Projet : détecteur de signatures

Openclassrooms

Cédric Soares

Octobre 2021

1. Contexte projet

Le présent projet a été réalisé dans le cadre de mon alternance au sein de l'opérateur de télécommunications Orange. De début novembre 2019 à fin octobre 2020 j'ai intégré l'équipe 3MU au sein de l'entité DEVRAP. Sur la même période Vitali Shchutski, autre apprenant de la formation développeur en intelligence artificielle à quant à lui intégré une autre équipe de DEVRAP : CATOP . Nous avons tous les deux travaillé sur le projet.

Les équipes 3MU et CATOP

L'entité DEVRAP (acronyme de développements rapides) fait partie de la Direction de la Technique et du Système d'Information (DTSI) d'Orange France. Composée de 850 personnes, les équipes assurent la maîtrise d'œuvre de projets de développement informatique sur des projets à échéance court voir moyen terme. L'échelle de temps s'échelonne de quelques semaines à quelques mois.

Les équipes 3MU et CATOP sont spécialisées en développement d'automates logiciels utilisant des technologies RPA (Robotic Automation Process) . 3MU développe des automates déployés sur des postes métiers. Ceux développés par CATOP sont déployés sur des serveurs.

À titre d'exemple, certains automates sont utilisés en boutique, lors de la souscription de nouveaux forfaits, pour récupérer les données clients dans différentes applications métiers.

Les équipes 3MU et CATOP sont positionnées comme des centres d'optimisation de coûts. Elles travaillent de concert et partagent le même management. Elles sont composées au total de 16 personnes. Dans le détail :

- 2 managers
- 9 développeu.r.se.s d'automates
- 2 développeur d'intelligences artificielles (Vitali et moi)
- 1 Administrateur système
- 1 DevOps
- 1 Product owner

Courant 2019, les équipes 3MU et CATOP ont fait le choix de monter en compétence sur des sujets d'intelligence artificielle. En effet, les automates sont limités dans leurs capacités au traitement d'informations existantes et à l'application de règles métier. Certains besoins remontés par les entités métier d'Orange France ont mis en lumière la nécessité d'implémenter des briques d'intelligence artificielles,

Vitali et moi-même avons été recrutés afin d'accompagner les équipes dans leur montée en compétence et réaliser des prototypes.

Besoin exprimé pour le projet de détecteur de signatures manuscrites

La direction DC2P France d'Orange souhaite optimiser son workflow de souscription de ses offres sur le segment Pro PME (professions libérales, autoentrepreneurs, PME). Ce segment se caractérise par un process de ventes à la fois moins direct et plus compliqué que le marché des particuliers. D'une part, le souscripteur d'une offre n'est pas forcément l'utilisateur de cette dernière. D'autre part, les contraintes réglementaires sont plus importantes. En résulte un nombre de documents à traiter plus important que pour la clientèle grand public. De plus, sur ce marché, plusieurs démarches, tel que la signature des contrats sont totalement numérisées. Ce qui n'est pas le cas sur le segment Pro PME.

Jusqu'à présent, l'activation des contrats est conditionnée par une vérification manuelle de la signature des documents. Étant donné le contexte, ce procédé génère à la fois un coût important mais aussi un goulot d'étranglement au niveau des étapes de vérification.

Afin d'optimiser l'existant, l'opérateur souhaite automatiser la détection des signatures sur les différents documents commerciaux utilisés sur le segment Pro PME. La direction commerciale d'Orange France a demandé aux DEVRAP (Développements Rapides) 3MU et CATOP de la direction de la technique et du système d'information de réaliser l'étude d'une solution possible et son prototypage.

L'expression de besoin émise la direction DC2P est mise à disposition en annexe I.

Préconisation de solution

Pour répondre à ce besoin, les équipes DEVRAP préconisent de développer un détecteur d'objets basé sur un algorithme de classification mis en œuvre par un modèle deep learning de Computer Vision (vision assistée par ordinateur). Sa finalité est de détecter si les documents scannés contiennent une ou plusieurs signatures. De plus, le cas échéant l'algorithme devra déterminer leur nombre et leur emplacement sur chaque document.

Le prototype sera livré sous forme d'API. Celle-ci pourra être interrogée via une application web ou un démonstrateur motorisé par un kit de développement Jetson Nano équipé d'une webcam.

La solution sera évaluée à la fois par la performance de ses résultats et sa réactivité.

2. Choix de la solution d'intelligence artificielle

Pour répondre au besoin de l'expression de besoin, nous avons préconisé de développer un détecteur d'objets basé sur un algorithme de classification mis en œuvre par un modèle deep learning de Computer Vision (vision assistée par ordinateur). La solution sera évaluée à la fois par la performance de ses résultats et sa réactivité. Pour déterminer le meilleur modèle, nous avons réalisé une étude de l'état de l'art.

Analyse des sources

Voici la liste des sources que l'équipe DEVRAP a utilisé pour procéder à l'étude :

- Papers with code
- arxIv
- Github

Dans le détail :

Papers with code : La plateforme destinée aux chercheurs en machine learning a été lancée en 2018. Selon ses fondateurs Robert Stojnic et Ross Taylor, elle regroupait plus de 18 000 publications et 1 500 classements lors de son acquisition par le laboratoire d'intelligence artificielle de Facebook, en décembre 2019. La source réalise des benchmarks évaluant les meilleurs modèles sur des thématiques spécifiques des différents domaines d'application du machine learning.

arXiv : Le site regroupe plus 1,7 million d'articles scientifiques et est géré par l'université Américaine de Cornell. Il servira à avoir accès à la publication de recherche du modèle retenu.

Github : Le service racheté par Microsoft en 2018 héberge 100 millions de repository et est utilisé par 50 millions d'utilisateurs (chiffres donnés par la plateforme en août 2019). Github sera utilisé pour vérifier l'adoption du modèle retenu par la communauté des développeurs et data scientists.

Que ce soit par leur objet, leur audience ou leur propriétaire, ces trois sources s'avèrent être un gage de confiance quant aux informations que l'équipe DEVRAP a pu analyser.

Benchmark des modèles de détection d'objets en temps réel

Pour évaluer le meilleur équilibre performance réactivité, l'équipe DEVRAP s'est basée sur le benchmark "Real-Time Object Detection on Coco" réalisé par Papers with code en mai 2020.

Le classement évalue les modèles selon trois indicateurs :

- mAP (mean Average precision): moyenne du rapport précision sur recall sur l'ensemble des classes du dataset étudié.
- FPS (Frame par seconde): Nombre d'images par seconde que le modèle peut traiter
- Inference Time: Temps de réponse du modèle en millisecondes.

En s'appuyant sur les critères d'équilibre l'équipe DEVRAP s'est fixé, YoloV4 apparaît être le modèle offrant le meilleur équilibre performance / réactivité.

En effet, d'après le benchmark réalisé sur le dataset Microsoft COCO, Yolov4, sur des images d'entrée de dimensions 512x512, arrive 7e du top 20 pour la métrique avec un mAP à 43 alors que le premier EfficientDet à 55.1.

Sur ce simple critère, le modèle fait moins bien que d'autres comme EfficientDet par exemple et ne paraît donc pas être le meilleur choix. Par contre sur les deux items que sont le FPS et l'Inférence Time, suivant Volov4 arrive premier du top 20

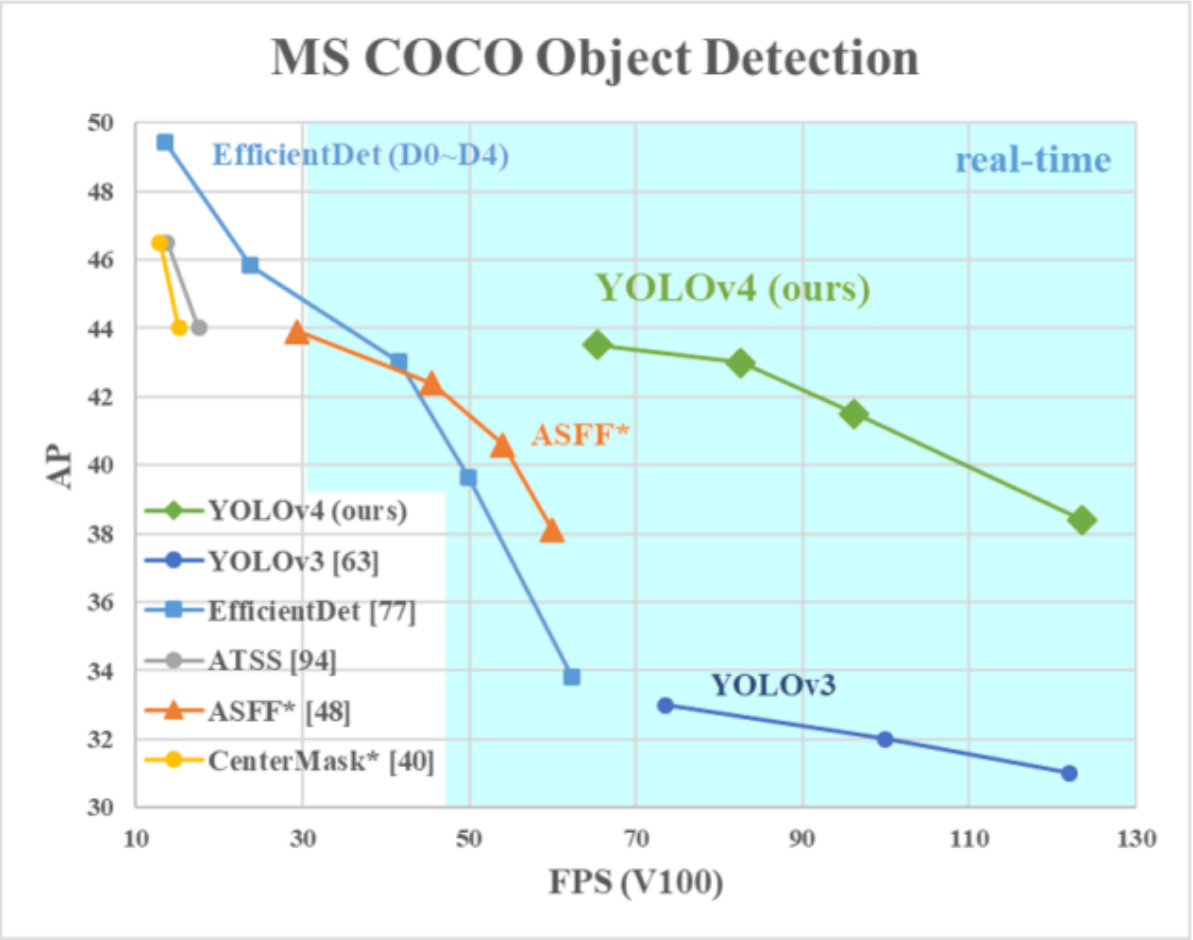
En considérant le nombre d'images par seconde, avec un FPS à 83 sur des images de 512x512, Yolov4 offre un écart significatif avec le modèle suivant dans le top 20 : CSPResneXt. Ce dernier affiche une performance de 58 FPS.

Voici le récapitulatif des informations extraites du becnchmark.

	Score mAP (Position)	**Score FPS (Position)**	**Score Inference Time (Position)**
EfficentDet-D7x	55.1 (1e)	6.5 (18e)	Pas de mesure dans le benchmark
CSPResneXt	33.4 (17e)	58 (3e)	17 (3e)
YolovoV4 - 512	43 (7e)	83 (1e)	12 (1e)

Comparaison des performances dans l'article de recherche

La publication des travaux d' Alexey Bochkovskiy, Chien-Yao Wang et Hong-Yuan Mark Liao, co-créateurs de YoloV4, a été soumise sur arXiv le 23 avril 2020.
Dans leur article, les auteurs ont détaillé le rapport performance / FPS entre YoloV4 et d'autres modèles à la pointe de la recherche dont notamment EfficentDet plus performant selon le benchmark de Papers with code.



Comparaison du rapport performance / FPS avec un GPU Tesla V100 entre YoloV4 et d'autres modèles de deep learning à la pointe de la recherche. *Source article de recherche* ***YOLOv4: Optimal Speed and Accuracy of*** *Object Detection (publié sur arXiv le 23 avril 2020)*

Il apparaît au vu de la courbe que les performances de Yolov4 baissent beaucoup moins vite avec l'augmentation du FPS que EfficentDet. Le modèle reconnaît deux fois plus à un niveau de performance équivalent selon la publication.

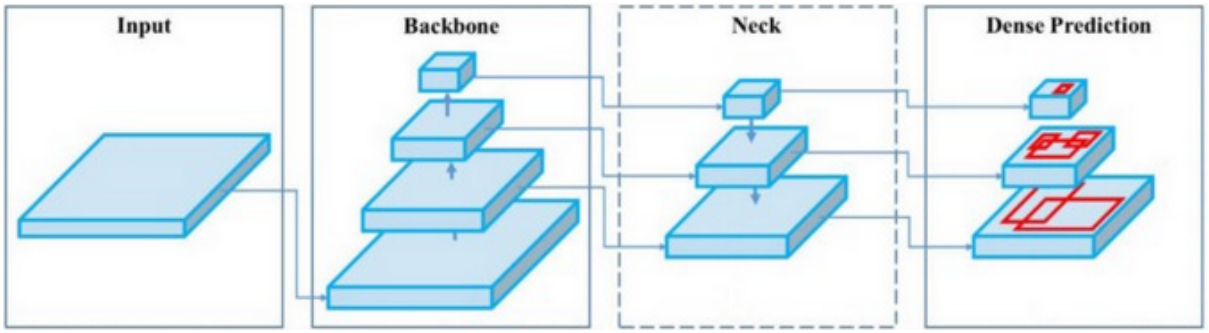
Adoption du modèle par la communauté

En plus de répondre au critère de sélection, le modèle est également largement utilisé par la communauté des data scientists. D'après Papers with code, hormis la publication originale, depuis avril 2020, Yolov4 a été utilisé dans 144 autres articles de recherche.

De plus, d'après les statistiques Github le repository lié aux travaux de recherche Alexey Bochkovskiy est sauvegardé par 13 600 utilisateurs de la plateforme et copié pour modifications par 14 900 utilisateurs.

Architecture de YoloV4

Yolov4 se décompose en trois blocs fonctionnels comme le détaille de le schéma tiré de la publication d' Alexey Bochkovskiy, Chien-Yao Wang et Hong-Yuan Mark Liao.



Architecture du modèle VoloV4. *Source article de recherche* ***YOLOv4: Optimal Speed and Accuracy of*** *Object Detection (publié sur arXiv le 23 avril 2020)*

- **Backbone :** extrait les features de l'image soumise en entrée. Il fournit en sortie une carte de l'ensemble des features présentes dans l'image.
- **Neck :** identifie les features pertinentes.
- **Dense prediction :** identifie la position des features utiles en traçant des bounding box autour. Pour chaque, il donne la nature de l'objet détecté.

3. Données utilisées pour l'entraînement du modèle

Construction de la base de données analytique

Dans le cadre du projet, la DC2P n'a pas fourni de scans de documents commerciaux. Ce choix s'explique par la présence de données personnelles liées à des clients d'Orange tel que : les coordonnées bancaires ou l'adresse postale par exemple. Il fait d'autant plus de sens que les entrainement des modèles n'ont pas été réalisés sur des matériels ou des machines virtuelles intégrées au système d'information de l'opérateur.

Nous avons donc recherché des sources de données répondants aux critères suivants:

- Données en accès publique
- Documents scannés de nature commerciale : contrats, baux
- Une partie des documents comporte des signatures manuscrites

La base de données analytiques que nous avons assemblées en compte trois :

- **Tobacco-800 :** Dataset de 1290 documents, dont 789 annotés, issus de l'industrie du tabac. Constitué par Guangyu Zhu, Yefeng Zheng, David Doermann and Stefan Jaeger, chercheurs au laboratoire LAMP (Language and Media Processing) de l'université du Maryland aux Etats-Unis. Sa première version a été mise en ligne en 2006 et la dernière mise à jour a été réalisée en 2011.

- **Nanonets :** Dataset de 174 documents issus des moteurs de recherche Google et Bing mis à disposition par Nanonets sur son compte Github. L'entreprise, spécialisée dans les OCR a utilisé le dataset dans un démonstrateur de son API en 2018.

- **GSA Lease documents :** Ensemble des baux commerciaux contractés avec l'administration des services généraux du gouvernement américain afin de fournir des bureaux aux agents fédéraux. Les documents sont classés régions regroupant plusieurs Etats. Nous avons utilisé les bases des 6 régions les plus à l'est du pays. Cet ensemble représente 772 documents.

Dans le cadre du projet nous avons entraîné YoloV4 à trois reprises avec des fichiers de poids et de configuration initiaux afin d'en améliorer les performances. Le premier entraînement a été réalisé sur le dataset Tobacco-800 seul. Nous l'avons ensuite enrichi avec les deux autres sources lors des entraînements suivants. Les détails des performances sont fournis dans la section du rapport.

Les documents fournis Nanonets et GSA Lease documents n'étant pas annotés, nous avons utilisé la librairie LabelImg.

Exploration des données

Pour les besoin de l'exploration des données, nous avons regroupé les métadonnées des documents suivantes :

- **name :** suite hexadécimale représentant le nom du document
- **height :** hauteur du document
- **with :** largeur du document
- **is_color :** indique si le scan document est en noir et blanc (0) ou en couleurs (1)
- **number_of_signatures :** nombre de signatures sur le document
- **source :** source où est tirée le document

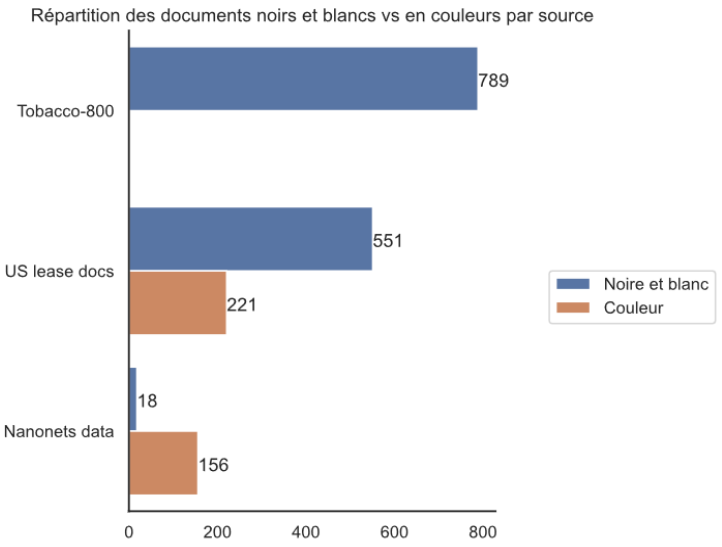
La dernière version de l'exploration présentée dans le rapport regroupe l'ensemble des sources utilisées.

En voici les statistiques descriptives.

	Unnamed: 0	height	width	is_color	number_of_signatures
count	1735.000000	1735.000000	1735.000000	1735.000000	1735.000000
mean	867.000000	2243.940634	1734.374640	0.217291	2.004035
std	500.995675	650.636099	507.187124	0.412521	1.215284
min	0.000000	408.000000	380.000000	0.000000	1.000000
25%	433.500000	2183.000000	1696.000000	0.000000	1.000000
50%	867.000000	2201.000000	1708.000000	0.000000	2.000000
75%	1300.500000	2292.000000	1728.000000	0.000000	3.000000
max	1734.000000	4200.000000	3506.000000	1.000000	8.000000

Statistique descriptive des documents du dataset d'entraînement. Source : notebook ad hoc

Premier constat : malgré la multiplication des sources, la proportion de documents en couleur est de 21,7%. Augmenter le nombre d'observations avec une modalité is_color = 1 est non seulement difficile mais surtout très chronophage. D'une part, le nombre de datasets de documents signés accessibles sont rares. D'autre part, le détail de la coloration des documents n'est généralement pas détaillé. Il nécessite une exploration empirique de chaque dataset. En examinant plus en détail cette caractéristique.



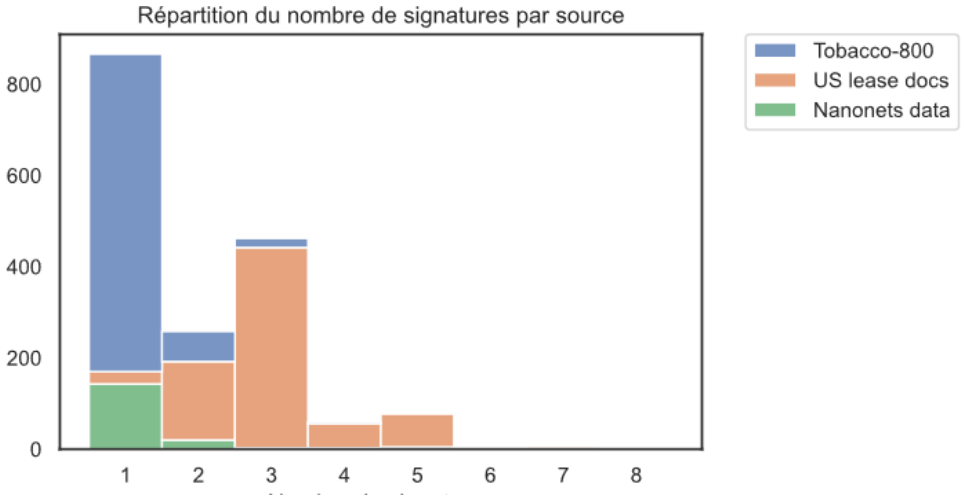
Répartition des documents noirs et blancs vs couleurs par source. *Source : notebook*

Sur le dataset nous avons initié ce travail de sélection d'images en couleurs. Notons l'absence de documents en couleur du dataset Tobacco-800. Nous avons émis deux hypothèse pouvant l'expliquer :

- Une normalisation en noir et blanc par l'équipe sceintifique qui a constitué le dataset
- Un dataset plus ancien constitué de documents b2b

Nos travaux ne nous ont pas permis de valider l'un ou l'autre des hypothèses.

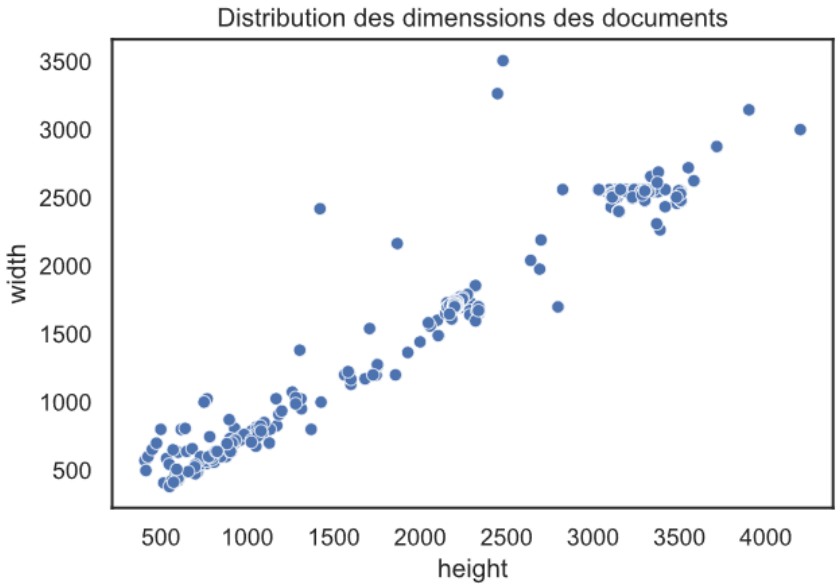
En observant la répartition du nombre de signatures par sources nous obtenons :



Répartition du nombre de signatures par sources. *Source : notebook*

Nous constatons un nombre de signatures par documents sensiblement plus élevé sur le dataset GSA US Lease. Le phénomène peut s'expliquer par la nature des documents. Les baux commerciaux nécessitent un nombre de signatures plus important par documents. De manière empirique nous avons également pu observer que le nombre de signataires par document est également plus important sur ce dataset.

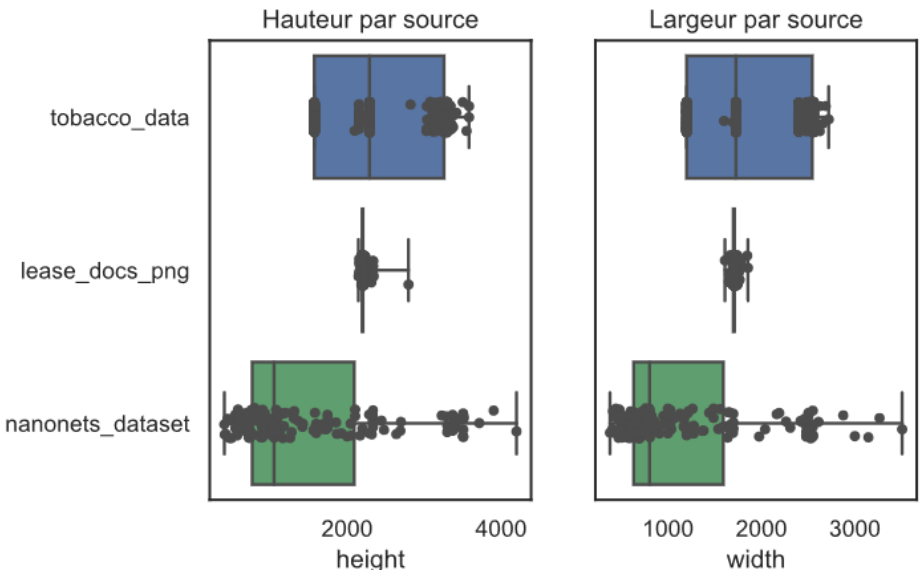
Observons maintenant la distribution des largeur par hauteur sur l'ensemble des documents.



Distribution des dimensions des documents. *Source: notebook*

On constate que la distribution du rapport hauteur / largeur est majoritairement linéaire. Cela s'explique par le fait, que malgré des dimensions différentes, les images, tirées de scans, sont quasiment toutes issues de documents au format A4.

Enfin observons les distribution des dimensions par source :



Distribution des dimensions par source. Source: notebook

Nous constatons une distribution très contenue des dimensions des documents issus du datatset GSA Lease. Le phénomène peut s'expliquer par le fait qu'une administration fédérale aurait tendance à normaliser la taille des documents qu'elle sauvegarde. A contrario la dispersion sur le dataset Nanonets, constitué à partir de résultats de moteurs de recherches, est beaucoup plus importante.

Le notebooks qui a servi à réaliser les visualisations est disponible en annexe IV.

Nettoyage des données pour l'entraînement

Pour les besoins de l'entraînement, les actions de nettoyage et préprocessing se sont limitées à un redimensionnement des images en 608 x 608 pixels comme le préconise le github officiel du modèle. Pour atteindre cet objectif nous avons utilisé la version compilée pour Python d'OpenCV.

Voici la portion de script utilisée:

```
import os
import cv2

cwd = os.getcwd()

width = 608
height = 608

for filename in os.listdir(cwd):
    source_image = cwd + '/'  +  filename + '.png'
    source_raw_image = cwd + '/' + filename + '.png'
    destination_resized_image = cwd + 'resized_' + filename + '.png'
```

```
    img = cv2.imread(source_image, cv2.IMREAD_UNCHANGED)
    resized = cv2.resize(img, (width, height))
    cv2.imwrite(destination_resized_image, resized)
```

4. Entrainement du modèle

Process mis en oeuvre

Le pilotage de l'entraînement réalisé par Darknet qui fait office de backbone de YoloV4. Le framework est appelé en lui passant les fichiers suivants :

- **detector :** Contenu dans le container du script d'entraînement. Objet contenant l'architecture de YoloV4..
- **obj.data :** Construit en extrayant l'ensemble des images des datasets train et valid de la collection images base de données. Il comprend : Des objets à détecter, le nombre de classes et les listings distincts des fichiers du dataset train et valid (images + labels).
- **culstom-yolov4-detector.cfg :** Extrait de la collection configs de la base de données. Fichier de configuration listant les hyperparamètres du modèles tels que les dimensions des images, leur angle de rotation, le learning rate ou le nombre maximum que le modèle va utiliser pour entraîner le modèle.

Darknet est compilé dans un container dédié à l'entraînement où sont également présent le notebook contenant le script. La base MongoDB contenant les images est hébergée dans un autre container.

Analyse des résultats

Sur l'ensemble du projet, nous avons entraîné à trois reprises la version Yolov4 préentrainée vierge :

- Premier entraînement sur la base dataset Tobacco-800 seul : 789 images
- Second entraînement sur la base dataset Tobacoo-800 (789 images) complété par le dataset Nanonets (174 images) et 150 images collectée pour la dataset USA GSA Lease soit un total de 1113 images
- Troisième entraînement sur l'ensemble des images Tobacco-800, Nanonets et USA GSA Lease que nous avons collecté soit 1735 images

L'ajout d'images, avec parmi celles-ci des scans en couleur, a permis d'augmenter les performances du modèle. Voici le tableau regroupant les résultats des tests réalisés sur le datatset de validation au terme de chaque entraînement :

	mAP	**Precision**	**Recall**	**F1 Score**
Premier entraînement	29.57%	44%	25%	32%

Deuxième entraînement	65.06%	52%	75%	62%
Troisième entraînement	68.20%	71%	69%	70%

On constate que chaque nouvel entraînement a vu augmenter le mAP. Pour rappel il s'agit du rapport Precision / Recall moyen. L'ajout de scans en couleur et le passage de un à trois canaux de couleurs dans les hyperparamètres lors du second entraînement ont permis d'augmenter très sensiblement les performances du modèle sur l'ensemble des critères d'évaluation.

L'augmentation significative du nombre de scans lors du troisième entraînement s'est certes traduite par une augmentation du mAP, de la précision et du F1 Score. Le Recall a toutefois baissé. On en déduit que le modèle est moins capable de détecter un document où il y a effectivement une ou plusieurs signature(s) sur l'ensemble des documents qui sont effectivement signés. En ramenant le prototype a son enjeu business, il y a un risque plus important que des processus de souscription soient bloqués faute de signature détectés alors qu'ils auraient dû se poursuivre.
Lors des tests que nous avons réalisé nous avons pu constater que le modèles avait tendance à se tromper la détection ou le nombre de signatures détectées dans deux cas précis :

- Les signatures qui ont fait l'objet d'un coup de tampon
- Les signatures entourées par des annotations manuscrites. Ce cas se présente dans le cas de mentions légales recopiées par exemple.

Faute de contact avec les équipes métiers à la source du besoin, nous ne sommes pas en mesure de dire si ces cas sont courants ou non en situation réelle. De plus, le jeu de données que nous avons constitué comporte un biais vis-à- vis des documents collectés. Pour ces deux raisons, nous avons décidé de ne pas poursuivre les entraînements. Il serait judicieux de réentraîner le modèle avec un nombre significatif de documents issus de la direction DC2P.

5. Intégration du modèle dans une API

Conformément à l'expression de besoins, nous avons développé une API contenant le modèle. Nous avons utilisé le framework python FastAPI pour le réaliser. Celui-ci propose plusieurs avantages :

- Une définition des payload attendus par requête simplifiée, sous forme de classes.
- Une documentation OpenAPI/swagger intégrée
- Un temps nécessaire au développement de la solution plus court que Flask

Fonctionnement de l'API

L'API prend en entrée un JSON contenant une image transformée en String Base64. Elle retourne sous forme de JSON les coordonnées des bounding box des signatures détectées et leur nombre. L'opération est réalisée via la route /predict.

La prédiction est réalisée à l'aide d'openCV qui utilise le modèle YoloV4 entraîné. Nous réalisé à la fois des tests unitaires et fonctionnels :

- Tests unitaires : api renvoie bien un code HTTP 200 ainsi qu'un JSON contenants des coordonnées lorsqu'elle est appelée
- Test fonctionnel : lorsqu'une image encodée est envoyée à l'API, l'application front est capable de décoder les bounding box renvoyées par l'API et les afficher sur l'image envoyée.

Annexes

Annexe I : Expression de besoin

Expression de besoin : détection automatisée de signatures manuscrites sur des documents commerciaux

Direction DC2P : Pro PME — Orange France

Mai 2019

Table des matières

Contexte

La DC2P — Direction des marchés Pro PME d'Orange France

La DC2P d'Orange France adresse les marchés Pro regroupant les professions libérales, autoentrepreneurs et celui des PME. Les terminaux et forfaits commercialisés sur ces segments de marché sont proches, voire similaires, par ceux proposés sur le marché grand public. Toutefois certaines particularités caractérisent ces marchés :

- Les niveaux de QoS (Qualité de service) et SLA (niveau de support) attendus sont supérieurs au marché grand public
- Les produits peuvent être commercialisés dans le réseau grand public, mais aussi dans des agences spécialisées
- Les process de ventes sont moins directs que sur le marché grand public. Le prescripteur, l'acheteur, l'utilisateur et le payeur ont une probabilité d'être distincts beaucoup plus forte.
- Les contraintes réglementaires sur ce marché sont plus élevées.

Le besoin

Les contraintes liées aux marchés Pro PME font qu'à date, le traitement des documents commerciaux relatifs à la signature de forfaits se fait exclusivement de manière manuelle à date. Le processus mis en œuvre est à la fois consommateur d'un nombre important d'ETP (équivalent temps plein) et peut engendrer un nombre significatif d'erreurs lors du traitement.

Dans ce cadre, la validation de la présence de signatures est un prérequis à l'initiation de la validation des contrats. L'opération est positionnée en amont et sur le chemin critique des autres opérations. Toute anomalie peut dès lors être synonyme d'allongement des process voire d'incohérences dans le système d'information.

La DC2P souhaite donc mandater les équipes DEVRAP 3MU et CATOP pour étudier la possibilité d'automatiser la détection des signatures.

Les objectifs

Les équipes DEVRAP devront étudier une solution et présenter un prototype répondant aux objectifs suivants :

- Présenter une solution technique retenue
- Développer un prototype de solution permettant de détecter les signatures manuscrites sur des documents
- La solution devra indiquer la position des signatures sur les documents, leur nombre et le cas échéant l'absence de signature.
- La solution devra pouvoir être supervisée par un opérateur ou être intégrée comme une brique de traitement dans le système d'information.

Les futurs utilisateurs

Le prototype devra pouvoir être utilisé par trois types de publics :

- Opérateur : utilise le prototype pour détecter la présence de signatures
- Administrateur : supervise le bon fonctionnement de l'application et assure le support de niveau 1
- Mainteneur : assure l'évolution du prototype, des mises à jour fonctionnelles et techniques et du support en cas d'impossibilité d'un administrateur de l'assurer.

Développement logiciel

Périmètre fonctionnel

- Une interface graphique est mise à disposition des opérateurs
- L'application propose des sessions d'utilisateurs distinctes
- Les opérateurs soumettant un document scanné peuvent visualiser l'emplacement de signature, le nombre de signatures présentes et l'absence le cas échéant dans l'interface graphique
- L'application peut être interrogée sans utiliser son interface graphique
- L'application peut être appelée depuis un autre applicatif métier
- Les administrateurs ont accès à une interface pour surveiller l'état de l'application et reçoivent des alertes
- en cas de dysfonctionnement de l'application

Contraintes techniques

- L'interface graphique doit pouvoir être interrogeable via les navigateurs web suivants : Edge, Chrome, Firefox, Safari
- L'interface répond aux standards W3C
- Le moteur est exposé sous la forme d'une API REST
- Les différents composants de l'application : interface graphique et API sont conteneurisées

Pilotage du projet

La maîtrise d'ouvrage est confiée au management des équipes 3MU et CATOP de l'entité DEVRAP.

La maîtrise d'œuvre devra être prise en charge intégralement par les équipes 3MU et CATOP de l'entité DEVRAP.

Planning et livrables

Livrables

- Les conteneurs de l'API et de l'interface graphique
- Le code source de l'application
- Des comptes-rendus de suivi des travaux. Le choix de la forme et périodicité de ces derniers sont laissés à l'appréciation aux équipes 3MU et CATOP. Les équipes devront néanmoins en valider la nature en début de projet.

Planning

Le prototype répondant à l'ensemble du périmètre fonctionnel et des exigences techniques devra être livré et présenté au 4 décembre 2020.

Annexe II : Notebook de visualisations sur le base de donnée analytique

In [1]:
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
sns.set_theme(style="white")
import math
import numpy as np

df = pd.read_csv('https://drive.google.com/uc?id=1-3TtbNmGJZeti05pX854G3RfI
```

In [2]:
```
df.head()
```

Out[2]:

	Unnamed: 0	name	height	width	is_color	number_of_signatures	source
0	0	b014c226	3300	2544	0	1	tobacco_data
1	1	bee1cedc	1575	1200	0	1	tobacco_data
2	2	6e367c98	1575	1200	0	1	tobacco_data
3	3	21f534e3	3300	2544	0	1	tobacco_data
4	4	cdfd883c	3150	2400	0	1	tobacco_data

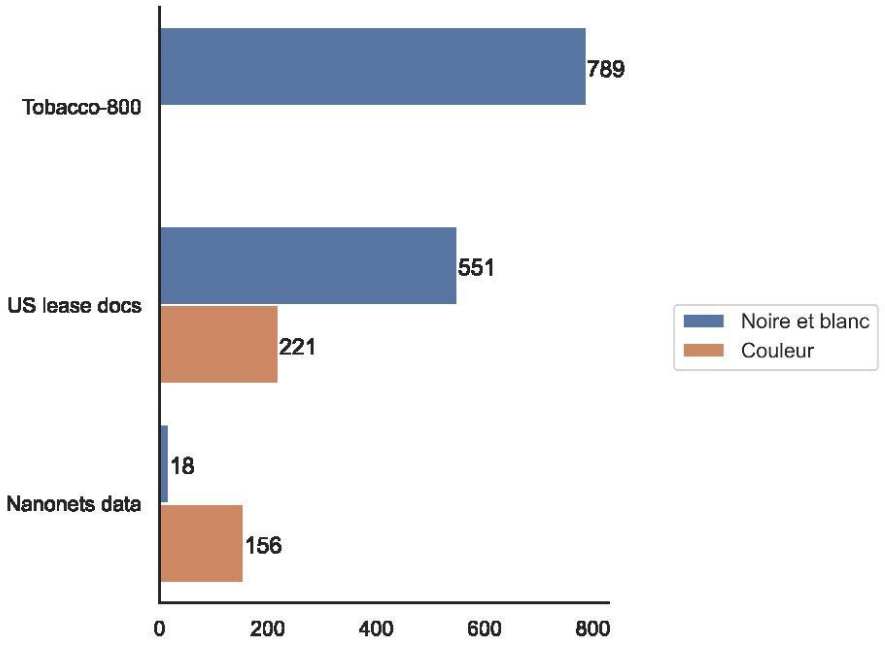
In [4]:
```
df.describe()
```

Out[4]:

	Unnamed: 0	height	width	is_color	number_of_signatures
count	1735.000000	1735.000000	1735.000000	1735.000000	1735.000000
mean	867.000000	2243.940634	1734.374640	0.217291	2.004035
std	500.995675	650.636099	507.187124	0.412521	1.215284
min	0.000000	408.000000	380.000000	0.000000	1.000000
25%	433.500000	2183.000000	1696.000000	0.000000	1.000000
50%	867.000000	2201.000000	1708.000000	0.000000	2.000000
75%	1300.500000	2292.000000	1728.000000	0.000000	3.000000
max	1734.000000	4200.000000	3506.000000	1.000000	8.000000

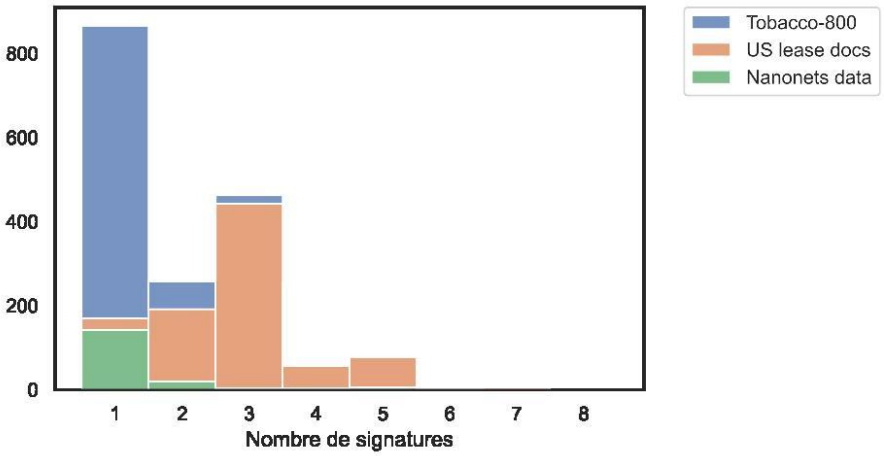
Nombre d'images en couleur et noire et blanc en fonction d'une source de données

In [6]:
```
plot_data = df.groupby('source')['is_color'].value_counts()
plot_data = plot_data.rename('count').reset_index()

g=sns.catplot(x='count', y='source', kind='bar', hue='is_color',
              order=['tobacco_data','lease_docs_png','nanonets_dataset'], c
g.set_yticklabels(['Tobacco-800','US lease docs', 'Nanonets data'])
g.set(xlabel="", ylabel = "")
handles, labels = g.ax.get_legend_handles_labels()
g.ax.legend(handles=handles,bbox_to_anchor=(1.6, 0.5),borderaxespad=0, labe

for p in g.ax.patches:
    width=p.get_width()
    if math.isnan(width):
      continue
    else:
      txt = str(int(width))
      txt_x = p.get_width()
      txt_y = p.get_y() + 0.25
      g.ax.text(txt_x,txt_y,txt)
```



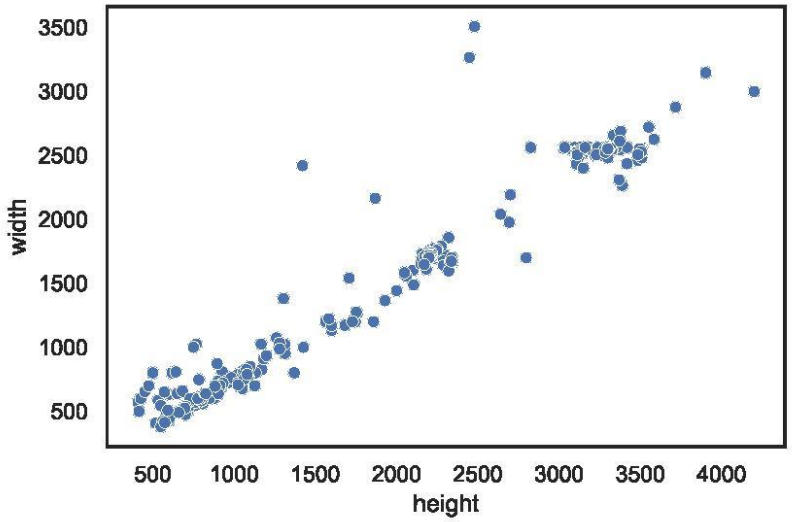
Le nombre de signatures sur les documents en fonction d'une source de données

In [7]:
```
g = sns.histplot(data=df, x='number_of_signatures', hue='source', discrete=
g.set(xlabel='Nombre de signatures', ylabel = '')
leg = g.axes.get_legend()
leg.borderaxespad=0
leg.set_bbox_to_anchor((1.4, 1))
leg.set_title('')
new_labels = ['Tobacco-800','US lease docs', 'Nanonets data']
for t, l in zip(leg.texts, new_labels): t.set_text(l)
```

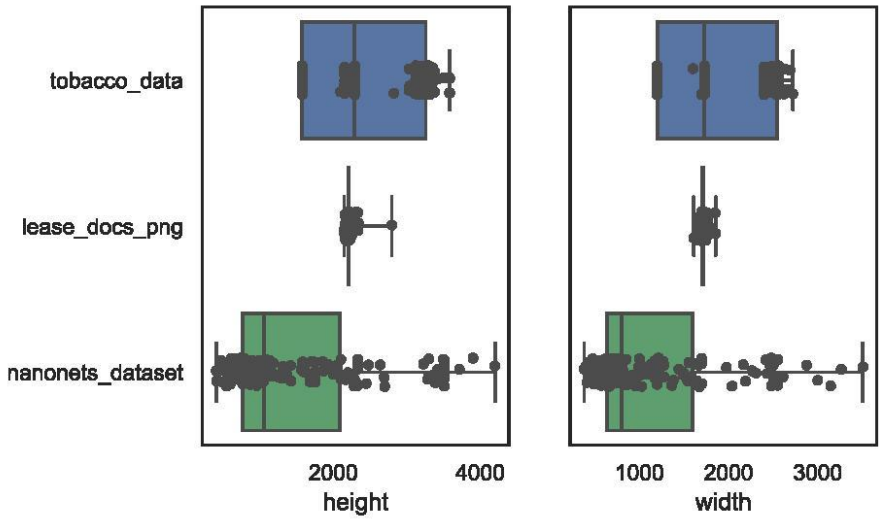


Les dimensions des documents

In [8]:
```
g = sns.scatterplot(data=df, x='height', y='width')
```



In [9]:
```
fig, (ax1, ax2) = plt.subplots(ncols=2, sharey=True)

sns.boxplot(x="height", y="source", data=df, whis=np.inf, ax=ax1)
g1=sns.stripplot(x="height", y="source", data=df, color=".3", ax=ax1)
g1.set(ylabel='')

sns.boxplot(x="width", y="source", data=df, whis=np.inf, ax=ax2)
g2=sns.stripplot(x="width", y="source", data=df, color=".3", ax=ax2)
g2.set(ylabel='')
plt.show()
```


tobacco_data
lease_docs_png
nanonets_dataset
2000
4000
height
1000
2000
3000
width

In [ ]: