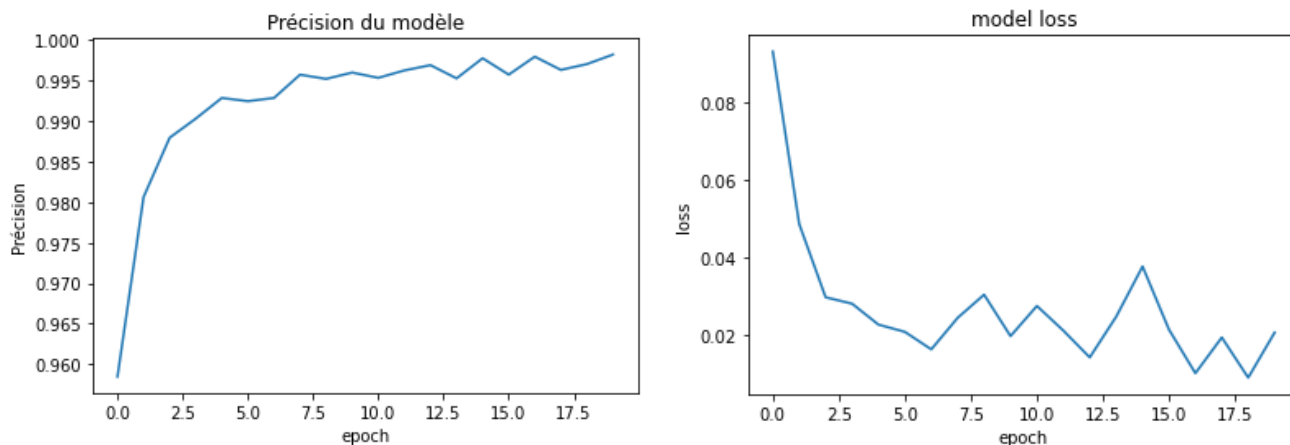
Observation du modèle construit



Nous avons réalisé une Cross Validation en utilisant le jeu de données d'apprentissage (environ 15 000 images). La courbe de Cross Validation indique que, pour les 20 epochs paramétrées, il n'y a pas d'overfitting. En effet, les courbes sont très rapprochées et fluctuent peu (0.035 d'amplitude moyenne de la courbe de test pour un nombre d'epochs supérieur à 12). Il est possible que le nombre d'epochs optimal soit supérieur à 20, mais le temps d'apprentissage de l'algorithme est avec cette valeur déjà très conséquent, et les résultats obtenus sont très bons. 20 epochs est donc la valeur finale que nous avons sélectionnée.

Résultats de l'entraînement

Après entraînement, le modèle est capable d'une prédiction à 99,8 % sur le même jeu de données.

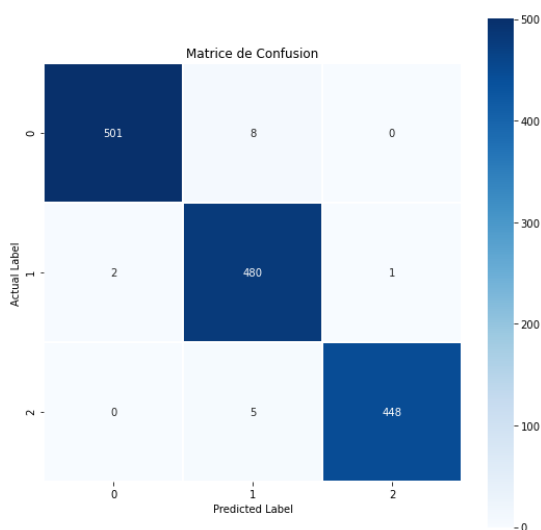


Résultats de l'entraînement
du modèle

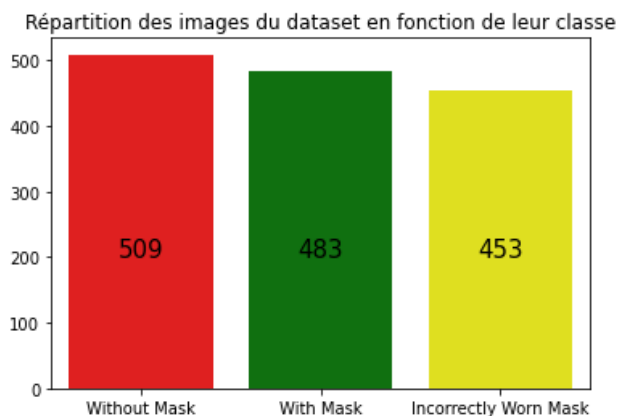
Prédictions sur le jeu de test

Le modèle entraîné est ensuite mis à l'épreuve avec le jeu de test prévu à cet effet, et composé de 1445 images réparties environ équitablement entre les 3 catégories *WithMask*, *WithoutMask* et *IncorrectMask*.

```
Prédiction :  
Nombre d'erreurs : 16 sur 1445 (1.107%)
```



Matrice de confusion

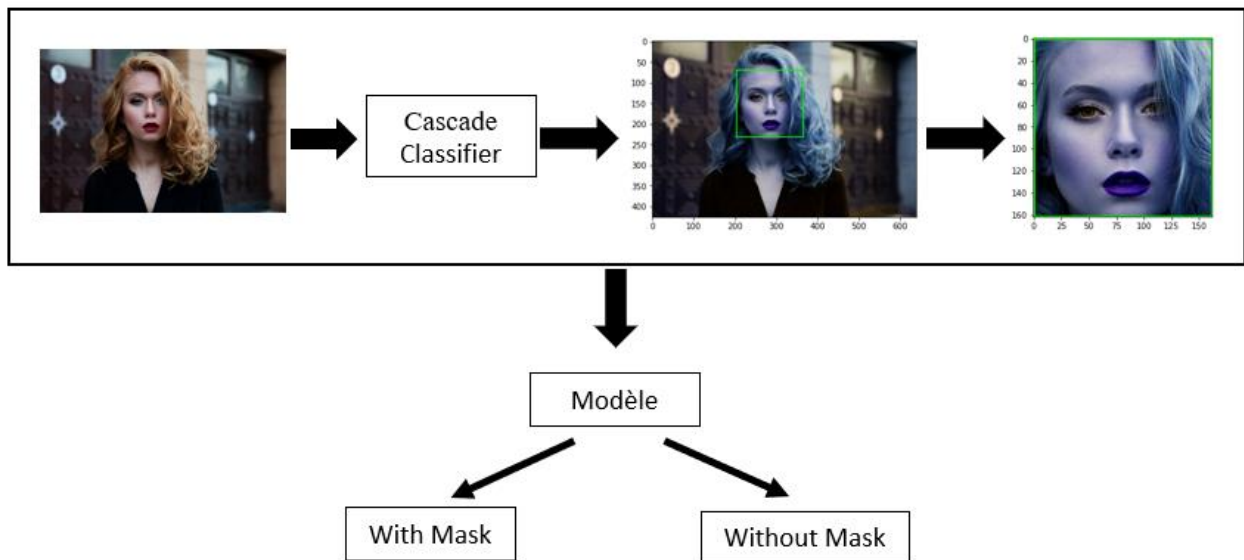


Répartition des données

Partie 4 – Implémentation sur flux vidéo

1. Démarche

L'objectif de cette partie est de lancer un flux vidéo et d'être capable d'afficher les prédictions du modèle [VGG16](#) sur la vidéo lancée en temps réel.

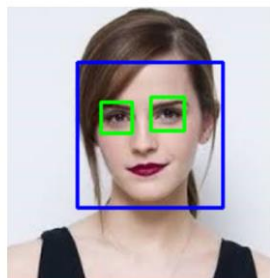


Processus appliqué à l'algorithme

Afin d'obtenir ce résultat, il est nécessaire de placer un détecteur pour la détection des visages en amont du modèle de Deep Learning. Ainsi, le réseau de neurones n'aura qu'à analyser les images avec des visages, et déterminer si ces derniers sont masqués.

2. Détecteur de visages utilisé

Le détecteur en amont du modèle [VGG16](#) est un classifieur [Viola et Jones](#). Ce classifieur est généralement appelé *Haarcascade*. Celui que nous utilisons est dédié à la détection de visages présents sur une image, ou un flux vidéo. Le fonctionnement d'un *Haarcascade* est détaillé en Partie 5.



3. Lancement du flux vidéo



L'objectif est d'afficher les prédictions du modèle en temps réel sur le flux vidéo. L'algorithme dédié à cet effet permet d'afficher des cadres autour des visages détectés, et de les légender de manière à afficher un cadre vert si l'individu porte un masque, un rouge s'il n'en porte pas, et un jaune s'il porte mal son masque. La probabilité associée à cette prédiction est également renvoyée.



Les différentes étapes du code sont les suivantes :

- Détection des visages en temps réel (Haarcascade Viola et Jones) ;
- Prédiction du port ou non du masque en temps réel (Réseau de neurones) ;
- Affichage du résultat sous forme de cadre légendé en temps réel (Label 0, 1 ou 2) ;
- Arrêt et sauvegarde de la vidéo.

Partie 5 – Utilisation d'un classifieur Viola et Jones

1. Principe de la méthode

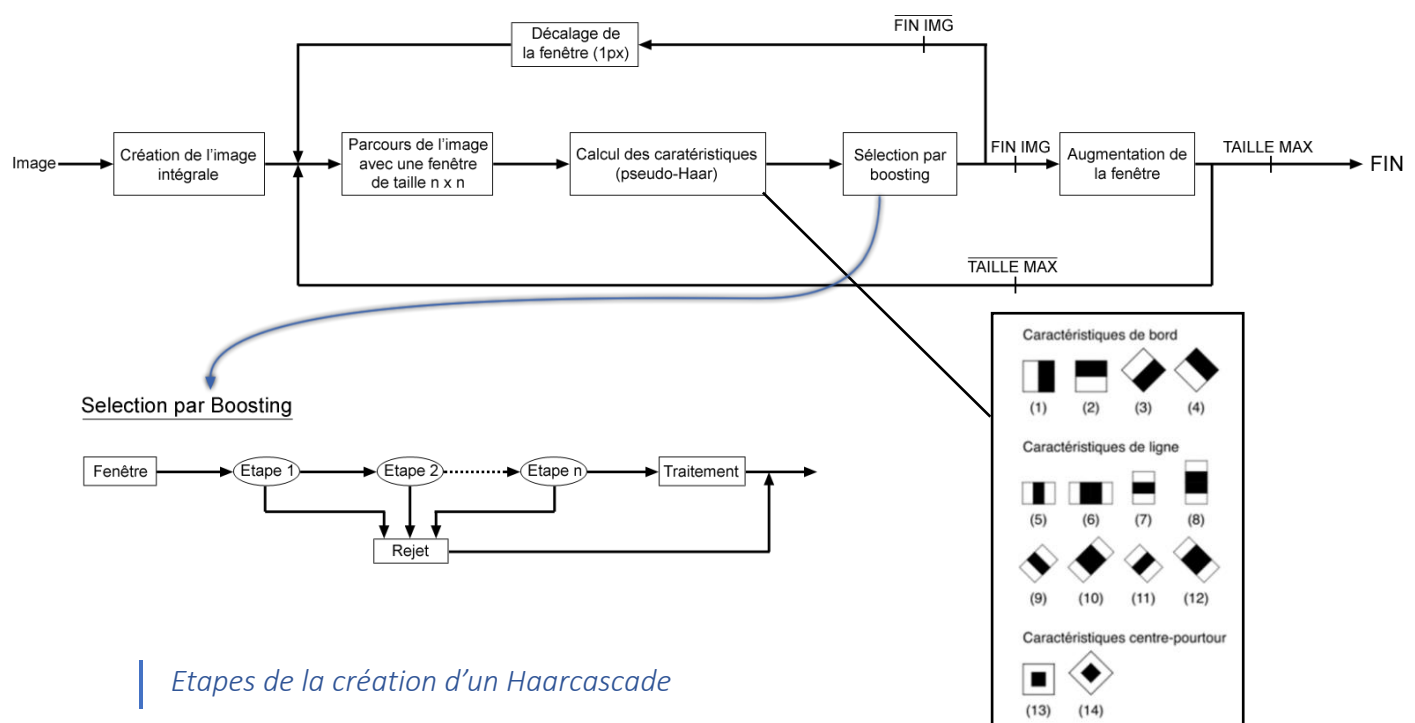
La méthode Viola et Jones consiste à réaliser l'apprentissage d'un classificateur ciblé sur l'élément que l'on souhaite détecter. Pour se faire, de larges bases de données (images positives ou négatives) sont utilisées. Il s'agit d'entraîner le classifieur afin de le sensibiliser à ce que l'on veut détecter. Pour cela, on lui présente d'une part une grande quantité de cas positifs, et d'autre part, une grande quantité de cas négatifs. Il en résulte un classifieur capable de détecter l'objet ciblé, sous la forme d'un fichier .xml.

Les images sont balayées à l'aide d'une fenêtre de détection. Lorsque l'image a été parcourue entièrement, la taille de la fenêtre est augmentée et le balayage recommence, jusqu'à ce que la fenêtre fasse la taille de l'image. Le balayage consiste simplement à décaler la fenêtre d'un pixel. Ce décalage peut être changé afin d'accélérer le processus, mais un décalage d'un pixel assure une précision maximale.

Une **caractéristique** est une représentation synthétique et informative, calculée à partir des valeurs des pixels. Les caractéristiques utilisées ici sont les caractéristiques pseudo-haar. Elles sont calculées par la différence des sommes de pixels de deux ou plusieurs zones rectangulaires adjacentes.

En parcourant l'ensemble de l'image, on calcule un certain nombre de caractéristiques dans des zones rectangulaires qui se chevauchent. On peut calculer les caractéristiques de différentes manières :

- L'utilisation d'images intégrales qui permettent de calculer plus rapidement les caractéristiques : elles contiennent en chacun de ses points la somme des pixels situés au-dessus et à gauche du pixel courant ;
- La sélection par boosting des caractéristiques : la sélection par boosting consiste à utiliser plusieurs classifieurs "faibles" mis en cascade plutôt que d'utiliser un seul classifieur "fort" ;
- La combinaison en cascade de classifieurs boostés, apportant un net gain de temps d'exécution.



2. Création d'un Haarcascade

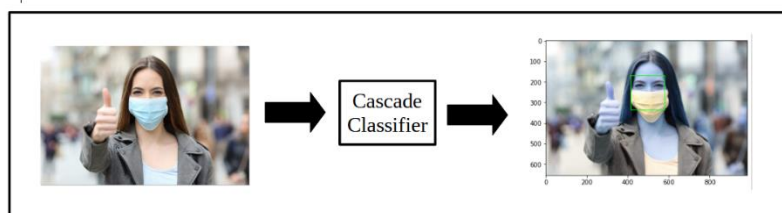
 [Projet-P5/ Partie5/](#)

 [Ressource : Dataset utilisé](#)

Au cours de ce projet, nous avons été amenés à utiliser des classifieurs de type Haar déjà entraînés à la détection de visages. Dans un souci de faire un algorithme de détection de visages de A à Z, procéder de nous-même à l'entraînement d'un classifieur de type Haar s'imposait.

Nous avons donc cherché à obtenir un classifieur qui détecte l'objet 'visage masqué'. La réponse donnée par un algorithme utilisant ce classifieur serait alors binaire : soit l'image étudiée comporte un visage masqué et alors l'objet est repéré et encadré (voir image ci-dessous), soit l'image ne comporte pas de visage masqué (même si dans l'image on trouve un visage ou un masque seul) et alors l'image est classée comme ne comportant pas l'objet recherché.

De ce fait, le modèle étudié ici n'est pas en lien direct avec les travaux précédents. Il est néanmoins important, puisque nous a permis de comprendre en détail le fonctionnement et la création d'un détecteur Haarcascade.

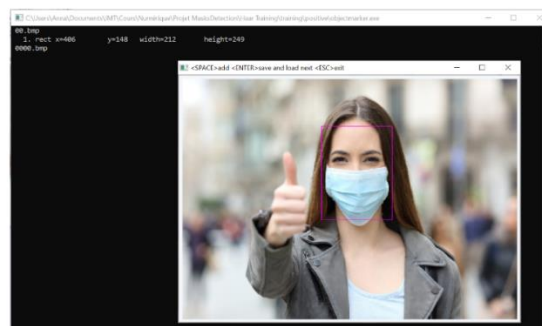


1. Entraînements de classifieurs Haarcascade

Nous suivons donc la trame du TP donné ci-joint :

<https://www.cs.auckland.ac.nz/~m.rezaei/Tutorials/Creating a Cascade of Haar-Like Classifiers Step by Step.pdf>

Un dossier comportant les fichiers exécutables et la structure, nécessaires à la création de notre classifieur, est fourni avec ce TP. Des dossiers 'positive' et 'negative' sont définis dans la structure de fichiers fournie et comportent les images utilisées pour l'entraînement du classifieur. Nous complétons donc ces dossiers avec les images sélectionnées pour la détection de visages masqués et nous saisissons à la main l'objet à détecter sur les images positives.



Nous avons finalement été amenés à entraîner trois classifieurs. En effet, nous trouvions des défauts dans les deux premiers classifieurs entraînés que nous étions en mesure de corriger. Les données utilisées pour l'entraînement des différents classifieurs sont décrites ci-dessous.

- Premier classifieur : 'mymaskedfacedetector0.xml'

Les données utilisées pour l'entraînement sont les suivantes :

- **Images positives** : 581 images de visages masqués données par le GitHub de *prajsnab*. Il s'agit d'images de visages sur lesquels un masque a été ajouté avec Photoshop.



- **Images négatives** : Nous rassemblons 469 images diverses ne comportant pas de visages masqués : des visages non masqués ainsi que des images de bruit diverses (voitures, fruits, fleurs, ...).



Après quelques tests non convaincants faits avec ce classifieur, nous cherchons une manière d'obtenir de meilleurs résultats. Nous décidons donc d'entraîner à nouveau le classifieur avec un jeu de données négatives composé d'images en nuances de même taille. Il semblerait en effet que les classifieurs de type Haar sont entraînés avec ce genre de données négatives.

- Deuxième classifieur : 'mymaskedfacedetector1.xml'

- **Images positives** : Nous utilisons les mêmes images positives que pour le premier test.
- **Images négatives** : Nous utilisons cette fois seulement les images négatives fournies avec le TP (200 images de même taille, en nuances de gris).



Des tests nous permettent d'observer que ce classifieur détecte à 92% les visages masqués lorsque le masque présent sur l'image est similaire au masque utilisé lors de l'entraînement. En revanche, le détecteur ne sait pas détecter les visages masqués lorsque le masque présent sur l'image est différent du masque utilisé pour l'entraînement. Ce défaut est problématique lorsque l'on sait que les masques utilisés sont très variés. Nous entraînons donc un troisième classifieur avec un jeu de données comportant des visages avec des masques variés.

- Troisième classifieur : 'mymaskedfacedetector2.xml'

- **Images positives** : 208 images de visages masqués avec des masques variés : formes, couleurs et motifs différents.

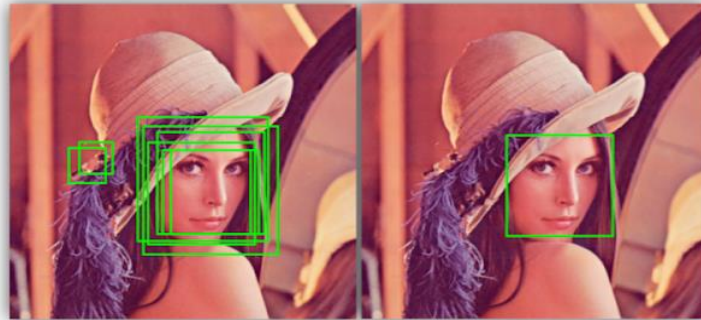


- **Images négatives** : Nous utilisons les mêmes 200 images négatives fournies avec le TP (images de même taille, en nuances de gris)

2. Tests et utilisations des trois classifieurs

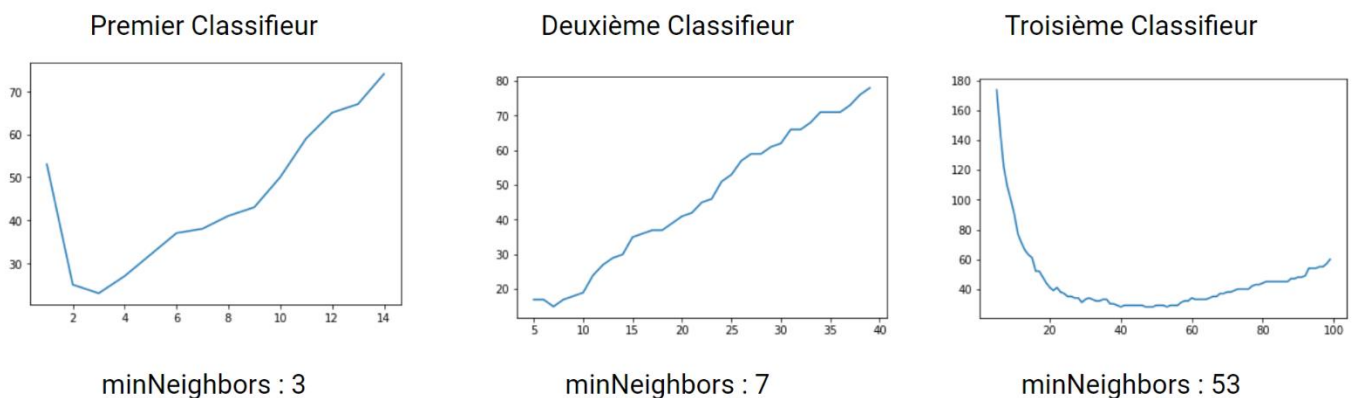
Après avoir entraîné nos trois classifieurs, nous cherchons à évaluer leurs performances. D'une part, leur capacité à détecter correctement un visage masqué lorsqu'il y en a un. D'autre part, leur capacité à discriminer les images ne comportant pas de visages masqués.

Nous utilisons le classifieur avec la bibliothèque *OpenCV*. La détection de 'visages masqués' se fait grâce à la fonction *classCascade.detectMultiScale*. Cette fonction comporte un paramètre *minNeighbors* qu'il est nécessaire d'adapter à chaque classifieur. En effet, ce paramètre spécifie le nombre de voisins minimum que doit avoir un rectangle défini par le classifieur pour être conservé.



Ainsi, si le paramètre *minNeighbors* défini est trop faible, le classifieur détectera plusieurs visages masqués sur une images qui n'en contient qu'un seul et risque de faire une détection sur des objets qui ne sont pas des visages masqués. Au contraire, si le paramètre *minNeighbors* définis est trop élevé, le classifieur ne détectera aucun visage masqué lorsque qu'il devra en repérer un.

Nous commençons donc par déterminer le paramètre *minNeighbors* optimal pour chacun des classifieurs entraînés. Nous cherchons à réduire la différence entre le nombre de visages masqués détectés par le classifieurs et le nombre de visages masqués réellement présents sur une image. Nous obtenons les résultats suivants :



Avec ces résultats, nous pouvons déjà faire des remarques sur le comportement des classifieurs. En effet le premier et le deuxième classifieur ont un *minNeighbors* nettement inférieur au troisième : cela signifie que les deux premiers classifieurs détectent généralement moins d'objets ce dernier. En revanche, cela ne nous donne pas d'informations sur les performances des classifieurs. Le troisième, s'il détecte beaucoup d'objets, peut cependant avoir une tendance à renvoyer beaucoup de faux positifs.

Finalement nous testons les classifieurs sur différents jeux de données :

- Un jeu de données comportant des images de visages masqués où le masque est identique sur toutes les images (voir données d'entraînement du premier classifieur).
- Un jeu de données où les masques portés sont variés : couleurs et formes différentes.
- Un jeu de données d'images variées (qui ont servies de données négatives pour l'entraînement du premier classifieur). On y trouve par exemple des visages non masqués, des fruits, des chats et des motos.

Nous obtenons alors les résultats suivants :

		Premier Classifieur	Deuxième Classifieur	Troisième Classifieur
Paramètre	<i>minNeighbors</i>	3	7	53
Score	Détection des visages masqués (masques identiques)	90%	92 %	76%
	Détection des visages masqués (masques variés)	11%	1%	26%
	Discrimination des images négatives	83%	95%	61%

Remarques :

Le premier classifieur n'identifie pas bien les visages masqués lorsque le masque présent sur l'image est différent du masque qui a été utilisé pour l'entraînement. De plus, il identifie des visages masqués sur 27% des images qui ne comportent pas de visages masqués. Ce résultat est particulièrement étonnant car les données négatives utilisées pour cette phase de test sont les mêmes que les données négatives utilisées pour l'entraînement de ce premier classifieur.

Le deuxième classifieur a une bonne réaction pour la détection du masque qui a servi à l'entraînement. En effet, il identifie à 92% les visages masqués avec ce masque et rejette bien toutes les autres images. Il est donc efficace pour identifier un objet bien précis mais n'a aucune capacité de généralisation : il ne détecte pas du tout les visages masqués lorsque les masques sont variés.

Le troisième classifieur, qui a été entraîné pour détecter des visages masqués avec des masques variés, obtient des meilleurs résultats que les deux premiers classifieurs pour la détection de masques variés mais la performance reste décevante. Ce mauvais résultat s'explique très certainement par la petite taille du jeu de données utilisé pour l'entraînement. En effet, nous pouvons penser que plus l'objet à détecter est variable, plus le jeu de données utilisé pour l'entraînement doit être conséquent.

Conclusion

Au terme des 4 semaines disponibles pour la réalisation de ce projet académique, nous avons été en mesure de mener à bien les objectifs que nous nous étions fixés initialement. Même si tous travaux n'ont pas forcément apporté les résultats espérés, nous sommes fiers et satisfaits d'avoir pu proposer un modèle final fonctionnel et des résultats relativement corrects.

En effet, la conception d'un Haarcascade a été plus difficile que prévu, et nous ne sommes pas en mesure de rendre un produit fini et exploitable sur ce point. A contrario, les travaux réalisés sur le CNN, et tout particulièrement avec le modèle pré-entraîné VGG16, permettent d'obtenir des prédictions sur flux vidéo fiables dans la majorité des cas simples, c'est-à-dire dans des conditions optimales pour la visualisation des visages (luminosité, contraste, distance, etc.).

Cependant, ce travail peut être largement amélioré, notamment sur plusieurs points précis : l'entraînement sur un jeu de données plus conséquent de masques mal portés, la conception d'un détecteur de visages qui reconnaît ceux de travers (de profil ou orientés), ou encore l'exploitation potentielle d'autres modèles de prédiction performants tel que le SVM étudié brièvement en partie 1.

Pour notre part, nous avons beaucoup apprécié travailler sur ce projet qui nous a beaucoup apporté, tant au niveau technique que managérial. Cela constitue en effet pour nous la première vraie expérience sur un projet de groupe de long terme et sans consignes préalables. Nous sommes reconnaissants envers Mr Mennesson pour nous avoir accordé sa confiance, son temps et ses précieux conseils, durant l'intégralité de ce projet. Nous pensons sincèrement que cela nous apportera beaucoup dans nos vies et futures carrières.

Nicolas GAUFFIN
Claire DELGOVE
Anna PASQUIER

Annexe : Inventaire des librairies



Bibliographie

- GITHUB** (2021) Disponible sur <https://github.com/sdhani/face-mask-detection> (Consulté en Février 2021).
- GITHUB** (2021) Disponible sur <https://github.com/chandrikadeb7/Face-Mask-Detection> (Consulté en Février 2021).
- GITHUB** (2021) Disponible sur <https://github.com/cabani/MaskedFace-Net> (Consulté en Février 2021).
- GITHUB** (2021) Disponible sur <https://github.com/amitgupta223/Real-Time-Face-Mask-Detection-using-OpenCV-on-MacOS> (Consulté en Février 2021).
- GITHUB** (2021) Disponible sur <https://github.com/FarhanSadaf/face-mask-detection> (Consulté en Février 2021).
- GITHUB** (2021) Disponible sur http://ethen8181.github.io/machine-learning/model_selection/model_selection.html (Consulté en Mars 2021).
- KAGGLE** (2021) Disponible sur <https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset> (Consulté en Février 2021).
- KAGGLE** (2021) Disponible sur <https://www.kaggle.com/andrewmvd/face-mask-detection?select=annotations> (Consulté en Février 2021).
- WIKIPEDIA** (2021) Disponible sur https://fr.wikipedia.org/wiki/M%C3%A9thode_des_k_plus_proches_voisins (Consulté en Février 2021).
- WIKIPEDIA** (2021) Disponible sur https://fr.wikipedia.org/wiki/Arbre_de_d%C3%A9cision (Consulté en Février 2021).
- WIKIPEDIA** (2021) Disponible sur https://fr.wikipedia.org/wiki/Classification_na%C3%AFve_bay%C3%A9sienne (Consulté en Février 2021).
- WIKIPEDIA** (2021) Disponible sur https://fr.wikipedia.org/wiki/Machine_%C3%A0_vecteurs_de_support (Consulté en Février 2021).
- WIKIPEDIA** (2021) Disponible sur https://fr.wikipedia.org/wiki/M%C3%A9thode_de_Viola_et_Jones (Consulté en Février 2021).
- WIKIPEDIA** (2021) Disponible sur https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework (Consulté en Février 2021).
- WIKIPEDIA** (2021) Disponible sur https://fr.wikipedia.org/wiki/R%C3%A9seau_neuronal_convolutif (Consulté en Février 2021).
- WIKIPEDIA** (2021) Disponible sur <https://en.wikipedia.org/wiki/Inceptionv3> (Consulté en Février 2021).
- WIKIPEDIA** (2021) Disponible sur https://fr.wikipedia.org/wiki/R%C3%A9tropropagation_du_gradient (Consulté en Février 2021).
- MEDIUM** (2021) Disponible sur <https://medium.com/@kenzaharifi/bien-comprendre-lalgorithme-des-k-plus-proches-voisins-fonctionnement-et-impl%C3%A9mentation-sur-r-et-a66d2d372679> (Consulté en Février 2021).
- MEDIUM** (2021) Disponible sur <https://medium.com/@CharlesCrouspeyre/> (Consulté en Février 2021).
- MEDIUM** (2021) Disponible sur <https://medium.com/analytics-vidhya/mtcnn-multi-task-cascade-convolution-network-a9a91755144d> (Consulté en Février 2021).
- MEDIUM** (2021) Disponible sur <https://medium.com/@godsonkkalipe/introduction-au-deep-learning-et-aux-r%C3%A9seaux-de-neurones-pour-les-nuls-de6f8a7b7a35> (Consulté en Février 2021).
- MEDIUM** (2021) Disponible sur <https://medium.com/dataseries/face-recognition-with-opencv-haar-cascade-a289b6ff042a> (Consulté en Février 2021).
- NEPTUNEBLOG** (2021) Disponible sur <https://neptune.ai/blog/hyperparameter-tuning-in-python-a-complete-guide-2020> (Consulté en Février 2021).
- JAVATPOINT** (2021) Disponible sur <https://www.javatpoint.com/machine-learning-naive-bayes-classifier> (Consulté en Février 2021).
- DOCPLAYER** (2021) Disponible sur <https://docplayer.fr/12553079-Le-traitement-d-images-detection-d-objets-par-le-classificateur-de-haar.html> (Consulté en Février 2021).
- JOHEL** (2021) Disponible sur <http://johel.m.free.fr/master/EVA/rdf/cours/Microsoft%20PowerPoint%20-%204-viola%20jones.pdf> (Consulté en Février 2021).
- DATASCIENTEST** (2021) Disponible sur <https://datascientest.com/convolutional-neural-network> (Consulté en Février 2021).
- LEMAGIT** (2021) Disponible sur <https://www.lemagit.fr/conseil/Reseaux-de-neurones-RNN-et-CNN-comment-les-differencier> (Consulté en Février 2021).

BLOG BUSINESS DECISION (2021) Disponible sur <https://fr.blog.businessdecision.com/tutoriel-deep-learning-le-reseau-neuronal-convolutif-cnn/> (Consulté en Février 2021).

TOWARDS DATASCIENCE (2021) Disponible sur <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> (Consulté en Février 2021).

MATHWORK (2021) Disponible sur <https://fr.mathworks.com/help/deeplearning/ref/inceptionv3.html> (Consulté en Février 2021).

OPENCCLASSROOMS (2021) Disponible sur <https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5097666-tp-implementez-votre-premier-reseau-de-neurones-avec-keras> (Consulté en Février 2021).

OPENCCLASSROOMS (2021) Disponible sur <https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5082166-quest-ce-quun-reseau-de-neurones-convolutif-ou-cnn> (Consulté en Février 2021).

NEUROHIVE (2021) Disponible sur <https://translate.google.com/translate?hl=fr&sl=en&u=https://neurohive.io/en/popular-networks/vgg16/&prev=search&pto=aue> (Consulté en Février 2021).

SOFTWAREINTEL (2021) Disponible sur <https://software.intel.com/content/www/us/en/develop/articles/inception-v3-deep-convolutional-architecture-for-classifying-acute-myeloidlymphoblastic.html> (Consulté en Février 2021).

DEEPLYLEARNING (2021) Disponible sur <https://deeplylearning.fr/cours-theoriques-deep-learning/fonctionnement-du-neurone-artificiel/> (Consulté en Mars 2021).

RESEARCHGATE (2021) Disponible sur https://www.researchgate.net/figure/Training-error-and-cross-validation-error-as-a-function-of-model-complexity_fig3_222344717 (Consulté en Mars 2021).

FIREFIY (2021) Disponible sur <https://www.firediy.fr/article/face-tracking-implementation-de-la-methode-de-viola-jones-en-c> (Consulté en Mars 2021).

STATS.STACKEXCHANGE (2021) Disponible sur <https://stats.stackexchange.com/questions/355774/how-to-know-if-model-is-overfitting-or-underfitting> (Consulté en Mars 2021).

STATS.STACKEXCHANGE (2021) Disponible sur <https://stats.stackexchange.com/questions/355774/how-to-know-if-model-is-overfitting-or-underfitting> (Consulté en Mars 2021).

MERIDIAN (2021) Disponible sur <https://meridian.allenpress.com/aplm/article/143/7/859/10038/Artificial-Intelligence-Based-Breast-Cancer-Nodal> (Consulté en Mars 2021).

