Détection de falsification de vidéo

Rapport de Stage

Master 2 IDM

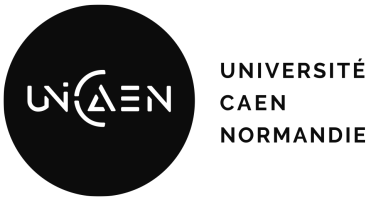
Université de Caen, Normandie

Hugo JEAN

Tuteurs : Christophe Charrier & Emmanuel Giguet

Tuteur université : Alexis Lechervy

Stage réalisé dans le laboratoire GREYC



Année 2021-2022

Table des matières

Résumé

Dans ce rapport, nous proposons plusieurs méthodes de détection de falsification de vidéo par analyse du bitstream encodé de vidéo ainsi que par analyse du contenu de la vidéo. La première partie de ce travail utilise une approche orientée machine learning (ML) en utilisant le flux vidéo non décodé pour classification de vidéo binaire. Les falsifications de vidéo utilisées dans ces travaux sont des modifications inter-frames de type copie, insertion, swap et suppression de frames.

Cette première partie est une reprise d'un projet réalisé par un précédent stagiaire utilisant la librairie Scikit learn de python pour proposer des modèles classiques de ML tels que des machines à vecteurs de support (SVM), arbres de décisions (Decision Tree) ou encore des forêts aléatoires (Random Forest). Les caractéristiques (features) utilisées sont extraites à partir du fichier encodé h264.

La deuxième partie utilise quant à elle des réseaux de neurones (DL) pour une classification multi-classe de vidéo. Dans cette partie, la vidéo décodée est utilisée. Deux approches ont été testées, la première en utilisant la prédiction de une ou plusieurs frames d'une vidéo à partir de frames précédentes. Et une seconde en utilisant le flux optique de la vidéo, puis en appliquant une transformation en signal sur celui-ci des réseaux récurrents tels qu'un simple réseau récurent (RNN) ou des réseaux récurrents plus évolués comme un GRU ou un LSTM.

Les datasets utilisés seront pour la première partie un ensemble de vidéos quelconques d'une durée de 10 secondes. Puis pour la deuxième partie le dataset Moving MNSIT et un dataset créé à la main à partir de vidéos de caméras de surveillance routière.

Les deux méthodes proposées donnent de bons résultats sur les datasets utilisés cependant l'indisponibilité de dataset de référence pour ce travail ne nous donne pas de métrique globale.

Remerciements

Je tiens à remercier tous les membres de l'équipe SAFE pour leur accueil convivial et leur disponibilité pour différentes questions, mais aussi pour leurs réunions/séminaires d'équipe qui permettent de ne pas se décrocher de l'équipe et de se tenir informé des projets de chacun.

Je remercie tout particulièrement mes deux tuteurs Christophe Charrier et Emmanuel Giguet pour leur disponibilité, leur expertise, leur soutien et la confiance qu'ils ont pu m'accorder pour ce projet.

1 Introduction

Mon stage s'est déroulé au sein du laboratoire GREYC à Caen, dans l'équipe de cybersécurité SAFE.

1.1 Présentation du laboratoire GREYC

Le GREYC, Groupe de Recherche en Informatique, Image, et Instrumentation de Caen, est un laboratoire de recherche en sciences du numérique dont les savoirs-faire sont centrés sur l'informatique et l'électronique. Il est dirigé par Christophe Rosenberger. Implanté en Normandie, le laboratoire, est associé au CNRS, à l'Université de Caen Normandie et à l'École Nationale Supérieure d'Ingénieurs de Caen.

Le GREYC est reconnu sur la scène nationale et internationale pour ses contributions scientifiques originales, ainsi que pour ses réalisations matérielles et logicielles. Près de 200 chercheurs, enseignants, ingénieurs, techniciens, post-doctorants, doctorants, et personnels administratifs contribuent à l'avancée des connaissances au travers de projets de recherche fondamentale et appliquée.

Les collaborations pluridisciplinaires avec les sciences humaines et sociales, les mathématiques et les sciences de l'ingénieur, ainsi que les partenariats industriels, sous forme de contrat ou de création de startups, contribue à son rayonnement.

Le GREYC est structuré en 6 équipes de recherche, dont 3 sont physiquement situées dans les locaux de l'Université de Caen, les 3 autres étant situées dans un bâtiment de l'ENSICAEN :

- l'équipe AMACC, dont l'expertise porte sur l'algorithmique, les modèles de calcul, l'aléa, la cryptographie, et la complexité (AmacC)
- l'équipe CoDaG, centrée sur les systèmes à base de contraintes, la fouille de données, les ontologies, et l'analyse à base de graphes
- l'équipe MAD travaille sur la mise au point de modèles pour le raisonnement, aux systèmes multi-agents et à la prise de décision

— l'équipe Image, spécialisée en traitement d'image et reconnaissance des formes
— l'équipe Electronique, spécialisée dans l'étude des capteurs et du bruit à basse fréquence,
— l'équipe SAFE, spécialisée en sécurité informatique, sciences de l'investigation et biométrie

C'est au sein de cette dernière équipe, SAFE, que mon stage s'est déroulé.

1.2 L'équipe SAFE

L'équipe SAFE, dirigée par Christophe Charrier , co-directeur de mon stage, mène des activités de recherche en sécurité informatique suivant trois axes qui traitent d'aspects à la fois théoriques et applicatifs de la sécurité :

— l'axe *Biométrie*
— l'axe *Architecture et modèles de sécurité*
— l'axe *Science de l'investigation numérique* (Forensique)

C'est au sein de l'axe Science de l'investigation numérique, coordonné par Emmanuel Giguet, co-directeur de mon stage, que j'ai mené mes travaux de stage. Cet axe de recherche s'intéresse plus particulièrement à la mise au point et à l'évaluation de méthodes et d'outils au service de l'investigation criminelle numérique.

Les techniques d'investigation numérique trouvant des applications naturelles en criminalistique, l'équipe SAFE entretient des relations avec l'IRCGN ou la Section de Recherche de la Gendarmerie Nationale à Caen. Mon stage, qui porte sur la détection de falsification vidéo, trouve des applications dans ce cadre.

1.3 Enjeux de la détection de falsification de vidéo

De nos jours, les contenus vidéos sont transmis dans des volumes en croissance exponentielle. Elles ont, pour la plupart, vocation à être partagées sur les réseaux sociaux plébiscités par le grand public. Cet essor a été favorisé par la création d'outils d'édition puissants, faciles à utiliser, qui permettent de personnaliser les contenus vidéos. Les montages n'ont jamais été aussi simples à réaliser : les vidéos sont assemblées, certains passages sont effacés ou au contraire dupliqués, certaines portions d'images peuvent être altérées pour faire disparaître ou au contraire apparaître des éléments, et cela, au gré des

envies ou des motivations. Les avancées technologiques en matière de manipulation d'images et de vidéos ont démocratisé les usages, qu'ils soient ludiques ou artistiques, mais également propagandistes ou complotistes. Dès lors, la légitimité, la fiabilité et l'authenticité des vidéos diffusées et relayées sur les réseaux sont devenues une préoccupation majeure, notamment pour détecter les tentatives de désinformation. En matière légale, les vidéos peuvent aujourd'hui servir de preuve devant la justice. La modification intentionnelle d'une vidéo à des fins de falsification, appelée contrefaçon vidéo, doit pouvoir être détectée. Le défi consiste à déterminer si la vidéo a été modifiée, et, dans la mesure du possible, à qualifier la nature des altérations.

1.4 Présentation du sujet de stage

Ce stage a pour objectif de prolonger le travail réalisé par Paul Canchon ancien stagiaire du GREYC. Celui-ci a travaillé sur une approche *bitstream* et classifieurs classiques de machine learning.

Enfin la deuxième partie de ce stage fut consacrée à la recherche de méthode de détection par utilisation de réseau profond.

2 Détection par analyse du bitstream

L'idée explorée dans cette méthode repose sur les informations codées dans le fichier vidéo, ici nous utilisons le protocole H.264 aussi appelé MPEG-4 AVC. Ce codec est aujourd'hui largement adopté par tous les particuliers et professionnels, il est par exemple utilisé par la majorité des diffuseurs de contenu vidéo en ligne comme Youtube (Voir figure 1), Netflix ou encore Prime Video.

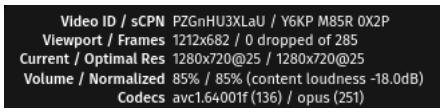


FIGURE 1 – Informations du lecteur Youtube, remarquez la ligne 'Codecs : avc...'

Il est important de noter que l'information n'est pas codée dans l'espace RGB mais dans l'espace Y'C_BC_R. Ce codec utilise ce qu'on appelle des GOP

(Group Of Images) découpés en 3 types d'images :

— Image I (Image de référence) : Elle n'est en soit pas codée, elle correspond à une image JPEG. Toutes les prédictions temporelles du GOP actuel se baseront sur celle- ci.

— Image P (Predictive frame) et image B (BiPredictive Image) : Dans ces deux images la prédiction est basée sur l'image I du début de GOP, pour la B elle utilise aussi l'image P précédente et la suivante.

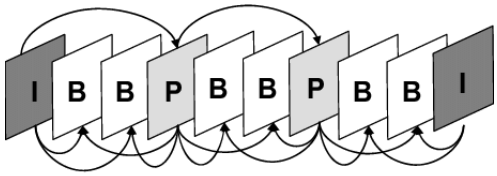


FIGURE 2 – Structure d'un GOP

Je ne vais détailler ici que les méthodes de codage utilisées qui nous intéressent dans ce codec, la redondance temporelle (prédiction Inter-Frame) et la redondance spatiale (prédiction Intra-Frame).

La redondance temporelle utilise des images de références dites P pour 'predictive frame', le principe de base est très simple, il part du fait qu'entre plusieurs frames d'une vidéo on retrouve généralement les mêmes évènements. Imaginons que nous filmions une vidéo de surf, le ciel est une caractéristique redondante et sera présenté à priori dans toutes les frames de la vidéo. On peut alors utiliser des macroblocs d'une frame de référence et encoder le reste à partir de ces macroblocs de la frame originale.

La redondance spatiale est basée sur le fait que dans une image nous retrouvons souvent des informations redondantes, reprenons notre vidéo de surf, le ciel qui occupe 50% de l'image est lui redondant, en effet si ce n'est quelques différences de luminosité et de contraste le ciel est majoritairement bleu. Nous pouvons donc encoder les macros blocs grâce à leurs voisins et ainsi éviter de dupliquer l'information.

2.1 Reprise d'un projet de détection orienté ML

Comme annoncé plus haut la première partie de mon stage s'est focalisée sur une reprise de projet d'un précédent stagiaire du GREYC.

2.1.1 Méthode de détection

Paul Canchon, ancien stagiaire au GREYC, a mis en place un protocole de falsification puis de détection de cette falsification. Il récupère des vidéos au format h264, applique un protocole de falsification 2.1.3 qu'il a conçu, récupère le fichier, utilise l'extracteur de caractéristiques. Enfin ces caractéristiques récupérées vont servir de données d'entrée pour les différents algorithmes de machine learning utilisés.

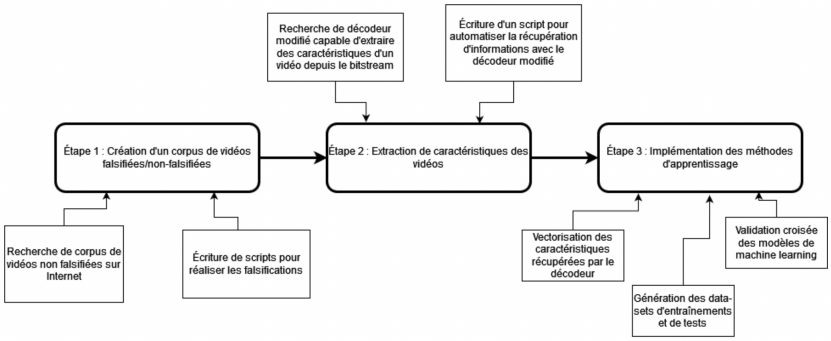


Figure 3 – Processus complet utilisé

Je vais détailler un peu plus la partie d'extraction de caractéristiques qui reste la même dans la méthode de Paul et la mienne. Une fois le fichier h264 créé, nous utilisons un extracteur de caractéristiques codé en C adapté d'un décodeur h264 et qui permet donc de récupérer des informations. Cet extracteur permet de récupérer les 27 caractéristiques que nous allons utiliser pour chaque image de la vidéo.

- c_1 : Le bitrate qui donne la quantité de donnée transmise
- c_2, c_3 : Valeur moyenne du pas de quantification (QP) ainsi que son delta.
- c_4, c_5 : Valeur moyenne et maximale des vecteurs de mouvement.
- c_6, c_7 : Valeur moyenne et maximale des erreurs de vecteurs de mouvement.
- c_8, c_9, c_{10} : Pourcentage de macro blocs codés dans l'image selon les codages suivants : Intraframe, InterFrame et skip.
- c_{11}, c_{12}, c_{13} : Pourcentage des macroblocs de référence selon leurs taille : 16x16,8x8 et 4x4.
- $c_{14}, c_{15}, c_{16}, c_{17}$: Pourcentage des macroblocs P selon leurs taille : 16x16

, 16x8 , 8x16 et 8x8

— c_{18}, c_{19}, c_{20} : Pourcentage des sous-macroblocs P selon leurs taille : 8x4, 4x8 et 4x4.

— c_{21}, c_{22}, c_{23} : Pourcentages des macroblocs B selon leur mode de partitionnement. Ces modes sont 16x16, 16x8, 8x16.

— $c_{24}, c_{25}, c_{26}, c_{27}$: Pourcentages des macroblocs de type P et B non codés : skip ou direct. Le décodeur s'occupe de déterminer leur valeurs en fonction des macroblocs voisins (mode skip) ou du macrobloc situé à la même position dans l'image de référence (mode direct).

Une fois ces 27 caractéristiques extraites pour chaque image de la vidéo nous récupérons un fichier .csv contenant 27 colonnes et $nbFrames$ lignes. Cela pose donc problème, nous avons besoin d'un vecteur de taille fixe en entrée de nos algorithmes d'apprentissage. Paul a donc choisi de prendre pour chaque colonne 4 statistiques : la moyenne, le minimum, le maximum et l'écart type. De mon côté, j'ai essayé de rajouter des statistiques telle que la médiane ou encore les quartiles, mais cela n'améliorait pas les résultats. J'ai donc décidé de rester sur les 4 métriques proposées par Paul. Cependant j'ai pu proposer une meilleure méthode d'extraction et d'agrégation du fichier .csv, Paul ayant proposé un code en utilisant seulement du Python pur. J'ai donc pu améliorer assez facilement les performances en utilisant Pandas une librairie de manipulation de base de données sous forme de fichier (.csv, .json ou même .txt).

2.1.2 Présentation du dataset

Paul a utilisé un premier dataset appelé : REWIND contenant 10 vidéos originales et 10 modifiées, puis un autre dataset appelé : VIFFD contenant lui 101 vidéos originale dont 15 en 1920x1080 et 86 en 720x404. Il n'a pas utilisé les vidéos modifiées de ce dernier dataset car il ne comprenait pas les 4 types de modifications visées dans ce travail, il a donc lui même créé des vidéos modifiées à partir de ces 101 vidéos.

De mon côté j'ai simplement récupéré une base de données de vidéos assez conséquente pour permettre une diversité dans le cadre et la taille, j'ai donc choisi la base VQC qui regroupe 585 vidéos avec des propriétés différentes.

2.1.3 Protocole de falsification

Protocole proposé par Paul : Pour créer des vidéos falsifiées à partir de vidéos originales, Paul a utilisé un script shell effectuant une modification fixée à un endroit fixe lui aussi.

- La suppression est réalisée à la marque des 2 secondes pour 2 secondes de contenu supprimé.
- L'insertion commence à 3 secondes pour 2 secondes.
- La copie commence à 2 secondes et dure 2 secondes.
- Le swap a pour première partie la deuxième seconde de la vidéo et pour deuxième la troisième seconde de la vidéo.

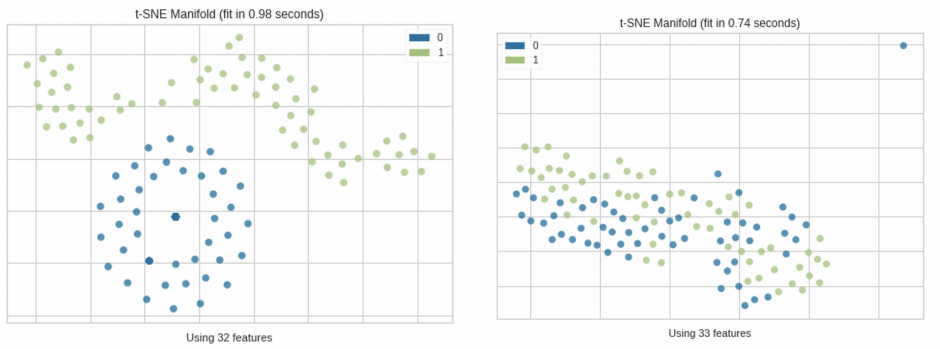
Ces modifications vont faire apparaître un biais dans la détection, les modifications étant centrées sur 2 secondes et n'étant pas de durée aléatoire, le modèle sera biaisé. Il m'a paru plus pertinent de créer un autre système de falsification utilisant l'aléatoire à la fois pour la localisation et à la fois pour la durée. La durée de la falsification joue un rôle important dans la modification du comportement de l'encodeur, en effet une grande modification entraînera sans doute un 'décrochage' dans la vidéo et donc une difficulté pour l'encodeur d'utiliser les frames I et P pour prédire le comportement de la vidéo.

Protocole modifié : Ainsi je propose le protocole suivant :

- Pour la suppression : Prendre un instant t_1 aléatoire et supprimer une durée t_2 aléatoire ne dépassant pas la durée de la vidéo.
- Pour la copie : Choisir un instant t_1 et une durée t_2 pour la copier à un endroit t_3 et tout ça choisi aléatoirement avec des conditions pour que $t_1 + t_2$ ne dépassent pas la durée de la vidéo et que t_3 ne soit pas compris dans $t_1 + t_2$.
- Pour l'insertion : Ici pas de restriction particulière, on choisit aléatoirement un instant t_1 et on insère t_2 secondes d'une autre vidéo choisie aléatoirement.
- Pour le swap : J'ai fait en sorte que la première partie se trouve dans les 45% de la vidéo et la deuxième partie entre 50% et 100% de la vidéo pour éviter de swapper deux fois une même partie et ainsi les remettre dans l'ordre (cas rare).

Ce protocole de falsification permet de générer des modifications aléatoires et ainsi de réduire au maximum le biais qui pourrait être induit dans la formation du dataset. Les vidéos sont ensuite ré-encodé avec un CRF (constant rate factor) qui permet de définir une qualité et non un bitrate. Ainsi le bitrate fluctue mais non la qualité.

Pour montrer le biais induit dans le dataset par la première méthode de falsification on peut réaliser une réduction de dimension pour essayer de visualiser nos vecteurs dans un espace 2D, ici la réduction est effectuée avec l'algorithme t-SNE :



(a) Visualisation 2D des vecteurs suppression/pristine du Dataset de Paul

(b) Visualisation 2D des vecteurs suppression/pristine du dataset après modification du protocole de falsification

On peut clairement apercevoir que le dataset de Paul contient un biais qui facilite la classification sur la figure ci-dessus.

2.1.4 Algorithmes de ML utilisés

Après avoir obtenu un vecteur de 27x4, nous pouvons utiliser des algorithmes de machine learning classiques. Ainsi, lors de son travail préliminaire Paul a pu tester les modèles suivants :

— Arbre de décision (Decision Tree)
— K plus proches voisins (KNN)
— Régression logistique
— Une classification bayésienne naïve
— Un SVM linéaire avec et sans pondération des classes
— Un SVM avec un noyau RBF avec et sans pondération

Il semble important de noter que le travail original ne proposait aucune validation croisée ou de méthode de recherche d'hyper-paramètres tel que GridSearch. Ainsi il était impossible pour moi de savoir comment les paramètres étaient choisis. L'autre problème était le biais introduit dans le dataset par le protocole de falsification proposé par Paul (2.1.3).

J'ai donc premièrement utilisé mon dataset avec des modifications aléatoires pour faire de la recherche d'hyper-paramètres pour les méthodes citées précédemment. L'ensemble des méthodes sont implémentées en Python3 en utilisant le package Scikit-Learn [1], référence en machine learning sous Python. Cette recherche d'hyper-paramètres est effectuée avec la méthode de ce package *GridSearchCV* [2]. Tous les modèles utilisés et résultats présentés sont issus de modèles avec hyper-paramètres optimisés avec cette méthode sauf contre-indication.

J'ai aussi pu utiliser le package *PyCaret* [3], un package de machine learning low code, testant les modèles les plus communs ainsi que quelques modèles plus optimisés tels que LightGBM (light gradient boosting machine), gbc (Gradient Boosting Classifier) et autres. Ce package permet aussi une utilisation très simple, il suffit de disposer de son dataset sous un format facilement utilisable sous Pandas (csv, txt) pour que ce module réalise de la réduction de dimension si elle est pertinente, de la stratégie k-fold et de la recherche d'hyperparamètres.

Ce module étant interactif il est intéressant de le lancer à travers des Notebook pour pouvoir avoir un retour à chaque étape.

L'entraînement des modèles se fait à base de ratio 75/25 entraînement validation, ce qui fait environ 750/250 vidéos pour chaque phase. Il faut cependant noter que la séparation se fait sur les vidéos originales, ensuite les vidéos modifiées sont ajoutés dans le dataset de la phase choisi. Cela permet d'assurer que les modèle n'ai jamais vu quelconques version de la vidéo lors de la validation.

2.2 Résultats

En utilisant les modèles et le dataset proposé par Paul avec 555 vidéos (111 de chaque modifications et 111 pristine), on obtient les résultats suivants :

1. https://scikit-learn.org/stable/
2. GridSearchCV
3. https://pycaret.org/

Classifieur	Exactitude	Précision	Rappel	F-score
Arbre de décision	0.78	0.89	0.82	0.85
k plus proches voisins	0.79	0.81	**0.96**	0.88
Régression logistique	0.89	0.93	0.94	0.93
Naives Bayes	0.62	0.84	0.65	0.73
SVM linéaire	**0.90**	0.94	0.93	**0.94**
SVM linéaire avec pondération	0.88	**0.96**	0.89	0.92
SVM rbf	0.89	0.93	0.93	0.93
SVM rbf avec pondération	0.86	0.95	0.88	0.91

FIGURE 5 – Tableau de résultats proposé par Paul

Ces résultats sont très bons : un F1 de 0.94 pour le SVM semble très prometteur. Pour rappel le score F1 permet d'avoir une appréciation générale du modèle en se reposant sur les métriques de *recall* et *precision*, ces deux métriques seules ne peuvent donner d'indication générale sur le modèle. Le F1 score est une moyenne harmonique et se calcule donc de la façon suivante :

$$F1 = 2 * \frac{recall * precision}{recall + precision} \tag{1}$$

Cependant il faut se rappeler que le protocole de falsification proposé plus tôt induit d'importants biais dans la création du dataset. On peut donc se demander comment les méthodes et algorithmes proposés par Paul se comporte sur un dataset généré de façon aléatoire :

	accuracy	precision	recall	f score
Support Vector Machine with linear kernel	0.665833	0.855896	0.722917	0.760904
Support Vector Machine with linear kernel and class weight	0.640833	0.869210	0.678125	0.724897
Support Vector Machine with rbf kernel	0.635000	0.842682	0.688542	0.726243
Support Vector Machine with rbf kernel and class weight	0.642500	0.888515	0.664583	0.720514
Naives Bayes	0.650000	0.881832	0.662500	0.734931
K-Nearest Neighbours	0.789167	0.857649	0.890625	0.867003
Logistic Regression	0.663333	0.850592	0.719792	0.753131
Decision Tree	0.700000	0.842513	0.785417	0.791741
Voting Classifier	0.742500	0.854851	0.831250	0.827312

FIGURE 6 – Résultats des modèles de Paul sur un dataset utilisant des modifications aléatoires

On remarque des résultats bien en deçà des résultats originaux proposés par Paul (perte de 20% sur le F1 pour notre SVM). Cela peut conforter l'idée que le protocole de falsification de Paul induit un biais conséquent dans les résultats de classification.

Par la suite, les résultats proposés découle du dataset aléatoire et de modèle optimisé avec recherche d'hyper-paramètres. Ainsi en utilisant les outils proposés par *PyCaret* il est facile d'avoir une synthèse des résultats et d'avoir une vue d'ensemble des différents modèles.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.9058	0.9246	0.9652	0.9216	0.9427	0.6778	0.6858	0.097
gbc	Gradient Boosting Classifier	0.8990	0.9171	0.9696	0.9125	0.9396	0.6331	0.6492	0.223
et	Extra Trees Classifier	0.8781	0.9090	0.9870	0.8772	0.9285	0.5227	0.5707	0.464
rf	Random Forest Classifier	0.8746	0.8863	0.9870	0.8751	0.9270	0.4969	0.5507	0.515
lr	Logistic Regression	0.8745	0.8909	0.9298	0.9141	0.9213	0.6065	0.6154	0.681
ada	Ada Boost Classifier	0.8741	0.9345	0.9174	0.9261	0.9211	0.6081	0.6125	0.133
lda	Linear Discriminant Analysis	0.8709	0.8849	0.9431	0.9014	0.9211	0.5628	0.5771	0.017
ridge	Ridge Classifier	0.8534	0.0000	0.9563	0.8731	0.9125	0.4644	0.4902	0.017
dt	Decision Tree Classifier	0.8191	0.7122	0.8911	0.8841	0.8869	0.4298	0.4353	0.018
dummy	Dummy Classifier	0.8009	0.5000	1.0000	0.8009	0.8894	0.0000	0.0000	0.011
qda	Quadratic Discriminant Analysis	0.7941	0.7866	0.8913	0.8583	0.8738	0.3023	0.3064	0.017
svm	SVM - Linear Kernel	0.7518	0.0000	0.8862	0.8188	0.8469	0.1098	0.1298	0.016
knn	K Neighbors Classifier	0.7486	0.6759	0.8911	0.8131	0.8498	0.0671	0.0698	0.117
nb	Naive Bayes	0.7102	0.8539	0.6820	0.9375	0.7871	0.3709	0.4184	0.015

FIGURE 7 – Résultats des différents modèles

Comme présenté dans le tableau plus haut on remarque que les meilleurs modèles sont les arbres de classifications et plus particulièrement les arbres couplés à des boosting machines. Très rapidement, une boosting machine permet un apprentissage à partir de classifieur dit faible ou fort, en adaptant le poids de ces classifieurs et en utilisant un apprentissage itératif, c'est-à-dire en utilisant le poids modifiés de l'itération d'avant. Ces types de modèles sont considérés comme les meilleurs aujourd'hui. Ainsi avec un F1 score de 94% on peut juger ces résultats très prometteurs. Il est aussi important de notifier que le recall est de 96%, cette métrique étant tout aussi importante pour un modèle de détection d'anomalies.

2.3 Perspectives et limites de la méthode

Ces résultats sur un dataset composé aléatoirement nous permet de penser que ces méthodes sont robustes. En plus d'avoir de bon résultats le temps d'inférence des modèles reste extrêmement bas, la plus grande partie du temps de calcul étant l'extraction des features. On peut quand même imaginer assez facilement que ceci puisse être fait en temps réel pendant la récupération du flux vidéo par le serveur central de vidéo surveillance. Cependant sans dataset de référence comme il peut en exister pour de la classification d'image avec ImageNet pour la classification d'image, il reste difficile de prouver que ces méthodes sont vraiment généralisables et éprouvées.

On peut aussi se poser la question des performances de tels modèles pour des problèmes de classification multiple. Ci dessous la matrice de confusion pour une classification multiple :

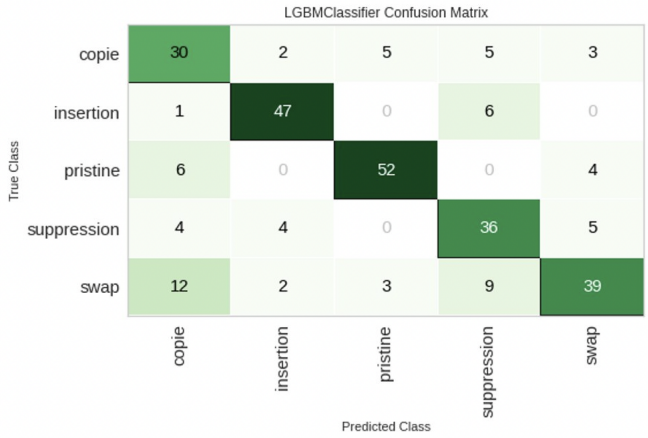


FIGURE 8 – Matrice de confusion d'un modèle LGBM

On remarque des résultats semblables pour toutes les classes à l'exception des classes copie/swap que le modèle n'arrive pas à différencier. Cependant l'idée originale est de proposer un outil d'aide à la décision et non un outil complètement automatique étant donné la particularité du sujet et son importance. En effet, l'utilisation de solutions de machine learning ou de deep learning comme oracle parfait sur des sujets aussi importants que celui-ci, l'analyse de preuve, est totalement déconseillé. Ce phénomène est parfaite-

ment illustré dans le domaine de la médecine où de multiples solutions d'aide à la décision des médecins sont apparues notamment pour la détection de cellules cancéreuses. Ici la problématique reste la même : nous voulons grandement diminuer le temps passé par les Forces de l'Ordre à regarder des vidéos de caméra de surveillance sans non plus prendre de décision à leur place. Il reste aussi des contraintes de format d'entrée comme pour beaucoup de modèles aujourd'hui, les nouvelles recherches laissent penser que les transformers, qui ont déjà fait leurs preuves en NLP [4] avec les modèles tels que GPT3 ou BERT, pourront être utilisé en vision par ordinateur et donc régler ce problème d'input.

3 Détection basée sur l'apprentissage profond

3.1 Présentation et état de l'art

Après avoir repris le projet d'utilisation du bitstream avec du machine learning et amélioré celui-ci, je me suis penché sur la question de l'utilisation de réseaux profonds pour la détection de falsification. Ici il n'es§ plus question d'utiliser le bitstream mais bien le contenu de la vidéo en elle-même. Aujourd'hui l'utilisation de deep learning sur vidéo est porté en avant par la détection d'action humaine principalement comme le montre les plus grands datasets de vidéo disponibles au public :

Nom du dataset	Type de dataset	Nombre de vidéos
HMBD51	Action Recognition	6766
UCF101	Sports	13320
Sports-1M	Sports	1.1M
Kinetics	Action Recognition	500k

TABLE 1 – Plus grand datasets vidéo publique

Cependant cette tâche reste assez semblable à de la détection d'action sur une image. Un des premiers modèles proposé dans [6] consiste à utiliser des convolutions mais en 3D et non pas en 2D, c'est-à-dire en empilant des frames successives en entrée. Cette méthode sur le dataset Sports-1M cité plus haut, donne les résultats suivant ($@k$ signifie que la bonne prédiction

4. Natural Language Processing (Traitement automatique des langues)

est dans les k prédictions du modèle) : 42% en Clip@1, 60% en Video Hit@1 et 80.2% en Video Hit@5. Ce type de modèle, aussi appelé 3D-CNN pose plusieurs problèmes, notamment un temps d'apprentissage énorme, 1 mois dans le papier, la nécessité d'un nombre important de vidéos ce qui va de paire avec le temps d'apprentissage. Il est cité dans le papier que le modèle a vu environ 500 millions de clips (paquet de frames). Bien que le papier soit sorti en 2014 et que les GPU ait gagné en puissance, les mêmes problèmes peuvent être observés aujourd'hui.

On peut se demander ce qu'un réseau du même type avec un backbone déjà entraîné pourrait proposer. C'est ce que propose le papier [11], en utilisant ResNet18 en tant que backbone. Les résultats sont alors améliorés pour atteindre : 48.8% en CLip@1, 65.6% en Video@1 et 87.8% en Video@5. Et cela tout en utilisant un réseau déjà entraîné avec finetuning (Ré-apprentissage des dernières couches pour la classification).

Plus tard d'autres méthodes ont émergé en utilisant le flux optique, le champ de vecteur quantifiant la quantité de mouvement entre deux frames consécutives. C'est ce que propose le papier [2], en utilisant en même temps un CNN sur les frames et un CNN sur le flux optique. Ainsi les résultats s'améliorent encore pour atteindre : 93.4% Video@5 sur le dataset UCF101.

Ce dernier modèle est très intéressant notamment pour son utilisation du flux optique comme donnée, cependant notre problème est bien différent d'une classification d'action humaine. En effet, aucune information venant d'une image seule n'est importante pour nous, un humain pour pouvoir donner une appréciation sur la falsification ou non d'une vidéo a besoin d'au moins 3 frames consécutives. Cela souligne que la dimension temporelle est bien supérieure au contenu, ce qui diffère bien de la reconnaissance d'action où a priori le contexte est bien plus important. À partir de ce constat, deux idées principales sont apparues :

— La première : pouvoir vérifier la continuité d'un groupe de frames conditionné à partir d'un autre groupe de frames. L'idée est donc à partir de k frames pouvoir prédire x frames et ainsi comparer les frames prédites avec les frames réelles de la vidéo.
— La deuxième : utiliser le champ de vecteurs de mouvement pour pouvoir détecter des mouvements anormaux. Prenons l'exemple d'une caméra filmant un homme marchant, celui-ci marche à une vitesse constante donc les vecteurs de mouvement sont semblables de frame en frame. Supprimons 5 frames, la longueur des vecteurs de mouve-

ment sera alors subitement 5 fois plus grande ou 5 fois plus petite si l'homme quitte le cadre.

3.2 Création de datasets

Comme annoncé précédemment il n'existe pas de dataset de référence pour notre problème. On peut noter l'existence d'un dataset de falsification de type *copy/move* appelé *REWIND* [1], cependant celui-ci ne comporte que 20 vidéos (10 originales et 10 modifiées) ce qui reste bien maigre pour espérer pouvoir entraîner un réseau dessus. Ainsi la solution choisie fût de prendre des vidéos quelconques et d'appliquer des modifications aléatoires sur celles-ci. Cela peut poser problème lorsqu'une vidéo est statique ou si la partie modifiée intervient dans un moment de la vidéo où il n'y a pas de mouvement ou d'action. Le cas le plus simple serait d'imaginer une vidéo dont toutes les frames sont noires. Une modification sur celle-ci n'affecterait en rien la vidéo et il serait impossible pour un humain de détecter cette modification. Pour palier ce problème, dans [12], ils réalisent eux mêmes leurs modifications, cette solution semble être parfaite pour créer des vrais cas d'utilisation mais reste très compliquée à mettre en place de par son coût en temps.

3.2.1 Identification de datasets originaux

Pour la prédiction vidéo, la majorité des papiers proposaient leurs modèles sur le dataset Moving MNIST [10], celui-ci n'est juste qu'une version vidéo avec mouvement du dataset de référence MNIST proposé par Yann LeCunn. Les caractéristiques sont les suivantes :

— Taille des vidéos : 64x64 pixels et sequences de 20 frames
— Espace couleurs : 0 - 255 sur 1 canal (Noir et blanc)
— Taille du dataset : 10 000 sequences (200 000 frames)

Pour la deuxième partie, l'utilisation du flux optique, nous n'étions en rien contraints, ainsi nous avons utilisé le dataset proposé dans [9]. Les caractéristiques sont les suivantes :

— Taille des vidéos : Résolution non fixée, les vidéos durent 10 secondes à différents framerate.
— Vidéos en couleurs : 0 - 255 sur 3 canaux (RGB)
— Taille du dataset : 585 vidéos uniques (5.5 GB)

Une autre solution explorée fut d'obtenir un flux live des caméras routières

de Sioux Falls, Dakota du Sud aux États-Unis, disponible librement ici [5].

3.2.2 Protocole de falsification

Le protocole de falsification proposé est le même que 2.1.3. Celui-ci est adapté pour modifier des suites d'images et non plus des vidéos directement pour un souci de facilité d'utilisation pour les réseaux de neurones. Cette modification permet de n'avoir à extraire qu'une seule fois les frames de la vidéo et donc ne travailler qu'avec des tableaux NumPy ce qui permet un traitement accéléré grâce à l'optimisation, possible lorsque l'on maîtrise le mode de fonctionnement mémoire de NumPy qui est plus proche du C que de l'idée originale des tableaux proposée dans Python.

3.3 Prédiction de vidéo

3.3.1 Principe

Comme expliqué rapidement précédemment, ici nous voulons pouvoir prédire des frames à partir de frames précédentes. Une fois ces frames prédites nous les utiliserons comme référence de ce que devrait être la vidéo, c'est-à-dire que nous faisons confiance en notre modèle et que les frames de la vidéo proposées sont celles mise en doute. En pratique nous appliquons la technique suivante, on réalise une MSE entre chaque frame successive pour obtenir un profil de MSE comme illustré ci-dessous :

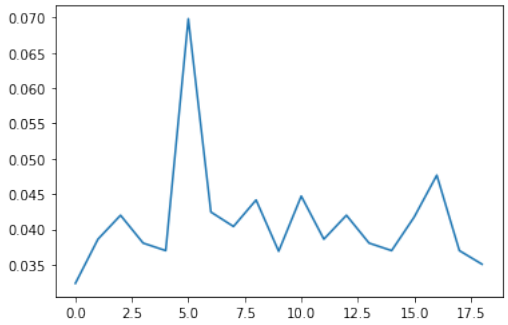


FIGURE 9 – Profil de MSE successives pour une vidéo modifiée

5. Lien pour le live des caméras routières : https://www.arcgis.com/apps/dashboards/

Une fois ce signal obtenu on trouve les valeurs aberrantes selon la formule suivante :

$$v_a = \forall x \in \{x_0, ..., x_n\} > \bar{x} + \sigma \tag{2}$$

Pour illustrer cela, ci-dessous une séquence avec les valeurs aberrantes mises en avant :

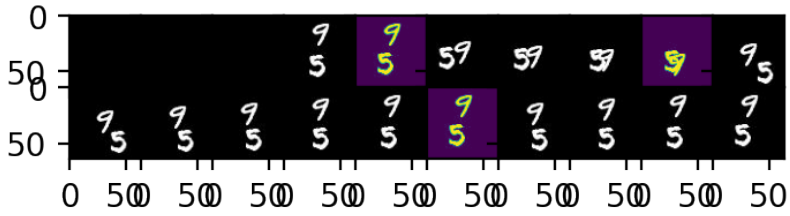


FIGURE 10 – Séquence vidéo avec frames aberrantes mises en avant

Pour chaque frame aberrante on applique un flag sur la séquence pour ainsi créer une sous séquence qui se termine sur la frame dite aberrante. Cela nous donnera notre groupe de frame qui sera l'entrée du réseau. Ensuite nous réaliserons des tests pour estimer la différence entre les frames predites et les frames de la vidéo originale et ainsi donner un pronostic sur la falsification ou non de cette vidéo.

3.3.2 Modèles utilisés

Le domaine de la prédiction vidéo est surtout poussé par trois technologies : la prédiction de météo, la prédiction de trafic routier et la conduite autonome. Les deux premières transforment leurs données en vidéo. C'est à dire qu'elles utilisent des cartes 2D des routes avec un niveau de congestion à des temps différents $(t_{-n}, ..t_{-1})$ pour conditionner leurs modèles. Ce qui permet de créer une 'video' dans le sens d'un empilement de prise de vue 2D avec un lien temporel. Un exemple ci-dessous extrait de [7]. Le principe est le même pour de la prédiction de météo comme présenté dans le papier [8].

Ces deux technologies ont tout d'abord été portées par l'utilisation de LSTM couplée avec des convolutions ce qui a engendré l'apparition de couches dites ConvLSTM. Disponible de base dans le package de deep learning TensorFlow, pour PyTorch une implémentation est nécessaire mais beaucoup

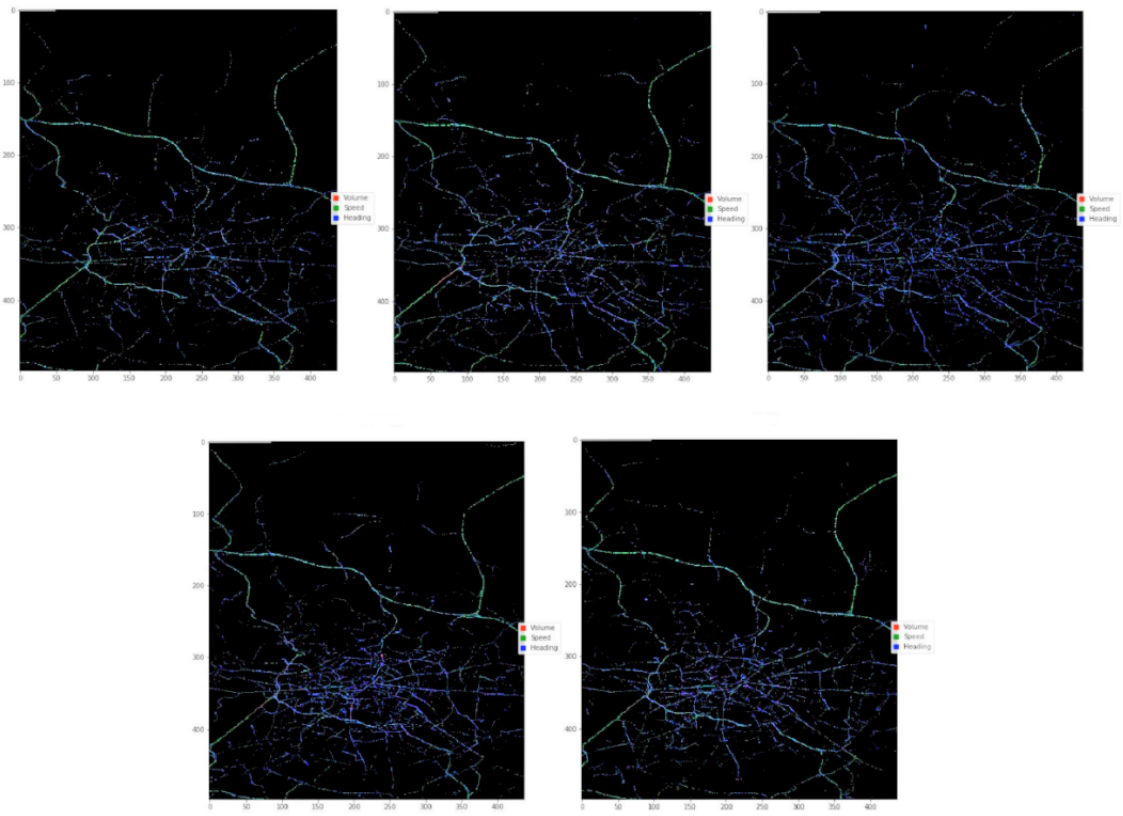


FIGURE 11 – Exemple de 'video' de trafic routier

sont disponibles sur GitHub. Une couche LSTM est l'une des 3 grandes couches récurrentes : RNN , LSTM et GRU. Les couches récurrentes sont faites pour fonctionner avec des séquences de données. La couche RNN est la première a apparaître mais souffre de problème de gradient quand la séquence de données est trop grande. La LSTM et la GRU ont été créé pour palier a ce problème, elles comportent des mécanismes permettant d'oublier une partie des données jugées non-importantes, là ou la couche RNN garde toutes les informations précédentes, ce qui pose problème pour le gradient. Ainsi j'ai pu essayé ConvLSTM du papier [8] pour cette tâche. Cependant depuis l'apparition de cette méthode, de meilleurs modèles sont apparus : on peut citer PredRNN(V2) [14] proposant les meilleurs résultats pour le problème de prédiction de trafic ou encore CrevNet [15] proposant lui aussi de très bons

résultats. Ces deux modèles se basent sur le principe d'encoder-decoder. Une architecture de modèle visant dans un premier temps à réduire la dimension dans ce qu'on appelle la phase d'encodage de l'entrée pour produire un vecteur de taille fixé dans un espace latent, puis ensuite décoder celui-ci pour la sortie. Ci-dessous un exemple d'architecture d'encoder-decoder :

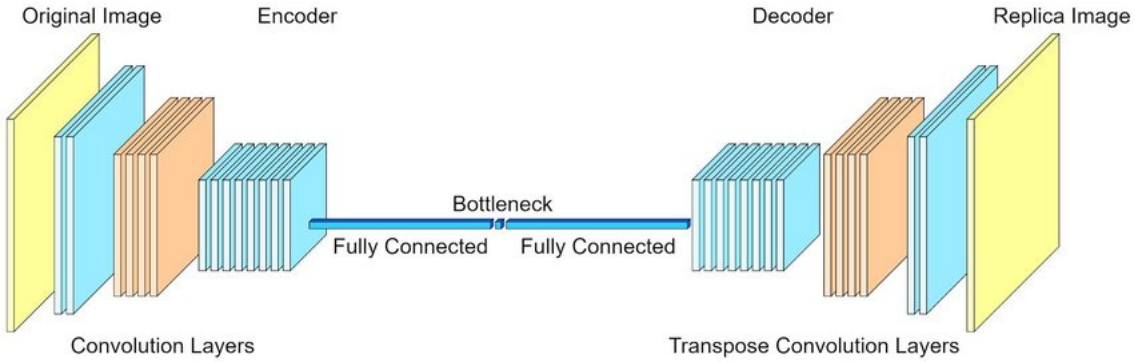


FIGURE 12 – Structure type d'un encoder decoder

Il faut noter ici que sur ce schéma l'entrée est une image et non un empilement d'images comme pour notre problème.

3.3.3 Entraînement et Résultats

Pour la suite j'ai choisi d'utiliser une version modifiée par moi-même du modèle [4], particulièrement performant pour la tâche de prédiction vidéo. Celui-ci utilise deux encoder-decoder. Le premier utilise des couches ConvLSTM pour apprendre les features dites inconnues telles que l'apparence, les détails et la texture. Alors que le deuxième permet d'apprendre en respectant certaines contraintes physiques comme par exemple ici dans notre problème la trajectoire des chiffres pour MovingMNIST ou encore la trajectoire d'un piéton.

Pour l'entraînement, notre problème n'étant pas une classification il n'existe pas de classe pour nos séquence. Ici nous voulons prédire le déroulement d'une vidéo, il n'est pas encore question de juger si une vidéo est modifiée ou non. Ainsi la proportion entraînement/validation est de 80/20, pour 8000/2000 séquences. Le modèle est entraîné avec une carte 1080 Ti disposant de 11G de mémoire vidéo sur 2000 époch.

Comme énoncé plus tôt dans 3.3.1, les frames prédites seront utilisées comme frames de références et comparées aux frames réelles de la vidéo. Pour illustrer le processus ci-dessous en parallèle deux séquences de frames :

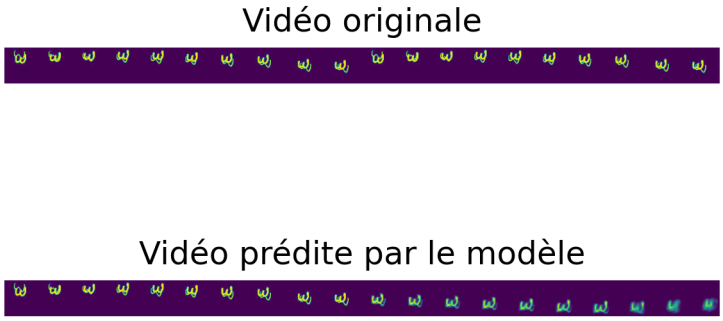


FIGURE 13 – La séquence originale et la séquence prédite par le modèle

Dans cet exemple la vidéo est en effet modifiée, on remarque aussi la qualité de prédiction de notre modèle et cela même sur plus de 10 frames futures. Cependant l'appréciation que l'on peut faire ici en tant qu'humain n'est pas aussi facile à faire pour un ordinateur et il n'est évidement pas question de regarder chaque séquence pour juger nous même de la séquence. Ainsi la nécessité d'une métrique pouvant quantifier la similarité des deux séquences est nécessaire, la première idée fût d'appliquer une MSE glissée paire par paire d'image de chaque séquence.

$$\forall i \in \{0, ..., nbFrames\}, MSE_i = \frac{1}{mn} \sum_{x=0}^{n-1} \sum_{y=0}^{m-1} F_i(x, y) - F'_i(x, y) \qquad (3)$$

Cependant même dans ce cas nous aurions besoin d'une métrique et décider d'un seuil arbitrairement semble contre-productif après avoir utilisé un modèle. On aurait pu directement utilisé cette méthode sur les frames de la vidéo originale.

3.3.4 Perspectives et limites de la méthode

La MSE est aussi bien trop sensible, en effet elle se réalise frame par frame et pixel par pixel, ce qui peut fortement pénaliser une prédiction même si celle-ci semble être assez bonne pour un humain. Une autre idée de test de similarité était tout simplement un autre modèle siamois pour ici encoder puis utiliser une distance sur les vecteurs dans l'espace latent. Cependant cette méthode nécessite plus de travail que de créer une réseau convolution qui lui regarde tout simplement la forme des chiffres et non leur position, bien que cela soit réalisable en modifiant la loss par exemple. Il existe un autre problème de taille, dans tous les papiers proposés et étudiés, aucun ne réalise de prédiction de frames en haute définition ou même une résolution raisonnable pour une caméra de vidéo-surveillance. C'est pourquoi j'ai choisi de revenir vers les idées de base du deep learning sur vidéo et d'utiliser le flux optique qui semble être très adapté pour cette tâche.

3.4 Utilisation du flux optique

3.4.1 Présentation flux optique

Le flux optique est une représentation du mouvement inter-frame, il est quantifié sous forme de champ de vecteurs, et permet de décrire le mouvement de chaque pixel dans une vidéo. Son calcul est souvent réalisé en vision par ordinateur par la méthode de $Lucas - Kanade$. Cette méthode résout les équations différentielles du flux optique. Cependant cette méthode pose problème, elle n'effectue qu'une résolution locale et ne peut donner de résultats pour des textures uniformes. Elle reste cependant parallèlisable ce qui fait d'elle la méthode de base dans le package OpenCV par exemple. Ci-dessous un exemple de flux optique :

Cependant, en 2015 la papier [3] propose un modèle d'estimation de flux optique ayant un temps d'inférence très bas du fait que le calcul soit réalisé sur GPU et non sur CPU pour les méthodes classiques. En 2016, une deuxième version appelé FlowNet2 est proposé dans le papier [5], améliorant grandement les performances en réduisant le temps d'inférence jusqu'à 7 ms par frame pour leur modèle FlowNet2-s. Dans la suite le modèle utilisé pour estimer le flux optique est FlowNet2.

Une fois notre flux optique obtenu pour l'ensemble de la vidéo, j'applique une technique utilisée dans le papier [13] visant à obtenir un facteur α pour

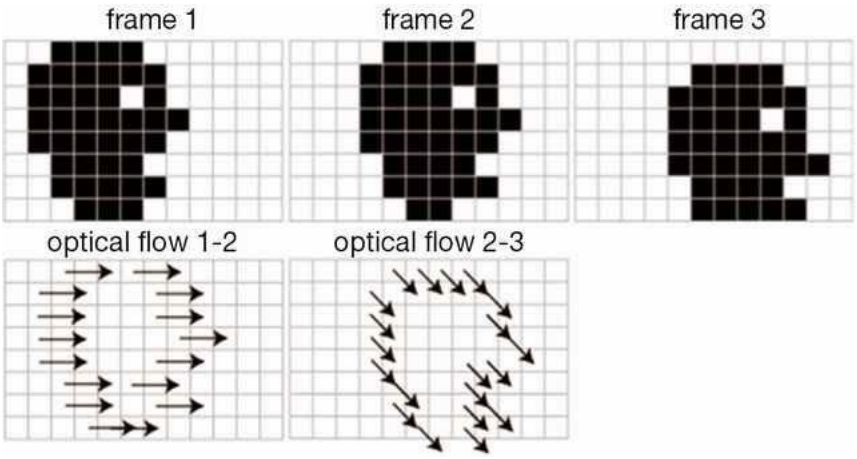


FIGURE 14 – Exemple de flux optique

une paire de frame, ce facteur α est obtenu de la façon suivante :

— Tout d'abord on obtient un scalaire pour chaque paire de frame de la vidéo :

$$\forall k \in \{0, ..., nbFrame - 1\}, OF(k) = \sum_x \sum_y \sqrt{(u^2 + v^2)} \qquad (4)$$

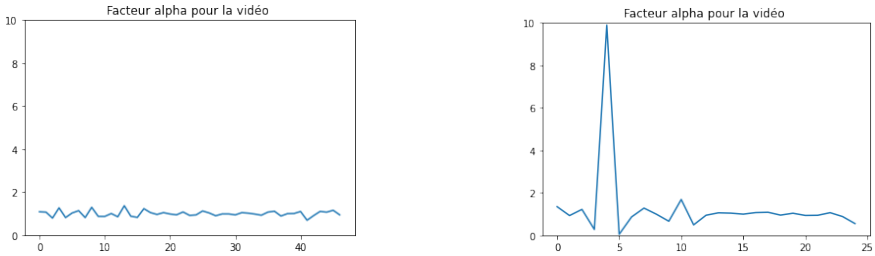
où u et v sont les vecteurs à la position (x,y) dans le champ de vecteur k.

— Enfin nous obtenons le facteur α de la façon suivante :

$$\forall k \in \{1, ..., nbFrame - 1\}, \alpha(k) = \frac{2 * OF(k)}{OF(k-1) + OF(k+1)} \qquad (5)$$

On remarque que si le flux optique est constant le facteur α se simplifie par $\frac{2}{2}$ ou 1. Ainsi nous pouvons tracer le profil de α tout le long d'une vidéo pour obtenir un signal 1D :

Le pic observable sur 15b correspond à un« décrochage »dans la vidéo. Ainsi cela nous permet même de pouvoir différencier les différentes modifications : pour 1 pic cela correspond à une suppression de frames, 2 pics soit une copie soit une insertion et 4 pics à un swap. Une autre propriété importante est la normalité des facteurs α pour une vidéo originale, ce qui permet pour une classification binaire de ne pas utiliser de modèle mais un simple test statistique. En l'occurrence ici le test de normalité *stats.normaltest* disponible dans le module *Scipy* pour Python basé sur le test D'Agostino-Pearson.



(a) Profil alpha pour une vidéo originale (b) Profil alpha pour une vidéo modifiée

FIGURE 15 – Exemple de profil alpha

3.4.2 Modèles utilisés

Concernant la classification multi-classes, nous voulons d'abord utiliser un test statistique pour différencier les séries modifiées des séries originales, puis ensuite réaliser une classification sur les séries identifiées comme modifiées. Pour cette classification nous utilisons un simple modèle récurrent codé sous PyTorch composé d'une couche de type LSTM, RNN ou GRU suivie d'une couche linéaire pour 4 sorties (les 4 types de modifications).

Le modèle FlowNet2 est fourni avec ses poids et est donc un modèle clé en main, il suffit juste de l'utiliser pas besoin d'entraînement.

3.4.3 Entraînement et Résultats

L'entraînement ici est beaucoup moins coûteux, la donnée d'entrée est un signal 1D de 100 valeurs. Nous disposons de 250 vidéos pour chaque modifications, cependant nous pouvons utiliser des techniques d'augmentation de données assez facilement. Un flip vertical change le signal mais en rien le label. Une autre technique est de réaliser un shift aléatoire qui là aussi ne change en rien au label mais permet de créer un « nouveau »signal. Le ratio entraînement/validation est 70/30 en gardant la même part de chaque label.

Je rappelle que les résultats de classification multiple sont effectués après un filtrage des vidéos non modifiées grâce à un test statistique, il est inutile de rajouter une 5 ème classe (vidéo originale) ce qui augmenterait ici l'accuracy et le F1 mais proposerait à coup sûr de moins bons résultats que le test statistique.

Résultats				
Type	Classif. Binaire		Classif. Multi Top@1 préd.	
	Acc (%)	F1 (%)	Acc (%)	F1 (%)
Test statistique (***)	90.2	-	-	-
Test statistique (**)	91.6	-	-	-
Test statistique (*)	93.9	-	-	-
LSTM	77	75	63	57
RNN	-	-	42	42
GRU	-	-	67	61

FIGURE 16 – Résultats de la méthode

3.4.4 Perspectives et limites de la méthode

Cette méthode semble prometteuse, les résultats obtenus avec le test statistique couplé à la rapidité inférence du modèle FlowNet permettent de penser qu'une application faisant ces tests en temps réel est facilement réalisable. Quant aux résultats proposés en classification multi-classes, le manque de dataset établi ne nous permet d'espérer un finetuning de modèle qui améliorerait sans doute les résultats. Le dataset utilisé ici comprend 250 vidéos de chaque classe et ne permet clairement pas de pouvoir obtenir les avantages du deep learning. Cependant le deep learning sur vidéo est bien plus contraignant que sur image de par la necessité de stocker, traiter et pour notre problème modifier puis ré-encoder les vidéos.

4 Bilan personnel

Pour conclure ce rapport, je voudrais évoquer ce que ce stage a pu m'apporter. Tout d'abord il m'a permis de découvrir le monde de la recherche bien différent du monde de l'industrie : j'ai eu un champ libre pour les méthodes utilisées et les sujets explorés. Ce stage m'a permis de mettre en pratique toute la théorie que j'ai pu assimiler durant mon master et m'a permis de mettre en contraste le monde académique et les problèmes réels rencontrés. On peut citer par exemple la difficulté à obtenir un dataset de qualité suffisante pour essayer d'utiliser à pleine efficacité les modèles de deep learning. Aujourd'hui seules les entreprises capables d'acheter un service de labellisation sont en mesure d'utiliser pleinement le deep learning et ce mal-

gré l'engouement général pour l'utilisation du deep learning que l'on peut rencontrer ces dernières années. Ce stage m'a aussi permis de découvrir le monde des conférences scientifiques auquel j'ai pu participer avec mon tuteur de stage M. Charrier, cette expérience est très enrichissante car elle permet de se tenir informer des dernières technologies à sortir, de pouvoir discuter avec des acteurs importants du monde de la recherche. J'ai aussi pu présenter mon travail lors de la journée du laboratoire à des collègues mais aussi à des industriels. J'ai aussi eu la chance de pouvoir participer à l'écriture d'un papier scientifique qui est sélectionné pour le colloque Gresti 2022 à Nancy.

Références

[1] Paolo Bestagini, Simone Milani, Marco Tagliasacchi, and Stefano Tubaro. Local tampering detection in video sequences. In *2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*, pages 488–493, 2013.

[2] Christoph Feichtenhofer, Axel Pinz, and Richard P. Wildes. Spatiotemporal residual networks for video action recognition, 2016.

[3] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet : Learning optical flow with convolutional networks. *CoRR*, abs/1504.06852, 2015.

[4] Vincent Le Guen and Nicolas Thome. Disentangling physical dynamics from unknown factors for unsupervised video prediction, 2020.

[5] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0 : Evolution of optical flow estimation with deep networks. *CoRR*, abs/1612.01925, 2016.

[6] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[7] David P Kreil, Michael K Kopp, David Jonietz, Moritz Neun, Aleksandra Gruca, Pedro Herruzo, Henry Martin, Ali Soleymani, and Sepp Hochreiter. The surprising efficiency of framing geo-spatial time series forecasting as a video prediction task – insights from the iarai $\widehat{4}$c competition at neurips 2019. In Hugo Jair Escalante and Raia Hadsell,

editors, *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, volume 123 of *Proceedings of Machine Learning Research*, pages 232–241. PMLR, 08–14 Dec 2020.

[8] Xingjian SHI, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. Convolutional lstm network : A machine learning approach for precipitation nowcasting. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[9] Zeina Sinno and Alan Conrad Bovik. Large-scale study of perceptual video quality. *IEEE Transactions on Image Processing*, 28(2) :612–627, 2019.

[10] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms, 2015.

[11] Du Tran, Jamie Ray, Zheng Shou, Shih-Fu Chang, and Manohar Paluri. Convnet architecture search for spatiotemporal feature learning, 2017.

[12] Wan Wang, Xinghao Jiang, Shilin Wang, Meng Wan, and Tanfeng Sun. Identifying video forgery process using optical flow. pages 244–257, 07 2014.

[13] Wan Wang, Xinghao Jiang, Shilin Wang, Meng Wan, and Tanfeng Sun. Identifying video forgery process using optical flow. pages 244–257, 07 2014.

[14] Yunbo Wang, Haixu Wu, Jianjin Zhang, Zhifeng Gao, Jianmin Wang, Philip S. Yu, and Mingsheng Long. Predrnn : A recurrent neural network for spatiotemporal predictive learning, 2021.

[15] Wei Yu, Yichao Lu, Steve Easterbrook, and Sanja Fidler. Crevnet : Conditionally reversible video prediction, 2019.