Mémoire

July 14, 2019

Contents

1 Sportagraph **3**

2 Introduction **4**

3 Détection des replays avec des approches "classiques" **5**
- 3.1 A Robust Replay Detection . . . 5
  - 3.1.1 Calcul du score de la luminance et filtrage des frames . 5
  - 3.1.2 Recherche du logo template parmis les frames filtrés . 5
  - 3.1.3 Recherche des logo . . . 6
  - 3.1.4 Recherche des replays . . . 6
  - 3.1.5 Résultats obtenus et conclusion sur cette méthode . . 6
- 3.2 Automatic summarization of soccer highlights using audio-visual descriptors . . . 7
  - 3.2.1 Algorithme . . . 7
  - 3.2.2 Conclusion . . . 7

4 Les approches proposées **8**
- 4.1 Détection des plans . . . 8
  - 4.1.1 Online, Simultaneous Shot Boundary Detection And Key Frame Extraction For Sports Videos Using Rank Tracing . . . 9
- 4.2 Première approche : ORB . . . 10
  - 4.2.1 Extraction des caractéristiques . . . 10
  - 4.2.2 KMeans . . . 10
  - 4.2.3 Expérimentation et résultat: . . . 10
  - 4.2.4 1 frame par shot . . . 11
  - 4.2.5 **TODO** meilleurs res . . . 11
  - 4.2.6 W frames par shot: . . . 11

4.2.7 1 fenêtre de frame par shot: . . . 12
4.2.8 1 fenêtre de frame par shot et différence des frames dans la fenêtre: . . . 14
4.3 Seconde approche : matching de contours . . . 15
4.3.1 Algorithme . . . 15
4.3.2 Mosaique de plan . . . 16
4.3.3 **TODO** resize img + put them together ie A + B = C 16
4.3.4 Résultats et limitation . . . 17
4.3.5 **TODO** : mettre les résultats ici . . . 17

5 Apprentissage profond : les bases théoriques 18
5.1 Réseaux de neurones récurrents (RNN) . . . 18
5.2 LSTM . . . 18
5.3 CNN . . . 19
5.3.1 Intéraction parcimonieuse . . . 20
5.3.2 Partage de paramètres . . . 21
5.3.3 Représentations équivariantes . . . 21
5.3.4 Pooling . . . 21

6 Apprentissage profond : état de l'art pour la reconnaissance d'action dans les vidéos 22
6.1 Two-Stream Convolutional Networks for Action Recognition in Videos . . . 22
6.1.1 Classifieur spatial . . . 22
6.1.2 Classifieur temporel . . . 23
6.1.3 Méthode d'évalutation et résultats obtenus . . . 23
6.2 Learning Spatiotemporal Features with 3D Convolutional Networks . . . 24
6.2.1 Apport de l'article . . . 25
6.2.2 Architecture et entraînement du réseau . . . 25
6.2.3 Résultat pour la tâche de reconnaissance d'action . . . 27
6.2.4 Conclusion . . . 27
6.3 Beyond Short Snippets: Deep Networks for Video Classification 28
6.3.1 Approche . . . 28

7 Collecte des données pour l'apprentissage profond 29
7.0.1 Architecture du scrapper . . . 29
7.1 Datasets . . . 29
7.2 Détection des frames logo . . . 29
7.2.1 Propre modèle . . . 29

7.2.2 VGG net 29
7.2.3 Transfert Learning 29
7.2.4 Comparaison résultat 29
7.3 Détection des séquences de frames logo 29

8 Appendice **29**

9 Bibliographie **30**
9.1 Articles relatifs à l'apprentissage profond 30
9.2 Articles relatifs à la détection de replays 30
9.3 Articles générals de vision par ordinateur 31

10 Table des figures **31**

1 Sportagraph

Sportagraph est une start-up filiale de Everspeed, fondée en 2013 et lancée en 2016 après trois années de recherche et développement (R&D). Son produit est un DAM (Digital Asset Manager) spéecialisé dans les événements sportifs. Le DAM permet aux utilisateurs de stocker et de gérer plusieurs types de fichiers (principalement des images, mais les vidéos et d'autres types sont aussi supportés). L'équipe travaillant sur ce projet est composée de :

- Alex Macris : Chief Experience Officer
- Edouard Binchet : Chief Strategy and Operation Officer
- Régis Jean-Gilles : Lead Server Developer (mon tuteur d'alternance)
- Benoit Hozjan : SaaS operation manager
- Francois Jacquier : Lead Developer Front
- Khedidja Almherabet : Développeur front-end
- Mate Gyomrei : Développeur front-end
- Daniel Denke : Développeur iOS
- Zoltan Tarsoly : Test Automation Engineer
- Youva Ammaouche : Assistant chef de projet
- Stéphane : Développeur front-end

- Ashant : Développeur front-end

En tant que développeur back end, je suis chargé de rajouter de nouvelles fonctionnalitées au DAM. L'une des missions qui m'a été affecté est la détection des replays dans les vidéos de sport. Cette fonctionnalité nous permettra de fournir à nos clients un résumé automatique des vidéos qu'ils uploadent sur le DAM.

2 Introduction

Au cours des dernières années, les techniques d'apprentissage automatique ont joué un rôle de plus en plus important dans les systèmes de reconnaissance automatique. Les avancements dans le domaine de l'apprentissage profond et l'accès à un volume massif de données ont permi de mettre en place des solutions de plus en plus performantes. En particulier, l'adoption globale des téléphones intelligents et de leur caméra intégrée a accru de manière exponentielle le nombre de vidéos disponibles sur Internet, à tel point qu'il est devenu impossible pour l'humain d'ingérer manuellement le contenu de toutes les vidéos disponibles sur la toile. Dans cet étude, nous allons nous intéressons à la tâche de "video summarization" ou, en français, de récapitulation de vidéo. Plus particulièrement, nous focalisons notre recherche sur les vidéos de match de football. Dans ces vidéos, notre objectif va être de parvenir à identifier les replays; car à partir de ces replays, nous serons capables de mettre en avant les moments importants du match.

Pourquoi les replays ? Un replay est la retransmission d'une action qui s'est déjà passé au cours d'une vidéo. Les replays sont intéressants car ils sont un indicateur d'un moment fort dans une vidéo. En effet, c'est l'équipe technique chargée du montage de la vidéo qui décide ou non de créer un replay pour une action, un replay est une annotation humaine sur une vidéo. Typiquement, un replay sera incrusté dans la vidéo après une action importante comme, par exemple, un but ou un pénaltie.

Caractéristique des replays Les replays sont introduits et se terminent par un logo ([1]). Ces logos ont en général une apparence qui se démarque facilement des autres images dans la vidéo. Les replays ne sont pas à confondre avec les ralentis. Les ralentis sont un type particulier de replays où l'action est montrée de nouveau en *slow-motion*, mais tous les replays ne sont pas des ralentis. C'est pourquoi, la vitesse de déplacement des objets dans l'image (pour détecter l'effet de *slow-motion*) n'est pas un bon critère pour la détection de replay.

L'objectif de notre recherche est double. Dans un premier temps, nous mettons en place un système de détection de replays en utilisant uniquement les méthodes "classiques" tels que ORB (**ref**) Ensuite, nous comparons plusieurs approches d'apprentissage automatique pour la tâche de détection dans les vidéos. Enfin, nous comparons l'efficacité des méthodes de vision par ordinateur "classique" aux méthodes par apprentissage profond (deep learning).

3 Détection des replays avec des approches "classiques"

L'un des objectifs de notre recherche est de mettre en place un modèle de détection de replays n'utilisant que des techniques ne faisant pas intervenir d'apprentissage automatique. Ce modèle servira de base de comparaison avec le modèle par apprentissage automatique que nous proposerons par la suite. Afin d'avoir le meilleur modèle, nous avons comparé plusieurs approches.

3.1 A Robust Replay Detection

Cette approche détecte les replays en trouvant les logos dans la vidéo. Les logos sont trouvés grâce à la luminance. Nous savons qu'un logo est présent pendant 0.8 secondes soit 18 frames pour une vidéo de 24 FPS. Tous les frames qui ont obtenu un score supérieur à un certain seuil sont considérés comme des "logo templates".

3.1.1 Calcul du score de la luminance et filtrage des frames

L'idée est de parcourir toute la vidéo et de calculer pour chaque frame la différence de luminance qu'il y a entre ce frame et les 17 frames précédents. Nous obtenons un score L_i pour chaque frame i dans la vidéo. Tous les frames dont le score est inférieur à un certain seuil sont écartés, les autres vont servir à trouver le logo template.

3.1.2 Recherche du logo template parmis les frames filtrés

Le logo template est le frame qui représente le mieux tous les logos dans la vidéo. Pour déterminer le logo template parmis les frames filtrés, l'algorithme K-means est utilisé pour séparer cet ensemble en deux (K = 2) en fonction de la luminance moyenne des frames. Pour trouver le logo template, nous allons chercher dans le cluster avec le centre de cluster le plus élevé, puis

sélection le frame m minimisant la distance avec tous les autres frames du cluster.

$$d_{mn} = \sqrt{\sum_{i=0}^{\text{binsize}} \left[\frac{H_m(i)}{\sum_{j=0}^{\text{binsize}} H_m(j)} - \frac{H_n(i)}{\sum_{j=0}^{\text{binsize}} H_n(j)} \right]^2} \quad (4)$$

Figure 1: Formule de la distance entre deux frame dans le cluster

3.1.3 Recherche des logo

Une fois que le logo template est déterminé, chaque logo trouvé en 1 va être comparé avec le logo template. La mesure de comparaison est la distance (équation 4) qu'il y a entre le frame et le template dans le cluster. Tous les frames qui ont une distance inférieure à un certain threshold sont considérés comme des logos.

3.1.4 Recherche des replays

Une fois que les logos sont détectés, nous pouvons trouver les replays en cherchant les paires de logos éloignés de moins de 80 seconde (durée maximum d'un replay).

3.1.5 Résultats obtenus et conclusion sur cette méthode

	Video					
Video width / Video height / Luminance threshold	Football : Premier League (150k frames)	Football: Ligue 1 (80k frames)	Football: Ligue 1 (150k frames)	Football: Liga (300k frames)	Tennis: Australia Open (400k frames)	American football: NFL (*) (200k frames)
100,100, 50000	>100,5/48,150	45,0/16,70	>100, 0/34, 200	>100,0/36,150	>100, 12/32, 280	**X**
100,100, 75000	>100,12/48,150	21,0/16,70	58,0/34,200	43, 0/36, 150	**>100, 12/32**, 280	**X**
100,100, 100000	52, 32/48, 150	7, 0/16, 70	16, 0/34, 200	23, 12/36, 150	53, 15/32, 280	**X**
100,100, 125000	12, 48/48, 150	**4, 0/16, 70**	**5, 0/34, 200**	13, 36/36, 150	33, 32/32, 280	**X**
100,100, 150000	5, 48/48, 150	2, 2/16, 70	2, 3/34, 200	4, 36/36, 150	23, 32/32, 280	**X**

Score : False Positive (FP), False Negative (FN), Time (rounded)

(*) Didn't work because in NFL games, the logo at the and at the beginning of a replay do not match

Après avoir implémenté cette méthode, nous avons constaté que celle-ci n'est pas efficace et ne fonctionne pas du tout sur notre ensemble de test. Cette approche est trop dépendant de la luminance et elle ne parvient pas à détecter les logos peu lumineux. De plus, les bases mathématiques (notamment la

manière de choisir le cluster et la mesure de distance) sont un peu douteuses. Enfin, celle-ci dépend trop du paramètre "seuil de luminance" affectant les logos détectés. Le seuil de luminance fournis par les auteurs ne produit pas de bons résultats sur toutes les vidéos. Nous n'avons pas réussi à trouver une valeur pour le seuil de luminance qui obtienne universellemnt de bons résultats. Un seuil à 100000 détecte les logos de Ligue 1 mais pas les logos de Liga. Un seuil de 75000 détecte les logos de Liga et de Ligue 1, mais laisse passer trop de faux positifs. Les logos de Premier League quant à eux ne sont pas tous détectés avec un seuil à 50000, alors que ce seuil accepte un grand nombre de faux positifs. Pour conclure, cette approche n'est pas celle qui va nous permettre de mettre en place un système de détection de replays robuste et efficace.

3.2 Automatic summarization of soccer highlights using audio-visual descriptors

Cette approche ressemble à la précédente, mais introduit néanmoins une différence importante : un pré-traitement sur la vidéo pour en extraire les plans.

3.2.1 Algorithme

```
S = Détecter tous les shots (plans) dans la vidéo
L = Pour chaque shot S_i:
1. L_i_start = La "luminance" des frames au début du shot
2. L_i_end = La "luminance" des frames à la fin du shot
3. L_template = Trouver le "logo template" dans L
4. Pour chaque logo l dans L:
   1. Diff l avec L_template = conversion grayscale puis somme de la soustraction pixel
   2. Si Diff l avec L_template < threshold => l est un logo
```

3.2.2 Conclusion

Cette méthode est trop semblable à l'approche "Robust Replay Detection" qui ne répond pas à nos besoin, cette approche ne fonctionnera pas dans notre cas (la luminance n'est pas un critère assez discriminant pour la reconnaissance de logo). Néanmoins, l'idée de découper la vidéo en "shot" (en plan) est intéressante et nous nous en servons par la suite.

4 Les approches proposées

Les méthodes "état de l'art" ne donnent pas d'assez bons résultats et ne peuvent pas être mises en production. Nous cherchons donc notre propre méthode.
Pour détecter les replays, nous faisons les hypothèses que :

- un replay a un logo de début (I)
- un replay a un logo de fin (II)
- les logos de début et de fin sont les mêmes (III)
- les logos ont une forme facilement reconnaissable qui se distingue des autres images dans la vidéo (IV)
- un replay dure entre 2 et 90 secondes (V)

Nous proposons plusieurs approches permettant de détecter les logo de replay dans les vidéo de sport. Dans cette partie, chacune de ces approches n'utilisent que des algorithmes de computer vision classique (flouttage, filtre de Canny, ORB, ...) et des algorithmes de machine learning non-supervisés (K-NN). Ces restrictions s'appliquent pour les raisons suivantes :

- le programme doit être le plus rapide possible (les réseaux de neurones sont en général trop lents, trop exigeant en ressource); d'où le choix d'algorithme plus simple.
- si la solution doit être mise en production, il est préférable de ne pas avoir d'ensemble d'apprentissage à obtenir ou maintenir; d'où le choix d'algorithme non-supervisé uniquement.

4.1 Détection des plans

Les approches que nous proposons itérent sur tous les frames de la vidéo, à la recherche des logo pouvant se trouver au début et à la fin des replays. Si nous faisons l'hypothèse qu'un replay entraînera toujours un changement de plan, alors au lieu de rechercher les logo parmi tous les frames de la vidéo, nous réduisons la recherche à tous les frames qui sont entre deux plans.

4.1.1 Online, Simultaneous Shot Boundary Detection And Key Frame Extraction For Sports Videos Using Rank Tracing

Cette méthode est proposée par W. Abd-Almageed en 2008.
Chaque frame est converti en HSV et les histogrammes H, S et V sont calculés. Un vecteur est formé pour chaque frame à partir de ces histogrammes. Ensuite, une matrice M de dimension N * L, représentant une fenêtre de N frames va être formée à partir de ces vecteurs, où L est la taille des histogrammes et N la taille de la fenêtre.
L'algorithme SVD (singular value decomposition) va être appliquée sur M. M = UWV, où W est la matrice de valeur singulière.
Les diagonales de la matrice W comportent des poids S ordonnées de manière non croissante. Le premier poid S_1 est le poid maximal. Ces poids représentent l'information contenu dans le vecteur V.
Nous allons assigner un rang à la matrice M, ce rang va être égal au nombre d'élement s dans S tel que s/S1 > threshold. Le rang va être calculé pour chaque fenêtre de frame dans la vidéo.
Si le rang d'une fenêtre est plus que grand que le rang de la fenêtre avant elle, alors le contenu visuel de la fenêtre est différent de la fenêtre précédente. À l'inverse, si le rang est inférieure à la fenêtre précédente, alors le contenu visuel se stabilise. S'il est de 1, alors c'est stable.
Le début d'un frame est celui qui maximise le rang parmis les fenêtres environnantes.

1. Résultats obtenus et conclusion Cette méthode pour trouver les plans dans une vidéo est très efficace, et constitue la base de la suite de notre recherche.

   En effet, avant de segmenter la vidéo en plan, nous comparions N frames , où N peut être aussi grand que 400000 (pour des vidéo de 120 minutes à 60 fps). Il est impensable d'utiliser un algorithme en $O(N^2)$, par exemple en comparant toutes les frames entre elles, avec un N aussi grand.

   Après avoir segmenter la vidéo en plan, nous obtenons un N' au alentours de 2000 pour une vidéo de 120 minutes à 60 fps. Nous pouvons donc nous permettre d'utiliser des algorithmes plus complexes que sans la segmentation en plan. De plus, la segmentation en plan réduit le champs de recherche des frames logo, et donc le nombre de faux positifs potentiels.

4.2 Première approche : ORB

Dans cette approche, nous cherchons à reconnaître les logo dans la vidéo. Pour ce faire, nous optons pour une approche de clustering. L'idée est de clusteriser la vidéo en deux groupe : un groupe pour les frames logo, et un autre groupe pour les frames non-logo.

4.2.1 Extraction des caractéristiques

OpenCV permet d'extraire des features à partir des images (détection des bords des objets dans l'image). A partir de ça, nous pouvons représenter l'image comme un vecteur de feature. Les méthodes d'extraction sont ORB et AKAZE.

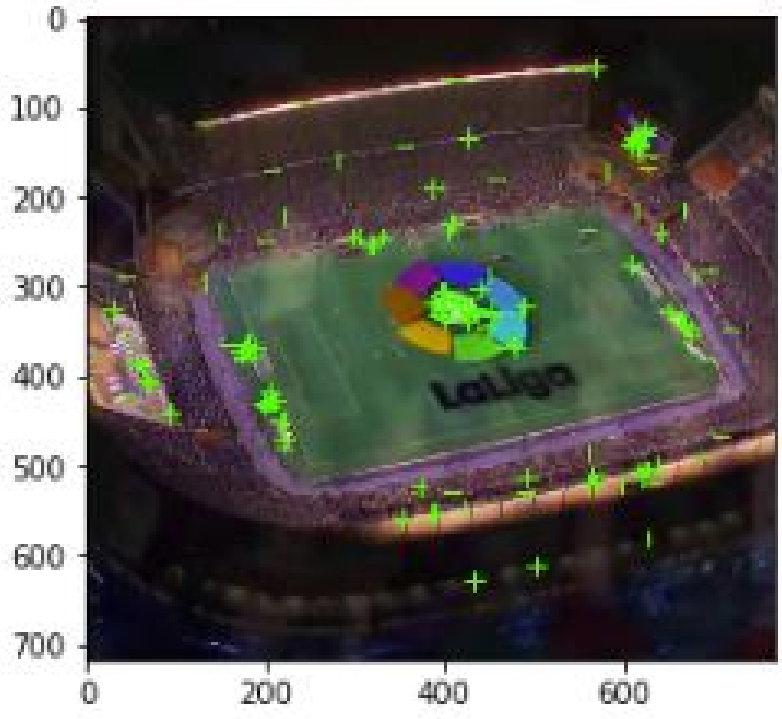
4.2.2 KMeans

OpenCV implémente aussi l'algorithme KMeans. Celui-ci permet de regrouper les objets similaires en fonction de leur feature. Dans notre cas, il va nous permettre de créer 2 groupes d'images : logo / non logo. L'avantage de KMeans est qu'il est est très rapide et assez efficace dans la plupart des cas. C'est l'un des algorithmes de clusterisation les plus utilisés.

4.2.3 Expérimentation et résultat:

Ensemble de test : une vidéo de ligue 1, une vidéo de liga, une vidéo de premier league et une vidéo NFL. Dans toutes les expérimentations, la vidéo est découpée en shot (plan). Soit S l'ensemble des shots.

4.2.4 1 frame par shot




```
- Récupérer le frame à la fin de chaque shot
  - nous obtenons |S| frame
- Pour chaque frame, calculer ses features (orb ou akaze)
  - Nous obtenons |S| vecteurs
- Utiliser KMeans avec K=2 pour séparer les vecteurs en deux groupes
  - le groupe le plus petit est le groupe des logo
```

Résultats :

4.2.5 TODO meilleurs res

Mauvais sur toutes les vidéos

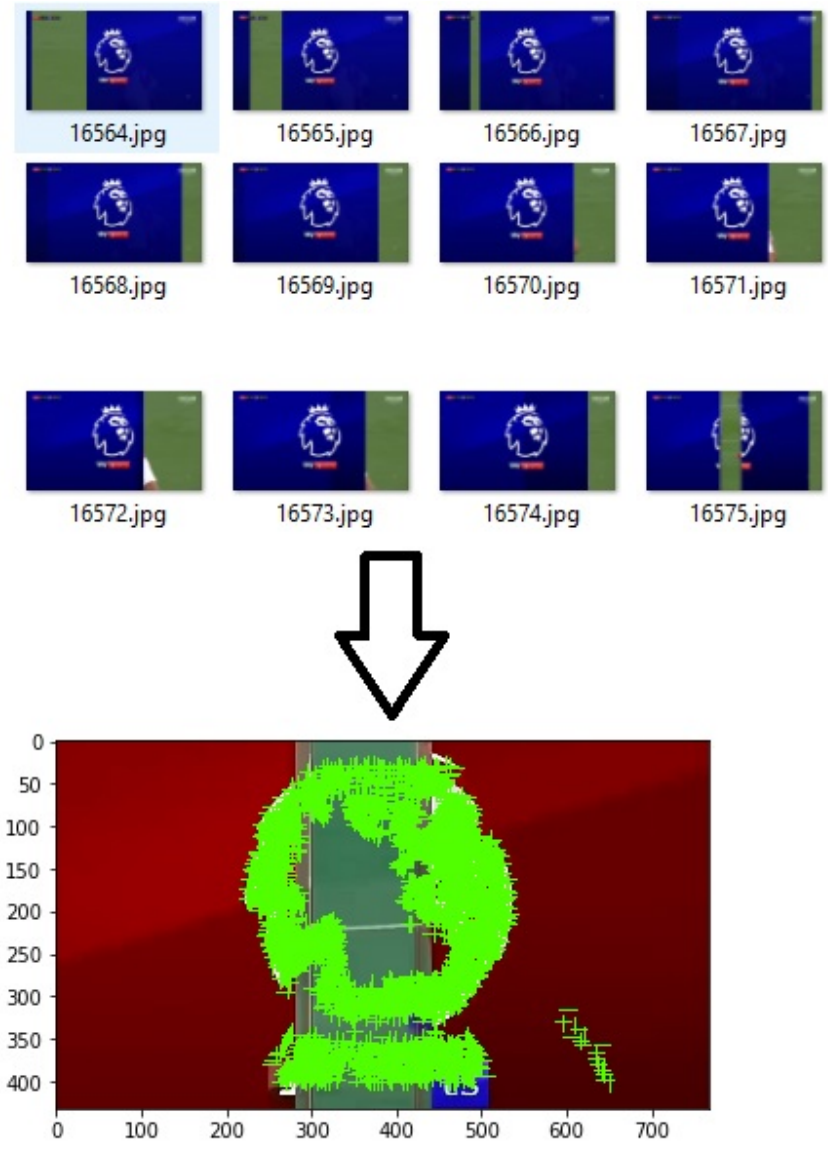
4.2.6 W frames par shot:

```
- Récupérer W frames pour chaque shot
  - nous obtenons |S*W| frame, où W est le nombre de frame
- Pour chaque frame, calculer ses features (orb ou akaze)
```

  - nous obtenons |S*W| vecteurs
- Utiliser KMeans avec K=2 pour séparer les vecteurs en deux groupes
  - le groupe le plus petit est le groupe des logo

Résultats : Mauvais sur toutes les vidéos

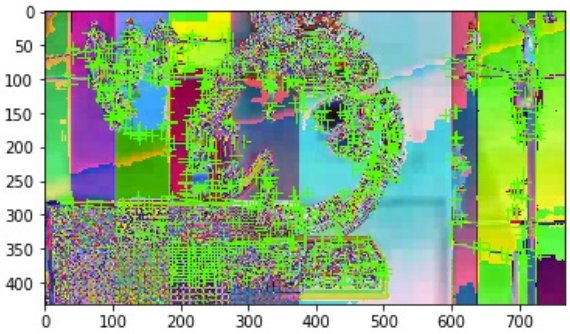
4.2.7 1 fenêtre de frame par shot:



```
- Récupérer W frames pour chaque shot, les régrouper en une fenêtre
  - nous obtenons |S| fenêtre de dimension W, où W est le nombre de frame
- Pour chaque fenêtre, calculer ses features (orb ou akaze)
  - Nous obtenons un vecteur de dimension |S*W|
- Utiliser KMeans avec K=2 pour séparer les vecteurs en deux groupes
  - le groupe le plus petit est le groupe des logo
```

Résultats: De bons résultats sur la vidéo de PL. Mauvais résultats sur les autres vidéos.

4.2.8 1 fenêtre de frame par shot et différence des frames dans la fenêtre:



- Récupérer W frames pour chaque shot, les régrouper en une fenêtre
  - nous obtenons |S| fenêtre de dimension W, où W est le nombre de frame
- Pour chaque fenêtre, calculer la matrice M égale à la différence de toute les autres frames dans la fenêtre
- Pour chaque matrice de différence, calculer ses features
  - nous obtenons |S| vecteur s

```
- Utiliser KMeans avec K=2 pour séparer les vecteurs en deux groupes
  - le groupe le plus petit est le groupe des logo
```

Résultats : De bons résultats sur la vidéo de PL. Mauvais résultats sur les autres vidéos.

4.3 Seconde approche : matching de contours

La méthode choisie différe avec les autres sur un point : au lieu de chercher à différencier les frames logo des frames non-logo, nous allons chercher les frames qui ont des formes en commun dans la vidéo. En effet, d'après l'hypothèse III, il est fort probable que si un frame à l'instant t a beaucoup de formes en commun avec un frame à l'instant t', avec $2 < t' < 90$ (hypothèse V), alors il y a un logo à l'instant t et un logo à l'instant t', et un replay entre t et '.

4.3.1 Algorithme

```
- Pré traitement sur les shots
  1. Redimensionner
  2. Cropper
  3. Supprimer le background (s'étendre la dessus)
  4. Détecter le contour (Canny Edge Detection)
  5. Génération des mosaiques
- Pour chaque mosaique de plan S_A :
  - Pour chaque mosaique de plan S_B après S_A :
    1. Contour_commun = C_A & C_B
    2. Contours_diff = Détection du contour de Contour_commun (cv2.findContours)
    3. Résultat = Ne garder que les contours qui sont assez longs (ceux qui ont au moi
    4. Si Résultat > Seuil : alors S_A et S_B sont des logos potentiels
- Pour chaque logo potentiel LP :
  1. Le comparer avec les autres logo L' (même procédure qu'en 2)
  2. Si au moins 2 logo L'  match, alors LP est un logo
- Trouver les replays grâce aux logos
```

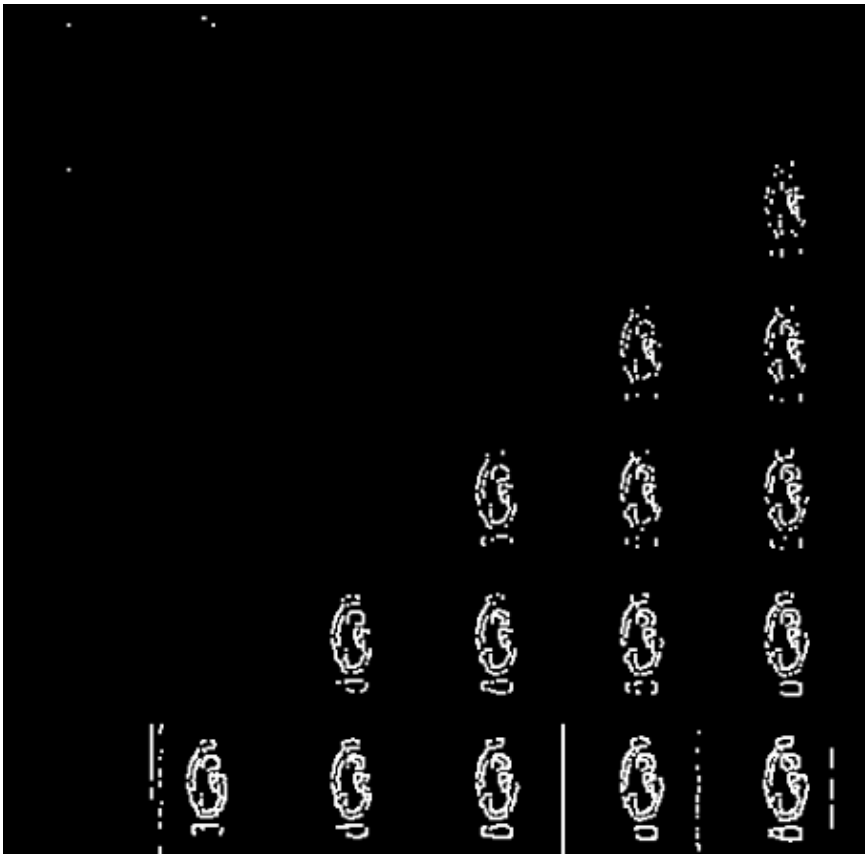
Pré traitement : Les frames sont resizé puis cropé vers le centre (pour ne pas avoir l'affichage en haut de l'écran etc...), puis un blur est appliqué (bilateralFilter, permet de filtrer certains faux positifs), et enfin on applique Canny Edge Detection.
Le point 3 de l'algorithme sert à filtrer les éventuels faux positifs. Notre algorithme est sensible au plan fixe et aux images avec beaucoup de bruits

(ces images ont beaucoup de contours détectés par l'algorithme de détection de contours). Beaucoup de ces faux-positifs peuvent être filtrer lors du prétraitement sur les plans, notamment en rajoutant du blur ou en supprimant le background, néanmoins, nous ne sommes pas parvenus à filtrer 100% des faux-positifs.

4.3.2 Mosaique de plan

4.3.3 TODO resize img + put them together ie A + B = C

Pour chaque shot deux images au format .png (pas au format jpg, car celui-ci prend trop d'espace disque) sont générées.
Chaque image est de dimension I * I * width * height où I est le nombre de frame dans le shot.
Ces images sont en faites des matrices d'image qui vont permettre de comparer rapidement deux shot. La première matrice a un décalage d'un frame par ligne, la seconde n'a pas de décalage.
Pour comparer deux shot, il suffit d'appliquer un ET binaire entre les matrices des mosaiques, puis de calculer la longueur du conteur dans cette matrice.



4.3.4 Résultats et limitation

Les résultats sans le filtrage des faux positifs (l'étape 3 de l'algorithme) sont un bon moyen d'évaluer l'efficacité de notre méthode.

4.3.5 TODO : mettre les résultats ici

Concernant le temps d'exécution, celui-ci est relié presque entièrement à la taille de la vidéo donnée en entrée, ainsi qu'à la taille des mosaiques.
Les limitations de notre méthode sont les suivantes :

- Dans certaines vidéos, il n'y a pas de logo pour les replays (simple fondu)
- Dans certaines vidéos, les logo de début et fin de replay ne sont pas les mêmes.

- Dans certains vidéos, il y a des logo au début des replays, mais pas de logo à la fin des replays (un simple fondu remplace le logo).

5 Apprentissage profond : les bases théoriques

Dans notre recherche, nous allons aborder plusieurs types de réseaux d'apprentissage automatique. Nous allons présenter dans cette partie les principes fondamentaux à la bonne compréhension de ces derniers.

5.1 Réseaux de neurones récurrents (RNN)

Les RNN (Recurrent Neural Networks), ou réseaux de neurones récurrents (RNR) en français, sont capables de répéter leur couche cachée, en utilisant comme entrée la sortie de toutes les couches précédentes et de générer une sortie pour chaque couche. Cela va leur permettre de prendre en entrée des séquences et de retourner des séquences. En effet, pour une entrée $[e_1, e_2, \ldots, e_n]$ et un initialiseur s_0, le RNN va répéter n fois sa couche cachée, de telle sorte à générer une sortie s_1 associée à la couche 1 et à l'entrée (e_1, s_0); puis il va générer une sortie s_2 associée à la couche 2 et à l'entrée (e_2, s_1), etc ... Pour finir, nous aurons en sortie la séquence $[s_1, s_2, \ldots, s_n]$.
Par exemple, appliqués à la génération de phrase, les RNN vont être capables de générer (mot par mot, ou n-gram par n-gram) des séquences de phrases de longueur arbitraire.
Pour apprendre un modèle, le RNN va avoir besoin d'un ensemble d'entraînement qui met en avant les propriétés qui nous intéressent dans le modèle.
La nature récursive de ces réseaux les rend particulièrement adaptés aux tâches de traitement du langage naturel.

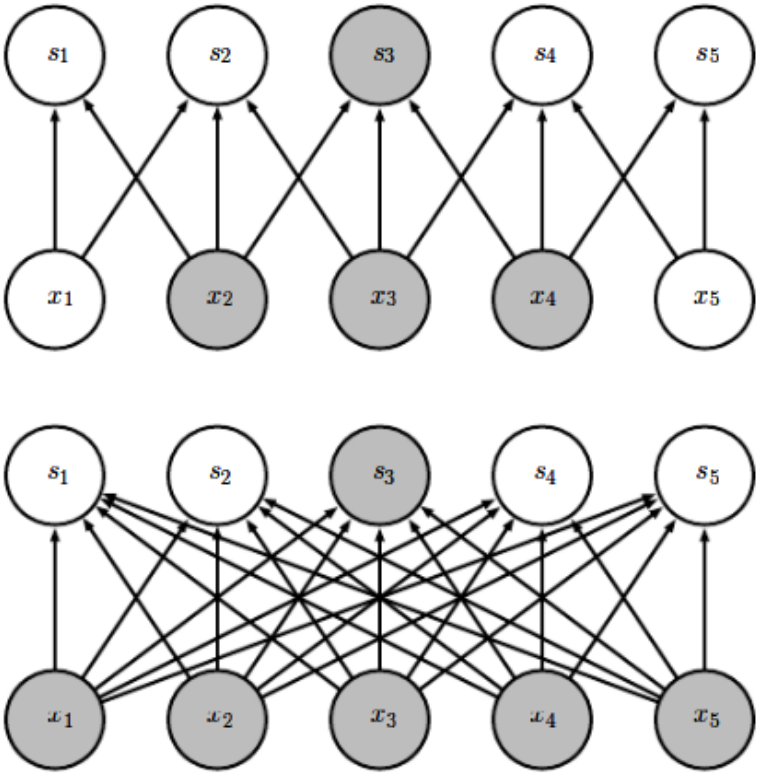
5.2 LSTM

Les LSTM (Long Short Term Memory) sont un type de RNN **à portes (gated RNN)**. Ces réseaux sont sur l'unité récurrente à portes. Ces portes vont permettre de stocker l'information apprise par le réseau à travers le temps. À la différence des RNN classiques, les LSTM sont capables d'oublier de l'information grâce à leur **leaky unit** afin d'éviter une explosion ou une disparition du gradient. Concrétement, cela va leur permettre de pouvoir capturer des dépendances à long terme de manière bien plus efficace que les RNN classiques.

5.3 CNN

Les CNN (Convolutional Neural Networks), ou réseaux de neurones convolutifs (RNC) en français, sont un type de réseau de neurones qui utilisent la convolution au lieu de la multiplication matricielle dans au moins une de leurs couches. La convolution est une opération qui prend en argument l'entrée (typiquement un vecteur représentant une donnée) et un **noyau** (les paramètres qui vont être appris par le CNN) et renvoie une **carte de caractéristiques** (feature map). Le noyau est une matrice qui va parcourir l'entrée et appliquer l'opération de convolution. Pour parcourir l'entrée, celle-ci va être divisée en plusieurs matrices carrées de même taille que le noyau (par ex 2x2 ou 6x6) en ajouteur si nécessaire du *padding* et du *striding*. La fonction de convolution a trois caractéristiques importantes : l'**intéraction parcimonieuse** ("sparse interaction"), le **partage de paramètres** et les **représentations équivariantes**. La couche de convolution est généralement composée de la fonction de convolution suivie d'une fonction d'activation non linéaire (par exemple, ReLU ou tanh) et d'une d'une fonction de **pooling**.

5.3.1 Intéraction parcimonieuse



À la différence des réseaux classiques où toutes les entrées intéragissent avec toutes les entrées, les réseaux à convolution ont des **intéractions parcimonieuses**. C'est à dire que la taille du noyau (donc de l'intéraction avec l'entrée), est plus petite que la taille de l'entrée. C'est pourquoi les réseaux à convolution sont très efficaces pour le traitement d'image. En effet, une image a une dimension en entrée de $c*l*w$ où c est le nombre de canaux de l'image (un seul pour une image en noir et blanc, trois pour une image en couleur), l la largeur en pixel de l'image et w la longueur en pixel de l'image. Une peite image couleur de dimension 100*100 aura 100*100*3 paramètres en entrée, ce qui provoque une explosion combinatoire avec les réseaux classiques qui n'ont pas d'intéraction parcimonieuse. Un réseau de convolution, quant à lui, aura un noyau d'une dizaine ou d'une centaine de pixel qui parcourt l'image à la recherche de caractéristiques significatives comme des contours. Ils ont besoin de stocker moins de paramètres, ils ont donc ont

besoin de moins de mémoire (pour la même tâche) et ont une meilleure efficacité statistique.

5.3.2 Partage de paramètres

Dans un réseau classique, un poid (un paramètre) est associé à chaque paramètre d'entré et ne sert qu'une fois. Tandis que dans un réseau convolutif, le noyau utilisé par une couche de convolution est le même sur toutes les matrices représentant l'entrée. Grâce à ce **partage des paramètres**, il n'y a qu'un les poids du noyau à apprendre au lieu d'un poid pour chaque neurone d'entrée. De plus, la taille du noyau est en général largement inférieure à celle de la couche d'entrée.

5.3.3 Représentations équivariantes

Une fonction est **équivariante** si, quand l'entrée change, la sortie change de la même manière. En terme mathématique, cela signifique que si $y = f(x)$ alors $g(y) = g(f(x))$. Les réseaux convolutifs sont équivariants à la translation. Dans le cas de l'image, celà signifie que le déplacement des pixels n'a pas d'influence sur le réseau.

5.3.4 Pooling

La fonction de pooling va modifier la sortie de la couche de convolution. Pour chaque valeur dans la carte des caractéristiques à la sortie de la convolution (après la fonction d'activation), la fonction de pooling va remplacer celle-ci en fonction de la valeur des cases voisines dans la carte. Une fonction de pooling usuelle est max pooling, qui va renvoyer la plus grande valeur dans un voisinage rectangulaire. L'utilité de la fonction de pooling est de rendre la représentation apprise par la couche de convolution **invariante** à de petites modifications sur l'entrée. Par exemple, dans le cas de la reconnaissance d'image, le réseau ne va pas chercher dans l'image en entrée les informations au pixel près. Si le réseau a appris à détecter les visages, il n'a pas besoin de retrouver l'emplacement des yeux au pixel près, une position approximative de ceux-ci lui suffira. Une autre utilité du pooling est de réduire la taille de la sortie de la couche de convolution. Nous pouvons voir le pooling comme un résumé de la carte des caractéristiques obtenue par convolution.

6 Apprentissage profond : état de l'art pour la reconnaissance d'action dans les vidéos

Nous nous intéressons à l'état de l'art concernant la détection d'action dans les vidéos. En effet, la transition d'un logo s'effectue sur plusieurs frames consécutives; il y a donc une composante temporelle à notre recherche, et nous pouvons considérer la transition d'un logo comme une action.

6.1 Two-Stream Convolutional Networks for Action Recognition in Videos

Cet article est écrit par Karen Simonyan et Andrew Zisserman. Dans celui-ci, ils proposent de séparer la tâche de reconnaissance d'action dans les vidéos en deux parties : une composante spatiale et une composante temporelle. La composante spatiale contient l'information concernant sur les objets dans la vidéo; tandis que la composante temporelle l'information sur les déplacements de ces objets et de la caméra. A partir de ces observations, les auteurs proposent d'entraîner un classifieur spatial (Spatial stream ConvNet) et un classifieur temporel (Temporal stream ConvNet). Ces classifieurs sont des réseaux de neurones convolutifs profonds.

6.1.1 Classifieur spatial

Ce réseau a une architecture de classifieur d'image classique. Il va permettre de donner un indice fort pour la prédiction, car certaines actions sont très liées à certains objets. De plus, la recherche dans le domaine de la classification est un domaine à part entière; toutes les avancées dans le domaine augmenteront l'efficacité de ce classifieur. Il n'est pas nécessaire d'apprendre ce réseau "from scratch" (de zéro), les approches de transfer learning sont efficaces.

6.1.2 Classifieur temporel

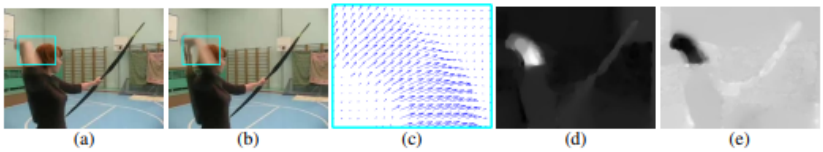


Figure 2: **Optical flow.** (a),(b): a pair of consecutive video frames with the area around a moving hand outlined with a cyan rectangle. (c): a close-up of dense optical flow in the outlined area; (d): horizontal component d^x of the displacement vector field (higher intensity corresponds to positive values, lower intensity to negative values). (e): vertical component d^y. Note how (d) and (e) highlight the moving hand and bow. The input to a ConvNet contains multiple flows (Sect. 3.1).

Source : Two-Stream Convolutional Networks for Action Recognition in Videos, Figure 2 L'innovation de l'article vient de l'introduction du classifieur temporel. L'idée est de détecter le mouvement des objets dans la vidéo, car un mouvement est la représentation d'un objet dans le temps. Les auteurs appellent leur approche "optical flow stacking" (empilement de flux optique). Dans celle-ci, ils utilisent la méthode "optical flow" pour détecter le mouvement des objets entre des frames consécutives. Ils définissent aussi un hyperparamètre L qui définit la distance maximum entre deux frames pour laquel il faut calculer le flux optique. Par exemple, si L=5, alors pour le frame t, il faudra calculer le flux entre le frame t et le frame t+1; entre t+1 et t+2; etc... jusqu'à t+4 à t+5. Chacun de ces flux servira d'entrée au classifieur temporel pour le frame t.

6.1.3 Méthode d'évalutation et résultats obtenus

Table 4: **Mean accuracy (over three splits) on UCF-101 and HMDB-51.**

Method	UCF-101	HMDB-51
Improved dense trajectories (IDT) [26, 27]	85.9%	57.2%
IDT with higher-dimensional encodings [20]	**87.9%**	61.1%
IDT with stacked Fisher encoding [21] (based on Deep Fisher Net [23])	-	**66.8%**
Spatio-temporal HMAX network [11, 16]	-	22.8%
"Slow fusion" spatio-temporal ConvNet [14]	65.4%	-
Spatial stream ConvNet	73.0%	40.5%
Temporal stream ConvNet	83.7%	54.6%
Two-stream model (fusion by averaging)	86.9%	58.0%
Two-stream model (fusion by SVM)	**88.0%**	**59.4%**

Source : Table 4: Mean accuracy (over three splits) on UCF-101 and HMDB-51.
Le classifieur spatial est pré-entrainé avec ImageNet, tandis que le temporel est entraîné de zéro (car il n'y a pas de réseau déjà entraîné pour cette tâche). Les dataset utilisés pour l'entraînement et l'évaluation sont UCF-101 et HMDB-51, contenant à eux deux près de 20000 vidéos annotées.

Note Pour calculer la classe d'un frame à l'instant t, les auteurs proposent deux méthodes :

- fusion par la moyenne (by averaging) : $y_t = y_{tspatial} + y_{ttemporal} / 2$
- fusion par SVM (by SVM) : un SVM multiclasse linéaire est entrainé pour

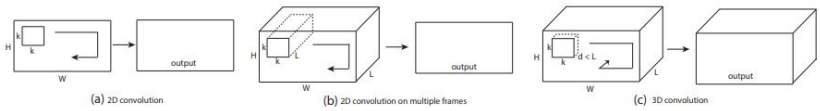
prédire la classe à partir du softmax des scores L2-normalisés.
Les résultats en **link table** montrent l'efficacité de leur méthode par rapport aux autres approches état de l'art.
Nous pouvons voir que leur approche two-stream avec fusion SVM est la plus efficace sur le dataset UCF-101, et qu'elle a aussi de bons résultats sur HMDB-51.
Ce qui est le plus intéressant dans cet article, c'est l'amélioration qu'apporte l'ajout de la composante temporelle. En effet, le classifieur d'image simple (spatial) n'a que 73.0% (UCF-101) et 40.5% (HMBD-51), tandis que le classifieur qui prend en compte l'image et la temporalité (two-stream model) atteint **88.0%** et 59.4%; ce qui est une nette amélioration. Cet article nous a renforcé dans l'hypothèse qu'il est nécessaire d'étudier une vidéo non pas comme une suite d'image indépendante, mais comme une suite de séquence avec un lien au sein de chaque séquence. Il semblerai que la temporalité a une très grande importance pour l'analyse de vidéos.

6.2 Learning Spatiotemporal Features with 3D Convolutional Networks

Dans cet article, Du Tran et. al proposent une approche pour apprendre les caractéristiques spatio-temporelles dans les vidéos grâce à un réseau de neuronnes profond. L'objectif est d'apprendre des caractéristiques qui soient :

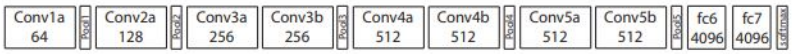
- générique : c'est à dire la capacité à representer différents types de vidéos
- compact : afin de pouvoir stocker un grand nombre de ces caractéristiques
- efficace (computationnellement): pour traiter les vidéos en temps réel
- simple (à implémenter) : afin de fonctionner même avec les modèles simples (comme un classifieur linéaire)

6.2.1 Apport de l'article



Source : Learning Spatiotemporal Features with 3D Convolutional Networks Figure 1 Dans l'approche Two-Stream Convolutional Networks for Action Recognition in Videos [2], deux réseaux à convolution 2D sont utilisés (un spatial et un temporel). L'apprentissage du réseau temporel est séparé de celui du réseau réseau spatial, ce qui ne permet pas d'avoir un réseau capturant l'information temporel relativement à l'information spatial. De plus, même si le réseau temporel prend plusieurs frames en entrée (paramètre L > 1), la convolution 2D va écraser la composante temporelle car sa sortie sera une image (2D). La figure **XXX** illustre la différence entre un réseau à convolution 2D (approche de K. Simonyan et A. Zisserman) et un réseau à convolution 3D (Tran et al.). Un réseau à convolution 2D aura en sortie une image (2D) même si l'entrée est une séquence d'images, tandis qu'un réseau à convolution 3D aura en sortie une image relative à une autre dimension (dans notre cas, le temps).
En choisissant d'entraîner leur réseau à partir de séquence d'images, les auteurs espèrent pouvoir apprendre la temporalité d'une manière "moins artificielle" que dans l'approche précédente (qui nécessite un pré-traitement sur les images pour pouvoir calculer les images d'optical flow servant au réseau temporel).

6.2.2 Architecture et entraînement du réseau



Source : Figure 3 L'entrée de ce réseau est de dimension c * l * h * w où c est le nombre de canal des images (3 pour la couleur, 1 pour les images en noir et blanc), l le nombre d'image dans les séquences, h la longueur et w la largeur. L'architecture conseillée par les auteurs est 8 couches de convolution et 5 couches de pooling, ainsi que 2 couches complètement connectées et la fonction softmax pour la couche de sortie. Le kernel recommandé par les auteurs est 3 * 3 * 3 avec un pas (stride) de 1 * 1 * 1 pour toutes les couches de convolution. Toutes les couches de pooling sont max pooling avec une taille de kernel 2 * 2 * 2 (sauf pour la première qui est 1 * 2 * 2) avec un

stride 2 * 2 * 2 (sauf pour la première qui a un stride de 1 * 2 * 2). Pour finir avec l'architecture, les deux couches complètement connectées ont 4096 sorties.
Ce réseau va être entraîné de zéro par descente du gradient à partir de séquences d'images annotées. Le taux d'apprentissage est de 0.003 et est divisé par 10 toutes les 4 epoch. L'entraînement s'arrête après 16 epoch.
Après l'entraînement, le réseau peut être utilisé comme un extracteur de caractéristique pour des tâches d'analyse vidéo. Pour se faire, la vidéo va être découpée en des clips de 16 frames (avec 8 frames de chevauchement entre deux clips consécutifs). Ensuite, chacun de ces clips va être passé au réseau et l'avant dernière couche complètement connectée (fc6) va contenir les caractéristisques du clip.
Qu'est-ce que ce réseau apprend ? Ce réseau apprend à se focalisant sur l'image des premiers frames, et à traquer leur déplacement dans les frames suivants.

6.2.3 Résultat pour la tâche de reconnaissance d'action

Method	Accuracy (%)
Imagenet + linear SVM	68.8
iDT w/ BoW + linear SVM	76.2
Deep networks [18]	65.4
Spatial stream network [36]	72.6
LRCN [6]	71.1
LSTM composite model [39]	75.8
C3D (1 net) + linear SVM	82.3
C3D (3 nets) + linear SVM	**85.2**
iDT w/ Fisher vector [31]	87.9
Temporal stream network [36]	83.7
Two-stream networks [36]	88.0
LRCN [6]	82.9
LSTM composite model [39]	84.3
Conv. pooling on long clips [29]	88.2
LSTM on long clips [29]	88.6
Multi-skip feature stacking [25]	89.1
C3D (3 nets) + iDT + linear SVM	**90.4**

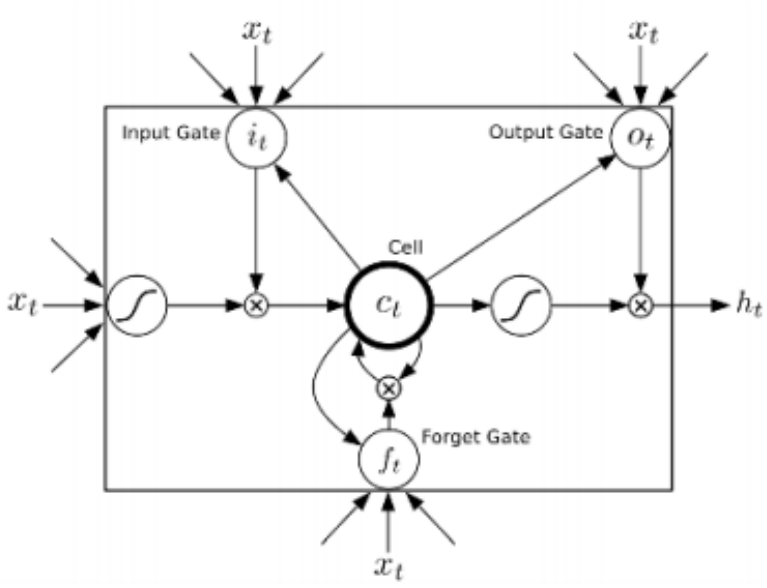
Ces résultats ont été obtenus par les auteurs pour la tâche de reconnaissance d'action sur le corpus de vidéo UCF101. Nous voyons que l'approche par réseau à convolution 3D est la plus efficace.

6.2.4 Conclusion

Dans cet article, les auteurs ont adressé le problème de la temporalité dans les vidéos. Ils ont montré qu'un réseau à convolution 3D est capable de modéliser l'information temporelle et spatiale simultanément, et donc d'obtenir de meilleurs résultats que les réseaux à convolution 2D sur plusieurs tâches d'analyse de vidéos. Pour ma part, je trouve cette approche très intéressante et élégante, de par sa simplicité. En effet, le réseau apprend la temporalité sans qu'il soit nécessaire d'injecter l'information temporelle de manière artificielle ou bien en compliquant l'architecture du réseau.

6.3 Beyond Short Snippets: Deep Networks for Video Classification

Dans cet article [3], les auteurs proposent d'utiliser une architecture hybride à base de CNN et de RNN (LSTM) pour l'analyse vidéo. Les CNN sont des réseaux particulièrement efficaces pour analyser les frames des vidéos, c'est le CNN qui va se charger de la composante spatiale de la vidéo.



Source : Beyond Short Snippets: Deep Networks for Video Classification; Figure 3

6.3.1 Approche

L'aspect le plus important de la recherche des auteurs concernent la manière dont le LSTM va recevoir l'information du CNN. Deux architectures de CNN sont utilisés par les auteurs : AlexNet et GoogLeNet. Afin que le LSTM puisse utiliser la sortie du CNN, il est nécessaire d'avoir une architecture de pooling efficace. Les auteurs en proposent cinq.

7 Collecte des données pour l'apprentissage profond

L'approche par matching de contours convient tout à fait à la tâche de scrapping. En effet, elle est :

- rapide : moins de **X** minutes sur une machine **machine de référence EC2 (ou autre)** pour une vidéo de 90 minutes à 60 fps
- précise : seulement **X** % de faux positifs sur **Y** logos scrappés

7.0.1 Architecture du scrapper

- requête HTTP avec une ID youtube => logo uploadé sur GCP
- image docker (avec le serveur à l'écoute des requêtes) déployée sur le cloud

Cette architecture est scalaire; ceci nous a permi de scrapper plusieurs vidéos en parallèle et d'obtenir un dataset conséquent.

7.1 Datasets

- Dataset non logo
- Dataset logo
- Dataset logo séparé en fonction du logo (ligue 1, premier league, ...)

7.2 Détection des frames logo

7.2.1 Propre modèle

7.2.2 VGG net

7.2.3 Transfert Learning

7.2.4 Comparaison résultat

7.3 Détection des séquences de frames logo

8 Appendice

Clustering : procédé permettant de regrouper des éléments Histogramme : représentation d'une image en fonction de ses canaux de couleurs (rouge,

vert, bleu) Frame : une image à l'instant t d'une vidéo Shot : un plan FPS : frame per second / image par seconde Cropper : sélectionner une partie continue des pixels l'image RNN : Recurrent Neural Networks, ou réseaux de neurones récurrents (RNR) en français CNN : Convolutional Neural Network, réseaux de neurones convolutifs (RNC) en français LSTM : Long Short Term Memory

9 Bibliographie

9.1 Articles relatifs à l'apprentissage profond

Gradient-Based Learning Applied to Document Recognition; Y. LeCun, L. Bottou, Y. Bengio, P. Haffner; 1998 Learning Hierarchical Features for Scene Labeling; C. Farabet, C. Couprie, L. Y. LeCun; 2013 Two-Stream Convolutional Networks for Action Recognition in Videos; K. Simonyan, A. Zisserman; 2014 Learning Spatiotemporal Features with 3D Convolutional Networks; D. Tran, L. Bourdev; R. Fergus; L. Torresani; M. Paluri; 2014 Beyond Short Snippets: Deep Networks for Video Classification; J.Y.H. Ng, M. Hausknecht; 2015 L'apprentissage profond; I. Goodfellow, Y. Bengio, A. Courville ; 2015

9.2 Articles relatifs à la détection de replays

A Robust Replay Detection Algorithm for Soccer Video; W. Xu, Y. Yi(2011). Replay and key-events detection for sports video summarization using confined elliptical local ternary patterns and extreme learning machine; A. Javed, A. Irtaza; 2019 Video Co-summarization: Video Summarization by Visual Co-occurrence; W. Chu; 2015 **??** Automatic Detection Of Replay Segments In Broadcast Sports Programs By Detection Of Logos In Scene Transitions; H. Pan; 2002 Mean Shift Based Video Segment Representation And Applications To Replay Detection; L. Duan; 2004 Online, Simultaneous Shot Boundary Detection And Key Frame Extraction For Sports Videos Using Rank Tracing; W. Abd-Almageed; 2008 On-line Key Frame Extraction and Video Boundary Detection using Mixed Scales Wavelets and SVD; A. Azeroual; 2016 Highlight Summarization in Soccer Video based on Goalmouth Detection; Z. Zhao; 2006 A General Framework for Automatic Online Replay Detection in Sports Video; B. Han, Y. Yan; 2009 An Efficient Framework for Automatic Highlights Generation from Sports Videos; A. Javed; 2016 A New Slow-Motion Replay Extractor For Soccer Game Videos; E. Farn, L. Chen; 2003 Fast Highlight Detection and Scoring for Broad-

cast Soccer Video Summarization using On-Demand Feature Extraction and Fuzzy Inference; M. Sigari, H. Soltanian-Zadeh; 2015 Automatic Summarization Of Soccer Highlights Using Audio-visual Descriptors; A. Raventós, R. Quijada, L. Torres, F. Tarrés; 2014 A Novel Method For Slow Motion Replay Detection In Broadcast Basketball Video; C. Chen; 2014

9.3 Articles générals de vision par ordinateur

ORB: an efficient alternative to SIFT or SURF; E. Rublee, V. Rabaud; 2011

10 Table des figures

1 Xu, W., & Yi, Y. (2011). A Robust Replay Detection Algorithm for Soccer Video. IEEE Signal Processing Letters, 18(9). Equation (4) **??** I. Goodfellow, Y. Bengio, A. Courville ; L'apprentissage profond; 2015; Chapitre 9, Figure 9.2

References

[1] Hao Pan, Baoxin Li, and Sezan. Automatic detection of replay segments in broadcast sports programs by detection of logos in scene transitions. *IIEEE International Conference on Acoustics Speech and Signal Processing*, 2002.

[2] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199, 2014.

[3] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015.