Apprentissage non-supervisé de représentations pour l'analyse de séquences vidéos

Guillaume Lorre

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité Informatique

Préparée au sein de INSA ROUEN NORMANDIE

Apprentissage non-supervisé de représentations pour l'analyse de séquences vidéos

Présentée et soutenue par

GUILLAUME LORRE

Thèse soutenue publiquement le 19/07/2021 devant le jury composé de		
Mme Catherine ACHARD,	Professeure à l'Université Sorbonne, Paris	Rapporteur
M. Louahdi KHOUDOUR,	Directeur de recherche Cerema, Toulouse	Rapporteur
M. Thierry CHATEAU,	Professeur à l'Université Clermont Auvergne	Examinateur
M. Stéphane CANU,	Professeur de l'INSA-Rouen, Normandie	Directeur de thèse
Mme Samia AINOUZ,	Professeure de l'INSA-Rouen, Normandie	Codirectrice de thèse
M Jaonary RABARISOA,	Chercheur scientifique au CEA, Saclay	Membre invité
Mme Astrid ORCESI,	Chercheuse scientifique au CEA, Saclay	Membre invité

Thèse dirigée par STÉPHANE CANU et Samia AINOUZ

INSA ROUEN NORMANDIE

MI∫IΣ

litis

Remerciements

Je remercie Catherine Achard et Louahdi Khoudour pour avoir accepté de rapporter ce manuscrit de thèse. Je remercie également Thierry Château pour avoir accepté d'être membre du jury.

Je tiens à remercier mes encadrants au CEA Jaonary Rabarisoa et Astrid Orcesi qui m'ont beaucoup aidé tout au long de la thèse. J'ai beaucoup appris en suivant leurs conseils autant d'un point de vue technique qu'organisationnel ou de présentation. J'ai notemment eu des discussions très intéressantes avec eux qui nous ont permis de proposer de nouvelles méthodes, tout celà dans une très bonne ambiance.

Un grand merci aussi à mes co-directrice et directeur de thèse Samia Ainouz et Stéphane Canu pour leur soutient et leur aide. Ils ont toujours des remarques intéressantes lors de nos points et au niveau de la rédaction d'articles et du manuscrit.

Je remercie aussi le laboratoire du LVA pour m'avoir accueilli pendant ces 3 années de thèse. Les personnes y sont très sympatiques et les présentations et discussions scientifiques très intéressantes.

Un grand merci aussi à mes parents qui m'ont soutenu pendant cette période.

Finalement, je souhaite aussi remercier mes amis et colocataires pour les temps passés ensemble en dehors du travail.

Résumé

Une faculté humaine importante est notre capacité à analyser notre environnement, notamment par la vue. Il nous est ainsi possible de reconnaître des objets, les localiser ou encore de comprendre les intentions et les activités des personnes qui nous entourent. Le domaine de la vision par ordinateur a pour but de créer des systèmes capables d'effectuer automatiquement ce même type de tâche à partir de capteurs, comme des caméras par exemple. Les systèmes « classiques » extraient des descripteurs visuels à partir d' images et les donnent en entrée d'un algorithme d'apprentissage statistique (de classification par exemple). A l'inverse, les réseaux de neurones prennent en entrée la donnée brute et déterminent par apprentissage les représentations adéquates grâce à leur structure composée d'une succession de couches. L'utilisation de réseaux de neurones a permis de largement surpasser les performances des méthodes « classiques ». Par exemple, le développement de réseaux convolutionnels a permis d'obtenir des performances similaires à celles des humains en reconnaissance d'images sur la base Imagenet. Les réseaux de convolution 3D permettent de différencier plus de 600 actions différentes avec une précision de plus de 90 % à partir de vidéos sur la base de données Kinetics. Ces réseaux sont appris de manière supervisée sur des bases de données annotées. Ces avancées ont été rendues possibles par l'augmentation significative de leur taille. La base de données Imagenet contient par exemple plus d'un million d'images et la base Kinetics plus de 500 000 vidéos. Malheureusement, l'annotation de ces bases est très coûteux et long. En effet, il est souvent effectué par du *crowd sourcing* avec *Amazon Mechanical Turk* par exemple. C'est une des limitation principale des méthodes d'apprentissage profond. En revanche, il nous est possible, à nous humain, de reconnaître des objets le plus souvent à partir d'uniquement quelques exemples contre plusieurs centaines ou milliers pour les réseaux de neurones. La raison la plus plausible de notre efficacité en termes de données annotées est que nous utilisons un mécanisme d'apprentissage non-supervisée en observant le monde qui

nous entoure. Cet apprentissage non-supervisé nous permet de transformer les stimuli visuels en une représentation haut niveau à partir de laquelle il est facile de classifier et raisonner. Par exemple, la théorie du *predictive coding* soutient que notre cerveau maintient à jour un modèle abstrait de notre environnement [HR11]. Celui-ci est utilisé pour prédire les stimuli futurs. L'erreur par rapport aux stimuli réels permet de corriger le modèle. Ainsi, tirant profit de toutes les données qui nous parviennent, nous pouvons apprendre de manière très efficace. Le contenu multimédia non annoté étant accessible de manière presque illimité, il serait donc intéressant de pouvoir les exploiter pour améliorer les performances de manière similaire au cerveau humain. Ainsi, la recherche en apprentissage profond s'est récemment intéressée aux méthodes d'apprentissage de représentations non-supervisées. Celles-ci permettent de réduire le nombre de données annotées nécessaires en pré-entraînant le modèle sur des données non-annotées.

Cette thèse propose de nouvelles méthodes d'apprentissage non-supervisé pour l'analyse de séquences vidéo. Le but est donc d'obtenir une représentation haut niveau d'une vidéo, pouvant par la suite être utilisée pour différentes tâches. Nous nous focalisons sur la tâche de reconnaissance d'actions qui est très utile pour la compréhension de vidéos. Il s'agit de déterminer l'action effectuée par une ou plusieurs personnes sur un clip vidéo pré-découpé. D'autres tâches plus complexes comme la détection d'actions, la détection d'anomalies dans les vidéos ou encore le vidéo *captionning* pourraient aussi être étudiées mais sont laissés pour des développements futurs.

Contributions et organisation du manuscrit

La partie 1 explique rapidement le fonctionnement des réseaux de neurones. On décrit notamment la notion d'apprentissage de représentations supervisé et non-supervisé ou pré-apprentissage. Différentes méthodes de pré-apprentissage non-supervisé y sont décrites. La partie 2 décrit les différentes méthodes d'apprentissage non-supervisé de l'état de l'art appliquées au domaine de la vision (sur des images et des vidéos). On note la différence entre 3 types de méthodes : les méthodes génératives, la prédiction de transformations et les méthodes contrastives. Les méthodes contrastives reformulent l'apprentissage non-supervisé en classification entre un exemple positif et des exemples négatifs. Leurs avantages par rapport aux autres méthodes y sont présentés. Nous nous intéressons à ce type de méthodes

dans nos contributions. Finalement, la partie 3 présente les méthodes de reconnaissance d'actions basées sur les réseaux de neurones. Typiquement, les différentes architectures, pré-traitements, modalités, utiles pour la reconnaissance d'actions y sont décrit. Ce sont les réseaux de neurones présentés que nous avons pour but de pré-apprendre de manière non-supervisée afin d'améliorer les performances en reconnaissance d'actions. Notre contribution principale est dans la proposition et l'application de nouvelles méthodes contrastives aux vidéos. Nous proposons trois méthodes différentes qui sont décrites par la suite.

Le *Predictive Coding* (prédiction du futur) est reconnu pour être une méthode intéressante d'apprentissage non-supervisé. Récemment, A. van den Oord et ses coauteurs [vdOLV18] ont proposé de d'allier le *Predictive Coding* aux méthodes contrastives pour effectuer les prédictions dans l'espace des représentations. La partie 4 décrit comment nous avons adapté cette méthode pour prédire les représentations de segments futurs de vidéos. Les points cruciaux pour obtenir des représentations utiles pour la reconnaissance d'actions y sont détaillés. Cette méthode surpasse les résultats précédents de l'état de l'art.

Cette méthode de *Predictive Coding* est assez contrainte par la structure du modèle autorégressif. Nous proposons ici une méthode ou l'agrégation de l'information de contexte est effectué par une architecture Transformer. Cette architecture est utilisée pour le traitement du langage naturel et notamment dans la méthode BERT. Cette méthode permet de pré-apprendre des modèles sur les données textuelles en prédisant des mots masqués dans le texte. Nous adaptons ici cette méthode aux séquences de données non catégorielles en ajoutant une fonction de coût contrastive. Dans l'application aux vidéos, on s'intéresse plus particulièrement à la stratégie de masquage qui joue un rôle essentiel pour obtenir de bonnes performances. Ces résultats sont présentés dans la partie 5.

Finalement, les méthodes contrastives nécessitent un nombre d'exemples négatifs important. Nous proposons d'augmenter ce nombre d'exemples au niveau des représentations grâce à l'opération de *mixup*. Cette solution est appliquée à un modèle non-supervisé plus simple (SimCLR) qui traite des paires plutôt que des séquences. Nous expérimentons sur les images et sur les vidéos et remarquons un gain conséquent en performance. Les résultats sont aussi au niveau de l'état de l'art pour la reconnaissance d'actions sur les vidéos. Cette méthode fait l'objet de la partie 6. Le chapitre 7 conclut le manuscrit et apporte différentes perspectives.

Publications

Les travaux présentés dans cette thèse ont fait l'objet de plusieurs publications :

Publications internationales

— Guillaume LORRE, Jaonary RABARISOA, Astrid ORCESI, Samia AINOUZ, Stéphane CANU. *Temporal Contrastive Pretraining for Video Action Recognition*. WACV 2020

— Guillaume LORRE, Jaonary RABARISOA, Astrid ORCESI, Samia AINOUZ, Stéphane CANU. *Contrastive Predictive Coding for Video Representation Learning*. Workshop on Self-Supervised Learning ICML 2019.

Publications nationales

— Guillaume LORRE, Jaonary RABARISOA, Astrid ORCESI, Samia AINOUZ, Stéphane CANU. *Reconnaissance d'actions et architectures à deux voies*. CAP 2018.

Publications en cours

— Guillaume LORRE, Jaonary RABARISOA, Astrid ORCESI, Samia AINOUZ, Stéphane CANU. *Contrastive Pre-training with Masked Video Modeling* en cours de soumission à ICIP 2021.

Table des matières

Table des matières **ix**

1 Pré-apprentissages supervisé et non-supervisé **1**

1.1 Classification supervisée et réseaux de neurones 2

1.1.1 Apprentissage supervisé 2

1.1.2 Réseaux de neurones 3

1.1.3 Les principales architectures utilisées dans ce travail 5

1.2 Pré-apprentissage supervisé 8

1.3 Pré-apprentissage non-supervisé 10

1.3.1 Principe général 10

1.3.2 Historique des méthodes de pré-apprentissage non-supervisé . 11

1.4 Conclusion 17

2 Méthodes d'apprentissage non-supervisé de représentations **19**

2.1 Applications des méthodes génératives 20

2.1.1 Application aux images 20

2.1.2 Application aux vidéos 21

2.2 Méthodes auto-supervisées 25

2.2.1 Application aux images 25

2.2.2 Application aux vidéos 27

2.3 Méthodes basées sur les pseudo-labels 28

2.4 Méthodes basées sur l'information mutuelle 30

2.4.1 Aspects théoriques 30

2.4.2 Applications pratiques 34

2.4.3 Expériences sur les méthodes contrastives 39

2.5 Conclusion 43

3 Reconnaissance d'actions **45**
3.1 Méthodes de reconnaissance d'actions supervisé 46
3.1.1 Réseaux de convolution 3D et variantes 46
3.1.2 Méthodes d'aggrégation à long terme 47
3.1.3 Utilisation du flot optique 49
3.1.4 Estimation de pose et reconnaissance d'actions 52
3.2 Bases de données 54
3.3 Résultats des méthodes de reconnaissance d'actions 56
3.4 Expériences effectuées sur la méthode *2 stream* 57
3.5 Conclusion 58

4 Apprentissage contrastif temporel **59**
4.1 Mise en contexte 59
4.2 Description des modèles 60
4.2.1 Modèle principal 61
4.2.2 Modèle sans auto-régressivité 66
4.2.3 Modèle bidirectionnel 67
4.2.4 Améliorations et réseaux 3D 68
4.2.5 Evaluation 69
4.3 Expériences 70
4.3.1 Détails d'implémentation 70
4.3.2 Entraînement 72
4.3.3 Résultats 74
4.4 Conclusion 84

5 Utilisation des modèles de langage **87**
5.1 Méthodes de langage 88
5.1.1 Transformer 88
5.1.2 Apprentissage non-supervisé de représentations en NLP 89
5.2 Modèles de langage contrastifs 92
5.2.1 Utilisation de BERT 92
5.2.2 Utilisation de XLNet 94
5.3 Applications 95
5.3.1 Application aux séquences d'images 96
5.3.2 Application aux vidéos 97
5.4 Conclusion 103

6 Augmentation du nombre d'exemples par mixup **107**
6.1 Pré-requis . . . 108
6.1.1 SimCLR . . . 109
6.1.2 Mixup . . . 110
6.2 Augmentation des exemples par mixup pour SimCLR . . . 112
6.3 Expériences sur les images . . . 115
6.3.1 Bases de données . . . 115
6.3.2 Evaluation sur les images . . . 115
6.4 Expériences sur les vidéos . . . 116
6.4.1 Détails d'implémentation . . . 117
6.5 Résultats sur les images . . . 117
6.6 Résultats sur les vidéos . . . 121
6.6.1 Résultats quantitatifs . . . 121
6.6.2 Résultats qualitatifs . . . 124
6.7 Conclusion . . . 127

7 Conclusion et perspectives **129**
7.1 Conclusion . . . 129
7.2 Perspectives . . . 130

8 Annexes **XXIII**
8.1 Résultats complémentaires CPC Vidéo . . . XXIII
8.1.1 Visualisations t-SNE pour différentes modalités . . . XXIII
8.1.2 Evolution des t-SNE en fonction du temps d'apprentissage non-supervisé . . . XXV
8.1.3 Evolution des scores de recherche de vidéos . . . XXVI
8.1.4 Résultats de recherche de vidéos . . . XXVI
8.2 Résultats complémentaires BERT Video . . . XXXVII
8.3 Résultats complémentaires SimCLR mixup . . . XLII

Table des figures **XLVII**

Liste des tableaux **LIII**

Chapitre 1

Pré-apprentissages supervisé et non-supervisé

Sommaire

1.1 Classification supervisée et réseaux de neurones 2

1.1.1 Apprentissage supervisé 2

1.1.2 Réseaux de neurones 3

1.1.3 Les principales architectures utilisées dans ce travail 5

1.2 Pré-apprentissage supervisé 8

1.3 Pré-apprentissage non-supervisé 10

1.3.1 Principe général 10

1.3.2 Historique des méthodes de pré-apprentissage non-supervisé 11

1.4 Conclusion 17

Cette partie définit la notion d'apprentissage non-supervisé de représentations aussi appelée pré-entraînement. Avant d'y parvenir, il est nécessaire d'expliquer les bases de l'apprentissage supervisé ainsi que les caractéristiques des réseaux de neurones. En effet, ces derniers permettent d'extraire des représentations intermédiaires contrairement aux autres méthodes d'apprentissage ce qui est le fondement du pré-entraînement. Les principes de pré-entraînement supervisé et non-supervisé sont introduits. Nous présentons par la suite les méthodes classiques de pré-apprentissage non-supervisé.

1.1 Classification supervisée et réseaux de neurones

1.1.1 Apprentissage supervisé

Etant donnée deux ensembles, un domaine **X** (par exemple $\mathbb{R}^d$) et un co-domaine **Y**, l'apprentissage supervisé a pour but de prédire une valeur $y \in \mathbf{Y}$ à partir d'une donnée d'entrée $x \in \mathbf{X}$, à l'aide d'une fonction de décision $f_\theta : \mathbf{X} \longmapsto \mathbf{Y}$ de paramètres θ. Pour estimer les paramètres θ, on utilise souvent une fonction de coût l, représentant l'erreur effectuée entre la prédiction et la vérité terrain, que l'on va chercher à minimiser. Pour un problème de classification, la vérité terrain prend des valeurs discrètes entre 1 et C, le nombre totale de classes à discriminer et $\mathbf{Y} = \{1, \ldots, \mathrm{C}\}$. Un encodage 1 parmi C est souvent utilisé pour y c'est à dire que $y_c = 1$ si c est la classe correcte et 0 sinon (que l'on apelle encodage *one-hot*). La sortie $\hat{y} = f_\theta(x)$ correspond à la probabilité que l'image appartienne aux différentes classes. La fonction de coût l utilisée en classification est le plus souvent l'entropie croisée

$$l(y, \hat{y}) = \frac{1}{\mathrm{C}} \sum_{c=1}^{\mathrm{C}} y_c \log \hat{y}_c \,. \tag{1.1}$$

On se place dasn un cadre probabiliste et on note X et Y les variables aléatoires correspondant aux entrèes et aux sorties et $\mathrm{P}(x, y)$ la densité de probabilité de leur loi jointe. Le but est de minimiser le risque défini par rapport à la fonction de coût

$$\mathrm{R}(f_\theta) = \mathbb{E}\Big[l\big(\mathrm{Y}, f(\mathrm{X})\big) \Big] = \int l\big(y, f_\theta(x)\big) d\mathrm{P}(x, y) \,. \tag{1.2}$$

En pratique, l'apprentissage s'effectue à partir d'une base de données annotée $\mathscr{D} = \{(x^{(i)}, y^{(i)}), i = 1, \ldots, \mathrm{N}\}$ et l'on cherche à minimiser le risque empirique

$$\mathrm{R}_{emp}(f_\theta) = \sum_{i=1}^{\mathrm{N}} l\big(y^{(i)}, f_\theta(x^{(i)})\big) \,. \tag{1.3}$$

Les méthodes d'apprentissage se différencient par leur fonction de coût l, leur fonction de décision f_θ ainsi que leur algorithme d'optimisation utilisé pour minimiser le risque empirique $\mathrm{R}_{emp}(f_\theta)$. Nous décrivons la spécificité des réseaux de neurone dans la partie suivante.

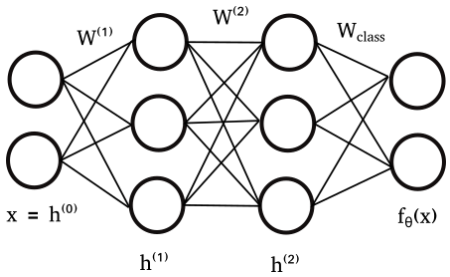


FIGURE 1.1 – Schéma d'un réseau de neurones multicouche (2 couches cachées de largeur 3 sont représentées).

1.1.2 Réseaux de neurones

Les réseaux de neurones se différencient des autres méthodes d'apprentissage par le fait qu'ils ne se basent pas sur des descripteurs faits à la main mais qu'ils prennent en entrée la donnée brute. Ils sont composés d'une succession de couches qui calculent des représentations intermédiaires de plus en plus abstraites [ZF13]. Par exemple, un réseau multicouche à $n+2$ couches, d'entrée $x = h^{(0)}$ et de sortie $f_\theta(x)$ est défini de la manière suivante

$$\begin{aligned} h^{(i)} &= a(\mathrm{W}^{(i)} h^{(i-1)} + b^{(i)}), \quad i = 1, \ldots, n \\ f_\theta(x) &= \mathrm{softmax}(\mathrm{W}_{class} h^{(n)} + b_{class}), \end{aligned} \tag{1.4}$$

où softmax indique une fonction d'activation dont la $i$ème composante s'écrit :

$$\mathrm{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)},$$

$a()$ est une fonction d'activation, $\theta = \left((\mathrm{W}^{(i)}, b^{(i)})_{i=1,\ldots,n}, \mathrm{W}_{class}, b_{class}\right)$, $\mathrm{W}^{(i)}$ et $b^{(i)}$ les poids de la couche i et $h^{(i)}$ les représentations intermédiaires. Une fonction d'activation populaire est la fonction linéaire rectifiée (ou ReLU pour *rectified linear unit*) définie par :

$$a(t) = \max(t, 0).$$

Les réseaux convolutionnels plus adaptés aux images sont basés sur le même type de structure mais utilisent comme opérations de base les convolutions et le *pooling* sur des cartes 2D de caractéristiques.

Les poids de ces réseaux sont appris de manière itérative par descente de gradient stochastique, ceux-ci calculés par rétropropagation. Stochastique indique ici

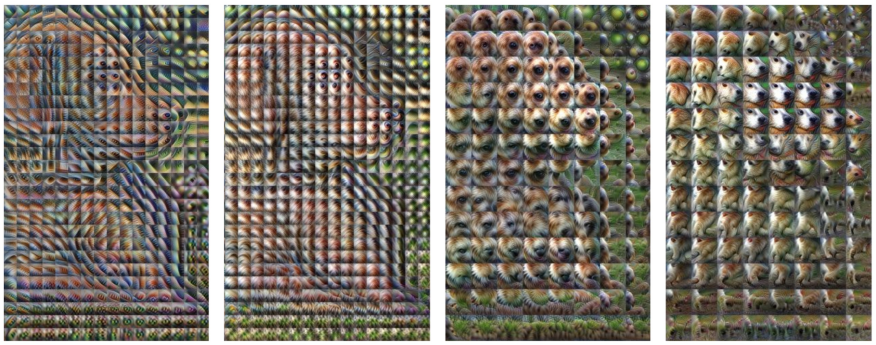


FIGURE 1.2 – Visualisation des activations d'un réseau de neurone convolutionnel (*Inception* [SVI+16]) pour différentes couches.

que l'optimisation lors d'une itération n'est effectuée en ne prenant en compte qu'une partie des exemples appelée *batch*. Ainsi, les poids des différentes couches sont optimisés pour résoudre la tâche en question. Les représentations intermédiaires apprises sont donc mieux adaptées à la tâche que les descripteurs faits à la main. Ces derniers sont par exemple les points *SIFT* [Low04] ou les histogrammes de gradients orientés [DT05] pour les images ou encore les points d'intérêt spatio-temporels pour les vidéos [LL03]. Notons $\mathrm{W}(t)$ la valeur des poids à l'itération t, la méthode de descente de gradient est décrite par l'algorithme 1, où le scalaire positif λ désigne le pas de l'algorithme.

Algorithm 1 Descente de gradients

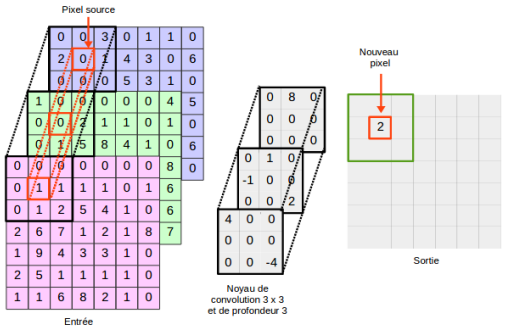
Initialiser aléatoirement $\mathrm{W}(0)$
for $t \in \{0, \ldots, n_{iter}\}$ **do**
  Sélectionner un batch B parmi $\mathscr{D}$
  Calculer $\sum_{i \in \mathrm{B}} \nabla_{\mathrm{W}(t-1)} l(y_i, f_\theta(x_i))$ par rétropropagation
  Mettre à jour les poids $\mathrm{W}(t) = \mathrm{W}(t-1) - \lambda \sum_{i \in \mathrm{B}} \nabla_{\mathrm{W}(t-1)} l(y_i, f_\theta(x_i))$
end for

La méthode de descente de gradients est appliquée pour tous les poids du réseau ($\mathrm{W}^{(i)}$ et $b^{(i)}, i = 1 \ldots, n$). Afin de ne pas surcharger les notations, les indices de couche ne sont pas notés dans l'algorithme 1.

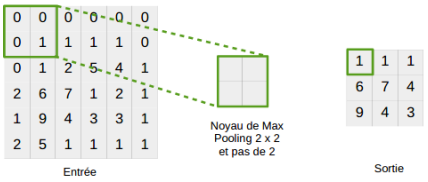
1.1.3 Les principales architectures utilisées dans ce travail

Dans la partie précédente, nous avons présenté le principe général d'un réseau de neurones et sa forme la plus simple (le réseau multicouche). Nous décrivons ici des réseaux plus complexes qui sont adaptés à différents types de données. Les réseaux convolutionnels ont par exemple été développés spécifiquement pour traiter des images, les réseaux récurrents pour traiter des séquences. Ces architectures seront utilisées par la suite dans ce manuscrit. Pour plus de détails et une liste plus complète des architectures de réseaux de neurones (*deep learning*) populaires, voir par exemple [KSZQ20].

Réseaux convolutionnels



(a) Schéma d'une convolution avec un noyau de taille 3 × 3. Un seul filtre est représenté en sortie



(b) Schéma de l'opération de *max pooling* avec un noyau de taille 2 × 2. Les dimensions spatiales de la carte de caractéristique sont divisées par 2.

FIGURE 1.3 – Illustration des opérations classiques utilisées dans les réseaux convolutionnels (repris de [Cha17])

Les réseaux convolutionnels sont utilisés pour extraire de l'information à partir d'images de dimensions (H × W × 3 avec H et W sa longueur et largeur). Leur principale composante est l'opération de convolution qui effectue un produit matriciel d'une zone de l'image avec un noyau (poids appris de taille K × K × 3 × C avec

K la taille du noyau et C le nombre de filtres (voir figure 1.3a). Leur sortie est appelée carte de caractéristiques et sont les représentations internes du réseau de convolutions. Nous avons pris ici l'exemple de la première couche qui s'applique sur l'image. Les couches de convolution suivantes sont appliquées sur la carte des caractéristiques en sortie de la couche qui la précède.

Les convolutions ont certaines propriétés qui les rendent performantes pour la reconnaissance d'images. En premier lieu, elles sont locales et permettent donc d'extraire des informations prenant en compte des zones de l'image de plus en plus importantes. C'est la notion de champ réceptif. De plus, les mêmes filtres étant utilisé dans toutes les zones de l'image, celà ajoute une propriété d'équivariance par translation. Des opérations de *pooling* peuvent être utilisées pour réduire la résolution des cartes de caractéristiques mais aussi augmenter plus rapidemment le champ réceptif. L'opération la plus utilisée est le *max pooling* qui va conserver uniquement la valeur maximale sur la région choisie au préalable (voir figure 1.3b). Cette opération permet aussi d'être invariants aux petites translations. Dans nos travaux, nous avons utilisé plus particulièrement un réseau nommé *Resnet* [HZRS15] que nous décrivons plus en détail ci-dessous.

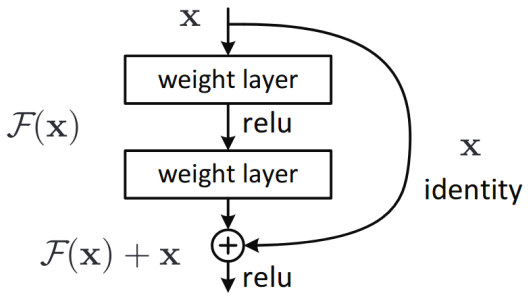


FIGURE 1.4 – Schéma d'une connexion résiduelle (repris de [HZRS15]). Le groupe est composé de 2 couches et l'entrée est additionnée à la sortie.

Réseau resnet La particularité des réseaux Resnet est l'utilisation de connections résiduelles. La connection résiduelle consisite à rajouter à la sortie d'un groupe de couches l'entrée (voir figure 1.4). Cette opération permet de préserver la norme du gradient lors de la rétropropagation et évite donc les problèmes de gradient évanescent. Il est ainsi plus facile d'apprendre des réseaux très profonds. La figure 1.5 présente l'architecture du modèle Resnet-18 qui est l'un des plus petits modèles de

la famille. Nous serons amené à l'utiliser par la suite.

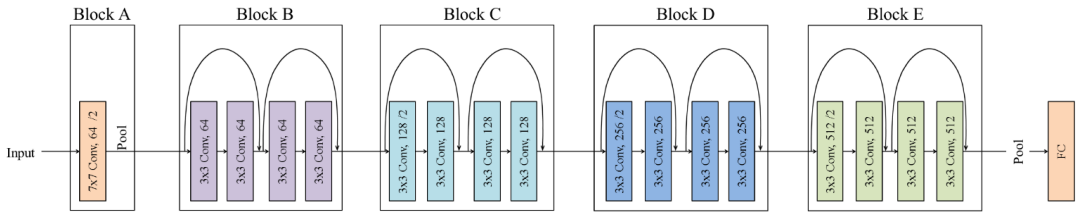


FIGURE 1.5 – Architecture d'un Resnet18 (repris de [GM19]). Il est composé d'une convolution suivie de 4 groupes de 2 blocks résiduels. A l'entrée de chaque groupe, la dimension spatiale est réduite par 2 dans les 2 dimensions.

Réseaux récurrents

Les réseaux récurrents (RNN) sont utilisés pour traiter des séquences (notées $x_1, x_2, \ldots$). Un réseau récurrent partage ses paramètres dans le temps, c'est à dire qu'il applique la même fonction à chaque x_t. Cependant, grâce à une cellule cachée notée h_t, celui-ci a une information sur le passé, c'est à dire les calculs effectués sur $x_1, \ldots, x_t$. Notons s_t sa sortie, un réseau récurrent basique a la formulation suivante :

$$\begin{aligned} h_t &= \tanh(\mathrm{W} h_{t-1} + \mathrm{U} x_t + b) \\ s_t &= \mathrm{W}_c h_t + b_c \,, \end{aligned} \tag{1.5}$$

avec $\mathrm{W}, \mathrm{U}, \mathrm{W}_c$ des poids appris et b, b_c des biais. Les réseaux *Long Short Term Memory* (LSTM) sont une amélioration de ce type de réseaux. En effet, ils évitent le problème des gradients évanescents à l'aide d'une cellule mémoire c_t supplémentaire. Celle-ci est accessible par des portes nommées *input* i_t, *forget* f_t et *output* o_t. L'architecture est présentée figure 1.6.

D'autres couches plus spécifiques à la reconnaissance d'actions comme les convolutions 3D, les convolutions (2+1)D ou encore les ConvLSTM sont présentés dans le chapitre 3.

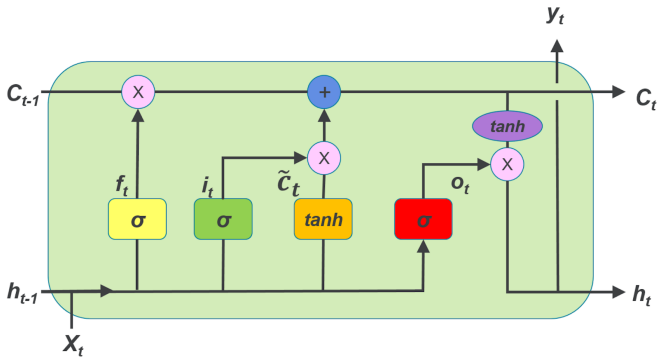


FIGURE 1.6 – Architecture d'un réseau récurrent LSTM.

1.2 Pré-apprentissage supervisé

Les méthodes décrites précédemment obtiennent de meilleures performances que celles basées sur les descripteurs faits à la main mais nécessitent un nombre de données annotées d'apprentissage important. En effet, pour un réseau profond, le nombre de paramètres à estimer est grand, ce qui favorise le sur-apprentissage. Il est alors impossible d'apprendre ce type de réseaux sur une petite base de données $\mathscr{D}_p$.

En faisant l'hypothèse que les représentations intermédiaires $h^{(i)}$ sont assez génériques et non spécifiques à une tâche particulière, il est possible d'apprendre les représentations intermédiaires sur une autre base de données plus importante $\mathscr{D}_g$ et de les réutiliser pour mieux apprendre sur la petite base de données $\mathscr{D}_p$. C'est ce qu'on appelle le pré-entraînement ou encore l'apprentissage de représentations. Il consiste à utiliser cette représentation (typiquement la sortie de l'avant dernière couche $h^{(n)}$) comme descripteur et à apprendre le classifieur à partir de celle-ci (un classifieur linéaire par exemple). Ce principe est illustré par la figure 1.7. Pour des meilleurs résultats, on peut utiliser les poids appris comme initialisation et continuer l'apprentissage (*finetuning*).

Notons $\theta_r = \{W^{(1)}, b^{(1)}, \ldots, W^{(n)}, b^{(n)}\}$ les paramètres du réseau exceptés ceux de la couche de classification et θ_r^* leur valeur optimale estimée sur la grande base de données. Les poids du réseau $\theta_r^n(0)$ sont initialisés par les valeurs θ_r^*. Les poids de la couche de classification W_{class} et b_{class} sont initialisés aléatoirement. En classification linéaire, la descente de gradients est effectuée uniquement sur les poids de la couche de classification. Autrement dit, $\theta_r^n(t) = \theta_r^n(0)$. Elle est effectuée sur tous les

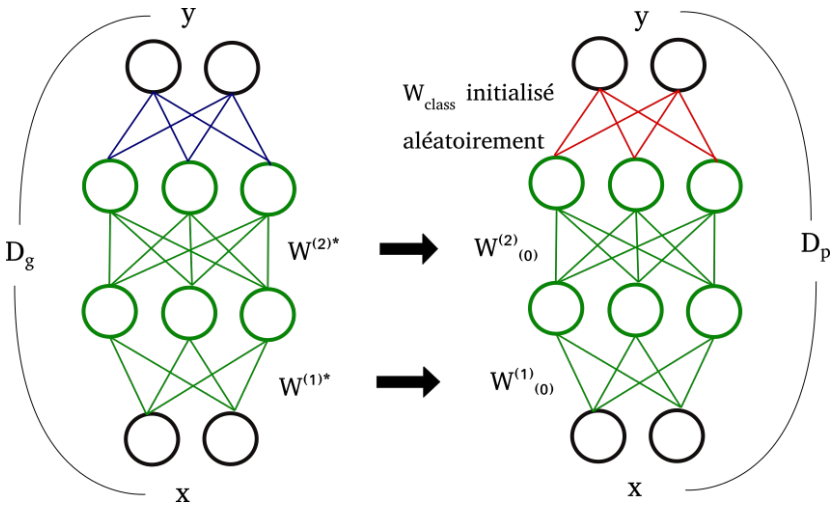


FIGURE 1.7 – Schéma représentant le principe de pré-apprentissage supervisé de réseaux de neurones.

Base de données	VOC 2007	*Flowers*	*Aircraft*
Pas d'initialisation	69.4	92.5	84.8
Evaluation linéaire	83.8	94.2	63.8
Finetuning	**86.5**	**98.0**	**88.0**

TABLEAU 1.1 – Comparaison des scores de reconnaissance sur différentes bases de données d'images entre sans initialisation, classification linéaire et *finetuning*. Le pré-apprentissage est effectué sur *Imagenet*. Les résultats sont repris de [CKNH20].

poids en ce qui concerne le *finetuning*.

Cette technique permet d'améliorer les performances sur de petites bases de données. En vision par exemple, dans la plupart des applications, les réseaux convolutionnels sont pré-entraînés sur la base de données *Imagenet* [RDS+14] avant d'être utilisés sur d'autres bases de données. Le tableau 1.1 montre que les résultats en *finetuning* sont toujours meilleurs que sans initialisation et en classification linéaire.

1.3 Pré-apprentissage non-supervisé

1.3.1 Principe général

Le pré-apprentissage supervisé a cependant l'inconvénient de nécessiter une base de données annotées $\mathscr{D}_g$ très importante sur laquelle effectuer le pré-apprentissage. L'annotation de ces larges bases de données labélisées doit se faire manuellement ce qui est très long et très coûteux. La plupart du temps du *crowd-sourcing* est utilisé permettant à des milliers d'annotateurs de contribuer. Les annotations doivent être vérifiées par plusieurs personnes afin de minimiser le bruit dans celles-ci. La création de ces bases de données n'est donc pas extensible de manière efficace. D'autres méthodes, proposent d'utiliser les tags des images sur Internet pour les transformer en vérités terrain. L'inconvénient est le bruit important dans les données.

L'apprentissage non supervisé de représentations a pour but d'apprendre les poids d'un réseau sans utiliser de données annotées. Cela permet notamment d'utiliser les larges quantités de données non annotées accessibles de manière presque illimitée. Pour cela, une tâche prétexte censée extraire de l'information haut niveau utile pour la classification est utilisée, celle-ci ne nécessitant aucune annotation. Considérons $\mathscr{U} = \{x_i, i = 1, \ldots, \mathrm{N}\}$ une base de données non-supervisée et l'objectif suivant

$$\min_{\theta} \sum_{j=1}^{\mathrm{N}} \mathrm{L}_{uns}\big(f_\theta(x_j), x_j\big), \tag{1.6}$$

où L_{uns} désigne un coût non supervisé comme le coût de reconstruction.

On estime de la même manière les paramètres optimaux θ^* qui minimisent la fonction de coût. Les poids du réseau pour la classification sur la petite base de données sont initialisés de la même manière que dans la partie précédente.

Notons la différence avec l'apprentissage semi-supervisé qui peut être considéré comme une formulation multitâche (classification sur les données annotées et une autre tâche sur les données non-annotées) avec un objectif qui peut s'écrire

$$\min_{\theta} \sum_{(x_i, y_i) \in \mathscr{D}_p} l\big(y_i, f_\theta(x_i)\big) + \sum_{x_j \in \mathscr{U}} \mathrm{L}_{uns}\big(f_\theta(x_j), x_j\big). \tag{1.7}$$

L'avantage de ce type de formulation est qu'il est possible d'utiliser l'information des données supervisées sur les données non-supervisées par l'utilisation de pseudo-labels ou encore la propagation de labels. Un seul pré-apprentissage non-

supervisé peut quant à lui être utilisé pour différentes tâches et bases de données, en n'ayant juste qu'à effectuer les classifications linéaires ou *finetuning*.

1.3.2 Historique des méthodes de pré-apprentissage non-supervisé

Les débuts du pré-apprentissage non-supervisé avec des approches couches par couches sont tout d'abord présentées, puis les méthodes sur les données discrètes comme le texte et enfin les méthodes génératives.

Premières méthodes couches par couches

Dans les premières applications de reconnaissance d'images utilisant les réseaux de neurone sur la base MNIST [LC10] par exemple, les apprentissages d'architectures profondes par rétropropagation n'étaient pas très performants. Différents articles comme [EBC+10] ou [BLPL06] montrent que l'on obtient de meilleurs résultats à l'aide d'un pré-entraînement non supervisé couche par couche. Typiquement, la couche supérieure capture les principaux facteurs de la couche courante. Les couches sont apprises les unes après les autres en partant de la couche la plus basse. [EBC+10] et [EMB+09] proposent différentes hypothèses sur l'utilité du pré-entraînement. La première est que le pré-apprentissage introduit une distribution à priori sur les paramètres. Une autre hypothèse est que le pré-apprentissage agit comme une régularisation qui réduit l'espace des paramètres. Finalement, l'apprentissage des principales variations de P(X) peut aussi se retrouver dans la distribution de P(Y|X). Les principaux modèles étudiés sont les *Restricted Boltzmann Machines* (RBM) [SMH07] ou les modèles auto-encodeurs (ceux-ci sont présentés par la suite avec les méthodes génératives). Les RBM sont un modèle graphique unidirectionnel définie par une fonction d'énergie J modèlisant une certaine forme de compatibilité entre une entrée x et des variables latentes notées h (celles-ci représentent la couche supérieure du réseau). Cette fonction d'énergie permet de représenter la distribution jointe entre l'entrée x et les variables latentes h.

$$\mathrm{P}(x,h) = \frac{1}{\mathrm{Z}} \exp^{-\mathrm{J}(x,h)}, \tag{1.8}$$

avec Z une constante de normalisation. Dans [Ben12], les auteurs indiquent que bien d'autres méthodes non-supervisées comme l'analyse en composantes princi-

Relation	Exemple 1	Exemple 2	Exemple 3
France-Paris	Italy-Rome	Japan-Tokyo	Florida-Tallahassee
Sarkozy-France	Berlusconi-Italy	Merkel-Germany	Koisumi-Japan

TABLEAU 1.2 – Exemples de relations haut niveau obtenues à partir des plongements lexicaux. La représentation du mot de droite est translatée par la relation et le mot sélectionné est le plus proche trouvé.

pales (PCA) peuvent être utilisées. Finalement, leurs expériences montrent que le pré-entraînement permet de mieux généraliser et d'être plus robuste à l'aléatoire. Cependant, aux vues des avancées sur l'apprentissage supervisé des réseaux de neurones, ces méthodes ne sont plus utilisées.

Historique des méthodes sur les données discrètes

Les premiers vrais succès de l'apprentissage de représentation non-supervisé ont eu lieu dans le domaine du *Natural Language Processing* (NLP). La première application est *word2vec* qui permet de trouver des représentations de mots intéressantes. Ces méthodes sont basées sur les co-occurences de mot et l'utilisation des mots de contexte. Considérons une séquence de mots $\mathrm{W} = w_1, \ldots, w_\mathrm{T}$, *Skip-Gram* [MCCD13] propose de prédire à partir d'un mot les mots aux alentours. *CBOW* [MCCD13] quant à lui propose de prédire un mot sachant son contexte. Dans les deux cas, l'objectif à maximiser est la log probabilité qui est détaillée dans les formules ci-dessous, à gauche pour *Skip-Gram* et à droite pour *CBOW* :

$$\sum_{t=1}^{\mathrm{T}} \sum_{j=-c}^{c} \log p(w_{t+j}|w_t), \qquad \sum_{t=1}^{\mathrm{T}} \log p(w_t|\sum_{j=-c}^{c} w_{t+j}). \tag{1.9}$$

Ainsi, les mots qui ont un contexte similaire ont une représentation similaire. Ces méthodes donnent des représentations de mots avec un haut niveau sémantique comme le montre les exemples de relations entre les représentations présentés table 1.2. L'étape suivante est l'apprentissage de représentations de phrases ou de séquences. GPT [RN18] propose d'utiliser la tâche de génération de texte et modélise la probabilité du mot suivant connaissant les précédents (voir schema 1.8)

$$\mathrm{L}(\mathrm{W}) = \sum_t \log p(w_t|w_{t-k}, \ldots, w_{t-1}). \tag{1.10}$$

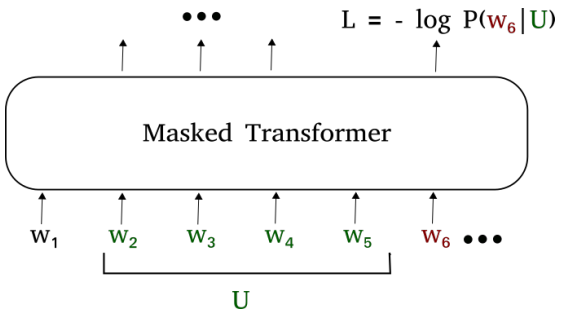


FIGURE 1.8 – Schéma de la méthode de pré-apprentissage GPT. U est l'ensemble des mots précédents utilisés pour prédire le mot courant (w_6 ici). *Masked transformer* est le réseau de neurone à apprendre.

Méthodes génératives

Les méthodes d'apprentissage de représentations se basent sur la notion de variables latentes et de variétés (*manifold*). L'hypothèse est que les images peuvent être générées à partir de variables latentes provenant d'un espace de petite dimension : $x = g(z)$ avec $z \in \mathbb{R}^d$ et d petit. L'objectif est alors de déterminer ces variables latentes qui sont une représentation intéressante des données. Elles contiennent en effet une information compressée peu bruitée et les notions de distance sont plus significatives sémantiquement dans cet espace. Nous décrivons ci-dessous différentes méthodes qui se basent sur ce principe, notamment les auto-encodeurs variationnels (VAE) et les réseaux génératifs adverses (GAN).

Auto-encodeurs et variantes Le but d'un modèle auto-encodeur est de reconstruire une donnée d'entrée x. Cette dernière est compressée par un encodeur E qui produit une représentation $z = \mathrm{E}(x)$. Le décodeur D permet la reconstruction de x à partir de z par :

$$r = \mathrm{D}(z) = \mathrm{D}\big(\mathrm{E}(x)\big). \tag{1.11}$$

La dimension de z est bien plus faible que la dimension d'entré x afin d'effectuer un goulot d'étranglement qui limite la quantité d'information. Le code z peut ensuite être utilisé pour effectuer différentes tâches. L'apprentissage de l'auto-encodeur minimise la différence entre x et r soit la fonction de coût $l(x, \mathrm{D}(\mathrm{E}(x)))$. L'erreur quadratique moyenne est le plus souvent utilisée :

$$l(a,b) = \frac{1}{\mathrm{N}} \sum (a_i - b_i)^2 . \tag{1.12}$$

Pour éviter que le réseau effectue uniquement une opération de « copier-coller », différentes régularisations sur le code ont été proposées. Les auto-encodeurs parcimonieux (*sparse*) par exemple proposent que les éléments de z soient nuls pour la plupart en ajoutant une régularisation L1. D'autres méthodes proposent de modifier l'entrée. Les auto-encodeurs débruitants ajoutent un bruit à x : $\tilde{x} + \epsilon$. La fonction de coût est donc $l(x, \mathrm{D}(\mathrm{E}(\tilde{x}))$. La méthode proposée dans [ZIE16b] découpe l'exemple en 2 parties x_1 et x_2 et propose de prédire l'une à partir de l'autre : $l(x_1, \mathrm{D}(\mathrm{E}(x_2))) + l(x_2, \mathrm{D}(\mathrm{E}(x_1)))$.

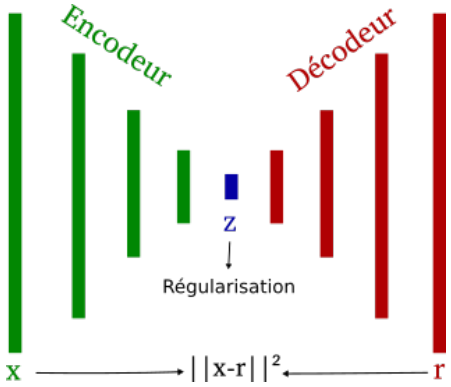


FIGURE 1.9 – Schéma d'un modèle auto-encodeur. L'erreur de reconstruction entre l'entrée x et la sortie r : $||x-r||^2$ est minimisée. Une régularisation est appliquée sur le code z en sortie de l'encodeur.

Dans [KW13], une adaptation des modèles auto-encodeurs qui permet de générer des données est proposée. Pour cela, une approche variationnelle pour maximiser la vraisemblance est utilisée. Ce modèle est appelé auto-encodeur variationnel (VAE) :

$$p_\theta(x) = \int p(z) p_\theta(x|z) dz. \tag{1.13}$$

Dans l'équation (1.13), l'*a priori* $p(z)$ est fixé, par exemple à une loi normale $\mathcal{N}(0, \mathrm{I})$. Estimer $p_\theta(x)$ tel que $\sum_{i=1}^{\mathrm{N}} p_\theta(x|z_i)$ avec $z_i \sim \mathcal{N}(0,1)$ est inefficace. En effet, il serait beaucoup plus intéressant de tirer les z selon la probabilité $p(z|x)$ qui est cependant aussi difficile à calculer que $p_\theta(x)$. Celle-ci est estimée par $q_\phi(z|x)$. Il est possible d'effectuer tout cela à partir de la formule variationnelle suivante :

$$\log p_\theta(\mathrm{X}) - \mathscr{D}_{\mathrm{KL}}\left[q_\phi(z|\mathrm{X})||p_\theta(z|\mathrm{X})\right] = \mathbb{E}_{z\sim \mathrm{Q}}\left[\log p_\theta(\mathrm{X}|z)\right] - \mathscr{D}_{\mathrm{KL}}\left[q_\phi(z|\mathrm{X})||\mathrm{P}(z)\right], \tag{1.14}$$

avec $\mathscr{D}_{\mathrm{KL}}$ la divergence de Kullback Leibler. L'expression (1.14) est une borne in-

férieure de la log probabilité. Sa maximisation permet aussi de réduire la distance entre $q_\phi(z|x)$ et $p_\theta(z|x)$ ce qui est souhaité. Il est possible de la maximiser par descente de gradient. De manière générale, on paramétrise les probabilités $p_\theta(x|z)$ et $q_\phi(z|x)$ de la façon suivante :

$$\mu, \log\sigma = \mathrm{E}_\phi(x) \text{ et } q_\phi(z|x) = \mathcal{N}(\mu, \sigma), \quad \mu_o = \mathrm{D}_\theta(z) \text{ et } p_\theta(x|z) = \mathcal{N}(\mu_o, \mathrm{I}). \tag{1.15}$$

La représentation z est définie par $z = \mu + \sigma\epsilon$ avec $\epsilon \sim \mathcal{N}(0, \mathrm{I})$ et $\sigma > 0$.

On peut alors considérer E et D comme des encodeurs et décodeurs probabilistes. La fonction de coût de reconstruction est $-\log p_\theta(\mathrm{X}|z)$ et la régularisation sur la représentation est $\mathscr{D}_{\mathrm{KL}}\left[q_\phi(z|\mathrm{X})||\mathrm{P}(z)\right]$. Lors de l'inférence, il est possible de tirer z selon l' *a priori* et de générer de nouveaux exemples similaires aux données grâce au décodeur.

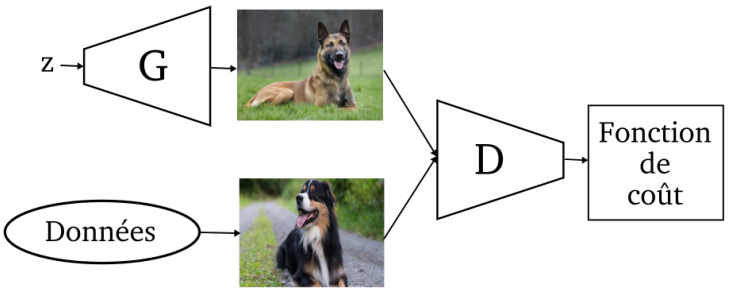


FIGURE 1.10 – Schéma général d'un *GAN*. Le réseau discriminateur D doit différencier les exemples provenant des données de ceux créés par le générateur G.

Méthodes génératives adverses (*GAN*) Les *GAN* [GPAM+14] (illustrés figure 1.10) sont composés de deux sous réseaux : un générateur G et un discriminateur D. Le modèle générateur G a pour but de générer des données similaires aux $\{x_i\}$ à partir d'un bruit $z \sim \mathrm{P}_z = \mathcal{N}(0, 1)$. Le réseau discriminateur D a pour but de différencier les exemples générés de ceux provenant des données. Le problème d'optimisation associé à l'apprentissage d'un GAN est formulée de la manière suivante :

$$\min_{\mathrm{G}} \max_{\mathrm{D}} \mathbb{E}_{x\sim p_{data}} \log\big(\mathrm{D}(x)\big) + \mathbb{E}_z \log\Big(1 - \mathrm{D}\big(\mathrm{G}(z)\big)\Big). \tag{1.16}$$

Les exemples générés suivent la distribution $\mathrm{P}_\mathrm{G} = \mathrm{G}\#\mathrm{P}_z$ [1]. Ainsi, le discriminateur apprend à différencier les images générées des autres et le générateur à travers

1. # indique ici l'opérateur *push forward*. P_G est donc la loi de probabilité des éléments $\mathrm{G}(z)$.

cela à améliorer sa génération. De manière plus formelle, le discriminateur optimal obtenu avec un générateur fixé est le suivant :

$$\mathrm{D}^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_{\mathrm{G}}(x)}. \tag{1.17}$$

Le discriminateur apprend donc un ratio de densités qui va permettre par la suite au générateur de mieux estimer p_{data}. En effet, en remplaçant le discriminateur par sa valeur optimale, il est possible de montrer que le générateur doit minimiser la *Jensen Schannon Divergence* (*JSD*) :

$$\mathrm{JSD}(p_{data}; p_{\mathrm{G}}) = \frac{1}{2}\left(\mathscr{D}_{\mathrm{KL}}\left(p_{data}; \frac{p_{data} + p_{\mathrm{G}}}{2}\right) + \mathscr{D}_{\mathrm{KL}}\left(p_{\mathrm{G}}; \frac{p_{data} + p_{\mathrm{G}}}{2}\right)\right). \tag{1.18}$$

De manière plus générale, le discriminateur estime une distance entre la distribution réelle et celle issue du générateur. Le générateur minimise cette distance pour que ses exemples se rapprochent de la distribution réelle. La distance choisie doit pouvoir être estimée uniquement à partir d'exemples des distributions, les modèles génératifs implicites ne donnant pas accès à autre chose. Les distances les plus utilisées sont les f-Divergences et les *Integral Probability Metrics* (*IPM*). Les f-divergences D_f se définissent par :

$$\mathrm{D}_f\big(p_d || p_\theta(x)\big) = \int p_\theta(x) f\left(\frac{p_d(x)}{p_{\mathrm{G}}(x)}\right) = \sup_{\mathrm{D}} \mathbb{E}_{x\sim \mathrm{P}_d}\big[\mathrm{D}(x)\big] - \mathbb{E}_{x\sim \mathrm{P}_{\mathrm{G}}}\Big[f^*\big(\mathrm{D}(x)\big)\Big], \tag{1.19}$$

avec f^* la conjuguée de Fenchel de la fonction f. Cette dernière formulation permet d'obtenir un problème de maximisation (en fonction de D pour estimer la distance D_f), minimisation (en fonction de G d'où proviennent les x) alternée. Les *IPM* se formulent de la manière suivante :

$$d_{\mathrm{F}}(p_d, p_\theta) = \sup_{f\in \mathrm{F}} \mathbb{E}_{x\sim \mathrm{P}_d}\big[\mathrm{D}(x)\big] - \mathbb{E}_{x\sim \mathrm{P}_\theta}\big[\mathrm{D}(x)\big]. \tag{1.20}$$

Ces distances se différencient en fonction de l'ensemble F utilisé. Parmi les *IPM*, la distance *Earth Mover* (*EM*) a été utilisée avec succès car elle permet un entraînement plus stable qu'avec les autres méthodes. Elle a en effet des propriétés intéressantes du fait qu'elle prenne en compte la distance dans l'espace des données. Ainsi, par exemple, 2 lois de probabilité ayant des supports disjoints ont une distance EM finie alors que ce n'est pas le cas pour la *JSD*.

Malheureusement, cette méthode ne permet pas d'accéder aux variables latentes z à partir d'une donnée x. L'article [DKD16] propose d'ajouter un encodeur E pour cela et d'utiliser le discriminateur sur le couple de variables x et z. L'objectif à optimiser est donc :

$$\min_{\mathrm{G,E}} \max_{\mathrm{D}} \mathbb{E}_{x\sim p_{data}} \log\Big[\mathrm{D}\big(x,\mathrm{E}(x)\big)\Big] + \mathbb{E}_z \log\Big[1-\mathrm{D}\big(\mathrm{G}(z),z\big)\Big]. \tag{1.21}$$

D'autres méthodes proposent d'utiliser l'avant dernière couche du discriminateur comme représentation.

1.4 Conclusion

Les réseaux de neurones, par leur structure, permettent d'apprendre des représentations intermédiaires intéressantes mais nécessitent un nombre important de données annotées. Afin d'utiliser peu de données annotées, un pré-apprentissage non-supervisé peut être utilisé. Les méthodes génératives sont à la base de l'apprentissage non-supervisé Nous decrivont dans la prochaine partie comment elles sont utilisées sur les images et vidéos ainsi que d'autres principes d'apprentissage non-supervisé.

Chapitre 2

Méthodes d'apprentissage non-supervisé de représentations

Sommaire

2.1 Applications des méthodes génératives 20
2.1.1 Application aux images 20
2.1.2 Application aux vidéos 21
2.2 Méthodes auto-supervisées 25
2.2.1 Application aux images 25
2.2.2 Application aux vidéos 27
2.3 Méthodes basées sur les pseudo-labels 28
2.4 Méthodes basées sur l'information mutuelle 30
2.4.1 Aspects théoriques 30
2.4.2 Applications pratiques 34
2.4.3 Expériences sur les méthodes contrastives 39
2.5 Conclusion 43

Nous décrivons dans cette partie les différentes méthodes d'apprentissage non-supervisé et détaillons comment elles sont appliquées aux images et aux vidéos. Nous exposons tout d'abord les applications des méthodes génératives décrites dans le chapitre précédent. Nous détaillons ensuite les autres types de méthodes pour l'apprentissage non-supervisé de représentations. Les méthodes auto-supervisées définissent une tâche prétexte que le réseau doit résoudre. D'autres

méthodes prédisent des pseudo-labels, générés par *clustering* par exemple. Finalement, les méthodes basées sur l'information mutuelle ont pour but de maximiser cette dernière entre différentes parties d'une même donnée. Les différents types de méthodes sont présentées dans la figure 2.1 et sont détaillées par la suite.

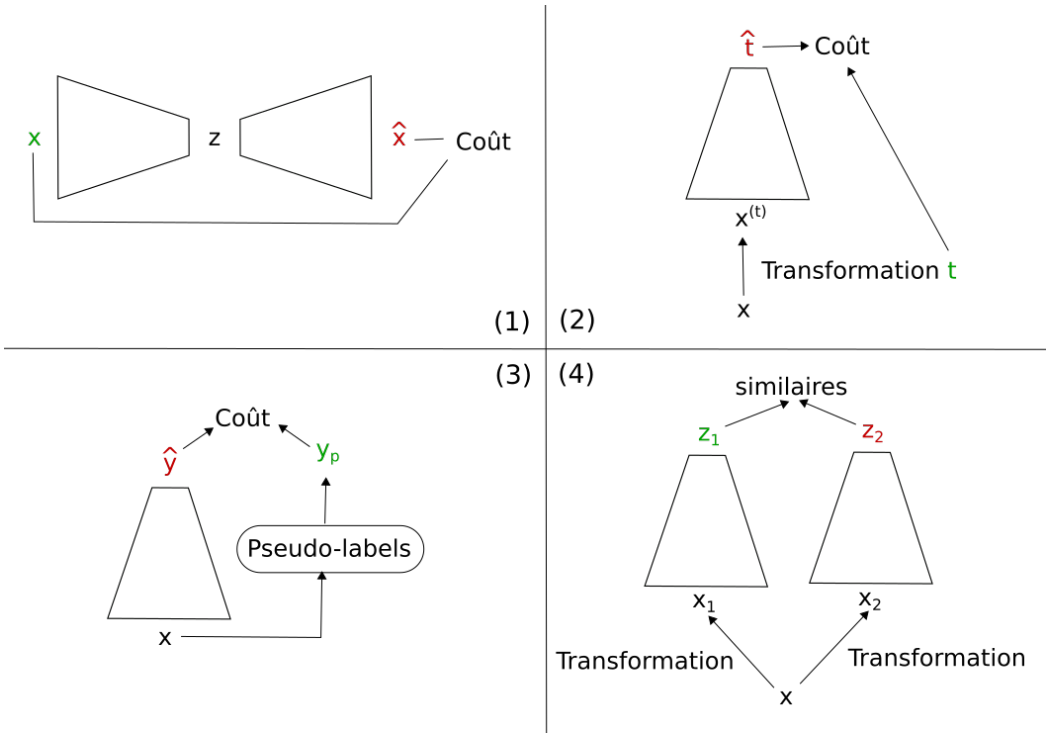


FIGURE 2.1 – Schéma général des différentes familles de méthodes d'apprentissage non-supervisé. Les prédictions sont notées en rouge et les valeurs à prédire en vert. (1) : méthodes génératives, (2) : méthodes auto-supervisées sur les transformations, (3) : méthodes basées sur les pseudo-labels, (4) : méthodes contrastives basées sur l'information mutuelle.

2.1 Applications des méthodes génératives

L'application des méthodes génératives (le plus souvent VAE ou GAN) aux images et aux vidéos est détaillée dans cette partie.

2.1.1 Application aux images

Un modèle auto-encodeur est appris dans [ZMGL15] afin de pré-entraîner l'encodeur pour des tâches de classification. L'encodeur donne à la couche correspon-

dante du décodeur l'information de localisation du pooling de manière à séparer l'information de position et l'information de contenu.

Dans [DS19], les auteurs combinent la technique de [DKD16] pour accéder aux variables latentes et aux performances impressionnantes de génération proposées dans [BDS18] afin obtenir des représentations d'images intéressantes. Il ajoute par ailleurs un discriminateur sur les x et un autre sur les z au discriminateur sur la paire (x, z) utilisée précédemment.

2.1.2 Application aux vidéos

Les méthodes génératives sur les séquences vidéo se séparent en deux applications, la génération de vidéo et la prédiction du futur de la vidéo. La génération de vidéo est très complexe et requiert beaucoup de données pour l'apprentissage. La plupart des articles traitent donc de la prédiction des séquences futures.

Génération de vidéo

Un GAN est utilisé dans [VPT16] pour générer des vidéos. Le générateur est composé de 2 voies, une pour les parties statiques (génère une image) et une pour les parties dynamiques (génère une vidéo). Les 2 parties sont combinées à l'aide d'un masque appris. Le discriminateur prend la vidéo générée afin de classifier si elle est réelle ou non. Ce discriminateur est un réseau composé de 5 convolutions 3D. Il est par la suite utilisé pour évaluer la méthode en apprentissage de représentations. Même si ce pré-entraînement améliore les performances de classification sur un apprentissage de zéro, celles-ci restent très faibles comparé à l'état de l'art.

Prédiction des séquences (*frames*) futures

Une tâche de pré-entraînement intéressante est la prédiction du futur de la vidéo à partir des frames précédents. Le réseau doit ainsi analyser les mouvements présents dans la vidéo et comprendre l'action en cours afin de générer la suite. La tâche est en quelques sorte similaire à la méthode décrite dans [ZIE16b] (voir partie 1.3.2) mais le découpage entre entrée et sortie se fait au niveau temporel et non au niveau des canaux des images. La prédiction du futur pose différents problèmes comme la multitude des futurs possibles qui peuvent engendrer des prédictions floues. Il est aussi nécessaire de prendre en compte le mouvement et l'information

présente dans les images précédentes. Nous voyons dans les paragraphes suivants comment ces problèmes sont pris en compte par les méthodes de l'état de l'art.

Prédiction de multiples futurs Un *VAE* est proposé par [BFE+17] pour pouvoir prédire plusieurs futurs. Un encodeur transforme toutes les images de la vidéo en une représentation stochastique $z \sim q_\phi(z|\mathrm{V}_{[0:\mathrm{T}]})$ avec V la vidéo. En notant μ et σ les sorties du réseau encodeur, la distribution est définie de la manière suivante :

$$q_\phi(z|\mathrm{I}_{[0:\mathrm{T}]}) = \mathcal{N}\big(\mu(\mathrm{V}_{[0:\mathrm{T}]}), \sigma(\mathrm{V}_{[0:\mathrm{T}]})\big).$$

Un réseau récurrent prédit l'image suivante en fonction des images précédentes et de la représentation z. La fonction de coût utilisée est similaire à celle utilisée dans les *VAE* de l'équation (1.14) mais considère une séquence :

$$\mathrm{L}(x) = -\mathrm{E}_{q_\phi}(z|\mathrm{V}_{[0:\mathrm{T}]})\Big[log\, p_\theta\big(\mathrm{V}_{[t:\mathrm{T}]}|\mathrm{V}_{[0:t-1]}, z\big)\Big] + \mathrm{D}_{\mathrm{KL}}\big(q_\phi(z|\mathrm{V}_{[0:\mathrm{T}]})||p(z)\big).$$

Par la suite, tirer aléatoirement à partir de la distribution $p(z)$ plusieurs z permet d'obtenir différents futurs possibles de la vidéo.

Prédictions floues L'utilisation de modèles auto-encodeurs pour la prédiction du futur de la vidéo génère des représentations floues car ils génèrent la moyenne de ces différents futurs possibles. Une fonction de coût adverse est proposée dans [MCL15] pour obtenir des images plus réalistes et donc moins floues. La fonction G essaie de prédire la fin de la vidéo $\mathrm{V}_{[t,\mathrm{T}]}$ à partir du début $\mathrm{V}_{[0:t]}$ le début de la vidéo. La fonction de coût est similaire à celle d'un *GAN* (voir equation (1.16)) mais est appliquée à une séquence d'images :

$$\mathrm{L}_{adv} = -log\big(\mathrm{D}(\mathrm{V}_{[0:t]}, \mathrm{V}_{[t,\mathrm{T}]})\big) + log\big(1 - \mathrm{D}(\mathrm{V}_{[0:t]}, \mathrm{G}(\mathrm{V}_{[0:t]}))\big).$$

La méthode [KHPG17] est similaire mais basée sur un Wasserstein GAN. Les structures statiques et dynamiques (faibles et hautes fréquences) sont gérées de manière séparée dans [XNL+18] de sorte à produire des prédictions de meilleure qualité. Le traitement des hautes fréquences permet de raffiner les prédictions floues issues des faibles fréquences.

Méthodes de transformation Deux images successives dans une vidéo sont très semblables. Il est donc intéressant de générer les images futures en transformant les précédentes. Cela permet au réseau de ne pas stocker l'information visuelle de bas niveau comme les textures et donc aussi de mieux généraliser aux objets non vus. Différentes transformations pour passer de l'image courante à la suivante sont présentées dans [FGL16]. A partir des images précédentes, le réseau détermine un noyau $m_{x,y}$ de largeur K qui est appliqué à l'image courante I_t pour obtenir la suivante I_{t+1}, de la manière suivante :

$$\mathrm{V}_t(x,y) = \sum_{k\in[-\mathrm{K},\mathrm{K}]} \sum_{l\in[-\mathrm{K},\mathrm{K}]} m_{x,y}(k,l)\mathrm{V}_{t-1}(x-k,y-l)\,.$$

La méthode proposée par Vondrick et al. [VT17] utilise la transformation décrite précédemment intégrée dans un modèle génératif adverse. Le générateur s'écrit sous la forme $f(x,\omega) = \gamma(g(x;\omega),x)$ avec γ la transformation, x l'image précédente et g un réseau donnant les paramètres de la transformation. Il n'applique pas de récurrence pour éviter l'accumulation d'erreurs (les images sont générées en une seule fois). Le discriminateur est un CNN spatio-temporel qui prend en entrée la concaténation de x et $f(x,\omega)$ de manière à ce que les frames générées soient en adéquation avec l'entrée x. L'article [PHC15] propose d'estimer le flot optique à l'aide d'un réseau récurrent pour transformer l'image courante en l'image suivante. La méthode [RLS+18] combine un déplacement par le vecteur de flots optiques avec une transformation de noyau M

$$\mathrm{V}_{t+1}(x,y) = \mathrm{M}(x,y) * \mathrm{V}_t(x+u,y+v)\,,$$

où $*$ indique l'opération de convolution.

Le flot optique est calculé par un réseau *flownet2* [IMS+17] et extrapolé (calculé entre V_{t-1} et V_t et transformé pour prédire V_{t+1} à partir de V_t). Une fonction de coût L1 et perceptuelle (à partir d'un VGG16) est utilisée. La fonction de coût perceptuelle compare les représentations de réseaux pré-entraînés au lieu de comparer directement au niveau des pixels.

Méthodes de séparation du contenu et du mouvement La séparation entre mouvement et contenu est intéressante car elle permet d'obtenir des représentations plus interprétables. [TLYK17] et [DB17] se placent dans l'espace latent des images où ils considèrent une partie fixe (le contenu) et une autre variable (la pose ou mou-

vement). En générant un contenu et une séquence de poses grâce à un réseau récurrent, ils sont capables de générer une vidéo.

Dans [TLYK17], un GAN composé du réseau récurrent générant la séquence de poses, un générateur d'images, un discriminateur au niveau de l'image et de la vidéo est utilisé. Le GAN apprend sans supervision à séparer le mouvement du contenu. Ce réseau permet aussi d'appliquer un mouvement différent à un certain contenu lors de la génération d'une vidéo.

Quand à [DB17], il utilise un auto-encodeur, avec une fonction de coût de reconstruction. Il ajoute un coût pour que la partie de contenu h_c soit fixe au cours du temps $h_c^t \approx h_c^{t+k}$ et une fonction adverse pour que les informations de pose ne contiennent pas d'informations constantes dans le temps.

Variantes de la prédiction des frames futures La prédiction dense de pixels ou flots optiques est complexe car il est nécessaire de garder en mémoire tous les détails. De plus, cela donne plus d'importance aux détails de bas niveau sémantique par rapport à l'information de haut niveau. Certains travaux choisissent donc des tâches plus simples afin d'avoir un entraînement plus facile tout en gardant une représentation intéressante. Une séquence de flots est prédite à l'aide d'un réseau récurrent dans [LPH+17]. Cependant, un *codebook* est utilisé pour réduire la dimension du signal à prédire. De plus, cela permet d'effectuer une tâche de classification, la régression avec une distance L_2 n'étant pas adaptée à des petites déformations. La méthode [ZZH+17] apprend un dictionnaire sur les images. Il utilise un réseau de séquence à séquence avec attention pour prédire l'élément du dictionnaire correspondant aux images suivantes et apprendre une représentation de la vidéo. L'article [VPT15] propose de prédire la représentation (issue d'un CNN pré-entraîné) des images suivantes à partir des images courantes. Il effectue une tâche de régression sur cette représentation. La représentation issue du CNN est de plus haut niveau que les pixels (information plus utile) mais n'a cependant pas d'information temporelle.

Les méthodes de prédiction des images futures a eu un intérêt important mais l'accent a plus été mis sur la qualité des prédictions que sur l'apprentissage de représentations intéressantes. L'amélioration des performances en reconnaissance d'actions reste limité.

2.2 Méthodes auto-supervisées

Les méthodes auto-supervisées (*self suppervised* en anglais) extraient une vérité terrain à partir d'un exemple et essaient de prédire celle-ci. Il s'agit par exemple de prédire l'angle d'une rotation appliquée à une image où plus généralement, d'appliquer une transformation à un exemple et essayer de prédire cette transformation. Si l'on note t la transformation et E un réseau encodeur, cela peut se formaliser à travers la minimisation du coût suivant :

$$\min_{\mathrm{E}} \sum_{(x,t)} l\Big(t, \mathrm{E}\big(t(x)\big)\Big).$$

Voyons plus en détails comment cela fonctionne dans le cas des images et des vidéos.

2.2.1 Application aux images

Les premières méthodes auto-supervisées sur les images utilisent l'information de position de patchs de l'image les uns par rapport aux autres. Par exemple, [DGE15] découpe l'image en une grille de 9 patchs. A partir du patch du milieu, le réseau doit prédire la position des autres patchs (8 positions possibles) (voir le cas (2) dans la figure 2.2). Pour éviter les solutions triviales, les images sont mises en noir et blanc et les patchs sont espacés. Dans [NF16], une tâche de *jigsaw puzzle* (voir (3) figure 2.2) est proposée. Le réseau apprend à retrouver la position de chaque pièce d'images. Pour cela, chaque patch est encodé par un réseau, la permutation utilisée sur les positions des patchs est prédite.

Une partie masquée de l'image est générée dans [PKD+16]. Pour cela, il utilise un réseau encodeur décodeur associé à des fonctions de coût L2 (sur la zone masquée) et adverse pour obtenir des générations plus réalistes.

Une autre méthode [ZIE16a] colorise des images en retrouvant les canaux a et b à partir du canal L dans la décomposition *Lab*. Les valeurs de couleur sont quantifiées et une tâche de classification est utilisée pour éviter les moyennages provoqués par la fonction de coût L2.

Dans [GSK18], les auteurs proposent de prédire l'angle de la rotation appliquée à une image comme tâche auto-supervisée (voir(1) figure 2.2). Une rotation d'angle 0, 90, 180 ou 270 est appliquée à l'image et le réseau effectue une classification à 4 choix. L'intérêt de cette méthode est que la rotation est une transformation globale

qu'il est difficile à estimer pour un CNN. Le réseau doit ainsi apprendre à reconnaître les différents éléments et leur orientation naturelle pour résoudre la tâche.

Finalement, [ZQWL19] propose d'encoder les transformations entre 2 images et de la prédire comme tâche auto-supervisée (voir (4) figure 2.2). L'idée est que le réseau doit extraire la structure des 2 images afin de les comparer pour retrouver la transformation. Un décodeur D est utilisé pour prédire la transformation et l'objectif s'écrit de la manière suivante :

$$\min_{\mathrm{E,D}} \mathbb{E}\, l\Big[t, \mathrm{D}\Big(\mathrm{E}(x), \mathrm{E}\big(t(x)\big)\Big)\Big].$$

Une amélioration est proposée par [QZCT19] en maximisant l'information mutuelle entre la transformation t et $\mathrm{E}(t(x))$ sachant $\mathrm{E}(x)$. Les résultats obtenus sont légèrement meilleurs en classification linéaire.

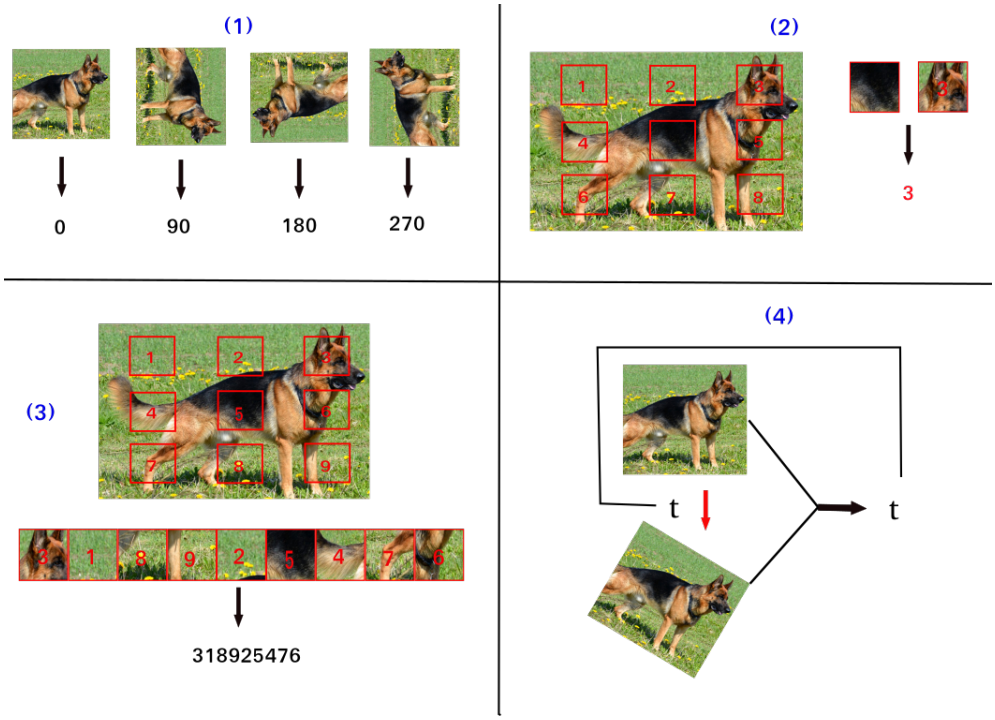


FIGURE 2.2 – Schéma regroupant les différentes méthodes d'apprentissage auto-supervisé pour les images.

Certaines méthodes utilisent l'information temporelle de vidéos pour apprendre des représentations sur les images. Par exemple, [PGD+16] utilise le flot optique pour déterminer les pixels qui ont un mouvement similaire et ainsi créer des

masques. La tâche auto-supervisée est de prédire ces masques. La méthode [JG15] impose que les représentations provenant de frames proches soient similaires. Pour cela, ils utilisent une distance L2 ainsi qu'une fonction de coût contrastive. Finalement, [WG15] traque un patch de l'image au cours de la vidéo et sélectionne les 2 extrémités comme une paire d'exemples. La fonction de coût triplet est utilisée pour rapprocher les représentations de cette paire et la distancier d'un exemple négatif choisi au hasard.

2.2.2 Application aux vidéos

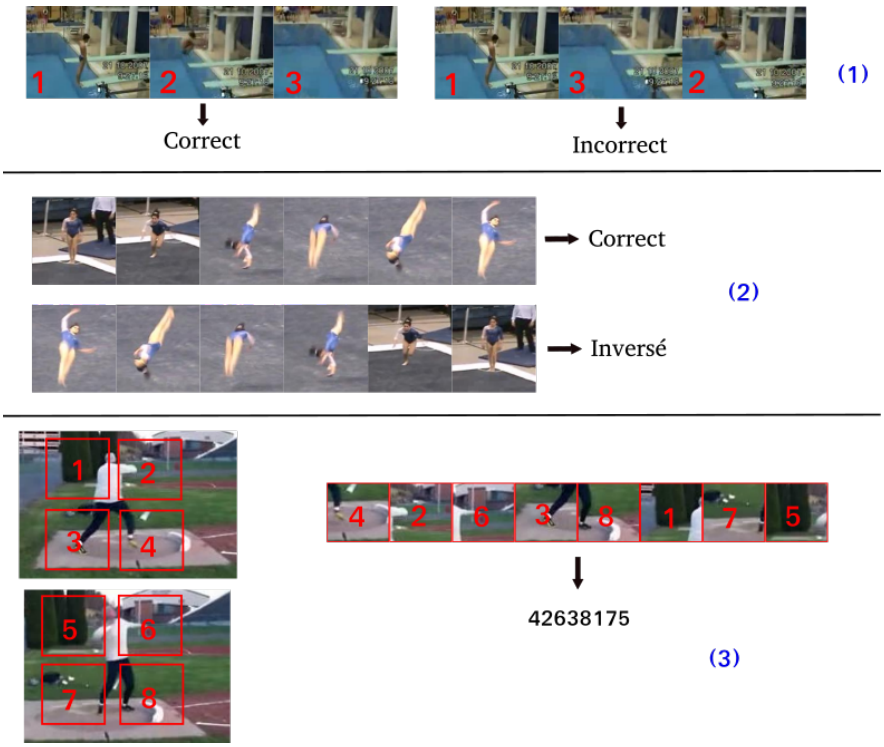


FIGURE 2.3 – Schéma regroupant les différentes méthodes d'apprentissage auto-supervisé pour les vidéos.

De nombreuses méthodes auto-supervisées sur les vidéos utilisent les mêmes principes vus précédemment mais appliqués au temps. Par exemple, [MZH16] propose un réseau qui permet de vérifier l'ordre temporel des frames. Trois images sont tirées au hasard dans la vidéo dans un ordre correct ou non et le réseau doit le déterminer en classifiant avec 0 ou 1 (voir (1) figure 2.3). Un réseau siamois tri-

plet [KZS15] est utilisé et les représentations sont concaténées avant la classification. La méthode précédente est améliorée par [FBGG16] en proposant un réseau à N+1 branches. Chaque branche encode une succession d'images. Parmi celles-ci, les images d'une est dans le mauvais ordre et le réseau doit déterminer cette branche. Cela permet d'encoder plusieurs images avec le même réseau ce qui est plus intéressant. La méthode décrite dans [LHSY17] prédit l'ordre des images de la vidéo. Pour cela, les représentations issues de chaque image sont utilisées pour obtenir des caractéristiques par paires d'image. Ces caractéristiques permettent ensuite de prédire la permutation des images utilisées. Cette méthode est étendue par [XXZ+19] en classifiant l'ordre de clips vidéos au lieu de simples images. Cela permet notamment d'enlever certaines ambiguïté entre les actions "ouvrir et fermer" par exemple.

Dans [AME18] les auteurs présentent une extension spatio-temporelle de la méthode de jigsaw puzzle. 4 patchs sont sélectionnés dans 3 frames de la vidéo soient 12 patchs au total. Le réseau doit prédire la permutation qui leur a été appliquée (voir (3) figure 2.3). Cette méthode est appliquée à des patchs cubiques (spatio-temporels) avec des réseaux 3D dans [KCK18].

La tâche auto-supervisée de [WLZF18] est la prédiction du sens temporel de la vidéo (sens naturel ou inversé) (voir (2) figure 2.3). L'inversement du sens de la vidéo la rend irréaliste et le réseau doit en comprendre la raison. Par exemple que les personnes marchent généralement vers l'avant ou encore les notions de physique de base comme la gravité (les objets tombent). La méthode montre un apport en reconnaissance d'actions à partir d'images et de flots optiques (voir chapitre 6). La méthode est donc similaire à RotNet [GSK18] sur les images. L'article [JT18] applique à cet effet la méthode RotNet à des clips vidéos en utilisant des réseaux 3D pour la prédiction de l'angle.

2.3 Méthodes basées sur les pseudo-labels

D'autres solutions se basent sur des méthodes d'association entre les exemples x_i et des pseudo-labels y_i. Le réseau a ensuite pour but de prédire ces pseudo-labels (voir figure 2.4) comme pour un apprentissage supervisé classique :

$$\min_{\theta,W} \frac{1}{N} \sum_{i=1}^{N} l\Big(g_W\big(f_\theta(x_i), y_i\big)\Big).$$

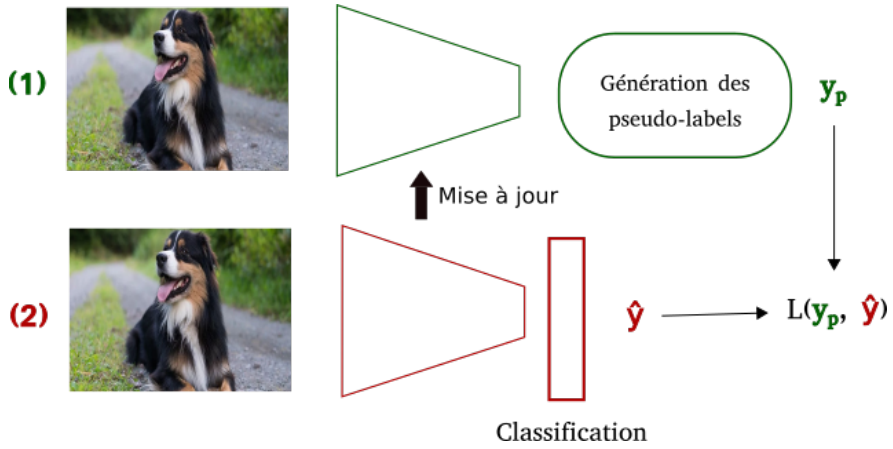


FIGURE 2.4 – Schéma général des méthodes basées sur les pseudo-labels. Les étapes 1 et 2 sont alternées. La génération des pseudo-labels dépend de la méthode, c'est par exemple un clustering avec k-moyennes pour la méthode Deep Cluster.

Bojanowski et Joulin [BJ17] proposent d'utiliser des vecteurs aléatoires fixes comme pseudo-labels. L'association des pseudo-labels aux exemples se fait par un problème d'assignation optimal pour minimiser les valeurs $l(g_W(f_\theta(x_i), y_i))$. Les étapes d'association et de prédiction sont alternées.

Deep Cluster [CBJD18] remarque que les représentations obtenues à partir d'un CNN initialisé aléatoirement sont bien meilleures qu'une prédiction aléatoire. Il propose d'utiliser un clustering sur ces représentations afin de produire les pseudo-labels. De même que dans la méthode précédente, les étapes de clustering et de prédiction sont alternées.

La méthode précédente est combinée avec des méthodes auto-supervisées présentées dans la section 2.2 dans [CBMJ19]. Pour celà, la méthode prédit un pseudo-label composé de l'angle de rotation appliqué à l'image et le numéro de cluster de l'image. Une fonction de coût hiérarchique est utilisée.

La méthode proposée dans [CMM+20] améliore celle de [CBJD18] en utilisant 2 augmentations de données différentes d'une même image dont on calcule une représentation z_1 et z_2. Le prototype (correspondant dans la précédente méthode au centroïde du cluster) le plus proche de chaque représentation leur est assigné. On les note ϕ_1 et ϕ_2. Une fonction de coût contrastive l est utilisée et les prototypes sont échangés :

$$L = l(z_1, \phi_2) + l(z_2, \phi_1)\,.$$

La méthode *Deep Cluster* est améliorée et adaptée pour la vidéo dans [AMT+19]. Elle utilise 2 modalités : la vidéo et le son. Des prédictions de modalités croisées (cross-modales) sont effectuées. Les pseudo-labels sur les vidéos proviennent d'un clustering des représentations de l'audio et vice-versa.

2.4 Méthodes basées sur l'information mutuelle

Les méthodes ci-dessous se basent sur la maximisation de l'information mutuelle entre les représentations de différentes parties des données ou entre entrées et sorties par exemple. Ces méthodes sont basées sur l'utilisation d'exemples négatifs introduits en premier dans la méthode *Noise Contrastive Estimation* (NCE) [GH10]. Nous abordons d'abord les aspects théoriques en 1ère partie suivis des aspects plus pratiques des architectures en deuxième partie.

2.4.1 Aspects théoriques

Nous nous intéressons tout d'abord à l'approche NCE puis à son lien avec l'information mutuelle. Nous présentons le principe général ainsi que les méthodes concurrentes et les critiques apportées à cette formulation.

NCE et information mutuelle

Notons s le score non normalisé en sortie d'un réseau par exemple. La probabilité résultante du score s'écrit alors de la manière suivante :

$$p_m(y) = \frac{\exp\big(s(y;\theta)\big)}{Z(\theta)},$$

avec $Z = \int_y \exp\big(s(y;\theta)\big)\,dy$ la fonction de partition et θ les paramètres du réseau. Si Z est trop complexe à calculer, il est impossible d'utiliser des probabilités et donc d'estimer θ par maximisation de la vraissemblance. Dans ce cas, nous sommes face à un modèle non-normalisé. Ses paramètres peuvent être estimés par la méthode NCE. Celle-ci consiste à classifier les exemples de cette distribution par rapport à une distribution de bruit : $p_n(x)$. La fonction de coût est donc l'entropie croisée binaire. On notant C la classe (1 si y provient de p_m et 0 s'il provient de p_n), cette fonction coût est :

$$\mathrm{L}(\mathrm{C}, y) = \mathrm{C} \ln \mathrm{P}(\mathrm{C}=1|y) + (1-\mathrm{C}) \ln \mathrm{P}(\mathrm{C}=0|y)\,.$$

Avec $\mathrm{P}(\mathrm{C}=1) = \mathrm{P}(\mathrm{C}=0) = \frac{1}{2}$, la probabilité de classifier correctement se calcule de la manières suivante :

$$\mathrm{P}(\mathrm{C}=1|y) = \frac{\mathrm{P}_m(y)\mathrm{P}(\mathrm{C}=1)}{\mathrm{P}(y)} = \frac{\mathrm{P}_m(y)\mathrm{P}(\mathrm{C}=1)}{\mathrm{P}_m(y)\mathrm{P}(\mathrm{C}=1) + \mathrm{P}_n(y)\mathrm{P}(\mathrm{C}=0)} = \frac{\mathrm{P}_m(y)}{\mathrm{P}_m(y) + \mathrm{P}_n(y)}\,.$$

On peut remarquer que NCE estime donc le même ratio de densité que les GAN. La différence est que P_n est une distribution fixe alors qu'elle est apprise de manière adverse dans les GAN. Ayant estimé $\mathrm{P}(\mathrm{C}=1|y)$ il est alors facile de retrouver $\mathrm{P}_m(y)$ si $\mathrm{P}_n(y)$ n'est pas nul. L'article [MC18] propose 2 nouvelles formulations pour les modèles conditionnels non-normalisés

$$p_m(y|x) = \frac{\exp\big(s(y|x;\theta)\big)}{\mathrm{Z}(\theta)}\,,$$

avec x la donnée d'entrée et y la valeur à prédire. Les paires positives sont tirées de la distribution jointe $p_m(y|x)p(x)$ et les paires négatives du produit des distributions marginales $p_m(y)p(x)$. Il propose aussi d'effectuer une classification multi-classe en tirant pour un x un exemple positif et plusieurs exemples négatifs

$$\mathrm{L}(x, y, y_{neg}) = -\log \frac{\exp\big(s(x,y)\big)}{\exp\big(s(x,y)\big) + \sum_k \exp\big(s(x, y^k_{neg})\big)}\,.$$

Ma et Collins [MC18] ont montré que c'est un estimateur consistant et efficace. Cette formule a aussi été proposée dans le papier Contrastive Predictive Coding (CPC) [vdOLV18] sous le nom de InfoNCE. Ils montrent par ailleurs que cette fonction est reliée à l'information mutuelle I entre X et Y :

$$\mathrm{I}(\mathrm{X};\mathrm{Y}) = \int\int p_{\mathrm{XY}}(x,y) \log\Big(\frac{p_{\mathrm{XY}}(x,y)}{p_{\mathrm{X}}(x)\,p_{\mathrm{Y}}(y)}\Big)\,dx\,dy = \mathrm{D}_{\mathrm{KL}}(\mathrm{P}_{\mathrm{XY}} || \mathrm{P}_{\mathrm{X}} \otimes \mathrm{P}_{\mathrm{Y}})\,.$$

Ceci s'explique par le fait que la fonction InfoNCE effectue en quelque sorte une classification entre la loi jointe et le produit des lois marginales et que l'information mutuelle calcule justement une distance entre ces deux lois de probabilité. D'autres méthodes d'estimation de l'information mutuelle [BRB+18] ont été proposées pour l'apprentissage de représentation par exemple en utilisant l'égalité de Donsker Varadhan qui exprime la distance KL ainsi :

$$D_{KL}(P_X, P_Y) = \sup_T \mathbb{E}_{P_X}[T] - \log\left(\mathbb{E}_{P_Y}[e^T]\right).$$

Principe général

Le but de l'apprentissage de représentations est de garder l'information utile tout en éliminant l'information inutile (le plus souvent bas niveau comme l'illumination, ou la position auxquelles les représentations doivent être indépendantes). On peut considérer par exemple que les images I sont générées par une fonction f à partir de variables latentes z contenant l'information haut niveau et d'un bruit b :

$$I = f(z, b).$$

Les méthodes basées sur l'information mutuelle ont pour but de maximiser cette dernière entre les représentations provenant de données contenant la même information haut niveau mais une information bas niveau différente. Ces images peuvent être générées en appliquant des transformations différentes à une même image par exemple, ou à 2 images prises à des moments différents d'une vidéo. On peut considérer qu'elles sont générées à partir des mêmes variables latentes mais d'un bruit différent :

$$\begin{aligned} I_1 &= f(z, b_1) \\ I_2 &= f(z, b_2). \end{aligned} \tag{2.1}$$

La maximisation de l'information mutuelle permet de garder l'information des variables latentes qui est partagée entre les 2 images et d'éliminer le bruit qui est indépendant entre chaque image. Cependant, le calcul de l'information étant lourd, ces méthodes proposent de maximiser la borne inférieure appelée InfoNCE présentée dans la partie précédente. Notons z_1 et z_2 les 2 représentations provenant des images I_1 et I_2

$$InfoNCE(z_1, z_2) = -\log \frac{\exp(z_1^\top z_2)}{\exp(z_1^\top z_2) + \sum_{n=1}^{N} \exp(z_1^\top z^{(n)})}, \tag{2.2}$$

où les $z^{(n)}$ sont des exemples négatifs. Cela revient en quelques sortes à effectuer une classification entre l'exemple positif z_2 et des exemples négatifs $z^{(n)}$. L'article [WXYL18] considère cela comme un classifieur non paramétrique avec une fonction

de coût de la forme :

$$\mathrm{L}(z_i, z) = -log\mathrm{P}(i|z) = -log\frac{\exp(z_i^\top z/\tau)}{\sum_{j=1}^{n}\exp(z_j^\top z/\tau)}, \tag{2.3}$$

avec z_j les représentations des différents exemples de la base de données et z et z_i les représentations provenant d'exemples similaires. Cette fonction de coût peut aussi être interprétée comme l'estimation de $p(\mathrm{I}_2|\mathrm{I}_1)$ en utilisant la méthode NCE conditionnelle. L'utilisation d'un modèle non normalisé est obligatoire car le calcul de Z demanderait d'effectuer une somme sur toutes les images possibles ce qui est difficilement réalisable.

Méthodes concurrentes

Des limitations à l'utilisation de l'information mutuelle et à son estimation sont présentées dans [TDR+20] et [MS20]. En effet, [TDR+20] montre qu'il est possible d'avoir des représentations avec une information mutuelle très importante entre haut et bas de l'image mais qui sont très peu utiles pour une tâche de classification en aval. Il attribue plutôt les bonnes performances de cette fonction de coût à sa ressemblance avec les techniques d'apprentissage de métrique. *SimCLR* [CKNH20] compare la fonction de coût InfoNCE par rapport aux fonctions classiques d'apprentissage de métrique, une fonction de coût triplet de type « marge maximale »

$$\mathrm{L}_{margin}(z, z^+) = \max(0, m + z^{\mathrm{T}}z^+ - z^{\mathrm{T}}z^-), \tag{2.4}$$

avec m une marge positive, et celle utilisant une « sigmoïde » $\sigma(t) = 1/(1+\exp^{-t})$

$$\mathrm{L}_{sigmoid}(z, z^+) = -log\,\sigma(z^{\mathrm{T}}z^+) + log\,\sigma(z^{\mathrm{T}}z^-), \tag{2.5}$$

et montre que InfoNCE donne les meilleurs résultats. Un seul exemple négatif est utilisé dans ces fonctions de coût, on note sa représentation z^-.

Lien avec l'apprentissage supervisé

L'article [AKK+19] montre un lien entre les tâches contrastives et la tâche de classification supervisée. Cela se fait par inégalité entre la fonction de coût de classification et la fonction de coût non supervisée. Ainsi, minimiser la fonction de coût non supervisée, minimise aussi la fonction de coût supervisée. Ils considèrent une

fonction de coût contrastive de la forme :

$$\mathrm{L}_{uns}(f_\theta) = l\Big(f_\theta(x)^{\mathrm{T}}\big(f_\theta(x^+) - f_\theta(x^-)\big)\Big).$$

On peut noter que cette fonction de coût est la même que précédemment si l'on considère 1 seul exemple négatif et $l(u) = -\log(1+e^u)$. Ils considèrent un problème de classification binaire où les exemples positifs sont tirés de la même classe et l'exemple négatif tiré au hasard. Ils se basent sur le classifieur moyen qui considère les colonnes w_c du poids de classification W comme la somme des représentations pour la classe c. Ils prouvent l'inégalité

$$\mathrm{L}^{\mu}_{sup}(f) < \frac{1}{1-\tau}\big(\mathrm{L}_{uns}(f) - \tau\big),$$

où τ est un terme qui dépend du nombre de collisions c'est à dire quand l'exemple négatif x^- est tiré de la même classe que x. Même si l'hypothèse de départ est peu réaliste et ne correspond pas aux moyens usuels d'extraire les exemples positifs, celà donne une première justification théorique aux approches contrastives.

2.4.2 Applications pratiques

Dans cette partie, on présente comment cette fonction de coût est utilisée dans les différents modèles d'apprentissage de représentations.

Application aux séquences

L'approche *CPC* (contrastive Predective Coding) [vdOLV18] a tout d'abord proposé ce type d'applications pour des séquences. Par exemple, une séquence de segments sonores ou encore des séquences de patchs d'images. Considérons $x_1, \ldots, x_{\mathrm{T}}$ une telle séquence. Les éléments sont encodés en représentations $z_t = f(x_t)$ et les représentations des éléments précédents agrégés en un contexte c par un réseau g : $c_t = g(z_1, \ldots, z_t)$. Les éléments suivants sont prédits à partir du contexte : $\hat{z}_{t+k} = \mathrm{W}_k c_t$, avec W_k une matrice de poids apprise. Le coût NCE est ensuite utilisé entre la prédiction $\hat{z}_{t+k}$ et l'exemple positif z_{t+k} : $\mathrm{L} = -Info\mathrm{NCE}(z_{t+k}, \hat{z}_{t+k})$. Cette méthode a été appliquée avec succès au traitement de l'audio, de l'image et du texte. En ce qui concerne les images, l'image est découpée en grille 7×7 de patchs de taille 64×64. Le modèle autorégressif agrège l'information de haut en bas et le modèle prédit les patchs en-dessous.

CPC-v2 [HSF+19] améliore le modèle précédent spécifiquement pour les images. L'augmentation de la taille des patchs, de la taille du réseau convolutionnel ainsi que l'utilisation d'augmentations de données plus fortes et une prédiction dans les 4 directions (au lieu d'uniquement de haut en bas) avec 4 modèles autorégressifs différents sont les principales améliorations.

Amélioration des transformations

Les méthodes [vdOLV18] et [HFW+20] ont montré l'importance d'utiliser des transformations sur les patchs ou images (notamment par l'utilisation d' *Autoaugment* [CZM+19]). *Autoaugment* est une augmentation de données apprise par renforcement pour maximiser les performances de test en classification. *SimCLR* [CKNH20] montre qu'il est nécessaire d'utiliser des augmentations plus fortes pour l'apprentissage des méthodes contrastives que pour l'apprentissage supervisé. En effet, l'augmentation de la complexité des augmentations entraîne une diminution des performances pour un apprentissage supervisé mais une augmentation en classification linéaire suite à l'apprentissage non-supervisé. Les augmentations utilisées sont basées sur des distorsions aléatoires sur les couleurs ainsi que sur une mise en niveau de gris aléatoire. *PIRL* [MM20] propose d'utiliser des augmentations plus complexes de type Jigsaw Puzzle ou rotations. Cette formulation s'oppose aux méthodes auto-supervisées classiques qui ont pour but de retrouver la transformation appliquée à l'image (l'angle de rotation par exemple), et propose d'être invariant par rapport à ces transformations. Différentes transformations sont combinées dans [TKI19] : *Autoaugment*, les transformations utilisées dans SimCLR ainsi que celles dans PIRL, ce qui permet de dépasser les performances des méthodes précédentes.

Paramétrage de la fonction de coût

MoCo [HFW+20] et *SimCLR* [CKNH20] montrent l'importance de normaliser les représentations avant le calcul de la similarité par produit scalaire : $z \leftarrow z/||z||$, ainsi que l'utilisation d'une température τ adéquate pour la fonction de coût

$$\mathrm{L_N}(z_i, z_j) = -\log \frac{\exp(z_i^\top z_j/\tau)}{\exp(z_i^\top z_j/\tau) + \sum_{n=1}^{\mathrm{N}} \exp(z_i^\top z^{(n)}/\tau)}.$$

SimCLR montre notamment qu'avec normalisation, une valeur de τ de 0.1 donne les meilleurs résultats pour l'apprentissage de représentations sur la base de données ImageNet.

Maximisation de l'information entre entrée et sortie

DIM [HFLM+18] propose de maximiser l'information mutuelle entre l'entrée et la sortie d'un réseau de neurones. Les représentations des différentes localisations d'une carte de caractéristiques bas niveau notée $C(x)$ est comparée avec la sortie notée $E(x) = f\big(C(x)\big)$ en utilisant la fonction de coût InfoNCE

$$L = \sum_{i,j} \text{InfoNCE}\big(E(x), C_{i,j}(x)\big).$$

AMDIM [BHB19] améliore le modèle précédent en utilisant 2 images augmentées différentes et plusieurs échelles. On note E_n le réseau encodeur limité aux n premières couches et x_1 et x_2 deux augmentations différentes d'une même image. L'objectif est alors

$$L = \sum_{n,m} \sum_{i,j,k,l} L\big(E_n(x_1)_{i,j}, E_m(x_2)_{k,l}\big).$$

Augmentation du nombre d'exemples négatifs

Ces méthodes ont besoin d'un nombre important d'exemples négatifs pour atteindre les meilleures performances. La plupart des méthodes tirent les exemples négatifs à partir des autres exemples du batch, ce qui limite leur nombre à cause des contraintes en mémoire GPU.

L'article [WXYL18] propose d'utiliser une banque mémoire où les représentations des images de toute la base de données sont stockées, afin de les utiliser comme exemples négatifs. Pour chaque exemple d'indice i, la représentation m_i est stockée dans la banque M. Ces vecteurs sont initialisés aléatoirement et mis à jour à chaque fois que l'image i est utilisée : $m_i = f_\theta(x_i)$. Cependant, les représentations stockées dans la banque mémoire ne sont pas à jour c'est à dire qu'elles ont été calculées à partir de paramètres θ différents. Pour diminuer cet effet, une régularisation $\lambda||m_i - f_\theta(x_i)||$, avec λ une constante, est proposée afin que les représentations varient peu au cours de l'entraînement.

Une banque mémoire est aussi utilisée dans [TKI19] mais les mises à jour sont

faites en utilisant un moment pour que celles-ci varient moins au cours de l'entraînement : $m_i \leftarrow \lambda m_i + (1-\lambda) f_\theta(x_i)$ avec λ proche de 1. Il n'est alors plus nécessaire d'utiliser la régularisation précédente.

MoCo [HFW+20] quant à lui propose de stocker les exemples négatifs dans une queue. Cela enlève la limitation qui vient du fait que les représentations de toutes les images de la base de données doivent être stockées. Les représentations une fois calculées sont stockées dans la queue et seront réutilisées comme exemples négatifs pour les itérations suivantes. Pour contrer ce problème de mise à jour, la méthode utilise un réseau dont les poids sont lissés au cours du temps en utilisant aussi un moment : $\theta_2 \rightarrow \lambda\theta_2 + (1-\lambda)\theta_1$.

Clustering

Les méthodes précédentes rapprochent les représentations d'exemples provenant d'une même image transformée de manière différente. [ZZY19] propose d'étendre celà aux images proches. Il détermine les images similaires à l'aide de plusieurs *clustering* sur les représentations. Notons C_i l'ensemble des éléments proches considérés comme similaires (appartenance au même *cluster*). La fonction de coût est basée sur $P(C_i|v) = \sum_{j \in C_i} P(i|v)$ avec $P(i|v)$ défini dans l'équation 2.3.

Quant à lui, [LZX+20] considère plutôt les centres des clusters notés c_j^m. Considérons c_s^m le centroïde du cluster contenant l'exemple v_i, la fonction de coût est alors définie de la manière suivante :

$$L_{ProtoNCE} = -\log \frac{\exp(v_i^\top v_j/\tau)}{\sum_k \exp(v_i^\top v_k/\tau)} + \frac{1}{M}\sum_{m=1}^{M} \log \frac{\exp(v_i^\top c_s^m/\phi_s^m)}{\sum_{k=0} \exp(v_i^\top c_k^m/\phi_k^m)}. \tag{2.6}$$

La première partie est la fonction de coût InfoNCE classique. La deuxième partie considère comme positif le centroïde du *cluster* contenant l'exemple et comme négatif les autres centroïdes. Ils dérivent leur formulation de la méthode *Expectation Maximization* [Moo96]. L'étape E (*Expectation*) correspond au clustering et l'étape M (*Maximization*) à la mise à jour des poids selon la fonction de coût définie dans l'équation 2.6.

Multimodalité

La méthode [TKI19] maximise l'information mutuelle entre 2 vues différentes des données. Cela permet au réseau d'apprendre ce qui est commun entre les 2 vues notamment la vérité terrain. Par exemple pour les images, la décomposition *Lab* est utilisée et les 2 vues proposées sont *L* et *ab*. Pour les vidéos, l'image et le flot optique sont considérés comme 2 vues différentes. Finalement, [TKI19] étudie les différentes vues possibles afin de découvrir celles qui partagent l'information de la classe de l'image mais contiennent le moins d'information en commun. Pour cela, ils proposent les vues qui minimisent l'information mutuelle.

Application aux vidéos

La méthode CPC est appliquée aux vidéos dans [HXZ19]. Les segments futurs de la vidéo sont prédits de manière contrastive à partir des précédents. Ils proposent 3 modifications principales qui sont :

— une formulation à la manière des modèles de séquence à séquence

— des prédictions spatio-temporelles

— un apprentissage par curriculum

La formulation de séquence à séquence consiste à intégrer dans le calcul du contexte c les prédictions des temps précédents :

$$\begin{aligned} \hat{z}_{t+1} &= \phi(c_t) \\ c_{t+2} &= g(z_1, \ldots, z_t, \hat{z}_{t+1}) . \end{aligned} \tag{2.7}$$

Dans CPC, uniquement les z_t sont utilisés dans le calcul du contexte. Cependant, des prédictions sont effectuées pour plusieurs segments futurs à partir de c_t.

La méthode [HXZ19] propose aussi de ne pas réduire par pooling spatial la carte de caractéristiques de sortie afin de prédire les représentations à différentes localisations au temps suivant. La fonction de coût L s'écrit donc

$$\mathrm{L} = -\sum_{i,k} \log \frac{\exp(\hat{z}_{i,k}^{\top} z_{i,k})}{\sum_{j,m} \exp(\hat{z}_{i,k}^{\top} z_{j,m})} .$$

L'indice k prend ici en compte la position spatiale dans la carte de caractéristiques pour chaque temps i.

L'article [SBMS19] applique la méthode BERT aux vidéos en utilisant la fonction de coût InfoNCE. Certains segments de la vidéo sont cachés et le modèle doit

apprendre à les prédire à partir des autres. Un réseau *transformer* est utilisé pour la prédiction. La principale différence est le caractère bidirectionnel de la méthode qui ne prédit donc pas uniquement à partir des segments passés. Cela se traduit par une modification au niveau des prédictions : $\hat{z}_t = g(z^{-t})$ ou z^{-t} représente la séquence privée de l'élément t. Une version multimodale permet d'intégrer de l'information textuelle lors de l'apprentissage non-supervisé.

La fonction de coût InfoNCE est utilisée pour rapprocher les représentations de vidéos avec des représentations de phrases correspondantes dans [MAS+20]. Ils extraient ces phrases à partir de vidéos instructionnelles. La principale contribution est le traitement d'un possible désalignement temporel entre les frames et le texte. Un apprentissage à instances multiples utilisant plusieurs exemples positifs est intégré à la fonction InfoNCE :

$$\mathrm{L} = -\log \frac{\sum \exp(\hat{z}^\top z_k)}{\sum \exp(\hat{z}^\top z_k) + \sum_n \exp(\hat{z}^\top z_n)}.$$

2.4.3 Expériences sur les méthodes contrastives

Nous détaillons ici les différentes expériences sur les méthodes que nous avons reproduites. Le pré-entraînement est effectué sur des images et la tâche d'évaluation est la classification d'images. Les expériences sont effectuées sur les bases de données Imagenette, Imagewoof et Imagewang. Les caractéristiques de ces bases de données sont détaillées ci-dessous.

Imagenette : Cette base de données est composé des 10 classes les plus faciles d'ImageNet. Elle contient 9469 images d'entraînement et 3925 en test.

Imagewoof : Elle contient quant à elle les 10 classes les plus difficiles à différencier d'ImageNet. Les classes représentent différentes races de chiens. Elle comptabilise 9025 d'entraînement et 3929 images en test.

Imagewang : C'est une combinaison des bases de données Imagenette et Imagewang. La partie d'entraînement est composée de 10 % de Imagewoof et de Imagenette. La partie de test est la même que pour Imagewoof, il n'y a pas d'images provenant de Imagenette. Les autres exemples de Imagewoof sont dans une partie appelée non-supervisée qui peut être utilisée mais sans vérités terrain.

Contrastive Predictive Coding (CPC)

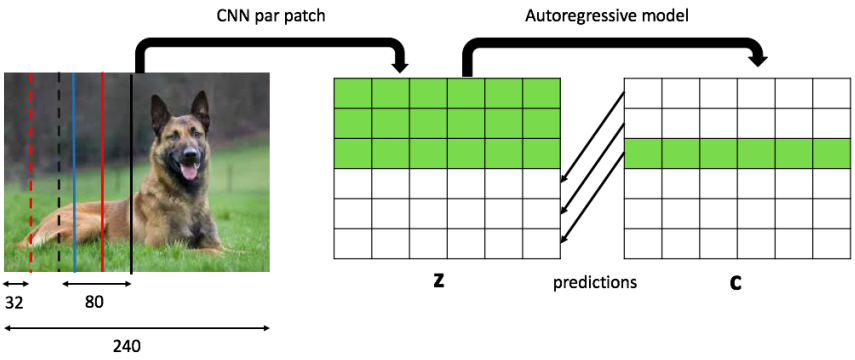


FIGURE 2.5 – Schéma de la méthode CPC appliquée à l'image en suivant l'article [HSF+19].

Description du modèle : L'image x est découpée en patchs $x_{i,j}$ où i indique la position verticale et j la position horizontale du patch. Chaque patch est encodé par une réseau f en $z_{i,j} = f(x_{i,j})$. Un modèle autorégressif a encode l'information provenant des patchs au-dessus en un contexte $c_{i,j} = a(z_{1,:}, ...z_{i,:})$. On compare par la suite avec la fonction de coût InfoNCE les prédictions $\hat{z}_{i+k,j} = W_k c_{i,j}$ aux représentations $z_{i+k,j}$.

Prétraitement : Les images de taille 256×256 sont découpées en une grille de taille 6×6 de patchs de taille 80×80. Le décalage entre les patchs est donc de 32. *Randaugment* [CZSL20] avec une magnitude de 5 est utilisé pour transformer les patchs indépendamment les uns des autres.

Architecture : Le réseau encodeur utilisé est un *Resnet50* classique. Le modèle autorégressif utilisé est de type *Pixel CNN* [vdOKE+16]. Celui-ci est composé de 5 blocs résiduels décrit dans la figure 2.5. Les représentations et les prédictions sont réduites à la dimension 128 par une couche dense avant la fonction de coût InfoNCE. Le modèle prédit deux lignes en dessous sans compter la première qui se chevauche trop avec les lignes d'au-dessus. Pour l'apprentissage supervisé après le pré-entraînement, deux choix d'architecture sont possibles. Il est possible d'effectuer le même découpage et de prendre comme représentation la moyenne des z sur les différents patchs ou bien de faire passer l'image entière à travers le réseau. Pour

la première méthode, il est compliqué d'effectuer un *finetuning*. On ne présentera donc que les scores en classification linéaire.

Apprentissage : Non-supervisé : Un batch de taille 16 est utilisé ce qui correspond à $16 \times 6 \times 6 = 576$ exemples négatifs. L'optimisation est effectuée avec Adam et un pas d'apprentissage de 0.001 pendant 500 époques.

Classification linéaire : L'apprentissage dure 200 époques avec Adam comme optimiseur et un pas d'apprentissage de 0.001. Les batchs sont de taille 128.

Résultats : Une fois le pré-entraînement effectué, il est possible de visualiser quels patchs ont été prédits par le modèle (voir figure 2.6). Nous remarquons que les patchs sont prédits le plus souvent de manière correcte même s'ils ne sont pas toujours placés exactement au bon endroit et que la méthode a des difficultés lorsque peu d'information est accessible.

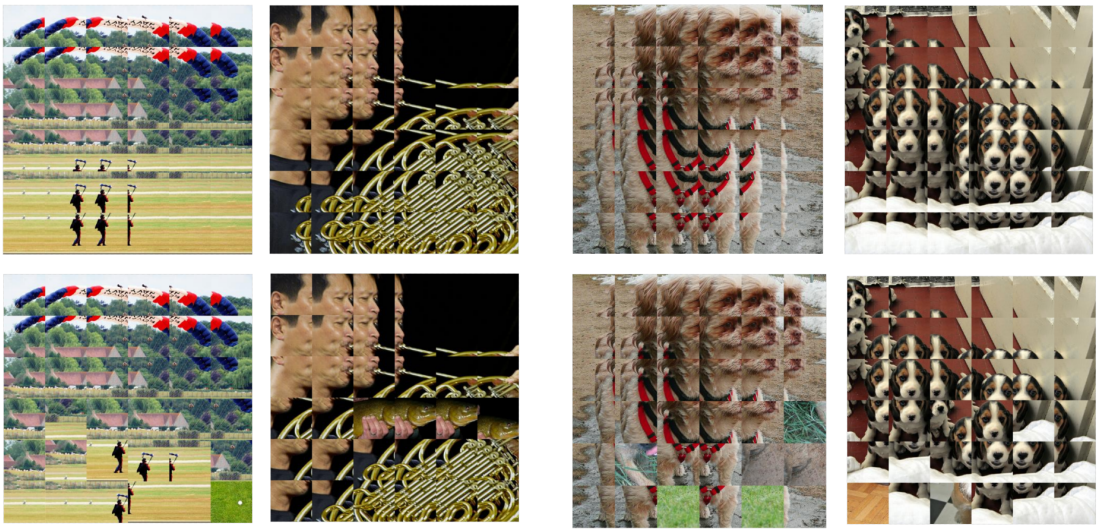


FIGURE 2.6 – Visualisation des reconstructions des 3 dernières lignes de l'image. La 1ère ligne d'images montre la vérité terrain et la 2ème ligne les prédictions de CPC. 2 lignes sont à chaque fois ignorées pour les prédictions, ainsi la 4ème ligne est prédite uniquement à partir de la 1ère, la 5ème à partir des 2 premières lignes et ainsi de suite. Les 2 premières colonnes proviennent du dataset Imagenette et les 2 autres du dataset Imagewoof.

La figure 2.7 montre que les images qui ont des représentations proches ont pour la plupart un contenu sémantique similaire. En effet, le premier groupe présente du contenu culinaire, le suivant des véhicules automobiles et le dermier des chiens. On

Méthode	Transfert	Imagenette	Imagewoof	Imagewang
CPC patch	Linéaire	90.2	73.6	58.7

TABLEAU 2.1 – Résultats de pré-entraînement par la méthode CPC sur les datasets Imagenette, Imagewoof et Imagewang.

peut remarquer quelques erreurs sur le dernier groupe où figurent aussi des images de chats et d'humains.

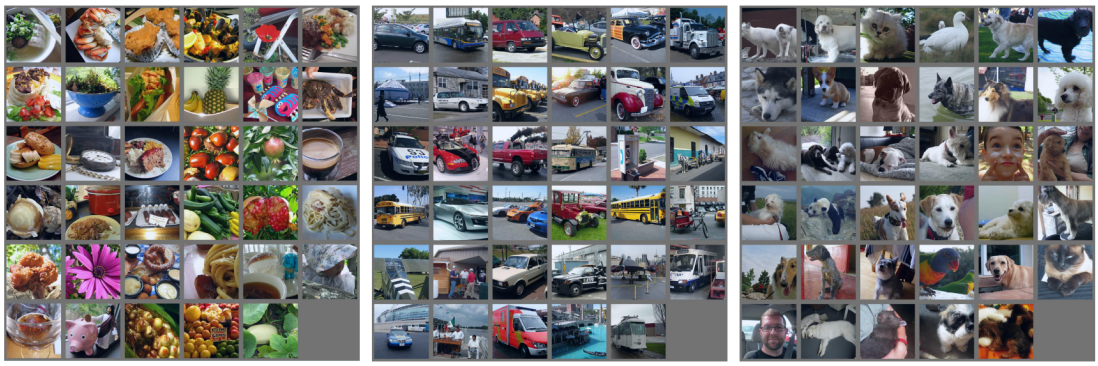


FIGURE 2.7 – Visualisation des résultats de recherche d'images à partir des représentations obtenues par l'apprentissage de CPC sur Imagenet. L'image en haut à gauche est la requête est les autres images sont les plus proches. Une similarité cosinus sur les représentations est utilisée.

Les résultats en classification linéaire sont présentés à la table 2.1. On remarque que les scores sont assez élevés (plus de 90 % sur Imagenette par exemple) et que les représentations apprises sont donc intéressantes. La table 2.2 montre que l'utilisation de la moyenne des représentations des patchs est plus efficace que de passer toute l'image à travers le réseau pour la classification linéaire. Les résultats en classification linéaire sont similaires à ceux obtenus par apprentissage de zéro. Finalement, les meilleurs résultats sont obtenus par *finetuning* sur les représentations apprises de manière non-supervisée. Nous arrivons donc à obtenir des représentations intéressantes sur les images grâce à la méthode CPC. Celles-ci permettent d'apprendre facilement une tâche de classification (reconnaissance d'images). Une de nos contributions consiste à appliquer cette méthode aux vidéos. Des expériences sur la méthode SimCLR ont également été effectuées et sont présentées dans la partie 6 où nous proposons une amélioration comme contribution.

Méthode	Transfert	Imagewoof
CPC patch	Linéaire	73.6
CPC	Linéaire	62.4
Aléatoire	*Finetuning*	62.8
CPC	*Finetuning*	72.2

TABLEAU 2.2 – Comparaison de différents types de transfert sur la base Imagewoof. Patch indique que la moyenne des représentations sur chaque patch est utilisée et non la représentation sur toute l'image.

2.5 Conclusion

Nous avons détaillé les méthodes d'apprentissage non-supervisé de représentations sur les images et les vidéos. Nous avons distingué quatres familles principales : les méthodes génératives, les méthodes basées sur la prédiction de transformations, les méthodes utilisant des pseudo-labels et les méthodes contrastives. Les méthodes contrastives sont plus particulièrement détaillées car elles obtiennent de meilleurs résultats. De même, nous avons reproduit des résultats de la méthode CPC sur les images. Il est intéressant de voir comment améliorer leur utilisation dans le cas de la vidéo. Le chapitre suivant décrit les méthodes d'apprentissage sur la vidéo, notamment pour la reconnaissance d'actions. Les architectures que nous allons utiliser y sont présentées.

Chapitre 3

Reconnaissance d'actions

Sommaire

3.1 Méthodes de reconnaissance d'actions supervisé 46
3.1.1 Réseaux de convolution 3D et variantes 46
3.1.2 Méthodes d'aggrégation à long terme 47
3.1.3 Utilisation du flot optique 49
3.1.4 Estimation de pose et reconnaissance d'actions 52
3.2 Bases de données 54
3.3 Résultats des méthodes de reconnaissance d'actions 56
3.4 Expériences effectuées sur la méthode *2 stream* 57
3.5 Conclusion 58

Etant donné l'explosion du contenu multimédia, il est utile, après avoir pris les précautions éthiques nécessaires, de pouvoir analyser des séquences vidéo de manière automatique dans le but d'indexer, rechercher ou encore proposer du contenu vidéo. Dans cette thèse, nous nous intéressons à l'apprentissage non-supervisé de représentations de vidéos. Le but est d'améliorer les performances en reconnaissance d'actions en utilisant peu de données annotées. La reconnaissance d'action est une tâche essentielle à la compréhension de contenu vidéo. Elle a de nombreuses applications comme la vidéo-surveillance (détection de comportements violents par exemple) ou encore dans le cadre d'applications smart home (toujours en conformité avec les exigences éthiques). Il s'agit d'associer une classe d'action $y \in \mathbf{Y}$ (domaine des actions) à une séquence vidéo $x \in \mathbf{X}$ représentant le domaine des vidéos. Les méthodes de l'état de l'art utilisent l'apprentissage supervisé de ré-

seaux de neurones profonds notés f_θ. Les réseaux sont appris de manière supervisée comme présenté dans le premier chapitre. La difficulté principale est la conception de modélisations spatio-temporelles efficaces. Différents choix comme les convolutions 3D, l'utilisation des flots optiques ou encore les méthodes d'aggrégation temporelles sont décrits dans la partie suivante. On présente par la suite les bases de données existantes et les scores obtenus par les méthodes de l'état de l'art.

FIGURE 3.1 – Illustration du problème de reconnaissance d'actions avec différents exemples associés à leur classe.

3.1 Méthodes de reconnaissance d'actions supervisé

Nous décrivons ici les principales méthodes de reconnaissance d'actions utilisant uniquement la vidéo [1]. Les convolutions 3D ainsi que les flots optiques permettent d'obtenir une représentation fine du mouvement alors que les méthodes d'aggrégation permettent d'incorporer une information temporelle plus long terme.

3.1.1 Réseaux de convolution 3D et variantes

Les réseaux de convolution 2D sont utilisés pour calculer des caractéristiques spatiales sur des images mais ne prennent pas en compte l'aspect temporel. Une extension aux vidéos consiste donc à utiliser des convolutions 3D qui extraient des représentations spatio-temporelles. Lubomir et al. proposent par exemple un réseau composé de 8 couches de convolution avec des noyaux de taille $3 \times 3 \times 3$ prenant en entrée 16 images consécutives [TBF+14]. Un paramètre important de ces

1. Le choix d'utiliser uniquement la vidéo est une hypothèse de base de notre travail. D'autres capteurs, fournissant par exemple des informations de profondeur, ont été utilisés dans d'autres contextes, voir par exemple [BWMT18].

méthodes est la sélection des *frames*. Quant à eux, Varol et ses coauteurs [VLS16] utilisent 60 images consécutives au prix d'une réduction de la résolution spatiale. Pour limiter la taille du réseau, un compromis entre résolution temporelle et spatiale est à effectuer. *SlowFast* [FFMH18] est un réseau avec deux voies, l'une avec une fréquence temporelle d'images élevé et l'autre faible. Cela permet de modéliser finement le mouvement mais aussi de prendre en compte des évolutions plus lentes. Peu de cannaux sont utilisés dans la voie haute fréquence afin d'avoir un modèle plus léger et rapide. Des connections latérales sont utilisées pour communiquer entre les deux voies. Beaucoup de travaux ont proposé d'adpapter les réseaux 2D en 3D et regardé si l'évolution des performances est similaire à celle de la classification d'images. Le modèle *I3D* [CZ17] étend le réseau *Inception* 2D [SVI+16] en un réseau 3D. Il initialise les poids de ce réseau en dupliquant temporellement les poids du réseau 2D pré-entraîné sur les images. Des *resnet3D* ont aussi été adapté pour la reconnaissance d'actions [HKS17]. Les performances de *resnet3D* de différentes tailles ainsi que les *resNext3D* et *DenseNet3D* sont comparées dans [HKS18]. Le réseau *ResNeXt3D-101* donne les meilleurs résultats. Un axe de recherche important a été la simplification des réseaux de convolutions 3D. Ainsi, Lin Sun et al. [SJYS15] a proposé de factoriser les noyaux 3D en un noyau spatial 2D suivi d'un noyau temporel 1D, ce qui réduit fortement le nombre de paramètres et augmente le nombre de non-linéarités. Une architecture multi-fibres [CKL+18] consiste à découper selon les cannaux les représentations intermédiaires en groupes et applique sur ceux-ci les convolutions 3D. Cela limite le nombre d'opérations et de paramètres, les sorties n'étant connectées qu'aux entrées du même groupe. La méthode proposée dans l'article de Du Tran et al. [TWTF19] va plus loin en utilisant des convolutions en profondeur (*depthwise*) (groupes de un seul cannal) lorsque le noyau est de taille $3 \times 3 \times 3$. La communication entre les cannaux se fait au niveau des convolutions $1 \times 1 \times 1$ classiques. *TSM* [LGH18] approxime des convolutions 3D en décalant temporellement les représentations de réseaux 2D sur différentes images sur une partie des cannaux. Ces méthodes ont permis de réduire le nombre de paramètres et d'augmenter la vitesse d'inférence des réseaux avec des performances similaires voire meilleures qu'avec les convolutions 3D.

3.1.2 Méthodes d'aggrégation à long terme

Les méthodes décrites ici ont pour but d'aggréger sur un temps assez long l'information spatiale provenant de réseaux 2D. Différentes images de la vidéo sont en-

codées par des réseaux 2D et leur information est aggrégée temporellement avant la classification.

Aggrégation par *pooling*

Les méthodes de cette partie se basent sur des aggrégations de type *pooling* (*max pooling*, *mean pooling*, convolutions à trous 1D). Les articles [KTS+14a] et [NHV+15] ont testé ces différentes méthodes de fusion ainsi que leur niveau d'application dans les réseaux (voir figure 3.2) : une fusion au niveau des pixels (*early fusion*), une autre avant les couches entièrement connectées (*late fusion*) et enfin une fusion progressive qui accorde une information sur une fenêtre temporelle de plus en plus grande aux couches les plus hautes (*slow fusion*). Les meilleurs résultats sont obtenus avec la *slow fusion* pour les convolutions 1D dans [KTS+14a] et avec la *late fusion* pour le *max pooling* dans [NHV+15].

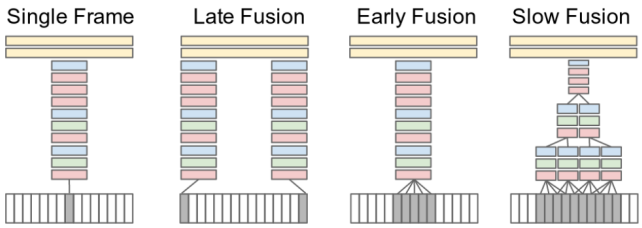


FIGURE 3.2 – Différentes localisations de la fusion (provenant de [NHV+15]).

Des opérations d'aggrégation plus complexes ont été proposées. Par exemple, *VLAD* [GRG+17] aggrège avec des scores d'attention le résidu par rapport à différents centroïdes de manière spatio-temporelle. Cela permet de garder plus d'informations car différents moments ou zones peuvent correspondre à différents centroïdes. La taille de sortie est cependant multipliée par le nombre de centroïdes choisis. Dans [DSG16], les cartes de caractéristiques issues d'un CNN sont aggrégées par une multiplication élément par élément. Un encodage bilinéaire est utilisé par la suite afin de mieux capturer les interactions des caractéristiques. Cette méthode permet d'avoir une représentation compacte et fortement discriminante et donne les meilleurs résulats.

Réseaux récurrents

Dans [DHG+14] et [NHV+15], un réseau récurrent encode la séquence de représentations (issues d'un *CNN*) des images de la vidéo. Ceci est naturel considérant

la vidéo comme une série temporelle et permet de prendre en entrée un nombre variable d'images. Cependant, les caractéristiques de haut niveau proposées en entrée du *RNN* ne permettent pas de modéliser le mouvement de manière fine, mais plutôt un changement global. Un réseau composé de couches de convolutions et de plusieurs couches récurrentes (*Gated Recurrent Units*) sur des cartes de caractéristiques est proposé par [BYPC15]. Les couches denses du réseau récurrent sont remplacées par des opérations de convolution afin de limiter le nombre de paramètres. La récurrence sur des couches plus basses du réseau permet d'améliorer la prise en compte du mouvement. Cependant, cette méthode reste moins performante que les réseaux 3D.

Opérations non locales

La méthode [WGGH17] propose une opération non locale semblable à de l'attention qui prend en compte les interactions entre toutes les positions spatio-temporelles. Ceci permet de calculer des dépendances long-terme sans passer par des opérations locales (comme les convolutions ou les cellules récurrentes). On note $x_{i,j,t}$, la position spatio-temporelle de l'entrée, la sortie $y_{i,j,t}$ associée est calculée de la manière suivante :

$$y_{i,j,t} = \frac{1}{\mathrm{N}} \sum_{i',j',t'} \varphi(x_{i,j,t}, x_{i',j',t'}) g(x_{i',j',t'}) .$$

ou φ est une fonction de relation ou similarité, g une fonction de représentation et N le nombre de termes dans la somme.

Ce type de méthode a une complexité faible et permet de prendre en compte une longueur de vidéo importante. Cependant, il ne modélise pas de manière fine le mouvement. Certaines apporches peuvent aussi être combinées avec un réseau convolutionnel 3D comme [WGGH17].

3.1.3 Utilisation du flot optique

Les méthodes utilisées avant les réseaux profonds en reconnaissance d'actions se basent sur des représentations pré-définies (des caractéristiques construites « à la main »). Les plus connues sont les trajectoires denses [WKSCL11], les histogrammes de flots et les descripteurs de frontière de mouvements [WKSL13]. Ces descripteurs sont basés sur le flot optique, son association avec les réseaux de neurone permet d'obtenir de très bons résultats en reconnaissance d'actions.

Flot optique et calcul

Le flot optique est une représentation classique du mouvement en vision par ordinateur. Il représente le vecteur de déplacement pour chaque pixel entre 2 images consécutives, comme l'illustre la figure 3.3. Notons $u(x,y)$ et $v(x,y)$ ses composantes horizontale et verticale. Le flot optique est souvent calculé à partir de l'hypohèse de constance de la luminosité

$$\mathrm{I}_{t+1}\big(x+u(x,y),y+v(x,y)\big)=\mathrm{I}_t(x,y)\,, \tag{3.1}$$

où I_t et I_{t+1} désignent 2 images consécutives. Une contrainte de régularité est aussi utilisée pour que le flot optique soit cohérent localement (notemment au niveau des zones uniformes). Elle est nécessaire car l'équation (3.1) seule ne suffit pas à déterminer les valeurs de u et v. C'est effectivement le cas pour la méthode TVL1 [SPMLF13] qui est très utilisée pour le calcul de flots optiques

$$\min_{u,v} \quad \left(\frac{\delta u}{\delta x}\right)^2+\left(\frac{\delta u}{\delta y}\right)^2+\left(\frac{\delta v}{\delta x}\right)^2+\left(\frac{\delta v}{\delta y}\right)^2.$$

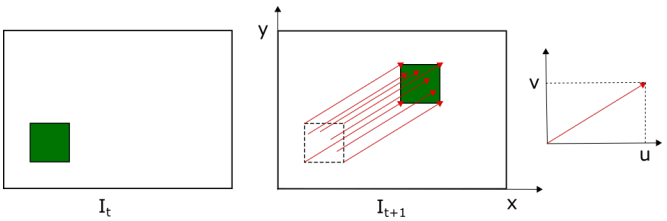


FIGURE 3.3 – Schéma du principe du flot optique. On considère ici le mouvement (flèches rouges) d'un carré entre 2 images consécutives I_t et I_{t+1}. u et v sont les 2 composantes du flot optique.

Récemment, le flot optique est calculé par des réseaux de neurone de convolution/déconvolution prenant en entrée deux images (comme dans [FDI+15]). Ces réseaux sont appris de manière supervisée sur des données synthétiques (*FlyingChairs* [IMS+17] par exemple).

Flot optique et reconnaissance d'actions

Une architecture à deux voies qui modélisent l'information spatiale pour l'une et l'information temporelle pour l'autre (illustrée figure 3.4) est présentée dans [SZ14].

La branche spatiale prend en entrée des images et la branche temporelle un empilement de flots optiques consécutifs. La prédiction finale est obtenue par moyenne des probabilités de sortie des deux réseaux.

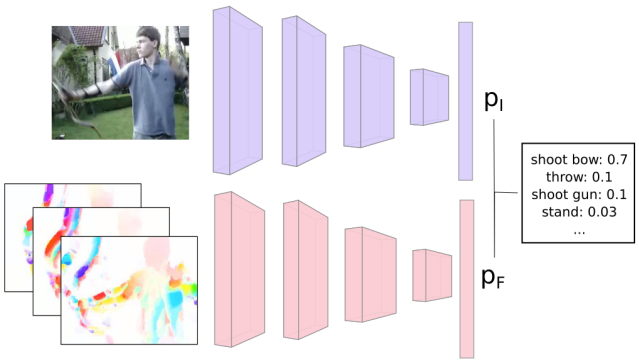


FIGURE 3.4 – Schema de l'architecture *2 stream*. Le réseau bleu constitue la branche spatiale et le réseau rouge la branche temporelle. La sortie est la probabilité pour les différentes classes.

En appliquant des solutions vues en sous partie 3.1.2, [WXW+16] et [MCKA17] permettent une modélisation plus long-terme. En effet, [WXW+16] propose de découper la vidéo en K segments et de calculer un score issu du consensus entre ceux-ci. Quant à lui, [MCKA17] aggrège (par *max-pooling*) les représentations d'un segment k en r_k. La séquence des r_k est combinée par un *LSTM* pour chacune des 2 branches. Il propose également un *CNN* 1D à 2 couches pour aggréger l'information temporellement. L'article [FPZ16] s'attaque au problème de fusion d'information entre la branche temporelle et spatiale. La fusion tardive ainsi que l'ajout de connections latérales entre les 2 branches donne les meilleurs résultats.

Un autre problème est le temps important de pré-calcul du flot optique et le fait que l'apprentissage ne soit pas de bout en bout. Un réseau (*MotionNet*) inspiré des méthodes *FlowNet* pour estimer celui-ci est proposé par [ZLNH17]. Le flot estimé est pris en entrée de la branche temporelle de la méthode *2 stream*. Il utilise la distance entre l'image suivante et la transformation par le flot estimé de l'image précédente comme fonction de coût pour l'estimation du flot optique. Ceci découle directement de l'hypothèse de constance de la luminosité :

$$\mathrm{L}_{flot} = \frac{1}{\mathrm{N}} \sum_{x,y}^{\mathrm{N}} \rho\Big(\mathrm{I}_t(x,y) - \mathrm{I}_{t+1}\big(x+u(x,y), y+v(x,y)\big)\Big),$$

avec ρ la distance de Charbonnier [CBFAB94]. La méthode [FHG+18] propose de dérouler l'algorithme TVL1 sous forme de réseau afin d'estimer le flot optique. Dans les 2 méthodes précédentes, la représentation obtenue peut être plus adaptée à la reconnaissance d'actions car elle est aussi apprise en fonction de l'erreur de classification (rétropopagation dans le réseau d'estimation du flot optique). Quant à lui, [PR18] applique l'adaptation de TVL1 sur des cartes de caractéristiques de réseaux 3D. Le multi-échelle ainsi que les multiples *warping* ne sont pas utilisés. Il montre notemment qu'enchaîner 2 opérations de calcul de flot séparées par une convolution améliore significativement les résultats. Finalement, [SKO+17] utilise comme représentation les gradients spatiaux et temporels de la vidéo $\left(\frac{\partial \mathrm{I}}{\partial x}(x,y,t), \frac{\partial \mathrm{I}}{\partial y}(x,y,t), \frac{\partial \mathrm{I}}{\partial t}(x,y,t)\right)$. Ce vecteur est perpendiculaire au flot optique. Un réseau combine ces représentations au niveau de différentes couches.

3.1.4 Estimation de pose et reconnaissance d'actions

Les méthodes vues jusqu'à présent se basent sur une donnée plutôt brute (images ou flots optiques). Il est aussi possible d'utiliser des représentations intermédaires. En reconnaissance d'actions, la plus utilisée est la pose ou le squelette de la personne. Une première étape consiste à estimer cette pose en 2 dimensions ou 3 dimensions. La seconde étape utilise cette information afin de prédire l'action effectuée par la personne.

Estimation de pose Le problème de l'estimation de pose se divise en différentes catégories : mono-personne ou multi-personne, 2D ou 3D. Les approches mono-personnes prennent en entrée des images avec une seule personne présente. Elles se basent sur des méthodes de régression ou la prédiction de cartes de chaleur. Les méthodes de régression comme [TS13] et [PSCZ14] prédisent directement les coordonnées $(x_i, y_i), i = 1, \ldots, \mathrm{K}$ des différents points du squelette. L'autre type construit une carte de chaleur différente pour chaque point du squelette. Ces cartes de chaleur vont être prédites par un CNN encodeur décodeur [BT16] [NYD16] [WRKS16].

Les méthodes multi-personnes se séparent aussi en 2 catégories : *bottom-up* et *top-down*. Les méthodes de type *bottom-up* prédisent d'abord les joints des différentes personnes puis les associent ensemble pour créer le squelette [BLPA19] [PIT+15] [ND16]. Les méthodes *top-down* prédisent d'abord les boîtes englobantes autour de chaque personne. Les points du squelette sont ensuite prédits [SXLW19] [BCL+20] [MCL18].

La plupart des méthodes de reconnaissance d'actions pour la pose considèrent

un seul squelette. Il est alors possible d'utiliser une méthode mono-personne. Il est aussi possible de combiner l'information des différents squelettes en entrée du réseau de prédiction ou encore de combiner à la fin les prédictions sur différents squelettes. Les différents prétraitements et architectures à partir des données de squelette pour la reconnaissance d'actions sont présentées dans le paragraphe suivant.

Reconnaissance d'actions à partir de la pose Les architectures pour la reconnaissance d'actions à partir de squelettes se découpent en 3 familles principales : les méthodes basées sur les réseaux récurrents (RNN), les méthodes basées sur les réseaux convolutionnels et les méthodes utilisant les réseaux sur des graphes. Nous détaillons rapidemment ces différentes approches ci-dessous.

Réseaux récurrents Les RNN sont intéressants pour modéliser des séquences. [ZLX+16] utilise 3 couches de LSTM bidirectionnel ainsi que des caractéristiques sur les co-occurences. Cependant, l'information de squelette est à la fois temporelle et spatiale, ce qui limite les performances de cette méthode. [WW17] propose un réseau à 2 branches récurrentes qui s'intéressent une à l'information temporelle et l'autre à l'information spatiale. [LSXW16] propose un LSTM spatio-temporel. La cellule cachée au niveau d'une articulation reçoit l'information de celle-ci au temps précédents mais aussi des articulations qui lui sont reliées.

Réseaux convolutionnels La première étape est de transformer l'information de squelette en une donnée sous la forme d'une image. Notons $S^t = \{J_1^t, \ldots, J_K^t\}$ les différents joints constituant le squelette dans l'image t de la vidéo. Chaque vecteur J est constitué de 3 coordonnées (x, y, z). On considère une séquence de T squelettes : $S = \{S^1, \ldots, S^T\}$. L'article [LZXP17] propose de créer à partir de celà une image de taille $T \times K \times 3$. Une image de mouvement est aussi crée en considérant les différences de coordonnées : $M^t = \{J_1^{t+1} - J_1^t, \ldots, J_K^{t+1} - J_K^t\}$. Une architecture à 2 voies avec des réseaux convolutionnels est utilisée. *SkeleMotion* [CdSB+19] génère des images à différentes échelles temporelles en considérant $M_c^t = \{J_1^{t+c} - J_1^t, \ldots, J_K^{t+c} - J_K^t\}$. Ces images sont cancaténées au niveau des canaux. Il sépare aussi l'information en magnitude et orientation. [PSK+19] propose une transformation de l'information de squelette en image plus complexe. Elle est composée des caractéristiques suivantes : Joint-Joint Distance (JJD) et Joint-Joint Orientation (JJO)

$$\begin{aligned} JJD_{i,j}^{t,t+1} &= ||J_i^t - J_j^{t+1}|| \\ JJO_{i,j}^{t,t+1} &= J_i^t - J_j^{t+1}. \end{aligned} \tag{3.2}$$

Les 2 dimensions utilisées pour former l'image sont le temps t et le spatial (les positions i,j des articulations). Des opérations supplémentaires de filtrage ou encore de renforcement du contraste sont appliquées.

Méthodes sur les graphes La méthode [YXL18] propose d'utiliser un graphe non-dirigé spatio-temporel pour modéliser la séquence des squelettes. Les noeuds sont les différents joints associés à leurs coordonnées comme caractéristiques. Les noeuds sont connectés s' ils sont au même temps et sont connectés par des parties du corps humain ou s'ils correspondent au même joint aux temps précdent et suivant. Un réseau convolutionnel sur les graphes est ensuite utilisé. [SZCL19] propose d'utiliser un graphe dirigé partant du tronc jusqu'aux extrémités. La modélisation spatio-temporelle est effectuée de manière similaire aux convolutions (2+1)D. La méthode [ZXT20] a pour but d'intégrer une information globale aux opérations de convolutions locales sur les noeuds du graphe. Pour celà, une sorte d'attention est utilisée afin de créer un contexte. Ces méthodes ont les meilleurs résultats pour la reconnaissance d'actions à partir de squelettes car elles incorporent l'information de la structure du squelette humain.

Conclusion De nombreuses méthodes de reconnaissance d'actions à partir de l'information de squelettes ont été développées. Elles nécessitent cependant d'estimer la pose des personnes à l'aide de réseaux appris de manière supervisée. Celà nécessite des annotations très coûteuses (annotation de toutes les articulations). Nous ne nous intéressons donc pas à ce type de méthodes dans cette thèse.

3.2 Bases de données

De nombreuses bases de données de reconnaissance d'action ont été créés au cours du temps. Les principales sont citées ci-dessous par ordre chronologique. Les premières bases de données contiennent peu de vidéos, peu de classes et peu de diversité. Kinetics, une des plus récentes, est de taille bien plus importante que les autres et a permis l'apprentissage de réseaux plus profonds et ainsi de repousser les performances de l'état de l'art.

KTH [LL04] (2004) : c'est l'une des premières base de données de reconnaissance d'actions. Elle est composée de vidéos montrant une personne réalisant une des 6 actions (marcher, courir, faire du jogging, boxer, applaudir et faire un signe de la main) avec un fond statique. Au total, 2391 séquences vidéo sont disponibles.

FIGURE 3.5 – Exemples de vidéos de UCF-101 en haut et HMDB51 en bas pour les classes High Jump et Shoot Bow. Une séquence d'image pour chaque vidéo est présentée.

HMDB51 [KJG+11a] (2011) : les 6849 vidéos de cette base, réparties en 51 classes proviennent de Youtube. Les catégories représentent des expressions faciales, des mouvements du corps ou encore des interactions avec des objets ou des personnes.

UCF-101 [SZS12a] (2012) : cette base de données est composé de 13320 vidéos réalistes provenant de Youtube (27 heures au total). Les vidéos appartiennent à une des 101 classes qui représentent des actions comme jouer d'un instrument de musique, pratiquer tel sport, ou encore effectuer différentes tâches. Malgré son nombre d'actions plus important, cette base est plus facile que HMDB51 car elle contient des vidéos plus longues et de meilleure qualité.

AcitvityNet [HEGN15] (2015) : c'est une base de données de 203 activités avec un total de 849 heures de vidéo. Contrairement aux bases précédentes, elle est plus équilibrée avec moins de classes de sport et les vidéos ne sont pas découpées.

Something something [GKM+17] (2017) : cette base de données décrit 174 classes de personnes effectuant des actions avec des objets du quotidien à travers 220 847 vidéos. Contrairement aux autres, les vidéos sont centrées sur l'objet et on voit peu la personne, uniquement ses bras par exemple.

Kinetics [KCS+17a] (2017) : les vidéos de cette base proviennent de YouTube comme pour UCF-101 et HMDB51 mais elle est de taille bien plus importante. Plusieurs versions sont disponibles (Kinetics 400, Kinetics 600 et Kinetics 700). Dans le cadre de cette thèse, nous utilisons Kinetics 600 qui contient 495 547 vidéos réparties en 600 classes. Chaque vidéo dure 10 secondes.

Une liste plus complète des différentes bases de données est présenté dans le tableau 3.1. Pour les expériences de cette thèse, nous utiliserons principalement les

			Action		Acteurs		Annotations			
Base de données	Années	Citations	Classes	Instances	H	N	C	T	S	Thème
KTH [SLC04]	2004	3,853	6	2,391	✓		✓			B/W, fond statique
Weizmann [BGS+05]	2005	1,890	10	90	✓		✓			mouvement humain
Coffee & Cigarettes [LP07]	2007	491	2	246	✓		✓	✓	✓	films et télévision
Hollywood2 [MLS09]	2009	1,312	12	3,669	✓		✓			films
VIRAT [OHP+11]	2011	536	23	~10,000	✓		✓	✓	✓	surveillance, vue aérienne
HMDB51 [KJG+11b]	2011	1,928	51	~7,000	✓		✓			mouvements hmain
UCF101 [SZS12b]	2012	2,470	101	13,320	✓		✓			vidéos youtube, étend UCF50
ADL [PR12]	2012	619	18	~1,200	✓		✓		✓	activités quotidiennes, vue égocentrique
THUMOS'13 [JLRZ+13, IZJ+17, SZS12b]	2013	146	*101	13,320	✓		✓		✓	vidéos Internet, étend UCF101
J-HMDB-21 [JGZ+13]	2013	458	51	928	✓		✓		✓	ré-annotation d'une partie de HMDB51
Sports-1M [KTS+14b]	2014	4,361	487	1,000,000	✓		✓			multi-classe, sports
MEXaction2 [Cru15]	2015	n/a	2	1,975	✓		✓	✓		actions culturelles
ActivityNet200 (v2.3) [CHEGCN15]	2016	797	200	23,064	✓		✓	✓		vidéos Internet non tronquées
Kinetics-400 [KCS+17b]	2017	810	400	306,245	✓		✓			vidéos Youtube
AVA [GSR+18]	2017	270	80	>392,416	✓		✓		✓	actions atomiques
Moments in Time (MiT) [MAZ+18]	2017	137	339	836,144	✓	✓	✓			variation intra-classe, vidéos Internet
MultiTHUMOS [YRJ+18]	2017	231	65	~16,000	✓		✓	✓		multi-classe, étend THUMOS
Kinetics-600 [CNBH+18]	2018	52	600	495,547	✓		✓			étend Kinetics-400
EGTEA Gaze+ [LLR18]	2018	52	106	10,325	✓		✓	✓	✓	cuisine, vue égocentrique
Something-Something-v2 [MBG+18]	2018	5	174	220,847	✓		✓			étend Something-Something
Charades-Ego [SGS+18]	2018	19	157	68,536	✓		✓	✓		vue égocentrique, activités quotidiennes
Jester [MBBM19]	2019	12	27	148,092	✓		✓			crowd-sourcing, mouvements
Kinetics-700 [CNHZ19]	2019	33	700	~650,000	✓		✓			étend Kinetics-600
Multi-MiT [MRA+19]	2019	1	313	~1,020,000	✓	✓	✓			multi-label, étends MiT
HACS Clips [ZTTY17]	2019	31	200	~1,500,000	✓		✓			vidéos Internet tronquées
HACS Segments [ZTTY17]	2019	31	200	~139,000	✓		✓	✓		étend et améliore SLAC
NTU RGB-D 120 [LSP+19]	2019	55	120	114,480	✓		✓			étend NTU RGB-D 60
EPIC-KITCHENS-100 [DDF+20]	2020	6	97	~90,000	✓		✓	✓	✓	étend EPIC-KITCHENS-55
AVA-Kinetics [LTR+20]	2020	5	80	>238,000	✓		✓		✓	AVA+Kinetics
AViD [PR20]	2020	0	887	~450,000	✓	✓	✓			personnes diverses, visages annonymisés

TABLEAU 3.1 – Tableau des principales bases de données de reconnaissance d'actions avec différentes informations comme l'année, le nombre de citations, le nombre d'actions et d'exemples. Les acteurs sont séparés entre humains (H) et non humains (N). (C) indique une annotation globale, (T) une annotation temporelle et (S) une annotation spatio-temporelle avec des boîtes englobantes. Adapté de la table 2 de [HG20].

bases de données Kinetics, UCF-101 et HMDB51. Des exemples de vidéos sont présentés en figure 3.5.

3.3 Résultats des méthodes de reconnaissance d'actions

Cette section présente des résultats permettant de comparer différentes méthodes de reconnaissance d'actions sur les bases de données présentées. Le tableau 3.2 montre que les résultats en reconnaissance d'actions sur UCF-101 et HMDB51 sont très élevés. L'association d'images et de flots optiques permet d'obtenir les meilleurs résultats. Cependant, toutes ces méthodes nécessitent un pré-entraînement sur Imagenet et/ou Kinetics suivi d'un *finetuning* sur les plus petites bases. Le tableau 3.3 compare les résultats de différents réseaux de l'état de l'art avec ou sans pré-entraînement sur Kinetics. On remarque qu'il y a une très grande différence de scores entre les deux (plus de 30 % dans la plupart des cas).

Réseau	Pré-entraînement	UCF-101	HMDB51
2-stream [SZ14]	Imagenet	88.0	59.4
I3D (Image) [CZ17]	Imagenet + Kinetics	95.1	74.8
I3D (Flot) [CZ17]	Imagenet + Kinetics	96.5	77.1
2-stream I3D [CZ17]	Imagenet + Kinetics	**98.0**	**80.7**
hidden 2-stream (I3D) [ZLNH17]	Kinetics	97.1	78.7
TSM [LGH18]	Kinetics	95.9	78.7

TABLEAU 3.2 – Résultats des méthodes de l'état en reconnaissance d'actions sur UCF-101 et HMDB51.

Réseau	Pré-entraînement	UCF-101	HMDB51
3D resnet [HKS17]	X	47.4	21.5
	Kinetics	**84.4**	**58.7**
resnet(2+1)D [SJYS15]	X	55.8	21.4
	Kinetics	**90.9**	**58.0**
SlowFast [FFMH18]	X	55.8	21.4
	Kinetics	**84.4**	**58.7**

TABLEAU 3.3 – Comparatif des résultats de reconnaissance d'actions sur UCF-101 et HMDB51 avec et sans pré-entraînement sur Kinetics

3.4 Expériences effectuées sur la méthode *2 stream*

Dans cette partie, les différentes expériences que nous avons effectuées sur l'architecture *2 stream* sont décrites. Nous avons testé différentes architectures de réseaux pour la branche spatiale et temporelle, différentes régularisations et techniques de transfert. La table 3.4 nous montre que les réseaux les plus profonds obtiennent les meilleurs résultats. Il y a cependant une exception pour l' *inception-resnet* sur la branche temporelle qui est moins performant que le *resnet-101*. Celà est sûrement du à une trop grande profondeur par rapport au nombre de données disponibles, sachant que les modèles sur les flots optiques ne sont pas pré-entraînés. Au niveau de la couche spatiale, on observe que le *finetuning* donne des meilleurs résultats que la classification linéaire simple (82.5% contre 77.5% pour un *resnet-101* par exemple). La base de données UCF-101 étant assez petite, il est important d'uti-

Modèle	Temporel	Spatial
Resnet-50	82.2	X
Resnet-101	**84.8**	82.5
Inception-resnet	83.5	**84.2**

TABLEAU 3.4 – Comparaison de différentes architectures de réseaux convolutionnels sur les branches temporelle et spatiale

liser une régularisation forte. On montre notemment qu'utiliser un taux de *dropout* de 0.8 contre 0.5 avec l'inception-resnet sur la branche temporelle permet de passer de 84.2% à 84.7%. Finalement, la combinaison des branches temporelle et spatiale améliore nettement les résultats par rapport aux branches utilisées séperemment comme le montre la table 3.5. Plus de 5 % sont gagné par rapport au meilleur résultat entre les 2 branches. Celà montre la complémentarité des images et des flots optiques.

Spatial	Temporel	Score
Resnet-101	Resnet-101	90.3
Inception-resnet	Resnet-101	**91.2**

TABLEAU 3.5 – Résultats de la méthode *2-stream*

3.5 Conclusion

Ce chapitre décrit les différentes architectures de réseaux pour effectuer la tâche de reconnaissance d'actions de manière supervisée. Nous mettons particulièrement en avant les réseaux basés sur les convolutions 3D ainsi que l'utilisation des flots optiques qui seront utilisé par la suite. Nous observons que les résultats dépendent beaucoup du pré-entraînement supervisé sur des bases de données importantes comme Kinetics. Dans le cadre de cette thèse, nous souhaitons ne pas utiliser de larges bases de données annotées. Le but principal est donc d'appliquer et de proposer des méthodes d'apprentissage non-supervisé de représentations pour s'approcher des scores de l'état de l'art en reconnaissance d'actions en utilisant peu de données annotées. Nous proposons donc d'utiliser uniquement les données annotées de la base sur laquelle nous nous évaluons (UCF-101 ou HMDB51) mais d'avoir accès aux vidéos des autres bases sans aucune annotation (Kinetics par exemple).

Chapitre 4

Apprentissage contrastif temporel

Sommaire

4.1 Mise en contexte 59
4.2 Description des modèles 60
4.2.1 Modèle principal 61
4.2.2 Modèle sans auto-régressivité 66
4.2.3 Modèle bidirectionnel 67
4.2.4 Améliorations et réseaux 3D 68
4.2.5 Evaluation 69
4.3 Expériences 70
4.3.1 Détails d'implémentation 70
4.3.2 Entraînement 72
4.3.3 Résultats 74
4.4 Conclusion 84

4.1 Mise en contexte

Comme nous l'avons vu précédemment dans l'état de l'art, de nombreuses méthodes d'apprentissage de représentation non-supervisé proposent de générer les *frames* suivantes de la vidéo à partir des précédentes. Cette tâche est intéressante sémantiquement car le modèle doit d'une certaine manière « comprendre » les actions en cours et les mouvements effectués par les différentes personnes afin de pouvoir estimer une suite possible à la vidéo. Cependant, la tâche de prédiction

dans l'espace des pixels est très complexe. En effet, plusieurs problèmes se posent : la nécessité de prendre en compte plusieurs futurs possibles, la prise en compte de critères bas niveau (luminosité, changement de textures) et la prédiction lointaine dans le temps. Différentes méthodes ont été employées pour résoudre ces problèmes comme notemment l'utilisation de modèles génératifs comme les *GAN* ou *VAE* (présentées section 2.1) pour prédire plusieurs futurs et qui sont plus réalistes. Des transformations des images précédentes, par exemple à base de flots optiques, ont été utilisées pour garder l'information bas niveau. Cependant, même si la qualité de ces méthodes en terme de prédiction s'est améliorée de manière significative, elles peinent à fournir des représentations utiles pour d'autres tâches comme la reconnaissance d'actions.

Nous proposons dans ce chapitre de prédire le futur de la vidéo sans génération en utilisant la méthode *Contrastive Predictive Coding* (CPC) [vdOLV18]. La tâche proposée consiste à prédire quel est le segment futur de la vidéo parmi différents choix. Les mauvaises réponses sont appelées « exemples négatifs » et la bonne réponse « exemple positif ». Cette tâche de classification est plus simple à réaliser que la génération et résout les différents problèmes cités ci-dessus. En effet, même si plusieurs futurs sont probables, parmi les choix proposés un seul est possible. Le réseau peut choisir les informations qui lui sont utiles pour résoudre la tâche et n'a pas à garder toute l'information bas niveau inutile pour obtenir des représentations de haut niveau sémantique.

De plus, la méthode *Contrastive Predictive Coding* a été utilisée avec succès pour apprendre des représentations sur d'autres types de données comme des images ou du son. Nous exposons dans ce chapitre quels sont les meilleurs moyens de l'utiliser pour apprendre des représentations de vidéos utiles. Une contribution importante est l'étude de différents modèles (autorégressif, non autorégressif, bidirectionnel) ainsi qu'architectures pour le réseau autorégressif (réseaux récurrents, réseaux 3D masqués, ...). La sélection des exemples négatifs ainsi que les prétraitements et modalités d'entrée sont aussi analysés.

4.2 Description des modèles

Nous détaillons dans cette partie les différents modèles de pré-entraînement non supervisé proposés basés sur la méthode *Contrastive Predictive Coding* ainsi que leur protocole d'évaluation.

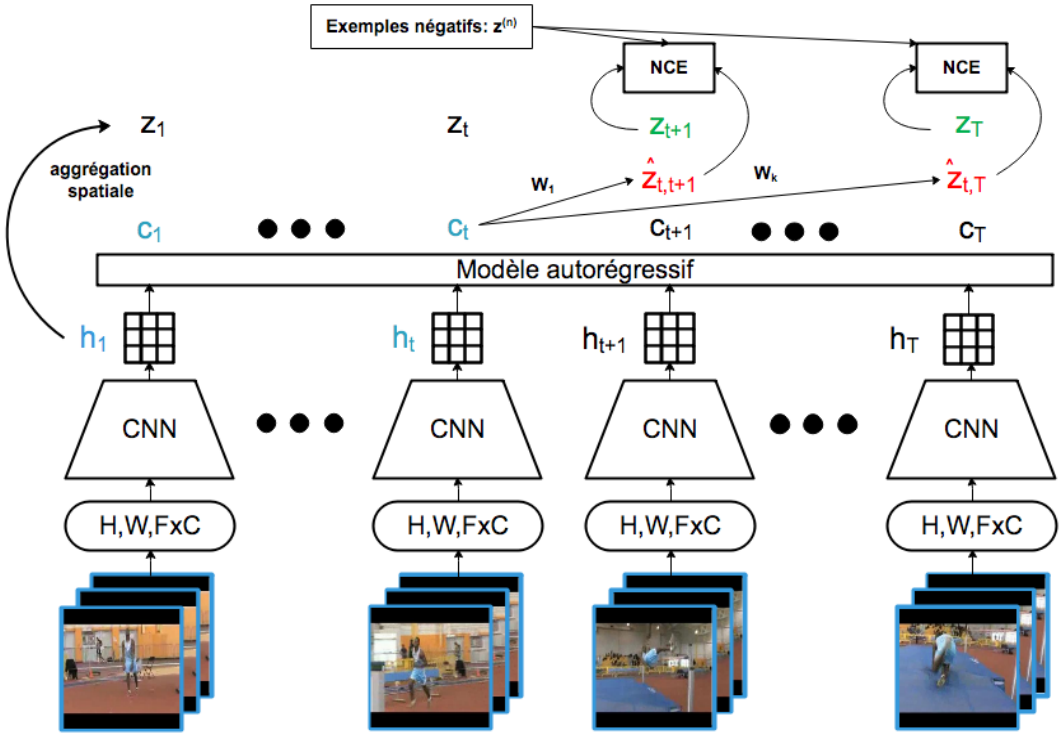


FIGURE 4.1 – Schema illustrant la méthode *Contrastive Predictive Coding* pour les vidéos. Les différents segments vidéos sont encodés par un CNN. Les cartes de caractéristiques passées en sortie sont aggrégées par un modèle autorégressif. Le contexte c_t permet de prédire les représentations des segments futurs

4.2.1 Modèle principal

La vidéo est tout d'abord découpée en T segments de F *frames* notés $x_1, \ldots, x_T$, comme illustré sur la figure 4.1. Ces *frames* peuvent être de différents types : images, différence d'images ou flots optiques. Les segments sont sélectionnés de manière à se chevaucher de la moitié de leur longueur. Pour rendre les segments moins semblables, différentes transformations aléatoires sont utilisées indépendemment sur ces derniers. Les mêmes transformations sont appliquées sur les différentes *frames* d'un même segment, comme par exemple le *cropping* ou encore le *color dropping*. Ces transformations sont souvent utilisées en apprentissage auto-supervisé afin d'éviter que le réseau ne converge vers des solutions « triviales » en se basant sur les informations bas niveau.

Afin d'utiliser un réseau 2D, les *frames* sont concaténées au niveau des canaux comme l'illlustre la figure 4.1. L'entrée est donc de dimension $(B \times T) \times H \times W \times (F \times C)$ où B désigne la taille du *batch*, T le nombre de segments de chaque exemple du

batch, F le nombre de *frames* par segment, H le nombre de pixels en hauteur, W le nombre de pixels en largeur et C le nombre de cannaux (3 pour les images, 2 pour les flots optiques et 1 pour les différences d'images). Chaque segment est encodé par le réseau convolutionnel (*CNN*) noté f_θ de paramètres θ. Les cartes de caractéristiques en sortie sont notées $h_1, \ldots, h_T$ et sont définies par :

$$h_t = f_\theta(x_t) \quad \text{pour } t \in [1, \ldots, T].$$

Notons H_o, W_o les dimensions spatiales et C_o le nombre de cannaux en sortie du réseau. h_t a donc une dimension $H_o \times W_o \times C_o$.

Pour aggréger temporellement les représentations des segments, un modèle autoregressif de paramètres α noté a_α est utilisé. Il transforme la séquence $h_1, \ldots, h_T$ en une séquence de contextes $c_1, \ldots, c_T$. C'est à partir de ces contextes que les prédictions du futur vont être effectuées. Ainsi, les contextes ne doivent pas contenir d'information des segments futurs. Le contexte c_t ne dépend donc pas de $h_{t+1}, \ldots, h_T$ et s'écrit de la manière suivante :

$$c_t = a_\alpha(h_1, ..., h_t) \quad \text{pour } t \in [1, \ldots, T].$$

Le modèle autorégressif prend en entrée des cartes de caractéristiques afin qu'il produise une modélisation spatio-temporelle. Cependant, nous n'effectuons que des prédictions temporelles. Le *CNN* prenant en entrée toute l'image et n'ayant pas de restriction spécifique au niveau du champ réceptif, il n'est pas forcément intéressant d'effectuer des prédictions spatiales. Ainsi, nous allons chercher à prédire l'aggrégation spatiale des représentations des segments : $z_t = p(h_t)$. Les z_t sont donc de dimension C_o.

Les prédictions sont faites à partir de $T-1$ fonctions linéaires de poids $W_1, \ldots, W_{T-1}$. Pour chaque contexte c_t, le réseau prédit $\hat{z}_{t,t+1}, \ldots, \hat{z}_{t,T}$ avec

$$\hat{z}_{t,t+k} = W_k p(c_t) \quad \text{pour } t \in [1, T-1] \text{ et } k \in [1, T-t],$$

prédictions qui seront comparés aux $z_{t+1}, \ldots, z_T$.

Les prédictions $\hat{z}_{t,t+k}$ sont comparées à z_{t+k} (l'exemple positif) et à N exemples négatifs. Le produit scalaire est utilisé comme fonction de similarité. Les scores obtenus sont utilisés pour faire une classification avec N+1 classes où la classe correcte correspond à l'exemple positif. La fonction de coût utilisée pour cette classification

est l'entropie croisée. La fonction de coût sera donc minimale pour une similarité importante entre la prédiction et l'exemple positif et une similarité faible avec les exemples négatifs. La fonction de coût InfoNCE [vdOLV18] est définie de la manière suivante :

$$\mathrm{L_N}(z_t, \hat{z}_t) = -\log \frac{\exp(\hat{z}_t^\top z_t)}{\exp(\hat{z}_t^\top z_t) + \sum_{n=1}^{\mathrm{N}} \exp(\hat{z}_t^\top z^{(n)})}, \tag{4.1}$$

où $z^{(n)}$ désigne les exemples négatifs.

La fonction de coût finale pour le modèle est la somme de toutes les prédictions (prédiction jusqu'à la fin de la séquence pour chaque t) pour toutes les vidéos dans le *batch* :

$$\mathrm{L} = \sum_{i=1}^{\mathrm{B}} \sum_{t=1}^{\mathrm{T}-1} \sum_{k=1}^{\mathrm{T}-t} \mathrm{L_N}(z_{t+k}^{(i)}, \hat{z}_{t,t+k}^{(i)}). \tag{4.2}$$

Les représentations de tous les segments dans les autres vidéos du *batch* ainsi que les autres segments de la vidéo sont utilisés comme exemples négatifs. Ces derniers sont appelés exemples négatifs « difficiles » car ils ressemblent plus à l'exemple positif. Il y a au total $\mathrm{B} \times \mathrm{T} - 1$ exemples négatifs avec $(\mathrm{B}-1) \times \mathrm{T}$ exemples négatifs « faciles » et $\mathrm{T}-1$ exemples négatifs « difficiles ».

Points principaux de la méthode Les quatre points principaux de la méthode sont illustrés dans la figure 4.2 et détaillées ci-dessous.

Réseaux auto-régressifs utilisés Différents modèles autorégressifs ont été testés. Les modèles les plus simples comme les *LSTM* utilisent des séquences temporelles de représentations vectorielles. Les autres (ConvLSTM, Conv3D masquées et Conv(2+1)D masquées) se basent sur une séquence de carte de caractérisitiques. Un *padding* temporel permet aux modèles basés sur les convolutions 3d de les rendre autorégressifs que l'on nomme convolutions masquées. Différents modèles autorégressifs et non-autorégressifs sont présentés dans la figure 4.3. La structure du modèle auto-régressif joue un rôle important comme présenté dans l'étude d'ablation 4.8.

Utilisation des exemples négatifs difficiles Les exemples négatifs difficiles proviennent de la même vidéo mais à des instants différents. Le modèle ne peut donc pas se servir d'information constante dans la vidéo pour les différencier

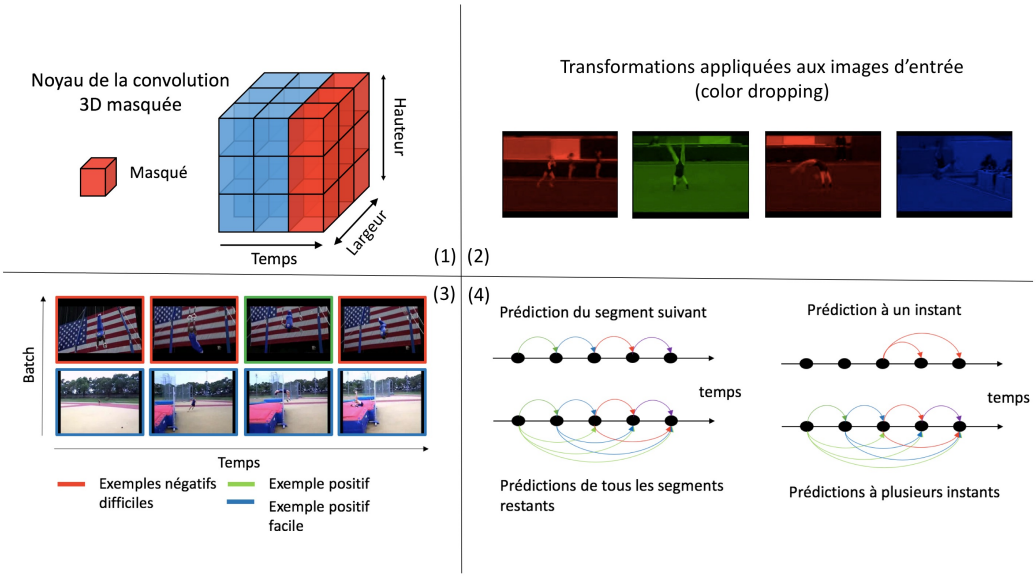


FIGURE 4.2 – Représentation des principaux points de la méthode CPC sur la vidéo. (1) Utilisation de réseaux autorégressifs basés sur les convolutions 3D masquées. (2) Utilisation d'augmentations différentes sur les segments. (3) Utilisation d'exemples négatifs difficiles. (4) Prédiction à partir de tous les instants de la vidéo et de tous les segments restants.

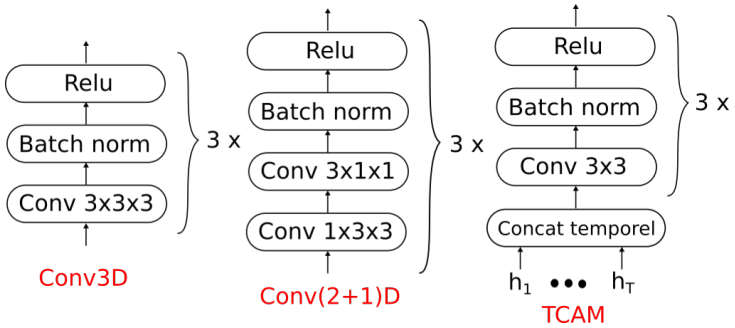


FIGURE 4.3 – Schéma de différents modèles d'aggrégation/autorégressifs. Pour les 2 premiers, avec des convolutions causales, le modèle devient autorégressif.

comme l'arrière plan. Il est ainsi obligé de comprendre l'ordre temporel de la vidéo pour résoudre la tâche. Comme l'a montré la littérature [MZH16, FBGG16, LHSY17, XXZ+19], l'apprentissage de l'ordre temporel permet d'apprendre des représentations intéressantes pour la reconnaissance d'actions dans les vidéos.

Augmentation sur les données d'entrée Les segments d'une même vidéo sont très similaires ce qui rend la tâche de prédiction assez simple. Le réseau peut avoir recours à l'information bas niveau partagée entre les deux segments pour effectuer la prédiction. Il est ainsi nécessaire de différencier les segments à un bas niveau sémantique en modifiant les couleurs par exemple. Ces transformations sont précisées dans les détails expérimentaux.

Prédiction long-terme De nombreuses méthodes de prédiction des *frames* suivantes se contentent de prédire la *frame* suivante ou un segment de quelques *frames*. Nous proposons de prédire tous les segments futurs restants. Il y a donc des prédictions plus long terme qui sont plus difficiles à apprendre mais qui permettent aussi d'obtenir de meilleures représentations (comme nous le montrerons grâce à l'étude d'ablation présentée au paragraphe 4.8).

Comparaison par rapport aux méthodes de l'état de l'art similaires Les deux méthodes de l'état de l'art les plus proches de notre méthode sont *Dense Predictive Coding* (*DPC*) [HXZ19] et *Contrastive Bidirectionnal Transformer* (*CBT*) [SBMS19]. Ces méthodes sont contemporaines à la notre. Nous mettons ici en évidence leurs principales différences par rapport à notre approche.

***Dense Predictive Coding* (*DPC*)** La méthode présentée ici est plus proche de *CPC* que la méthode *DPC*. En effet, on effectue la prédiction de plusieurs segments futurs à partir d'un contexte alors que DPC intègre les prédictions dans le calcul du contexte futur. Notre méthode est plus simple car elle ne nécessite pas d'effectuer des prédictions spatio-temporelles, uniquement des prédictions temporelles. Les prédictions spatio-temporelles sont restrictives car elles ne permettent pas d'utiliser des crops différents pour les segments. De plus, le champ receptif à la fin du réseau étant très étendu spatialement, chaque élément de la carte de caractéristique contient l'information de toute l'image et non une information locale ce qui n'est pas souhaitable. *DPC* utilise des techniques d'apprentissage par curriculum ce qui n'est pas nécessaire dans notre méthode. Finalement, nous proposons d'appliquer notre méthode sur d'autres modalités comme les différence d'images et les flots optiques contrairement à *DPC*.

***Contrastive Bidirectionnal transformer* (*CBT*)** *CBT* se base sur la méthode *BERT* [DCLT19] pour prédire un segment masqué à partir des autres segments de la vidéo. Pour celà, un modèle transformer est utilisé. Un avantage de la méthode *CBT* est le côté bidirectionnel par rapport à notre méthode qui utilise un modèle autorégressif. L'autorégressivité permet d'effectuer un grand nombre de prédictions à chaque itération ce qui rend l'apprentissage plus rapide. L'utilisation d'un modèle autorégressif basé sur des convolutions 3D masquées a le bénéfice de prendre en entrée des cartes de caractéristiques contrairement au transformer. Finalement, les réseaux utilisés dans *CBT* sont de taille beaucoup plus importante et nécessitent beaucoup plus de moyens de calcul (comme par exemple le Cloud TPU de Google).

4.2.2 Modèle sans auto-régressivité

Les méthodes de génération du futur de vidéos peuvent être divisées en deux catégories selon qu'elles soint autorégressives ou non. Les méthodes autorégressives prédisent le futur à chaque instant de la vidéo. Les méthodes non-autorégressives considérent une partie de la vidéo comme le passé et une autre partie comme le futur à prédire. On s'intéresse à l'utilité de l'autorégressivité dans le modèle défini précédemment pour l'apprentissage de représentations. Pour celà, nous le comparons à un modèle non-autorégressif défini ci-dessous.

Les segments vidéos sont encodés de la même manière que précédemment. Cependant, pour un instant b donné, on va prédire les segments futurs $x_{b+1}, \dots, x_{\mathrm{T}}$ à partir des segments passés $x_1, \dots, x_b$. Le contexte c est l'aggrégation des représentations $h_1, \dots, h_b$ par un réseau d'aggrégation a_α. Comme l'on ne prédit qu'à un seul instant, on ne considère ici qu'un seul contexte c. Le réseau n'a donc as besoin d'être autorégressif temporellement contrairement au modèle précédent

$$c = a_\alpha(h_1, \dots, h_b)\,.$$

Les représentations des segments futurs sont prédites à partir du contexte c de la même manière que précédemment

$$\hat{z}_{b+k} = \mathrm{W}_k p(c) \text{ pour } k \in [1, \mathrm{T} - b]\,.$$

La fonction de coût finale est la somme des fonctions de coût pour la prédiction des segments futurs à partir de c pour toutes les vidéos dans le *batch* :

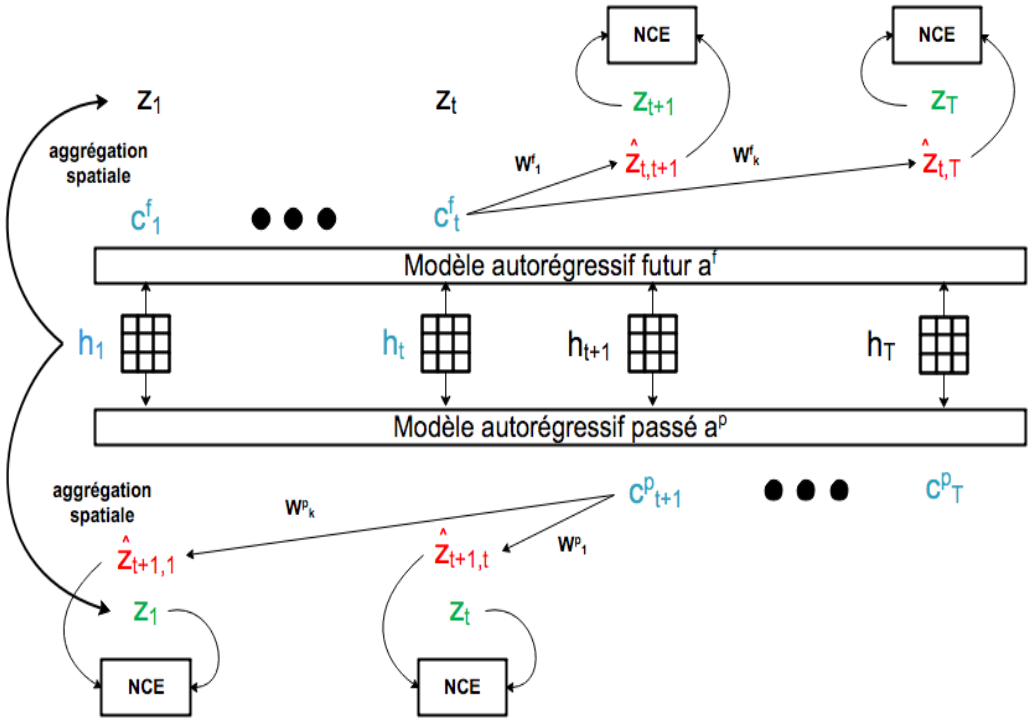


FIGURE 4.4 – Schema de la variante bidirectionnelle de la méthode CPC pour les vidéos. Deux modèles autorégressifs sont utilisés pour effectuer des prédictions dans les deux sens : vers le futur et vers le passé.

$$\mathrm{L} = \sum_{i=1}^{\mathrm{B}} \sum_{k=1}^{\mathrm{T}-b} \mathrm{L}_{\mathrm{N}}(z_{b+k}^{(i)}, \hat{z}_{b+k}^{(i)}) \,. \tag{4.3}$$

En pratique, on utilise ici des séquences de taille $\mathrm{T} = 8$ et l'on prédit les 4 derniers segments à partir des 4 premiers ($b = 4$).

4.2.3 Modèle bidirectionnel

Le modèle principal effectue des prédictions dans le futur mais il est tout autant intéressant d'effectuer des prédictions dans le passé. En effet, l'utilisation de ce type de méthode sur des données textuelles a montré qu'un modèle bidirectionnel améliore le plus souvent les performances. Les *LSTM* bidirectionnels permettent d'apporter plus d'information que les *LSTM* unidirectionnels. Nous proposons donc un modèle bidirectionnel qui prédit les segments dans les deux sens temporels.

En ce qui concerne les prédictions futures, rien ne change par rapport au modèle principal. Pour la prédiction du passé, un autre modèle autorégressif est utilisé

pour aggréger l'information future. On ajoute l'indice p pour indiquer les valeurs et réseaux utiles à la prédiction dans le passé. Le contexte est donc le suivant :

$$c_t^p = a_{\alpha^p}^p(h_\mathrm{T}, \ldots, h_t)\,.$$

Les prédictions utilisent d'autres poids W_k^p et s'effectuent vers le passé

$$\hat{z}_{t,t-k}^p = \mathrm{W}_k^p\, p(c_t^p) \text{ pour } t \in [\mathrm{T}, 2] \text{ et } k \in [1, t-1]\,.$$

La fonction de coût compare les prédictions $\hat{z}_{t,t-k}^p$ aux représentations z_{t-k} :

$$\mathrm{L}^p = \sum_{i=1}^{\mathrm{B}} \sum_{t=\mathrm{T}}^{2} \sum_{k=1}^{t-1} \mathrm{L}_\mathrm{N}(z_{t-k}^{(i)}, \hat{z}_{t,t-k}^p)\,. \tag{4.4}$$

La fonction de coût finale est la somme des fonctions de coût pour les prédictions du futur et du passé : $\mathrm{L}_{bidir} = \mathrm{L} + \mathrm{L}^p$.

4.2.4 Améliorations et réseaux 3D

Les méthodes *CPCv2* [HSF+19] et *SimCLR* [CKNH20] (introduits dans la section sur les méthodes contrastives du chapitre 2) ont apporté des améliorations pratiques sur l'utilisation des méthodes contrastives pour l'apprentissage de représentations. Ces modifications se focalisent sur la fonction de coût *InfoNCE* introduit équation (4.1), ainsi que sur les transformations appliquées aux données d'entrée. Nous nous intéressons aussi à l'utilisation de réseaux de taille plus important plus précisemment des réseaux 3D (resnet3D-18 [HKS17] et resnet(2+1)D-18 [TWT+18]). Ceux-ci sont beaucoup plus longs à entraîner et nécessitent plus de mémoire.

Fonction de coût Les représentations z_t ainsi que les prédictions $\hat{z}_t$ sont normalisées avant de calculer la similarité. La normalisation permet d'éviter d'avoir des scores de similarité très faibles ou très élevés et associée à une température τ adaptée (usuellement 0.1) permet d'obtenir de meilleurs représentations. La fonction de coût utilisée est donc la suivante :

$$\mathrm{L}_\mathrm{N}(z_t, \hat{z}_t) = -\log \frac{\exp\left(\frac{\hat{z}_t^\top z_t}{\tau||\hat{z}_t||\,||z_t||}\right)}{\exp\left(\frac{\hat{z}_t^\top z_t}{\tau||\hat{z}_t||\,||z_t||}\right) + \sum_{n=1}^{\mathrm{N}} \exp\left(\frac{\hat{z}_t^\top z^{(n)}}{\tau||\hat{z}_t||\,||z^{(n)}||}\right)}. \tag{4.5}$$

Architecture La dimension des z_t est aussi réduite à une taille 128 par un réseau *MLP* après le réseau 3D. Les opérations de *batch normalisation* sont remplacées par des couches de *normalisation* de manière similaire à la 2ème version de *CPC*. En effet, nous avons remarqué des différences de score importantes avec l'utilisation de la *batch normalisation* entre le mode d'entraînement et celui de test. Les statistiques sur le *batch* permettraient donc de faire passer de l'information entre différents segments ce qui n'est pas souhaitable.

Partage des exemples négatifs Les réseaux 3D étant beaucoup plus demandeurs en mémoire, la taille du *batch* et donc le nombre d'exemples négatifs sont réduits. La solution proposée est de partager les exemples négatifs entre les GPU. Cela permet en effet de multiplier le nombre d'exemples négatifs par le nombre de GPU utilisés pour l'expérience.

Prétraitement Le prétraitement des données est modifié. Des segments de durée temporelle plus longue sont sélectionnés. Cela permet aux réseaux 3D d'avoir un contexte plus important. La 2ème version de *CPC* montre que l'utilisation de patchs plus grands améliore les résultats. Les augmentations utilisées dans le papier *SimCLR* sont appliquées à la place du *color dropping* (voir la partie 4.3.1 pour plus de détails).

4.2.5 Evaluation

Le pré-apprentissage est évalué sur la tâche de reconnaissance d'actions. Pour cela, le réseau de reconnaissance d'actions est initialisé avec les poids appris grâce au pré-entraînement non-supervisé décrit ci-dessus. La moyenne des z_t (sorties du CNN) sur différents segments sélectionnés uniformément dans la vidéo est utilisé comme représentation. Cette représentation est normalisée (batch normalization bn) avant de passer à travers une couche de classification linéaire (de poids W_c et de biais b_c avec une activation softmax) :

$$o = \mathrm{softmax}\Big(\mathrm{W}_c\, bn\Big(\frac{1}{\mathrm{T}}\sum_{t=1}^{\mathrm{T}} p\big(f_\theta(x_t)\big)\Big) + b_c\Big).$$

Lors de l'entraînement supervisé, la fonction de coût utilisée est l'entropie croisée.

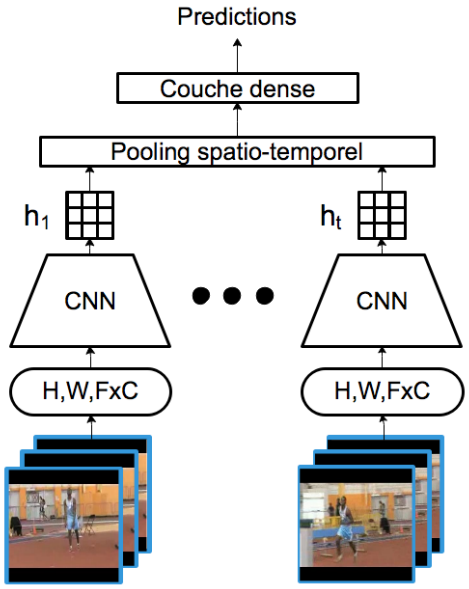


FIGURE 4.5 – Schéma de la méthode d'évaluation de l'apprentissage de représentations sur les vidéos. La partie CNN est la partie pré-entraînée.

Deux critères différents sont utilisées pour évaluer la qualité de la représentation fournie par notre méthode : la classification linéaire et le *finetuning*. Pour la classification linéaire, uniquement les poids de la couche de classification sont mis à jour (les paramètres W_c et b_c). Au contraire, pour le *finetuning*, tous les poids du réseau sont optimisés c'est à dire W_c, b_c et θ.

4.3 Expériences

4.3.1 Détails d'implémentation

Cette section décrit les détails d'implémentation, notemment les différents prétraitements, l'obtention des différentes modalités ainsi que le choix des réseaux et les paramètres d'apprentissage.

Modalités d'entrée

Différentes modalités d'entrée (les images, différence d'images et les flots optiques ont été testés). Les flots optiques ont été extrait en utilisant les implémentations d'OpenCV. On s'est intéressé aux méthodes TVL1 [SPMLF13] pour sa précision

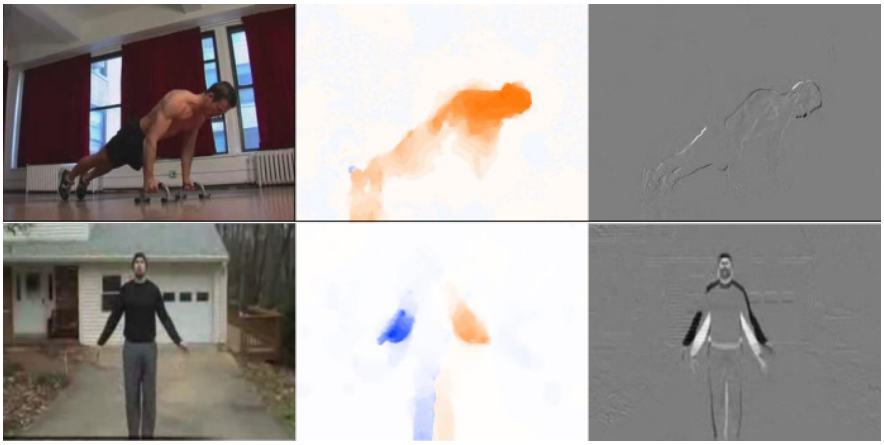


FIGURE 4.6 – Visualisation des différentes modalités utilisées pour cette méthode (images, flots optiques et différence d'images)

et Disflow [KTDG16] pour sa rapidité. Les valeurs des flots optiques sont sueillées entre -10 et 10 avant d'être quantifiées en 256 valeurs pour obtenir une image jpeg. Pour les différences d'images, les images sont d'abord transformées en niveau de gris et la différence entre deux frames consécutives est utilisée.

Prétraitement

Pour l'entraînement non-supervisé, chaque segment est rogné (l'opération de rognage (appelée *cropping* en anglais) est une augmentation de donnée où l'on sélectionne aléatoirement une partie de l'image) de manière différente (des segments différents ont des *crops* différents mais les images du même segment le même). Les images de taille 256×342 sont *rognées* à la taille 224×224. La moyenne spatiale est soustraite lorsque les flots optiques sont utilisés comme proposé dans [SZ14]. Pour la modalité image, l'augmentation *color dropping* est utilisée pour réduire la similarité des segments sur une même vidéo. Cela consiste à mettre à zéro deux cannaux de couleur parmi les trois.

Pour l'apprentissage supervisé (de zéro c'est à dire sans pré-apprentissage, classification linéaire et *finetuning*), le même *crop* est appliqué et un *flipping* horizontal est ajouté. Pas de *color dropping* est utilisé sur les images.

Réseaux et hyperparamètres

Nous utilisons dans les expériences suivantes un resnet2D 18 (présenté figure 1.5). Comme c'est un encodeur efficace et rapide et peu demandeur en mémoire. Le

modèle auto-régressif est composé de trois couches de convolutions 3D masquées avec batch normalization et activation ReLU. La sortie du réseau est une carte de caractéristiques de taille $7 \times 7 \times 512$. Celle-ci est moyennée spatialement avant la couche de classification pour l'apprentissage supervisé. Pour le pré-entraînement non-supervisé, on utilise 10 segments de 10 frames (T = F = 10). La taille de batch est fixée à 25 ce qui correspond à 249 exemples négatifs.

Modifications pour les réseaux 3D

En ce qui concerne les réseaux 3D, seulement des expériences avec la modalité image ont été effectuées. En ce qui concerne le prétraitement, des segments de 16 images avec un pas de deux sont sélectionnés ce qui correspond environ à une seconde. Six segments sont sélectionnés par vidéo et ceux-ci sont espacés de 15 frames.

On utilise l'augmentation *Color Distort* comme dans l'article original de *SimCLR*. Des *crops* plus aggressifs que précédemment (allant jusqu'à une couverture de 10 % de l'image) sont effectués. Les réseaux resnet3D 18 et resnet(2+1)D 18 sont employés. Les resnet(2+1)D 18 sont plus demandeurs en mémoire, les tailles de *batch* ne sont donc pas les mêmes. Nous utilisons dans les deux cas 8 GPU avec cinq exemples par GPU pour les resnet3D et trois exemples par GPU pour les resnet(2+1)D.

4.3.2 Entraînement

Réseaux 2D

Les réseaux sont entraînés avec la descente de gradient stochastique (SGD) avec un moment de 0.9 et une régularisation L_2 de 0.0001.

Apprentissage non supervisé : L'entraînement dure 150000 itérations. Le pas d'apprentissage initial est fixé à 0.01 et est diminué aux itérations 20000, 40000 et 60000 lorsque l'on s'entraîne sur UCF-101. La taille du *batch* et le pas d'apprentissage sont multipliés par 2 sur Kinetics.

Classification linéaire : 8 segments sont utilisés pour l'apprentissage et 15 pour le test. La taille du batch est fixée à 16. La couche de classification est entraînée durant 20000 itérations. Le pas d'apprentissage initial est de 0.005 et est diminué à

0.001 après 10000 itérations. Un dropout of 0.5 est utilisé. Les poids du CNN sont fixés pour la classification linéaire (seulement les poids de la couche de classification sont appris).

Apprentissage de zéro : Comme l'entraînement de zéro nécessite de voir beaucoup plus d'exemples, on augmente la taille du batch à 64 mais on utilise un seul segment pour l'entraînement. Le nombre d'itérations est augmenté à 50000. Le pas d'apprentissage initial est de 0.01 et est diminué à 0.005 après 15000 itérations et à 0.001 après 30000 itérations. Un dropout de 0.9 est utilisé pour éviter le sur-apprentissage. Comme précédemment, 15 segments sont utilisés pour l'évaluation.

***Finetuning* :** UCF-101 et HMDB51 sont des bases de données assez petites comparé à Kinetics. Effectuer un finetuning sur ces bases de données est compliqué car avec des paramètres peu adaptés, les avantages du pré-entraînement peuvent être réduits et le sur-apprentissage peut être conséquent. Ainsi, une régularisation L2 avec comme origine les poids pré-entraînés est utilisé et un dropout de 0.9. Les mêmes paramètres (taille du batch et nombre de segments) que pour la classification linéaire sont utilisés. Le pas d'apprentissage initial est de 0.01 et est diminué à 0.005 après 15000 itérations.

Réseaux 3D

Tous les entraînements sont effectués avec Adam que nous avons trouvé plus pratique à utiliser que SGD.

Apprentissage non supervisé : L'entraînement est de 250 époques sur UCF-101 et de 20 époques sur Kinetics. Le pas d'apprentissage est fixé à 10^{-5}.

Classification linéaire : De même que précédemment, 8 segments sont utilisés pour l'apprentissage et 15 pour le test. Le pas d'apprentissage est de 0.001 et un *dropout* de 0.5 est utilisé. L'apprentissage dure 20 époques.

Finetuning : Pour éviter de perturber trop rapidemment les poids pré-entrainés, nous utilisons un pas d'apprentissage différent pour les poids de la couche de classification $p1$ que pour les autres poids, $p2$. Le pas p1 est fixé à 10^{-3} alors que le poids $p2$ augmente de 0 à 10^{-5} lors de l'apprentissage. 100 époques sont effectuées. Le même nombre de segment que pour la classification linéiare est utilisé.

4.3.3 Résultats

Cette partie montre les résultats obtenus par notre méthode. On s'intéresse tout d'abord à la réussite de la tâche non-supervisée. Par la suite, on étudie nos résultats principaux de classification linéaire et *finetuning* sur UCF-101 et HMDB51 qui utilisent les flots optiques comme entrée. D'autres modalités (images et différence d'images) sont analysées par la suite. Une étude d'ablation montre les différents points forts de notre méthode. Différentes visualisations sont montrées pour apprécier les résultats. Finalement, les résultats des autres modèles proposés (sans autorégressivité, bidirectionnel, avec réseaux 3D) sont présentés à des fins de comparaison.

Tâche non-supervisée

La tâche non-supervisée peut être vue comme une tâche de classification où la classe correcte est l'exemple positif et les classes incorrectes les exemples négatifs. Il est donc possible de définir une métrique de précision qui va quantifier la réussite de la tâche non-supervisée. On remarque que plus la prédiction est lointaine, plus la précision est faible car la tâche est plus compliquée. De même, si des exemples négatifs difficiles sont utilisés, ou des transformations sur les entrées sont ajoutées, la précision est beaucoup plus faible (voir figure 4.7). En effet, plus les segments sont rendus dissimilaires, plus la tâche proposée est compliquée. Ceci est bénéfique dans la plupart des approches auto-supervisées. C'est notemment le cas pour notre méthode comme on peut le voir dans l'étude d'ablation 4.8.

Classification linéaire et finetuning avec les flots optiques

Les flots optiques sont très utilisés en reconnaissance d'actions car ils éliminent l'information d'arrière plan peu utile et se concentrent sur le mouvement. Ils donnent d'ailleurs les meilleurs résultats dans nos expériences et nous présentons ces résultats dans la table 4.1. Le pré-entraînement est effectué sur UCF-101 et la méthode est évaluée en classification linéaire et *finetuning*. Les résulats en classification à partir du pré-entraînement donnent de biens meilleurs résultats qu'à partir d'un réseau initialisé aléatoirement. La méthode proposée dépasse les résultats de l'état de l'art publiés dans Arrow of Time (AOT) [WLZF18] de 19.8 p.p sur le 1er *split* d'UCF-101. La précision en classification linéaire atteint presque les performances d'un apprentissage supervisé sur UCF-101. Cela prouve que sans utiliser

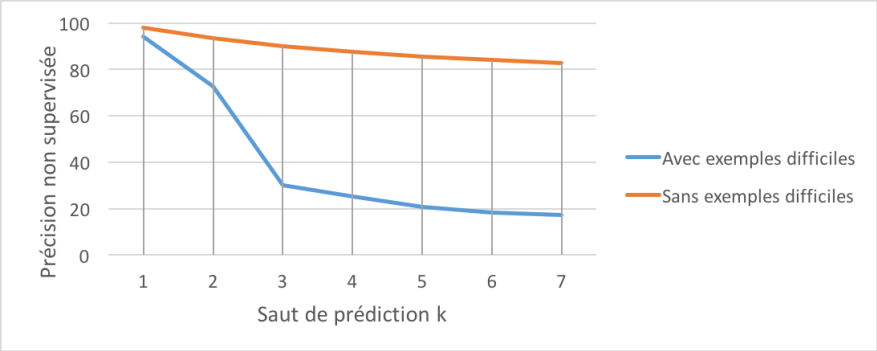


FIGURE 4.7 – Précision non-supervisée (sur la classification entre l'exemple positif et les exemples négatifs) en fonction du pas de prédiction (k lorsque l'on prédit z_{t+k} à partir de c_t). Pour la courbe en bleu, des exemples négatifs difficiles sont utilisés ainsi que des transformations différentes entre les segments et non pour la courbe en orange.

les vérités terrain, nous obtenons des représentations qui sont presque linéairement séparables par rapport aux classes.

Le pré-entraînement améliore la précision par rapport à un apprentissage de zéro sur les bases de données UCF-101 et HMDB51. Les performances sont bien meilleures que les résultats de l'état de l'art présentés dans [WLZF18] sur HMDB51 mais légèrement moins bonnes sur UCF-101.

Apprentissages avec peu de données annotées

Dans cette partie, nous nous intéressons au cas où peu de données annotées sont accessibles. Pour cela, on n'utilise qu'un pourcentage des données accessibles pour chaque classe (18% et 5%) et compare les résultats de notre pré-entraînement par rapport à un apprentissage de zéro dans la figure 4.8. On remarque notemment qu'en classification linéaire, notre méthode a de bien meilleures performances qu'avec un réseau initialisé aléatoirement. En *finetuning*, le pré-apprentissage apporte aussi par rapport à un apprentissage de zéro ce qui montre son efficacité. De plus, la différence de performance entre le *finetuning* et l'apprentissage de zéro augmente quand moins de données annotées sont utilisées. Cette différence atteint 19.1 p.p quand uniquement 5% des données sont utilisées. Il est aussi intéressant de noter qu'en utilisant uniquement 5% des données annotées, on peut obtenir 80% de la précision obtenue par un apprentissage de zéro sur toutes les données annotées.

PRÉ-ENTRAÎNEMENT	FINETUNING	SPLIT1	SPLIT2	SPLIT3	MOYENNE	HMDB51
SANS	NON	21.1	23.6	22.4	22.4	10.3
CPC	NON	**78.4**	**80.1**	**80.6**	**77.9**	**45.3**
AOT [WLZF18]	NON	58.6	X	X	X	X
SANS	OUI	80.5	83.5	82.5	82.2	52.0
CPC	OUI	84.6	88.1	88.4	87.0	**58.9**
AOT [WLZF18]	OUI	**86.3**	**88.6**	**88.7**	**87.9**	55.4
3D ST-PUZZLE [KCK18]	OUI	X	X	X	65.8	33.7
DUAL MOTION GAN[LLDX17]	OUI	55.1	X	X	X	X
VGAN [VPT16]	OUI	X	X	X	52.1	X
SHUFFLE AND LEARN [MZH16]	OUI	50.9	X	X	50.2	18.1

TABLEAU 4.1 – Performance de pré-entraînement sur les différents splits de UCF-101 en utilisant les flots optique. La première partie montre les résultats de classification linéaire et la seconde les résultats de finetuning (délimitées par des lignes doubles). Ces parties sont séparées entre nos résultats et ceux de l'état de l'art. Les résultats sur les 3 splits de UCF-101 et sur la moyenne des 3 splits de HMDB51 sont détaillés comme dans [WLZF18]. La dernière partie montre les résultats des méthodes utilisant les images comme modalité d'entrée.

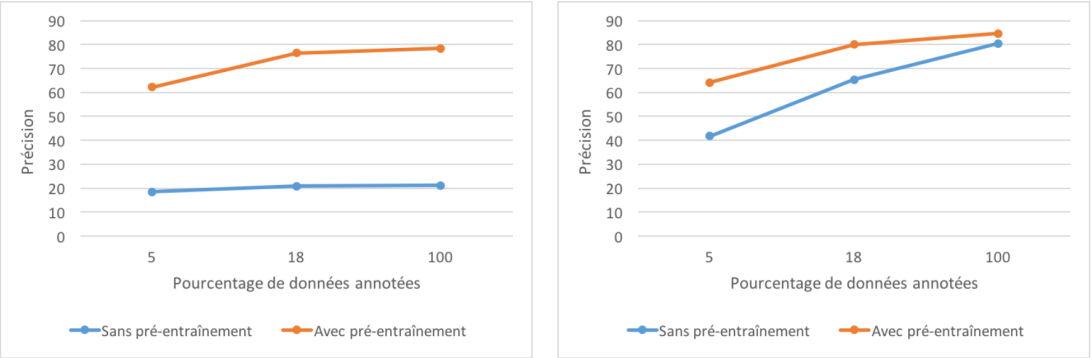


FIGURE 4.8 – Performances de reconnaissance d'actions avec les flots optiques pour différentes quantités de données labélisées utilisées. Les résultats avec et sans pré-entraînement sont comparés. Le premier diagramme montre les résultats en classification linéaire et le deuxième en finetuning

Résultats avec d'autres modalités

Notre méthode donne de bons résultats en utilisant les flots optiques mais ceux-ci peuvent être longs à calculer en pratique. Il est donc intéressant d'utiliser d'autres modalités comme les images ou différence d'images même s'ils donnent de moins bons résultats. Les résultats de ces expériences sont décrits dans la table 4.2. Notre

MODALITÉ	PRÉ-ENTRAÎNEMENT	UCF-101	HMDB51
TVL1	SANS	80.5	52.0
	UCF-101	**84.6**	**58.9**
	AOT	**84.6**	**58.9**
DISFLOW	SANS	76.7	46.7
	UCF-101	**79.2**	**55.6**
DIFF IMAGE	SANS	73.5	37.5
	KINETICS	**78.6**	**46.8**
	UCF-101	77.7	44.6
	ROTATION [JT18]	74.3	42.5
	OPN [LHSY17]	71.4	37.5
IMAGE	SANS	48.6	17.9
	KINETICS	**70.5**	**41.1**
	UCF-101	64.8	34.7
	ROTATION [JT18]	66.0	37.1
	PUZZLE [KCK18]	65.8	33.7
	OPN [LHSY17]	59.8	23.8

TABLEAU 4.2 – Evaluation du pré-entraînement sur différentes modalités et en transférant sur différentes bases de données. Les résultats de l'état de l'art sont présentés à la fin de chaque partie de la modalité.

pré-entraînement améliore les résultats sur toutes les modalités sur les 2 bases de données UCF-101 et HMDB51. On compare au début 2 méthodes différentes de calcul des flots optiques. La méthode TVL1 est plus précise mais le temps de calcul est bien plus important. Les résultats sont cependant bien meilleurs avec TVL1 qu'avec Disflow. Même si les résultats sont moins bons avec les images, le pré-entraînement améliore la précision de beaucoup (29.6 p.p de plus par rapport à un apprentissage de zéro). Un pré-entraînement sur une base de données plus grand donne de meilleurs résultats sur UCF-101 et HMDB51 (Kinetics par rapport à UCF-101). Finalement, notre méthode améliore significativement les résultats de l'état de l'art. Par exemple, nous obtenons 4.3 p.p. avec les différences d'image et 4.0 p.p. avec les images de plus comparé à [JT18] sur HMDB51. Les résultats sont aussi supérieurs sur UCF-101 avec un gain de 4.5 p.p sur les images et de 4.3 p.p sur les différences d'images.

MODALITÉ	UCF-101
FLOT + IMAGE	85.7
IMAGE DIFF + IMAGE	80.6
FLOT + IMAGE DIFF	**86.5**

TABLEAU 4.3 – Evaluation du pré-entraînement en combinant plusieurs modalités

Résultats avec plusieurs modalités

Les résultats précédents utilisent les différentes modalités séparemment. Il est ici proposé de regarder les performances avec notre pré-apprentissage lorsque plusieurs modalités sont utilisées. Les prédictions des différentes modalités sont combinées de la même manière que pour les architectures à 2 voies. Notons ici x^m et x^n 2 modalités différentes d'une vidéo x. On généralise ces notations sur les réseaux appris sur chaque modalité :

$$o^m = softmax(\mathrm{W}_c^m bn(\frac{1}{\mathrm{T}}\sum_{t=1}^{\mathrm{T}} p(f^m(x_t^m,\theta))) + b_c^m)\,.$$

La prédiction finale est alors une combinaison linéaire des prédictions sur chacune des modalités.

$$o = \lambda^m o^m + \lambda^n o^n\,.$$

Comme le montre le tableau 4.3, ajouter une autre modalité améliore significativement les résultats avec des améliorations entre 1 et 2 %. Les meilleurs résultats sont obtenus en combinant les flots optiques avec les différences d'image avec un score final de 86.5 %. L'association des images et différence d'images donnent aussi de très bons résultats et permettent de se passer du calcul des flots optiques qui peut être problématique.

Résultats sans auto-régressivité

Le tableau 4.4 montre les résultats des expériences sans modèle auto-régressif avec les flots optiques sur UCF-101. En comparant au tableau 4.8, nous constatons que l'utilisation des modèles autorégressifs améliore beaucoup les représentations et les performances en reconnaissance d'actions par la suite. Celà montre l'importance d'effectuer des prédictions à différents temps dans la vidéo. Nous remarquons

AGGREGATION	FINETUNING	100	18	5
SANS	NON	21.1	20.9	18.5
LSTM	NON	32.8	31.6	23.4
TCAM	NON	49.6	47.8	34.7
CONV3D	NON	**55.4**	**51.6**	**37.5**
SANS	OUI	76.4	65.3	41.8
LSTM	OUI	78,3	68,7	45,8
TCAM	OUI	**80.5**	70.8	**50.0**
CONV3D	OUI	80	**71.4**	49.5

TABLEAU 4.4 – Résultats de classification linéaire sur UCF-101 après pré-entraînement sur UCF-101 avec le modèle sans autorégressivité. Différents modèles d'aggrégation sont comparés ainsi que sans pré-entraînement. Différents pourcentages de données annotés sont utilisés

MÉTHODE	UCF-101
UNIDIRECTIONNEL	**80.1**
BIDIRECTIONNEL	76.8

TABLEAU 4.5 – Résultats du modèle bidirectionnel en classification linéaire sur les flots optiques

comme pour le modèle principal que le pré-apprentissage améliore les résultats en classification linéaire et en finetuning. De même, le gain de performances augmente quand peu de données labélisées sont utilisées pour le finetuning. Différents réseaux d'aggrégation ont été testé et les meilleurs résultats sont obtenus en utilisant les convolutions 3D.

Résultats du modèle bidirectionnel

Les résultats du modèle bidirectionnel sont présentés dans le tableau 4.5. Malheuresement, les résultats sont moins bons que pour le modèle unidirectionnel pour l'apprentissage sur les flots optiques. Un porblème du modèle bidirectionnel est la nécessité d'utiliser 2 modèles autorégressifs ce qui augmente le nombre de poids à apprendre. De plus, le modèle étant plus lourd en mémoire, nous avons légèrement réduit la taille du batch par rapport à la méthode de référence.

PRETRAIN	RÉSEAU	FINETUNING	UCF-101	HMDB51
RANDOM	RESNET3D	OUI	47.4	21.5
UCF-101	RESNET3D	NON	66.6	36.2
RANDOM	RESNET(2+1)D	OUI	55.8	21.4
UCF-101	RESNET(2+1)D	NON	**67.5**	**37.9**
CBT [SBMS19]	S3D	NON	54.0	29.5
MEMDPC [HXZ20A]	R-2D3D	NON	54.1	30.5

TABLEAU 4.6 – Résultats de reconnaissance d'actions suite au pré-entraînement de CPC en utilisant les améliorations présentées dans la partie 4.2.4 et les réseaux convolutionnels 3D

Résultats avec les réseaux 3D

Les performances en reconnaissance d'actions des modèles entraînés avec les améliorations proposées en partie 4.2.4 sont présentés dans le tableau 4.6. On observe que le pré-entraînement permet d'obtenir de meilleurs performances en classification linéaire q'un modèle entraîné de manière supervisée de zéro pour les deux architectures. Le réseau resnet(2+1)D-18 donne des scores légèrement meilleurs que le resnet3D-18 en classification linéaire et en finetuning. Les résultats en finetuning sont meilleurs que ceux présentés sur le tableau 4.2 pour les images. Celà met en évidence l'intérêt des améliorations proposées ainsi que l'utilisation de réseaux 3D plus adaptés aux données vidéos. Les performances de classification linéaire sont bien meilleures que les résultats des autres approches même si ils utilisent des réseaux plus importants.

Les k plus proches voisins

Cette partie propose une autre méthode d'évaluation des représentations apprises. Il s'agit d'utiliser la méthode des k plus proches voisins sur les représentations apprises. L'avantage de cette méthode est qu'elle ne requiert pas d'apprentissage et n'a qu'un seul paramètre k. Cette méthode ne dépend pas d'augmentations de données de pas d'apprentissage utilisé lors de classification linéaire ou finetuning ce qui rend les comparaisons plus aisées.

Les représentations z_t de différents segments sont moyennés pour obtenir la représentation de la vidéo $r = \sum_t f_\theta(x_t)$. Soit x une vidéo de test, on sélectionne l'ensemble A des K vidéos les plus proches avec la distance L2 sur les représentations :

$$\mathrm{A} = \underset{\mathrm{A}' \in \mathrm{X} \,|\, |\mathrm{A}'|=k}{\arg\max} \sum_{a \in \mathrm{A}'} ||r - \sum_{t} f_\theta(a_t)||^2 .$$

La prédiction $\hat{y}$ est la classe qui est la plus représentée parmi les vidéos de A

$$\begin{aligned} &\mathrm{P}(y = j|\mathrm{X} = x) = \tfrac{1}{k} \textstyle\sum_{i \in \mathrm{A}} \mathrm{I}(y^{(i)} = j) \\ &\hat{y} = \arg\max_j \mathrm{P}(y = j|\mathrm{X} = x) . \end{aligned} \tag{4.6}$$

DATASET		UCF-101				HMDB51			
MODALITÉ	DATASET	1	5	10	20	1	5	10	20
3D	UCF-101	48.4	49.4	49.7	50.1	19.2	21.7	22.3	**24.8**
(2+1)D	UCF-101	49.5	50.2	50.3	50.3	**20.2**	**22.3**	**23.7**	24.7
IMAGE DIFF	KINETICS	40.9	40.0	42.1	42.4				
FLOT	UCF-101	**50.8**	**52.5**	**53.7**	**52.9**				

TABLEAU 4.7 – Résultats de reconnaissance d'actions en utilisant les k plus proches voisins sur les représentations obtenues par apprentissage non-supervisé.

Les résultats de reconnaissance d'actions en utilisant les k plus proches voisins sont présentés en table 4.7. Le modèle CPC utilisant les flots optiques obtient les meilleurs résultats avec plus de 50 % de précision. Le modèle utilisant les réseaux 3D et les nouvelles améliorations a des scores un peu moins bons mais est bien meilleur que la première version sur l'image et un peu meilleur qu'avec les différences d'images. Ces résultats montrent que la représentation apprise de manière non-supervisée peut même être utilisée directement par la suite sans apprentissage avec des performances intéressantes.

Etude d'ablation

L'effet des éléments principaux de notre méthode sur les scores de reconnaissance d'actions sont analysés dans la table 4.8. Nous nous intéressons tout d'abord au nombre de segments futurs et donc à quel point le modèle prédit loin dans le temps. Les performances s'améliorent plus on prédit loin dans le temps. La prédiction de 3 segments futurs améliore les résultats. Le modèle principal prédit à un maximum de 9 segments futurs. L'architecture du modèle auto-régressif est aussi une composante clé du modèle. La table 4.8 montre qu'utiliser un modèle autorégressif plus expressif améliore de beaucoup les résultats. Les convolutions 3D mas-

	Linéaire
Modèle principal	**80.1**
1 prédiction	50.3
2 prédictions	69.9
3 prédictions	77.6
Conv(2+1)D	77.0
ConvLSTM	73.5
LSTM	69.7
LSTM $_{\text{sans}}$ transformation	50.0
LSTM $_{\text{sans}}$ transformation $_{\text{sans}}$ négatifs difficiles	37.9

TABLEAU 4.8 – Etude d'ablation du modèle : La 1ère partie correspond aux résultats du modèle de référence. La 2ème à l'ablation sur le nombre de segments futurs prédits (pour une prédiction, seulement z_{t+1} est prédit à partir de c_t). La 3ème compare les différents modèles autoregressifs. Finalement, les 2 dernières lignes montrent les résultats sans transformations différentes sur les entrées et sans exemples négatifs difficiles. La précision en classification linéaire utilisant la représentation pré-apprise sur UCF-101 avec les flots optiques est présentée (sur le split 2).

quées temporellement donnent les meilleurs résultats devant les Conv(2+1)D masquées et les ConvLSTM. Utiliser un LSTM sur le pooling spatial des cartes de caractéristiques h_t donne de moins bons résultats comme ils ne prennent en considération uniquement l'information temporelle. Finalement, il est crucial pour obtenir de bonnes performances de rendre la tâche difficile. En effet, l'application de différentes transformations sur les segments ainsi que l'utilisation d'exemples négatifs difficiles améliorent les résultats de façon significative comme le montrent les expériences conduites avec le LSTM.

Visualisations t-SNE

On visualise les représentations obtenues par l'apprentissage non-supervisé dans la figure 4.9. Pour celà, de manière similaire à ce qui est fait pour l'évaluation, la moyenne des z_t obtenus à partir de 10 segments de flots optiques sélectionnés à travers la vidéo sont utilisés. On compare ici les visualisations t-SNE entre l'utilisation du réseau pré-entraîné et du réseau initialisé aléatoirement. Pour le réseau pré-entraîné, le regroupement des exemples en classe est plutôt correct. Les classes Ice

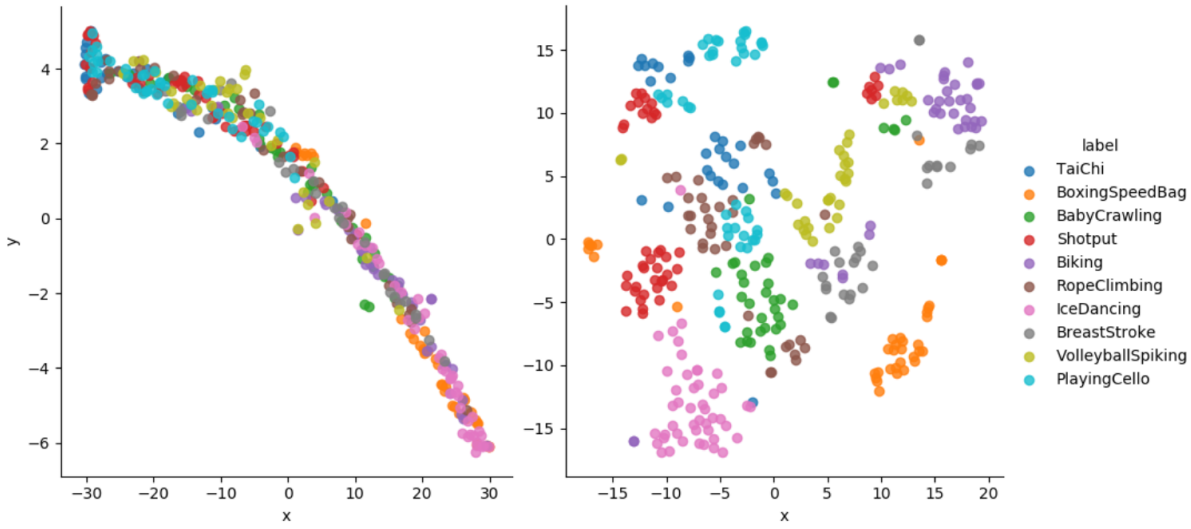


FIGURE 4.9 – Visualisations t-SNE des représentations obtenues à partir des flots optiques sur le test de UCF-101. Seulement 10 classes sur les 101 sont présentées pour une meilleure visualisation. La figure gauche montre la visualisation pour un réseau initialisé aléatoirement et la figure droite avec notre pré-entraînement.

Dancing, Boxing Speed Bag, Biking et Baby Crawling sont bien séparées des autres classes. Au contraire, la visualisation pour le réseau non-entraîné ne donne aucune information et ne sépare pas les classes correctement. Plus de visualisations avec les différentes modalités ainsi que différents temps d'apprentissage non-supervisé sont présentés en annexe.

Recherche de vidéos

Les représentations obtenues à la suite du pré-entraînement peuvent aussi être utilisées pour de la recherche de vidéos. Etant donné une vidéo requête, on s'intéresse à retourner les vidéos les plus similaires. Pour celà, on utilise les mêmes représentations que pour la classification avec les K plus proches voisins. Cependant, on utilise la similarité cosinus pour classer les différentes vidéos. Il est possible d'évaluer numériquement les résultats de recherche obtenus. Pour celà, on se met dans le même protocole que [JMF20]. Les vidéos requêtes sont sélectionnées parmi les vidéos de test et les vidéos retournées parmi celles d'entraînement. On considère différents nombres de vidéos retournées K. Pour l'évaluation, on considère que la réponse est correcte si au moins une vidéo retournée appartient à la même classe

que la vidéo requête. Les résultats sont présentés dans la table 4.9.

Dataset		UCF-101				HMDB51			
Modalité	Dataset	1	5	10	20	1	5	10	20
3D	UCF-101	51.8	67.7	75.3	82.5	**21.5**	45.3	**59.7**	**73.1**
(2+1)D	UCF101	**54.5**	69.3	76.9	83.2	21.1	**45.7**	**59.7**	72.3
Image diff	Kinetics	41.5	62.9	72.0	79.9				
Flot	UCF101	52.5	**75.0**	**82.1**	**88.7**				
MemDPC [HXZ20a]	UCF101	20.2	40.4	52.4	64.7	7.7	25.7	40.6	57.7
Clip Order [AME18]	UCF101	14.1	30.3	40.0	51.1	7.6	22.9	34.4	48.8
SpeedNet [BEL+20]	Kinetics	13.0	28.1	37.5	49.5				
Temp Trans [JMF20]	UCF101	26.1	48.5	59.1	69.6				

TABLEAU 4.9 – Scores de recherche de vidéos basés sur les représentations obtenues par pré-apprentissage non-supervisé.

Les meilleurs résultats sont obtenus avec l'utilisation des flots optiques même si les résultats sur les images avec les réseaux 3D sont très proches. Les méthodes que l'on propose obtiennent toutes des meilleurs résultats que l'état de l'art. Ceci montre un intérêt supplémentaire à notre méthode d'apprentissage non-supervisé de représentations de vidéos.

Dans la figure 4.10, les 3 vidéos les plus similaires par rapport à une vidéo requête sont présentées. Les représentations sur les flots optiques sont utilisées ici. Les vidéos retournées sont très similaires à la vidéo requête (toutes des vidéos d'escrime) ce qui illustre bien les performances de notre méthode en recherche de vidéo. Les recherches de vidéo pour d'autres modalités et d'autres vidéos requête sont proposées en annexe 8.1.4.

4.4 Conclusion

Cette partie décrit l'utilisation de la méthode *Contrastive Predictive Coding* pour prédire le futur de vidéos de manière latente. Elle permet d'obtenir des représentations utiles pour la tâche de reconnaissance d'actions. On montre notemment qu'on obtient des performances meilleures qu'avec un apprentissage de zéro sur différentes modalités et des résultats compétitifs avec l'état de l'art. Nous avons mis en évidence les points principaux de la méthode qui sont : l'utilisation de modèles autorégressifs expressifs de manière à prédire des segments à chaque moment, l'uti-

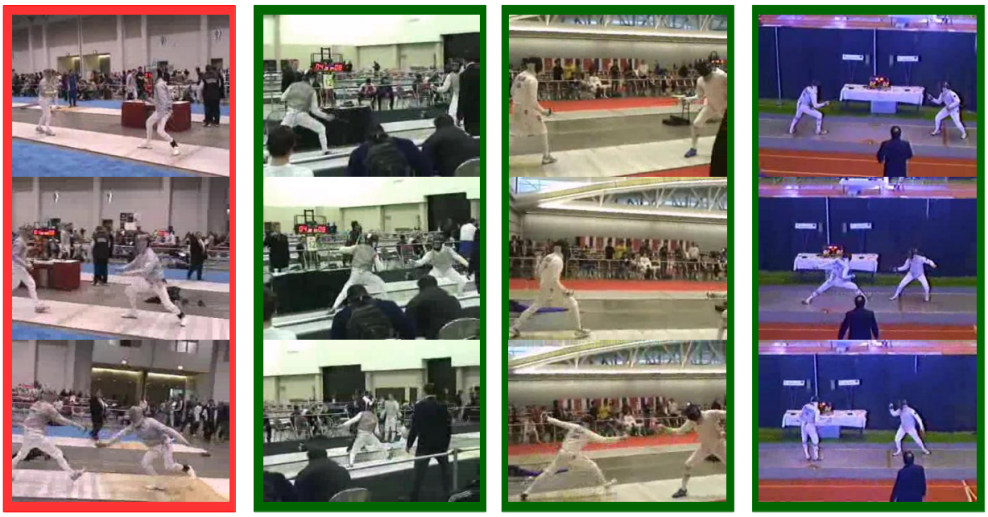


FIGURE 4.10 – Vidéos les plus similaires à la requête en utilisant la distance cosinus sur les représentations apprises de manière non supervisée. Les flots optiques sont utilisés et nous considérons les représentations comme la moyenne des valeurs z_t sur 10 segments. La requête est en rouge et les résultats sont en vert.

lisation d'exemples négatifs difficiles et de transformations importantes sur les entrées. Finalement, nous avons mis en évidence l'avantage d'utiliser des réseaux 3D ainsi que différentes améliorations proposées dans les articles SimCLR et CPCv2 comme la normalisation des z par exemple.

Différentes améliorations peuvent encore être apportées à cette méthode. Les meilleurs résultats étant obtenus à partir des flots optiques, il peut être intéressant de les calculer de manière implicite dans le réseau de la même manière que [FHG+18] pour éviter de les précalculer. La prédiction se fait uniquement de manière temporelle et à un pas de temps régulier. Il peut être intéressant d'utiliser des prédictions spatio-temporelles et à plusieures échelles.

Chapitre 5

Utilisation des modèles de langage

Sommaire

5.1 Méthodes de langage 88
5.1.1 Transformer 88
5.1.2 Apprentissage non-supervisé de représentations en NLP . . 89
5.2 Modèles de langage contrastifs 92
5.2.1 Utilisation de BERT 92
5.2.2 Utilisation de XLNet 94
5.3 Applications 95
5.3.1 Application aux séquences d'images 96
5.3.2 Application aux vidéos 97
5.4 Conclusion 103

Afin de résoudre le problème d'apprentissage de représentation, l'approche vue au quatrième chapitre utilise un modèle autorégressif pour agréger l'information du passé afin de prédire les segments futurs. Les prédictions se font donc dans un seul sens prédéfini, ce qui n'est pas optimal. Dans le domaine du traitement du langage naturel (NLP), les approches bidirectionnelles ont permis d'améliorer significativement les résultats. C'est notamment le cas dans le cadre du pré-entraînement non supervisé avec la méthode *BERT* [DCLT19]. Nous présentons dans ce chapitre comment l'appliquer à la vidéo. *BERT* a pour but de prédire des mots masqués à partir d'autres mots de contexte. Nous l'associons à une fonction de coût contrastive pour gérer le domaine continu des vidéos. Nous nous intéressons particulièrement à la stratégie de masquage qui joue un rôle important dans l'adaptation pour les vidéos. Finalement, nous montrons que cette méthode permet d'obtenir des résultats supé-

rieurs à l'état de l'art en classification linéaire pour la reconnaissance d'actions ainsi qu'en recherche de vidéos. Nous présentons tout d'abord l'architecture *Transformer* utilisée par les méthodes *BERT* et *XLNet* (une amélioration de BERT [YDY+19]). Nous nous intéressons ensuite à la formulation de ces méthodes avec une fonction de coût contrastive. Des expériences sur des séquences de chiffres et sur les vidéos sont finalement présentées.

5.1 Méthodes de langage

5.1.1 Transformer

Le *transformer* est un modèle initialement introduit en 2017 dans le domaine du traitement des langues [VSP+17], très utile par exemple pour les tâches de traduction. Il traite des séquences notées $z_1, \ldots, z_\mathrm{T}$ avec $z_t \in \mathbb{R}^\mathrm{D}$. On notera la correspondance matricielle :

$$\mathrm{Z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_\mathrm{T} \end{bmatrix} \in \mathbb{R}^{\mathrm{T}\times\mathrm{D}}.$$

Le *transformer* est basé sur la notion d'auto-attention (*self-attention*). L'attention est utilisée pour modéliser des dépendances à long-terme difficilement réalisables par d'autres opérations plus locales comme les convolutions. On note q, k et v des fonctions apprises (respectivement requête, clé et valeur). L'attention est modélisée à travers un score calculé entre chaque couple d'éléments de la séquence : $s_{i,j} = \frac{\exp\left(q(z_i)^\mathrm{T} k(z_j)\right)}{\sum_m \exp\left(q(z_i)^\mathrm{T} k(z_m)\right)}$. La sortie est la somme pondérée par les scores obtenus d'une représentation de chaque élément de la séquence : $o_i = \sum_j s_{i,j}\, v(z_j)$. Le transformer utilise une attention normalisée (les scores s sont normalisés en fonction de la dimension de $q(z_i)$). La formulation matricielle est la suivante :

$$\mathrm{attention}(\mathrm{Q},\mathrm{K},\mathrm{V}) = \mathrm{softmax}\Big(\frac{\mathrm{QK}^\mathrm{T}}{\sqrt{d_\mathrm{K}}}\Big)\mathrm{V}, \tag{5.1}$$

où Q, K et V sont les notations matricielles de $q(z_1), \ldots, q(z_\mathrm{T})$, $k(z_1), \ldots, k(z_\mathrm{T})$ et $v(z_1), \ldots, v(z_\mathrm{T})$ et d_K est la dimension des $k(z_i)$ et $q(z_i)$.

Nous détaillons ci-dessous l'architecture du réseau. En premier lieu, un encodage positionnel noté pos est ajouté à l'entrée afin que le réseau ait une information

sur la position de chaque élément :

$$\mathrm{Z}_{pos} = \mathrm{Z} + \mathrm{pos}\,,$$

avec pos une matice de poids apprise de taille T × D. Les opérations de normalisation et de décrochage (*layernorm* et *dropout*) ne sont pas précisées ici pour ne pas surcharger les notations.

Notons les représentations intermédiaires H^l avec l le numéro de la couche. On a donc $\mathrm{H}^0 = \mathrm{Z}_{pos}$. A chaque couche, plusieurs opérations d'attention sont utilisées simultanément et les sorties sont concaténées [1] :

$$\begin{aligned} \mathrm{I}_i^l &= \mathrm{attention}(\mathrm{H}^{l-1}\mathrm{W}_i^{\mathrm{Q}}, \mathrm{H}^{l-1}\mathrm{W}_i^{\mathrm{K}}, \mathrm{H}^{l-1}\mathrm{W}_i^{\mathrm{V}}) \\ \mathrm{I}^l &= \mathrm{concat}(\mathrm{I}_i^l; i \in [1, h])\,. \end{aligned} \tag{5.2}$$

Finalement, un réseau complètement connecté FFN est appliqué à la sortie ainsi qu'une connexion résiduelle :

$$\begin{aligned} \mathrm{FFN}(x) &= \max(0, x\mathrm{W}_1 + b_1)\mathrm{W}_2 + b_2 \\ h_t^l &= \mathrm{FFN}(i_t^l) + i_t^l\,. \end{aligned} \tag{5.3}$$

De même, les vecteurs h_t^l et i_t^l proviennent des matrices H^l et I^l et les opérations de *layernorm* et *dropout* sont ignorées.

5.1.2 Apprentissage non-supervisé de représentations en NLP

Le domaine du traitement des langues est le premier à démontrer que des méthodes d'apprentissage non-supervisé de représentations permettent d'améliorer les performances de l'état de l'art sur différentes tâches. On peut notamment citer l'apprentissage de représentations de mots (*word embedding*) ou de phrases (on s'intéresse plus particulièrement à ces derniers). La plupart de ces méthodes sont des modèles de langage et se basent sur la prédiction des mots suivants. Considérons une séquence $x_1, \ldots, x_\mathrm{T}$, le but est d'estimer la probabilité jointe :

$$p(x_1, \ldots, x_\mathrm{T}) = \prod_{t=1}^{\mathrm{T}} p(x_t | x_1, \ldots, x_{t-1})\,. \tag{5.4}$$

1. concat indique ici l'opération de concaténation selon les canaux (deuxième dimension sans prendre en compte le *batch*

Ces modèles n'utilisent malheureusement l'information que dans une seule direction. BERT [DCLT19] propose une solution à ce problème en utilisant un système de masquage et le modèle *transformer*. Nous étudions cette méthode plus en détail ainsi qu'une amélioration XLNet dans les parties suivantes.

BERT

Le modèle BERT propose de prédire des mots de la phrase qui ont été masqués à partir des autres. Notons $\bar{x}$ les mots masqués et $\hat{x}$ les mots non masqués. En prenant $m_t = 0$ pour un mot masqué et 1 autrement, le but est de modéliser :

$$\log p_\theta(\bar{x}|\hat{x}) \approx \sum_{t=1}^{\mathrm{T}} (1 - m_t) \log p_\theta(x_t|\hat{x}) \,. \tag{5.5}$$

Les mots masqués sont sélectionnés aléatoirement. L'intérêt de cette formulation est que les mots masqués sont prédits à partir de tous les mots non masqués et non uniquement à partir des mots précédents. Cela permet une modélisation bidirectionnelle qui donne une meilleure représentation.

Voyons plus en détail les différentes étapes du modèle dans le cadre du traitement des langues. Considérons une séquence de mots $x_1, \ldots, x_\mathrm{T}$. 15% des mots sont masqués aléatoirement par le symbole [MASK] de la façon suivante :

$$\begin{aligned} &m_1, \ldots m_\mathrm{T} \sim \mathscr{B}(0.85) \\ &x_t^m = \begin{cases} x_t, & \text{si } m_t = 1 \\ [\mathrm{MASK}], & \text{sinon} \end{cases}, \end{aligned} \tag{5.6}$$

avec $\mathscr{B}$ une loi de Bernouilli. Dans la méthode originale, les mots sont soit masqués soit remplacés par un autre mot aléatoirement. La séquence est ensuite plongée lexicalement en associant à chaque mot un vecteur (*word embedding*), on note cette séquence : $w_1, \ldots, w_\mathrm{T}$. Les dépendances de cette séquence sont modélisées par un *transformer*. Une couche dense de poids W_c et de biais b_c va finalement prédire les mots masqués :

$$\begin{aligned} &o_1, \ldots, o_\mathrm{T} = \mathrm{transformer}(w_1, \ldots, w_\mathrm{T}) \\ &p_t = \mathrm{softmax}(\mathrm{W}_c o_t + b_c) \,. \end{aligned} \tag{5.7}$$

La fonction de coût utilisée est la somme de l'entropie croisée sur les mots masqués. Notons $e(x_t)$ le *one hot encoding* du mot x_t, la fonction de coût s'écrit de la

forme suivante :

$$\mathrm{L} = \sum_t (1 - m_t) e(x_t)^{\mathrm{T}} \log(p_t)\,. \tag{5.8}$$

Pour simplifier ici, nous ne détaillons pas la tâche de prédiction de la phrase suivante qui ne nous intéressera pas par la suite. Cette fonction de coût est utilisée pour apprendre les poids du *transformer* sur un corpus de texte non annoté de taille importante. Par la suite, ces poids peuvent être transféré sur différentes tâches et une amélioration significative est observée par rapport à un apprentissage de zéro.

XLNet

XLNet propose d'améliorer la méthode BERT en s'attaquant à différents problèmes de la méthode précédente. Le premier est que le modèle utilise une approximation $\log p_\theta(\bar{x}|\hat{x}) \approx \sum_{t=1}^{\mathrm{T}} (1 - m_t) \log p_\theta(x_t|\hat{x})$ qui n'est pas forcément justifiée. De plus, la méthode introduit un caractère spécial [MASK] supplémentaire qui n'apparaît pas dans les textes et donc dans les tâches d'évaluation. Finalement, contrairement aux modèles de langage, BERT n'est pas autorégressif. Afin que le modèle soit en même temps bidirectionnel et autorégressif, ils proposent de prédire le mot suivant mais selon une permutation aléatoire. L'objectif est le suivant :

$$\max_\theta \mathbb{E}_{\sigma \in \Sigma^{\mathrm{T}}} \log p_\theta\left(x_{\sigma(t)} | x_{\sigma(<t)}\right). \tag{5.9}$$

Cependant, cette modélisation est impossible à réaliser en utilisant l'architecture classique du transformer. En effet, pour une couche intermédiaire $o_{\sigma(t)}$, le réseau ne peut pas avoir d'information sur $x_{\sigma(t)}$ car cette valeur doit être prédite. Mais $o_{\sigma(t)}$ doit avoir l'information sur $x_{\sigma(t)}$ pour la prédiction des mots suivants selon la permutation : $x_{\sigma(>t)}$. C'est pourquoi ils ont recours à une architecture à 2 voies : la voie du contenu h et la voie de la requête r. $h_{\sigma(t)}$ dépend de $x_{\sigma(t)}$ et ne servira donc pas à prédire celui-ci, mais les mots suivants. Au contraire, $r_{\sigma(t)}$ ne dépend pas de $x_{\sigma(t)}$ et sera utilisé pour faire la prédiction.

Considérons une séquence $x_1, \ldots, x_{\mathrm{T}}$ et leur encodage positionnel $q_1, \ldots, q_{\mathrm{T}}$. Prenons une permutation σ et fixons t_{pred} l'indice à partir duquel on effectue les prédictions. Les 2 voies h et r sont initialisées de la manière suivante :

$$\begin{aligned} h^0_{\sigma(t)} &= q_{\sigma(t)} + \mathbb{1}_{t<t_{pred}} x_{\sigma(t)} \\ r^0_{\sigma(t)} &= q_{\sigma(t)} + x_{\sigma(t)}\,. \end{aligned} \tag{5.10}$$

Ainsi, la voie h a accès au contenu de x_t uniquement si on n'effectue pas de prédiction pour l'indice t. Les couches suivantes sont construites de la manière suivante :

$$\begin{aligned} r_{\sigma(t)} &= \text{attention}\big(r^{l-1}_{\sigma(t)}\mathrm{W}^{l-1}_q, h^{l-1}_{\sigma(<t)}\mathrm{W}^{l-1}_k, h^{l-1}_{\sigma(<t)}\mathrm{W}^{l-1}_v\big) \\ h_{\sigma(t)} &= \text{attention}\big(h^{l-1}_{\sigma(t)}\mathrm{W}^{l-1}_q, h^{l-1}_{\sigma(\leq t)}\mathrm{W}^{l-1}_k, h^{l-1}_{\sigma(\leq t)}\mathrm{W}^{l-1}_v\big)\,. \end{aligned} \tag{5.11}$$

De cette manière, $r_{\sigma(t)}$ n'a pas accès à l'information provenant de $x_{\sigma(t)}$ mais $h_{\sigma(t)}$ y a accès. La dernière couche g est utiliseé avant une couche dense (de poids W_c et de biais b_c) pour la prédiction des mots

$$p_{\sigma(t)} = \text{softmax}(\mathrm{W}_c r_{\sigma(t)} + b_c)\,. \tag{5.12}$$

Le coût est la somme des entropies croisées sur les mots $x_{\sigma(t)}$ tels que $t > t_{pred}$:

$$\mathrm{L} = \sum_t \mathbb{1}_{t>t_{pred}} e\big(x_{\sigma(t)}\big)^{\mathrm{T}} \log\big(p_{\sigma(t)}\big)\,. \tag{5.13}$$

5.2 Modèles de langage contrastifs

Dans le cadre du NLP, l'élément de base est discret (un mot est considéré comme un indice dans le vocabulaire). Cependant, ce n'est pas le cas dans les applications à la vision. On propose ici une manière d'appliquer ces méthodes au domaine de la vision à l'aide d'une fonction de coût contrastive. Les formulations de la méthode pour BERT et pour XLNet sont détaillées ci-dessous.

5.2.1 Utilisation de BERT

La méthode est illustrée dans la figure 5.1. Notons $x_1, \ldots, x_{\mathrm{T}}$ une séquence d'éléments d'un exemple. Celà peut être des segments successifs d'une vidéo ou encore différents patchs d'une image. Un CNN f est utilisé pour encoder chaque élément en une représentation : $h_t = f(x_t)$. Chaque segment vidéo x_t est associé à un masque binaire $m_t \in \{0,1\}$. Les éléments masqués h_t (avec $m_t = 0$) sont prédits à partir des éléments non masqués (avec $m_t = 1$). La séquence masquée est définie de la manière suivante : $h^m_t = m_t * h_t$. Le *transformer* encode cette séquence afin de prédire les éléments masqués :

$$o_1, \ldots, o_{\mathrm{T}} = \text{transformer}(h^m_1, \ldots, h^m_{\mathrm{T}})\,. \tag{5.14}$$

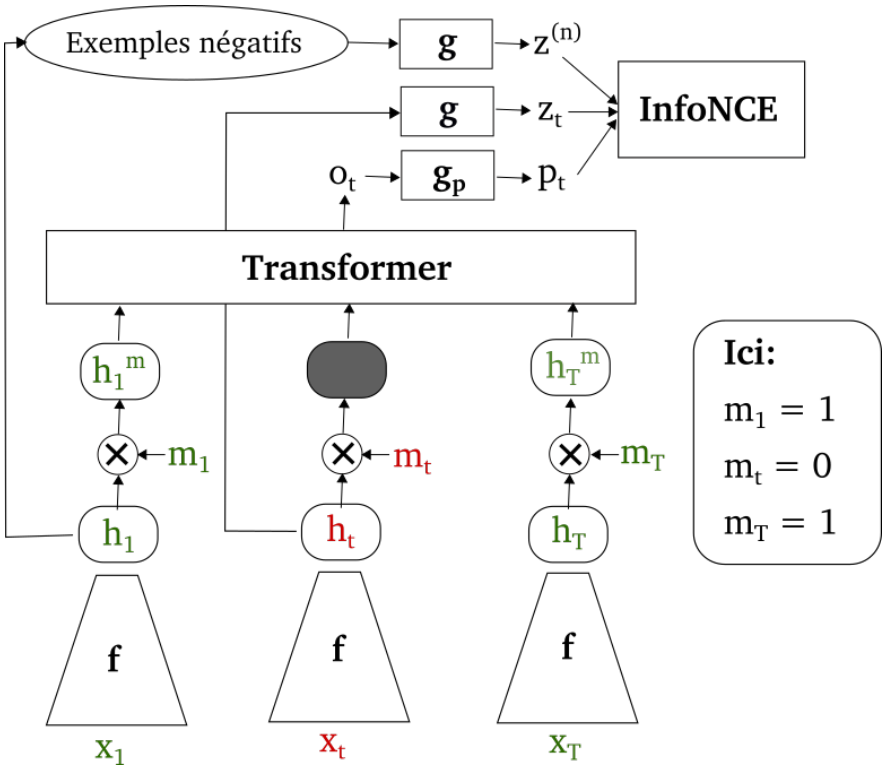


FIGURE 5.1 – Schéma représentant la méthode BERT associé à la fonction de coût InfoNCE. Dans cet exemple, l'élément x_t est masqué ($m_t = 0$) et va être prédit à partir des autres éléments. La prédiction p_t est claculée à partir de la sortie o_t du transformer. L'exemple positif est alors z_t et les exemples négatifs sont les autres éléments de la séquence (comme h_1 et h_T par exemple) mais aussi du *batch*

.

Deux réseaux multicouches (MLP) g_p et g sont utilisés respectivement pour la prédiction et pour réduire la dimension des représentations h_t :

$$\begin{aligned} p_t &= g_p(o_t) \\ z_t &= g(h_t)\,. \end{aligned} \tag{5.15}$$

La fonction de coût InfoNCE maximise la similarité entre les prédictions p_t et l'exemple positif z_t et la minimise entre la prédiction et les exemples négatifs $z^{(n)}, n = 1, \ldots, \mathrm{N}$

$$\mathrm{L}_{\mathrm{NCE}}(p_t, z_t) = -\log \frac{\exp(p_t^\top z_t)}{\exp(p_t^\top z_t) + \sum_{n=1}^{\mathrm{N}} \exp(p_t^\top z^{(n)})}\,. \tag{5.16}$$

La fonction de coût NCE est ensuite utilisée sur les éléments masqués :

$$\mathrm{L} = \sum_{t=1}^{\mathrm{T}} (1 - m_t)\mathrm{L}_{\mathrm{NCE}}(p_t, z_t)\,. \tag{5.17}$$

Une fois le pré-apprentissage effectué, le CNN f peut être utilisé pour différentes tâches.

5.2.2 Utilisation de XLNet

Cette section reprend les notations de la partie 5.1.2 sur XLNet. La méthode est illustrée dans la figure 5.2. La séquence $x_1, \ldots, x_{\mathrm{T}}$ est encodée de la même manière que dans la partie précédente 5.1 en éléments $h_1, \ldots, h_{\mathrm{T}}$. La première couche du réseau XLNet est composée des h_t avec l'encodage positionnel :

$$\begin{aligned} h^0_{\sigma(t)} &= q_{\sigma(t)} + \mathbb{1}_{t<t_{pred}} h_{\sigma(t)} \\ r^0_{\sigma(t)} &= q_{\sigma(t)} + h_{\sigma(t)}\,. \end{aligned} \tag{5.18}$$

Les réseaux MLP introduits précédemment sont utilisés de la même manière, la prédiction s'effectue à partir de la dernière couche de la branche des requêtes :

$$\begin{aligned} p_{\sigma(t)} &= g_p\big(r^{\mathrm{L}}_{\sigma(t)}\big) \\ z_{\sigma(t)} &= g\big(h_{\sigma(t)}\big)\,. \end{aligned} \tag{5.19}$$

La fonction de coût InfoNCE n'est pas appliquée pour les t_{pred} premières valeurs

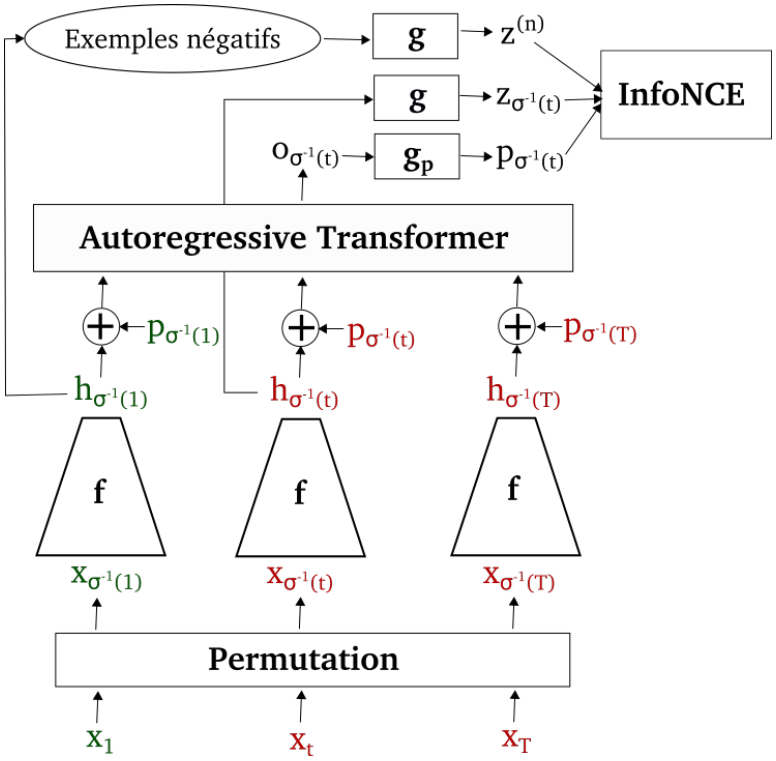


FIGURE 5.2 – Schéma représentant la méthode XLNet associée à la fonction de coût InfoNCE. La séquence est permutée aléatoirement. L'élément suivant est prédit à partir des éléments précédents dans l'ordre de la permutation. Un encoding positionnel est ajouté et l'architecture à 2 voies de XLNet est utilisée. La sortie $o_{\sigma^{-1}(t)}$ est utilisée pour effectuer la prédiction $p_{\sigma^{-1}(t)}$. La fonction InfoNCE est utilisée pour comparer la prédiction avec $z_{\sigma^{-1}(t)}$ et les exemples négatifs.

de la permutation :

$$\mathrm{L} = \sum_t \mathbb{1}_{t>t_{pred}} \mathrm{L}_{\mathrm{NCE}}\left(p_{\sigma(t)}, z_{\sigma(t)}\right). \tag{5.20}$$

5.3 Applications

On considère dans cette partie l'application des méthodes BERT et XLNet au pré-entraînement sur des séquences d'images de chiffres ainsi que des vidéos. L'application sur les séquences d'images de chiffres a pour but de vérifier la méthode sur une application assez proche du texte. La méthode n'a pas besoin d'être adaptée pour obtenir de bonnes performances. La méthode sur la vidéo est adaptée à la structure des données d'entrée au niveau du masque. Ces différentes tâches sont

Méthode	Longueur des séquences	Acc train	Acc test
Aléatoire		32.9	33.8
Supervisé		**98.4**	**98.2**
BERT	6	95.4	95.5
BERT	10	95.5	95.7
XLNet	6	95.2	95.4
XLNet	10	96.2	96.5

TABLEAU 5.1 – Resultats de classification linéaire après un pré-entraînement avec BERT et XLNet comparé à l'apprentissage supervisé

décrites dans les parties suivantes.

5.3.1 Application aux séquences d'images

Description de la tâche

On considère des séquences consécutives de chiffres (1, 2, 3, 4 ou encore 7, 8, 9, 0). Pour chaque chiffre, on choisit aléatoirement une image de ce chiffre dans la base MNIST. Cette image est transformée en ajoutant un fond étant un bout aléatoire d'une image. Le but est de retrouver les chiffres manquants dans la séquence. Dans le cas de BERT, certains chiffres de la séquence sont masqués et devront être retrouvés. Dans le cas de XLNet, les chiffres sont mélangés selon une permutation aléatoire. Le modèle devra prédire le chiffre suivant à partir des précédents selon l'ordre défini par la permutation. Cette tâche de pré-entraînement est ensuite évaluée sur la classification des images de chiffres MNIST.

Résultats

Les méthodes décrites précédemment sont utilisées pour pré-apprendre un réseau de classification de chiffres. Le tableau 5.1 montre que les pré-apprentissages non-supervisés utilisant BERT et XLNet permettent d'obtenir des performances de classification linéaire bien meilleures qu'avec un réseau initialisé aléatoirement et proches des performances obtenues par apprentissage supervisé. Cela prouve que les méthodes BERT et XLNet associées à la fonction de coût InfoNCE permettent d'obtenir des représentations intéressantes et de haut niveau sémantique. Les visualisations t-SNE présentées dans la figure 5.3 le confirment. En effet, les repré-

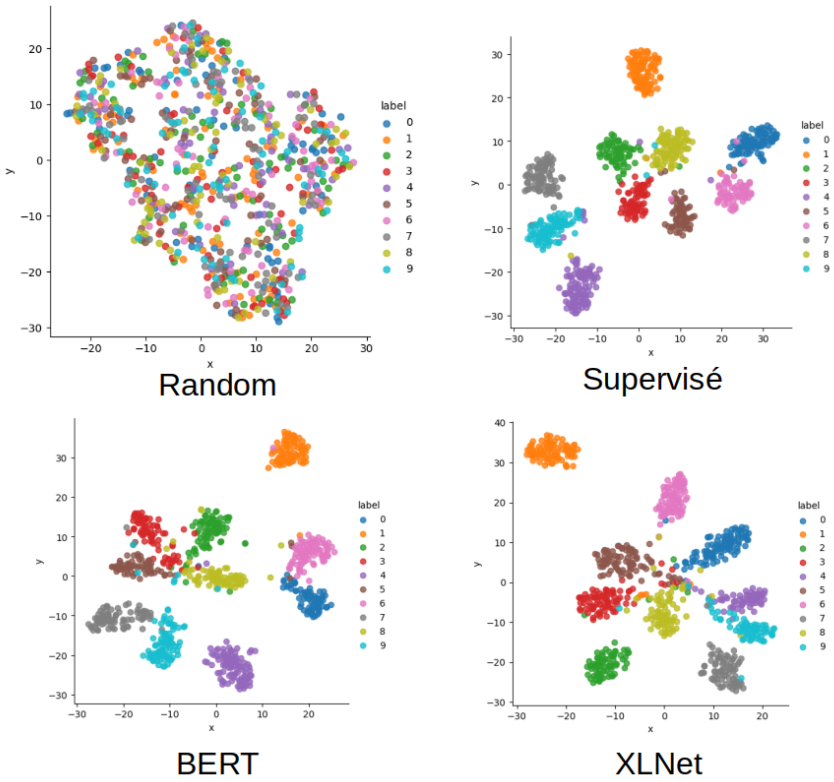


FIGURE 5.3 – Visualisations t-SNE des représentations obtenues par BERT et XLNet comparées au supervisé et à un réseau initialisé aléatoirement

sentations obtenues sont bien regroupées par classes et sont organisées de manière similaire à celles obtenues en supervisé.

5.3.2 Application aux vidéos

Description de la méthode

Nous proposons une nouvelle méthode ayant pour but de prédire différents segments de la vidéo (segments masqués) à partir des autres segments de la vidéo. Cette méthode a des similitudes avec la méthode CPC pour la vidéo présentée dans la section précédente mais est inspirée de BERT et non de CPC. Elle se base sur un modèle *transformer* pour l'agrégation du contexte et les prédictions comme vu précédemment. Un désavantage par rapport au modèle CPC est que le modèle d'agrégation n'est pas autorégressif et l'impossibilité d'utiliser des cartes de caractéristiques (en entrée du *transformer*). Cependant, le transformer est plus souple que

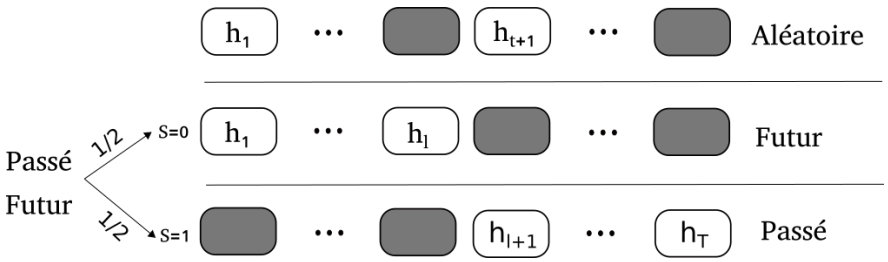


FIGURE 5.4 – Schéma représentant les différentes stratégies de masquage pour la vidéo

les modèles autorégressifs masqués et permet ainsi de choisir différentes configurations pour le masquage. De plus, les prédictions sont effectuées directement en sortie du transformer ce qui permet d'avoir plus d'expressivité. Nous décrivons plus en détail la méthode ci-dessous.

La vidéo est découpée temporellement en T segments $x_1, \ldots, x_T$ de F images. De la même manière que pour la méthode CPC Vidéo décrite dans le quatrième chapitre, les segments sont sélectionnés de manière à se chevaucher de la moitié de leur longueur et les images sont concaténées au niveau des canaux. Le terme image décrit ici différentes modalités : des images RGB, des différences temporelles de celles-ci ou encore des flots optiques. A partir de cette séquence, la méthode BERT NCE décrite précédemment est appliquée. Le réseau f utilisé est un réseau de convolution 2D.

Choix du masquage

Le masquage dans la méthode BERT est effectué de manière aléatoire. Typiquement, un mot est masqué avec 15% de chance. Cependant, ce type de masquage a un inconvénient important lorsqu'il est appliqué à la vidéo. En effet, un élément masqué x_t serait la plupart du temps entouré d'éléments non masqués (x_{t-1} et x_{t+1}). Ces éléments étant très similaires, la tâche non-supervisée devient alors assez facile ce qui est néfaste pour la qualité de la représentation. De plus, nous avons montré dans la méthode précédente que la prédiction des segments sur le long terme était nécessaire pour apprendre de bonnes représentations sur les vidéos.

C'est pourquoi nous proposons deux autres stratégies de masquage illustrées sur la figure 5.4. La première prédit les segments futurs à partir des précédents. Elle est nommée « futur ». Le nombre d'éléments passés visibles n'est pas fixe et varie à chaque itération d'entraînement, il est noté ℓ. De manière plus concrète, ℓ est tiré

aléatoirement selon une loi uniforme $\mathscr{U}$ et prend ses valeurs entre 1 et T − 1. Le nombre de segments futurs à prédire est alors T − ℓ. Le masque est défini de la façon suivante :

$$m_t = \mathbb{1}_{\{t \leq \ell\}} \quad \text{avec} \quad \ell \sim \mathscr{U}[1; \mathrm{T}-1]\,. \tag{5.21}$$

De la même manière, il est possible de s'intéresser à la prédiction des segments passés. Les résultats de ce type de masquage ne sont pas présentés ici car cette stratégie est assez similaire avec la précédente (« futur ») et moins conventionnelle. Par contre, la deuxième stratégie prédit soit le passé soit le futur et est nommée « passé/futur ». A certaines itérations d'entraînement, le passé est masqué et le futur est masqué pour les autres. Plus théoriquement, on tire une variable de Bernouilli S pour le déterminer. Comme précédemment, la limite ℓ est aussi sélectionnée aléatoirement :

$$\begin{aligned} &m_t = (1-\mathrm{S})\,\mathbb{1}_{\{t \leq \ell\}} + \mathrm{S}\,\mathbb{1}_{\{t > \ell\}}\,, \\ &\text{avec} \quad \ell \sim \mathscr{U}[1; \mathrm{T}-1] \quad \text{et} \quad \mathrm{S} \sim \mathscr{B}(0.5)\,. \end{aligned} \tag{5.22}$$

Ces différentes techniques de masquage sont évaluées dans les expériences présentées par la suite. Des masquages plus complexes pourraient être utilisés comme une combinaison du masquage aléatoire et « passé/futur » mais ils ne sont pas étudiés dans ce mémoire.

Expériences

Les bases de données utilisées sont les mêmes que pour CPC pour la vidéo : UCF-101 et HMDB51. Tous les pré-apprentissages sont effectués sur UCF-101. Les modèles sont ensuite évalués sur UCF-101 et HMDB51. Le protocole d'évaluation est décrit dans la partie suivante.

Evaluation Une fois les représentations apprises de manière non-supervisée, elles sont évaluées sur la tâche de reconnaissance d'actions et de recherche de vidéos. La reconnaissance d'actions est effectuée par classification linéaire et par la méthode des K plus proches voisins. Les modalités précises de toutes les évaluations sont décrites dans les parties 4.2.5 et 4.3.3.

MÉTHODE	MODALITÉ	UCF-101	HMDB51
OUR ALÉATOIRE	FLOT	69.2	40.1
OUR FUTUR	FLOT	75.1	45.7
OUR PASSÉ/FUTUR	FLOT	**78.7**	**47.4**
OUR PASSÉ/FUTUR	IMAGE DIFF	67.4	33.2
OUR PASSÉ/FUTUR	IMAGE	56.3	28.9
CBT [SBMS19]	IMAGE	54.0	29.5
MEMDPC [HXZ20A]	IMAGE + FLOT	58.5	33.6
COCLR [HXZ20B]	IMAGE	70.2	39.1
COCLR [HXZ20B]	IMAGE + FLOT	72.1	40.2

TABLEAU 5.2 – Performances de reconnaissance d'actions par classification linéaire sur les représentations obtenues par notre méthode

Architecture et hyperparamètres De manière similaire à CPC Vidéo, nous utilisons T = 10 segments de F = 10 images. Les différentes modalités (flots optiques, différence d'images) sont calculées de la même manière. Un *resnet18* avec des images de taille 224 × 224 est aussi utilisé. Pour le masquage aléatoire, la moitié des éléments de la séquence sont masqués c'est à dire 5. Pour les deux autres méthodes de masquage, entre 1 et 9 éléments sont masqués. Pour le *transformer*, 2 couches sont utilisées avec 8 têtes d'attention et une dimension de 512 pour la largeur des couches.

Entraînement Les réseaux sont entraînés de manière non-supervisée sur UCF-101 pour 1000 époques. L'optimisation s'effectue avec Adam et un pas d'apprentissage de 0.0001 qui est réduit par un facteur de 10 aux itérations 50000 et 60000. La taille de *batch* utilisée est fixée à 20.

Résultats

Nous comparons tout d'abord les différentes méthodes de masquage présentées puis les résultats sur différentes modalités. Les résultats sont ensuite comparés aux autres méthodes de l'état de l'art. Différentes visualisations sont finalement présentées pour obtenir une meilleure compréhension de la qualité des représentations apprises.

DATASET		UCF-101			HMDB51		
MÉTHODE	MODALITÉ	1	5	10	1	5	10
OUR ALÉATOIRE	FLOT	46.5	47.7	48.3	22.2	22.7	25.3
OUR FUTUR	FLOT	56.6	56.1	56.7	26.4	**28.1**	28.0
OUR PASSÉ/FUTUR	FLOT	**60.8**	**60.4**	**61.1**	**27.3**	**28.1**	**29.0**
OUR PASSÉ/FUTUR	IMAGE DIFF	45.6	46.1	46.6	16.6	17.2	18.6
OUR PASSÉ/FUTUR	IMAGE	39.7	40.8	40.7	14.5	16.4	18.2

TABLEAU 5.3 – Résultats de reconnaissance utilisant les k plus proches voisins à partir des représentations non-supervisées.

Comparaison des méthodes de masquage La première partie des tableaux 5.2, 5.3 et 5.4 montrent les résultats en reconnaissance d'actions (classification linéaire et k plus proches voisins) et de recherche de vidéos pour les différents masquages proposés en utilisant les flots optiques comme entrée. Dans la plupart des cas, les résultats sont les meilleurs pour le masquage "passé/futur" et les moins bons pour le masquage aléatoire. Par exemple, en classification linéaire, le masquage « passé/futur » permet de gagner 3.6 p. et 1.7 p. par rapport au masquage "futur" sur UCF-101 et HMDB51 respectivement. Le gain est encore beaucoup plus important par rapport au masquage aléatoire (9.5 p. et 7.3 p.). Celà montre bien qu'un masquage aléatoire comme appliqué dans BERT pour le texte n'est pas optimal pour la vidéo. Il est donc important d'intégrer une information de structure au masque de manière similaire aux résultats obtenus sur les images. Finalement, la prédiction dans les 2 directions donne de meilleurs résultats ce qui est compréhensible.

Résultats sur les différentes modalités La deuxième partie des tableaux 5.2, 5.3 et 5.4 montrent les évaluations pour différentes modalités d'entrée (image, différence d'images et flots optiques). On remarque que les meilleurs résultats sont obtenus avec les flots optiques puis avec les différences d'image et ensuite avec les images. Par exemple, pour la classification linéaire, les flots optiques obtiennent +11.3 p. et +14.2 p. par rapport aux différences d'images sur UCF-101 et HMDB51 respectivement. Les différences d'images ont des scores supérieurs de 11.1 p. et 4.3 p. par rapport aux images. Ces résultats sont attendus car les flots optiques et les différences d'image représentent le mouvement dans la vidéo ce qui en fait de meilleures entrées pour un réseau de reconnaissance d'actions. Les résultats sont d'ailleurs en adéquation avec ceux obtenus sur la méthode CPC pour la vidéo.

DATASET		UCF-101			HMDB51		
MÉTHODE	MODALITÉ	1	5	10	1	5	10
OUR ALÉATOIRE	FLOT	47.1	70.0	78.9	22.8	47.5	59.8
OUR FUTUR	FLOT	57.4	75.9	83.3	26.3	52.4	63.4
OUR PASSÉ/FUTUR	FLOT	**61.2**	**80.0**	**86.0**	**29.0**	**54.2**	**65.1**
OUR PASSÉ/FUTUR	IMAGE DIFF	47.1	66.0	74.6	16.4	38.2	52.0
OUR PASSÉ/FUTUR	IMAGE	40.3	58.6	67.6	15.2	37.8	51.3
COCLR [HXZ20B]	FLOT	53.3	69.4	76.6	23.2	43.2	53.5
COCLR [HXZ20B]	IMAGE	51.9	68.5	75.0	23.9	47.3	58.3
COCLR [HXZ20B]	IMAGE + FLOT	55.9	70.8	76.9	26.1	45.8	57.9
MEMDPC [HXZ20A]	IMAGE	20.2	40.4	52.4	7.7	25.7	40.6
CLIP ORDER [AME18]	IMAGE	14.1	30.3	40.0	7.6	22.9	34.4
SPEEDNET[BEL+20]	IMAGE	13.0	28.1	37.5			
TEMP TRANS [JMF20]	IMAGE	26.1	48.5	59.1			

TABLEAU 5.4 – Résultats de recherche de vidéos se basant sur la similarité cosinus sur les représentations

Comparaison avec l'état de l'art Les résultats de classification linéaire et de recherche de vidéos sont bien meilleurs que l'état de l'art pour la modalité des flots optiques. Par exemple, les performances sont supérieures de 6.6 p. et 7.2 p. en classification linéaire sur UCF-101 et HMDB51 par rapport à CoCLR à 2 voies (images et flots optiques) alors que nous utilisons juste les flots optiques. Les résultats sur les images et les différences d'images sont compétitifs par rapport à l'état de l'art. Les résultats sur les images sont assez similaires par rapport à CBT (+2.3 p. sur UCF-101 mais −0.6 p. sur HMDB51). Une différence est que CBT est appris sur une bien plus grande base de données (Kinetics) et avec des réseaux beaucoup plus importants (S3D). Les capacités de calcul nécessaires à effectuer leurs apprentissages sont très importantes (utilisation de CloudTPU) comparées à celles nécessaires pour obtenir nos résultats. Finalement, les résultats sur les images de CoCLR sont bien meilleurs que les nôtres ce qui peut s'expliquer par le fait que les flots optiques sont quand même utilisés lors du pré-entraînement dans leur méthode. En effet, c'est à partir des flots optiques que les exemples positifs pour les images sont déterminés. Cela apporte un avantage conséquent par rapport à notre méthode qui n'utilise lors du pré-entraînement que la modalité sur laquelle elle est évaluée.

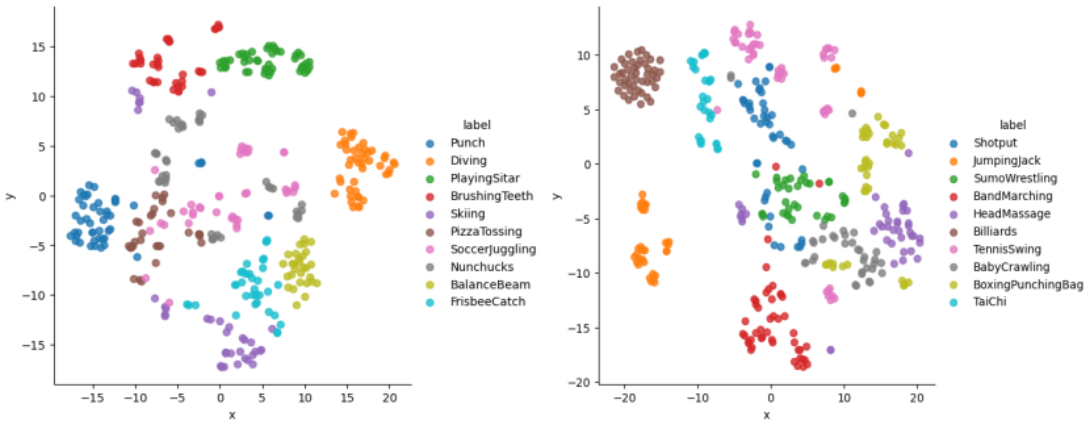


FIGURE 5.5 – Visualisations du tsne des représentations obtenues avec BERT en utilisant les flots optiques et un masquage bidirectionnel

Visualisations La figure 5.5 visualise les représentations obtenues par la méthode BERT sur les flots optiques en utilisant le masquage bidirectionnel. Les différentes classes sont assez bien séparées, comme par exemple les classes *punch*, *diving* ou encore *billards*. Les visualisations de recherche de vidéos sont présentées sur la figure 5.6. Les vidéos retournées appartiennent la plupart du temps à la même classe que la requête. Par exemple, toutes les vidéos retournées pour la première requête sont bien des vidéos de boxe. Ces visualisations permettent de mettre en avant l'utilité des représentations apprises par notre méthode non-supervisée.

5.4 Conclusion

Nous avons montré dans ce chapitre comment il est possible d'adapter les modèles de langage comme BERT à des données comme les vidéos en utilisant la fonction de coût InfoNCE. Nous nous sommes particulièrement intéressé à la méthode de masquage utilisée pour l'adapter au domaine de la vidéo. Nos résultats expérimentaux montrent qu' un masquage bidirectionnel « passé/futur » marche bien mieux qu'un masquage aléatoire. Finalement, des résultats sur différentes modalités pour la vidéo sont présentés et sont compétitifs par rapport à l'état de l'art.

Il serait intéressant d'essayer d'améliorer cette méthode en profitant de la souplesse des *transformers* pour varier les prédictions. Par exemple en combinant des prédictions à la fois spatiales et temporelles qui utiliseraient des encodages posi-

tionnels différents. Il est aussi possible de réaliser des prédictions temporelles à plusieurs échelles de temps différentes. Chaque échelle de temps aurait de la même manière un encodage positionnel différent. Finalement, lors du pré-entraînement, chaque modalité (image, différence d'images, flots) a été utilisée séparément. Des prédictions multimodales (de flots optiques à partir d'images par exemple) pourraient améliorer les performances du modèle. Cela pourrait notamment permettre de transmettre l'information de mouvement contenue dans les flots optiques aux images et donc d'améliorer les performances sur les modèles utilisant les images qui ont pour l'instant des résultats plus faibles.

FIGURE 5.6 – Visualisations des résultats de recherche de vidéos. Les 2 premières vidéos en haut sont les requêtes (5 images de la vidéo sont représentées) et celles en dessous sont les vidéos retournées. Les vidéos retournées entourées en rouge sont incorrectes (elles n'appartiennent pas à la même classe que la requête) et les vertes sont correctes.

Chapitre 6

Augmentation du nombre d'exemples par mixup

Sommaire

6.1 Pré-requis 108
6.1.1 SimCLR 109
6.1.2 Mixup 110
6.2 Augmentation des exemples par mixup pour SimCLR 112
6.3 Expériences sur les images 115
6.3.1 Bases de données 115
6.3.2 Evaluation sur les images 115
6.4 Expériences sur les vidéos 116
6.4.1 Détails d'implémentation 117
6.5 Résultats sur les images 117
6.6 Résultats sur les vidéos 121
6.6.1 Résultats quantitatifs 121
6.6.2 Résultats qualitatifs 124
6.7 Conclusion 127

Les méthodes contrastives présentées section 2.4 offrent les meilleures performances d'apprentissage de représentation de l'état de l'art [HSF+19] [HFW+20] [CKNH20]. Nous nous intéressons à deux problèmes des méthodes contrastives qui sont le besoin d'un grand nombre d'exemples négatifs et le temps important d'apprentissage. La nécessité d'utiliser un très grand nombre d'exemples négatifs

a été montrée dans l'article SimCLR [CKNH20] ou des tailles de *batch* très importantes sont utilisées (jusqu' à 4096). Malheureusement, les accélérateurs classiques comme les GPUs ne permettent pas d'utiliser des *batchs* aussi conséquents à cause de limitations mémoires. D'autres accélérateurs comme les TPUs sont alors nécessaires. L'augmentation du nombre d'exemples négatifs rend la tâche non-supervisée plus complexe ce qui permet d'apprendre des représentations d'une meilleure qualité. L'amélioration des performances avec le nombre d'exemples négatifs N s'explique aussi théoriquement. En effet, la borne entre la fonction de coût InfoNCE et l'information mutuelle dépend de log(N). Les expériences conduites dans l'article SimCLR [CKNH20] montrent aussi qu'il est nécessaire d'apprendre plus longtemps qu'en supervisé (jusqu'à 1000 époques contre environ 100 époques en supervisé pour l'apprentissage sur Imagenet par exemple). Cela rend ce type d'apprentissage très coûteux : il peut durer plusieurs semaines voire mois sur des GPUs. Notre idée est d'augmenter le nombre d'exemples négatifs de manière peu coûteuse en mémoire et de diminuer les temps d'apprentissage. Pour atteindre cet objectif, nous proposons de créer de nouveaux exemples en faisant une combinaison linéaire entre deux représentations d'exemples existants. On propose d'utiliser cette approche pour augmenter les exemples positifs et les exemples négatifs dans la méthode SimCLR, et ainsi dimunuer la taille des *batchs* et le temps nécessaire à l'apprentissage. Nous avons choisi SimCLR comme méthode de base pour sa simplicité et ses bonnes performances au niveau de l'état de l'art. Les approches contrastives présentées précédemment sont plus complexes car elles se basent sur une séquence temporelle alors que SimCLR se base uniquement sur deux augmentations d'un même exemple. Nous appliquons cette nouvelle approche à l'apprentissage de représentations d'images et de vidéos. L'apport est évalué par la suite sur la classification d'images et la reconnaissance d'actions.

6.1 Pré-requis

Cette partie présente les méthodes sur lesquelles nous nous basons. Nous détaillons tout d'abord SimCLR qui est notre méthode de référence et que l'on propose d'améliorer grâce à la création de nouveaux exemples positifs et négatifs. On décrit ensuite la méthode d'augmentation *mixup* [ZCDL17] à partir de laquelle ces nouveaux exemples positifs ou négatifs vont être créés.

6.1.1 SimCLR

SimCLR est une méthode de pré-apprentissage non-supervisé qui a pour but de rendre similaire les représentations provenant d'exemples proches et inversement. Soient $\tilde{x}_i$ et $\tilde{x}_j$ deux images provenant de la même image x mais ayant subi des transformations ou augmentations de données différentes. On nomme requête les valeurs $\tilde{x}_i$ et clés les valeurs $\tilde{x}_j$. Le but de SimCLR est d'obtenir des représentations z_i et z_j des images $\tilde{x}_i$ et $\tilde{x}_j$ respectivement qui sont très similaires, et en même temps éloignées des N représentations $z^{(n)}$ de N exemples négatifs $x^{(n)}, n = 1, \ldots, \mathrm{N}$. Notons qu'a partir d'une image x, sa représentation z est obtenue par son passage à travers la composition d'un CNN f et d'un perceptron multi-couches g. On a donc $z = g(h)$ avec $h = f(x)$. La représentation intermédiaire h est utilisée par la suite pour la tâche de classification supervisée pour évaluer la représentation apprise. La similarité utilisée entre z_i et z_j ainsi qu'entre z_i et les différentes $z^{(n)}$ est le cosinus :

$$\mathrm{sim}(z_i, z_j) = \frac{z_i^\top z_j}{||z_i||.||z_j||}.$$

Un autre choix possible est le produit scalaire mais sa valeur n'est pas bornée comme le cosinus (entre -1 et 1). Celà permet d'éviter d'avoir des valeurs trop extrêmes et améliore la qualité des représentations obtenues. La fonction de coût InfoNCE utilisée est la suivante :

$$\mathrm{L_N}(z_i, z_j) = -\log \frac{\exp\big(\mathrm{sim}(z_i, z_j)/\tau\big)}{\exp\big(\mathrm{sim}(z_i, z_j)/\tau\big) + \sum_{n=1}^{\mathrm{N}} \exp\big(\mathrm{sim}(z_i, z^{(n)})/\tau\big)}, \tag{6.1}$$

avec τ un paramètre appelé température.

Application de SimCLR à la vidéo Soit $x = x_1, \ldots, x_\mathrm{T}$ une vidéo d'entrée constituée de la concatenation de T images (*frames*) $x_t, t = 1, \ldots, \mathrm{T}$. Les exemples $\tilde{x}_i$ et $\tilde{x}_j$ sont obtenus de la manière suivante (voir figure 6.1). Un segment vidéo de F images successives est sélectionné en tirant un instant t de manière aléatoire suivant une loi uniforme discrète par :

$$t \sim \mathscr{U}[0, \mathrm{T} - \mathrm{F}], \qquad \tilde{x}_i \leftarrow x[t : t + \mathrm{F}].$$

Par la suite, un *cropping* spatial aléatoire est appliqué. Des opérations d'aug-

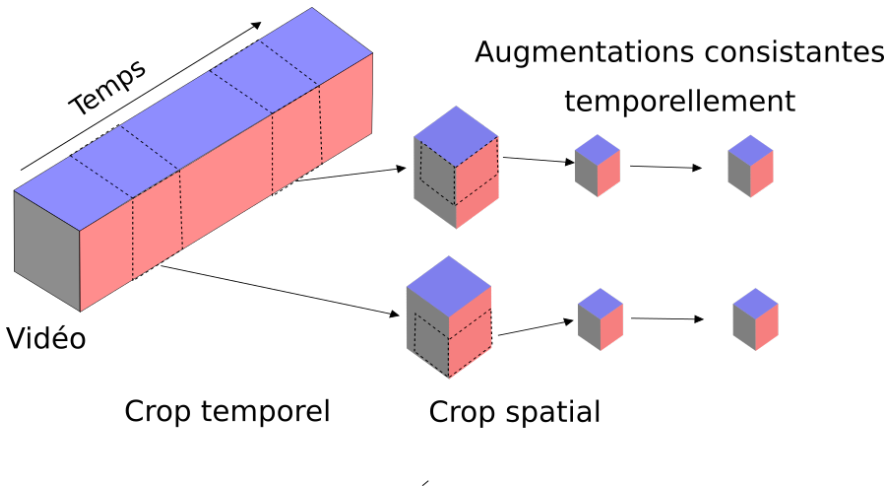


FIGURE 6.1 – Schéma des augmentations utilisées sur les vidéos pour générer la requête et la clé dans SimCLR. Un crop temporel puis un crop spatial et d'autres augmentations (*color jittering* par exemple) sont appliquées.

mentation sont effectuées (opérations sur les couleurs, contraste, saturation). Le *crop* spatial et les augmentations sont consistants temporellement c'est à dire qu'ils sont appliqués de la même manière sur les différentes images d'un même segment. Le but est de garder la cohérence spatio-temporelle pour que le réseau puisse extraire de manière plus efficace le mouvement. Le même processus est appliqué pour $\tilde{x}_j$. Il en résulte que les exemples $\tilde{x}_i$ et $\tilde{x}_j$ peuvent représenter des moments temporels différents dans la vidéo mais aussi des zones spatiales différentes. Les entrées du réseau $\tilde{x}_i$ et $\tilde{x}_j$ sont des blocs spatio-temporels. Des réseaux convolutionnels 3D sont donc utilisés au lieu des réseaux 2D sur les images. Un *pooling* global spatio-temporel est effectuée à la fin du réseau au lieu d'un *pooling* global seulement spatial. Le reste de la méthode reste similaire.

6.1.2 Mixup

Le *mixup* [ZCDL17] est une technique d'augmentation de données qui crée un nouvel exemple en interpolant linéairement entre deux exemples existants (voir figure 6.2). Considérons une base de données annotée $D = \{(x_i, y_i)\}$ et f un réseau de classification. On considère l'opération de mixup suivante

$$\text{mix}_\lambda(a,b) = \lambda.a + (1-\lambda).b\,.$$

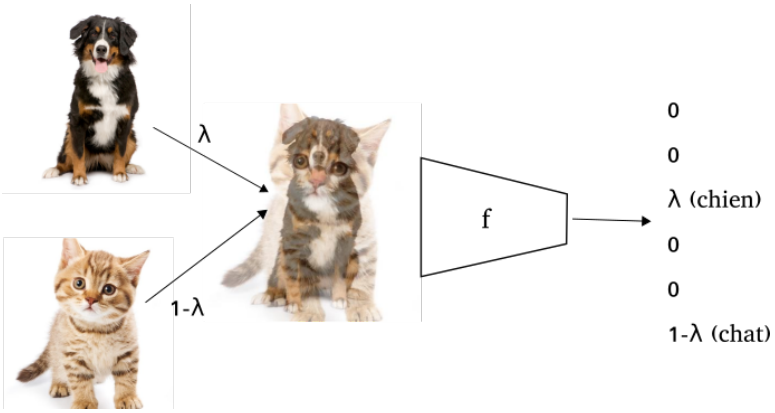


FIGURE 6.2 – Schéma de la méthode de mixup sur les entrées. Le réseau f a pour entrée l'interpolation de deux images et doit prédire l'interpolation des deux vérités terrain.

Notons l la fonction de coût. La perte à minimiser de manière classique est

$$\mathrm{L} = \sum_i l\big(f(x_i), y_i\big).$$

La méthode de *mixup* propose de classifier les exemples $\tilde{x} = \mathrm{mix}_\lambda(x_i, x_j)$ avec comme vérité terrain $\tilde{y} = \mathrm{mix}_\lambda(y_i, y_j)$. La fonction de perte s'écrit donc de la manière suivante :

$$\mathrm{L}_{mix} = \mathbb{E}_\lambda \sum_{i,j} l\Big(f\big(\mathrm{mix}_\lambda(x_i, x_j)\big), \mathrm{mix}_\lambda(y_i, y_j)\Big).$$

Cette approche permet d'avoir un comportement linéaire entre les exemples d'entraînement ce qui est une propriété intéressante. En effet, c'est le comportement le plus simple en dehors de la base d'entraînement et limite les oscillations. [ZCDL17] montre que cette méthode permet de régulariser de manière efficace et ainsi d'avoir des meilleurs performances en test sur différents problèmes de classification.

[VLB+19] montre qu'il est plus approprié d'utiliser cette technique sur une couche intermédiaire du réseau plutôt que sur l'entrée (voir figure 6.3). Décomposons le réseau f en 2 sous parties $f(x) = f_{>k}\big(f_{<k}(x)\big)$. L'interpolation linéaire va s'effectuer sur les représentations intermédiaires $f_{<k}(x)$. La fonction de perte pour le *manifold mixup* est la suivante :

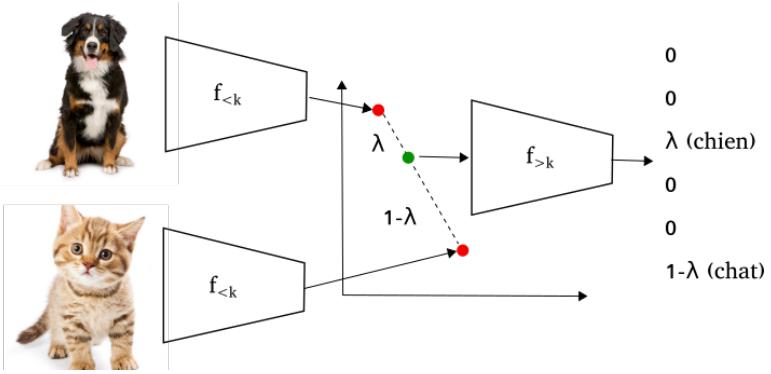


FIGURE 6.3 – Schéma de la méthode de mixup sur les représentations intermédiaires. Le réseau de classification est découpé en deux. L'interpolation se fait au niveau d'une représentation intermédiaire (à la fin de la première partie de réseau de classification). Le but est encore de prédire l'interpolation des vérités terrain.

$$\mathrm{L}_{mix} = \mathbb{E}_{\lambda} \sum_{i,j} l\Big[f_{>k}\Big(\mathrm{mix}_{\lambda}\big(f_{<k}(x_i), f_{<k}(x_j)\big)\Big), \mathrm{mix}_{\lambda}(y_i, y_j)\Big].$$

Cette approche est intéressante car elle effectue les interpolations sur des représentations haut niveau ce qui a plus de sens que sur les images. En effet, de nombreuses méthodes d' apprentissage de représentation montrent des propriétés intéressantes d'arithmétique sur les représentations. [VLB+19] montre que cette méthode permet d'obtenir des frontières de décisions moins tranchées et d'aplanir les zones de représentation des différentes classes. Il a été vérifié expérimentalement que cette méthode améliore la généralisation.

6.2 Augmentation des exemples par mixup pour SimCLR

La méthode proposée a pour but d'augmenter le nombre d'exemples positifs et négatifs en effectuant des combinaisons linéaires de manière similaire à la méthode présentée précédemment. On sépare la fonction de coût en 2 parties, la fonction de coût InfoNCE classique (equation 6.1) et la fonction de coût sur les exemples interpolés. L'interpolation entre un exemple positif et un exemple négatif $mix_{\lambda_p}(z_j, z^{(p)})$ donne un exemple positif dont la fonction de coût est pondérée par λ_p. Cette pondération est similaire à l'interpolation des vérités terrain dans le *mixup*. L'interpo-

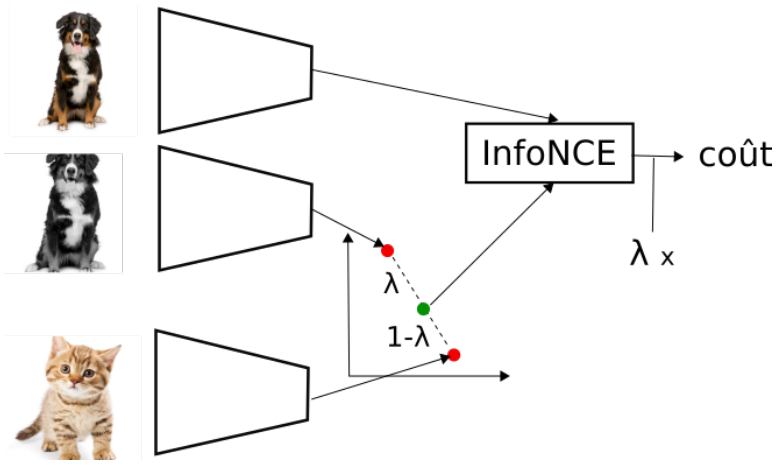


FIGURE 6.4 – Schéma de la méthode SimCLR avec utilisation du mixup pour augmenter les exemples positifs et négatifs. La clé est interpolée au niveau des représentations avec d'autres exemples. La fonction de coût InfoNCE est appliquée entre la clé et ces exemples mixés. Les exemples négatifs sont aussi issus de l'opération de *mixup*. La fonction de coût est ensuite pondérée par λ.

lation linéaire entre 2 exemples négatifs donne un exemple négatif. La fonction de coût obtenue pour les exemples interpolés est donc la suivante :

$$\mathrm{L}_{\mathrm{N}}^{mix}(z_i, z_j) = -\sum_{p}^{\mathrm{P}} \lambda_p \log \frac{\exp\left(\mathrm{sim}\left(z_i, \mathrm{mix}_{\lambda_p}(z_j, z^{(p)})\right)/\tau\right)}{\exp\left(\frac{\mathrm{sim}\left(z_i, \mathrm{mix}_{\lambda_p}(z_j, z^{(p)})\right)}{\tau}\right) + \sum_{n,m} \exp\left(\frac{\mathrm{sim}\left(z_i, \mathrm{mix}_{\lambda_{n,m}}(z^{(n)}, z^{(m)})\right)}{\tau}\right)} \quad . \tag{6.2}$$

La fonction de coût finale est la somme de la fonction de coût InfoNCE classique et de celle avec le *mixup* :

$$\mathrm{L} = \mathrm{L}_{\mathrm{N}}(z_i, z_j) + \mathrm{L}_{\mathrm{N}}^{mix}(z_i, z_j)\,.$$

L'utilisation d'exemples positifs mixés permet en quelque sorte de voir plus d'exemples, même si ceux-ci sont issus d'une interpolation d'autres exemples du *batch*. Cela permet de réduire le temps d'entraînement. L'utilisation du *mixup* sur les exemples négatifs permet d'augmenter le nombre d'exemples négatifs ce qui améliore les performances des méthodes contrastives. Contrairement à l'augmentation de la taille du *batch*, notre méthode se fait avec peu de coût de calcul supplémentaire. En effet, la création des nouveaux exemples positifs ou négatifs se fait au niveau des représentations, donc après que les exemples soient passés à travers le

réseau convolutionnel qui représente la plus grosse partie des calculs.

Autres formulations du mixup D'autres formulations du *mixup* ont été testées mais donnent de moins bonnes performances que la formulation principale. Elles sont décrites dans ce paragraphe.

Mixup uniquement sur les négatifs Dans cette formulation, on augmente uniquement le nombre d'exemples négatifs par interpolation. On rajoute donc aux exemples négatifs classiques des exemples issus de l'opération de *mixup*. La fonction de coût est donc la suivante :

$$\mathrm{L_N}(z_i, z_j) = -\log \frac{\exp\left(\mathrm{sim}_{(z_i, z_j)/\tau}\right)}{\exp\left(\mathrm{sim}(z_i, z_j)/\tau\right) + \sum_{n=1}^{\mathrm{N}} \exp\left(\mathrm{sim}(z_i, z^{(n)})/\tau\right) + \sum_{n,m} \exp\left(\mathrm{sim}\left(z_i^{\top} \mathrm{mix}_{\lambda_{n,m}}(z^{(n)}, z^{(m)})\right)/\tau\right)}. \quad (6.3)$$

La différence principale est qu'il n'a pas de mixup sur les exemples positifs. Un inconvénient est que l'exemple positif est classique mais les exemples négatifs ajoutés sont interpolés. Cela provoque une différence entre l'exemple positif et les exemples négatifs qui n'est pas forcément souhaitable.

Mixup classique Cette formulation est l'application stricte de la technique de *manifold mixup* à SimCLR. L'opération d'interpolation va s'effectuer sur la représentation des requêtes. Nous considérons z_{i1} et z_{j1} les 2 augmentations d'un premier exemple et z_{i2} et z_{j2} celles d'un second exemple. Le *mixup* sur la requête sera donc $mix_\lambda(z_{i1}, z_{j1})$. La fonction de coût InfoNCE peut être considérée comme une classification paramétrique. On peut alors considérer les exemples positifs comme les vérités terrain. On pondère donc par λ la fonction de coût InfoNCE avec comme exemple positif z_{j1} et par $(1-\lambda)$ la fonction de coût avec comme exemple positif z_{j2}. On obtient donc la fonction de coût suivante :

$$\begin{aligned} \mathrm{L_N} \;=\; & -\lambda \log \frac{\exp\left(\mathrm{sim}\left(\mathrm{mix}_\lambda(z_{i1}, z_{i2}), z_{j1}\right)/\tau\right)}{\exp\left(\mathrm{sim}\left(\mathrm{mix}_\lambda(z_{i1}, z_{i2}), z_{j1}\right)/\tau\right) + \sum_{n=1}^{\mathrm{N}} \exp\left(\mathrm{sim}\left(\mathrm{mix}_\lambda(z_{i1}, z_{i2}), z^{(n)}\right)/\tau\right)} \\ & -(1-\lambda) \log \frac{\exp\left(\mathrm{sim}\left(\mathrm{mix}_\lambda(z_{i1}, z_{i2}), z_{j2}\right)/\tau\right)}{\exp\left(\mathrm{sim}\left(\mathrm{mix}_\lambda(z_{i1}, z_{i2}), z_{j2}\right)/\tau\right) + \sum_{n=1}^{\mathrm{N}} \exp\left(\mathrm{sim}\left(\mathrm{mix}_\lambda(z_{i1}, z_{i2}), z^{(n)}\right)/\tau\right)} \end{aligned}. \quad (6.4)$$

Le principal désavantage de cette formulation est que l'on n'augmente ni le nombre d'exemples positifs ni le nombre d'exemples négatifs car l'on effectue l'opération de mixup sur la requête.

6.3 Expériences sur les images

Cette partie présente les expériences menées sur les images. Elle décrit notamment les bases de données utilisées, comment la méthode est évaluée ainsi que les paramètres utilisés dans la méthode.

6.3.1 Bases de données

Les expériences sont effectuées sur des bases de données d'image de taille modérée qui sont Imagenette, Imagewang et STL10 [CNL11]. Les deux premières bases de données ont été utilisées auparavant et présentées dans la partie 2.4.3. Les caractéristiques de la base STL10 sont détaillées ci-dessous.

STL10 : Elle contient des images de taille 96x96 séparées en 10 classes. Cette base de données est spécialement adaptée pour l'apprentissage non-supervisé et semi-supervisé car il contient peu d'exemples dans l'ensemble d'entraînement (800) mais beaucoup d'exemples non-annotés (100000).

6.3.2 Evaluation sur les images

Les pré-entraînements effectués sont ensuite évalués sur la tâche de classification d'images. On initialise les poids du réseau f avec les poids obtenus par la méthode non-supervisée. On ajoute une opération de *batch normalization* bn et une couche dense pour effectuer la classification. L'entropie croisée sur les prédictions o est utilisée pour l'entraînement :

$$o = \text{softmax}\Big(\text{W}_c \text{bn}\big(f(x,\theta)\big) + b_c\Big).$$

La plupart des résultats présentés sont issus de la classification linéaire c'est à dire que les poids θ du réseau sont fixés. Seuls les paramètres W_c et b_c sont appris. S'il n'est pas indiqué autrement, le pré-entraînement et la classification linéaire sont effectués sur la même base de données.

Détails d'implémentation

L'architecture utilisée est un resnet50. La *batch normalization* synchronisée entre les GPUs est utilisée. La valeur de τ est fixée à 0.1 comme dans l'article SimCLR. Le réseau g est une couche linéaire comme nous avons remarqué dans nos expériences qu'un MLP apporte peu. Les images utilisées sont de taille 224×224 sauf pour STL10 pour laquelle elles sont de taille 112×112.

Prétraitement Pour le pré-apprentissage non-supervisé et l'apprentissage supervisé, un cropping avec une couverture minimale de 0.1 suivi d'un redimensionnement de l'image à la taille d'entrée du réseau sont effectués. Pour l'apprentissage non-supervisé, les 2 *crops* sont différents pour les différentes versions. Des augmentations plus fortes sont aussi utilisées pour le pré-entraînement. Nous en avons testé deux, *Rand Augment* et l'augmentation de distorsion des couleurs de SimCLR nommée *color distort*. Pour l'évaluation, un seul *crop* de la partie centrale de l'image est utilisé dans notre implémentation.

Entraînement Les paramètres d'entraînement pour les différentes étapes (pré-apprentissage et classification linéaire) sont présentés ci-dessous.

Pré-entraînement Le réseau est pré-appris pendant 500 époques. On utilise des *batchs* de taille 64. L'optimisation se fait avec la méthode Adam et un pas d'apprentissage de 0.001. Une régularisation L2 de 10^{-6} est appliquée.

Classification linéaire 50 époques de classification linéaire sont effectuées. Les résultats présentés sont la moyenne et l'écart-type des scores sur 10 apprentissages. La méthode Adam avec un pas d'apprentissage de 0.001 est aussi utilisée.

6.4 Expériences sur les vidéos

Nous décrivons ici les paramètres des expériences que nous effectuons sur le pré-apprentissage de vidéos. Les représentations sont ensuite principalement évaluées sur la tâche de reconnaissance d'actions. Les bases de données UCF-101 et HMDB51 sont présentées dans la partie reconnaissance d'actions 3.2. Le protocole d'évaluation est le même que pour la méthode CPC Vidéo (présenté parties 4.2.5 et 4.3.3).

6.4.1 Détails d'implémentation

Les réseaux utilisés pour ces expériences sont le resnet18-3D et le resnet18-(2+1)D. La *batch normalization* synchronisée est aussi utilisée. Les réseaux plus importants demandent une mémoire plus importante ce qui limiterait la taille du batch en entrée et pourrait nuire aux performances.

Prétraitement 16 *frames* avec un *stride* de 2 sont sélectionnées par segment ce qui représente environ une seconde de vidéo. Un crop avec une couverture minimale de 0.1 et un redimensionnement de l'image sont effectués. Les images sont de taille 224×224 en entrée du réseau. L'augmentation de distorsion des couleurs est utilisée pour le pré-entraînement.

Entraînement

Pré-apprentissage non-supervisé Le réseau est pré-appris pendant 500 époques. On utilise des *batchs* de taille 28. L'optimisation se fait avec la méthode Adam et un pas d'apprentissage de 0.001. Une régularisation L2 de 10^{-6} est utilisée.

Classification linéaire 20 époques de classification linéaire sont effectuées. La méthode Adam avec un pas d'apprentissage de 0.001 sont aussi utilisée. 6 segments sont utilisés lors de l'entraînement et 10 pour l'évaluation.

6.5 Résultats sur les images

Nous présentons tout d'abord les résultats prélimilaires sur la méthode SimCLR. Les performances avec et sans application du *mixup* sont ensuite comparées. La supériorité de la formulation principale du *mixup* est enfin mise en évidence et certains de ses paramètres sont étudiés.

Expériences préliminaires

La table 6.1 compare différents prétraitements et paramètres de la méthode sur les bases Imagenette et Imagewang. Nous montrons ici que l'utilisation de l'augmentation de distorsion des couleurs améliore les résultats par rapport à Rand Augment. Ceci s'explique par le fait que ces augmentations sont plus fortes et per-

Base de données	Transformation	Normalisation	Train Accuracy	Test Accuracy
imagenette	rand augment	non	71.54	76.36
imagenette	color distort	non	80.79	81.70
imagenette	color distort	oui	**90.58**	**91.57**
imagewang	rand augment	non	55.79	34.54
imagewang	color distort	non	68.79	40.27
imagewang	color distort	oui	**73.03**	**58.64**

TABLEAU 6.1 – Etude des différents paramètres de la méthode SimCLR sur les images : normalisation, type de transformation. Les résultats de classification linéaire sont présentés.

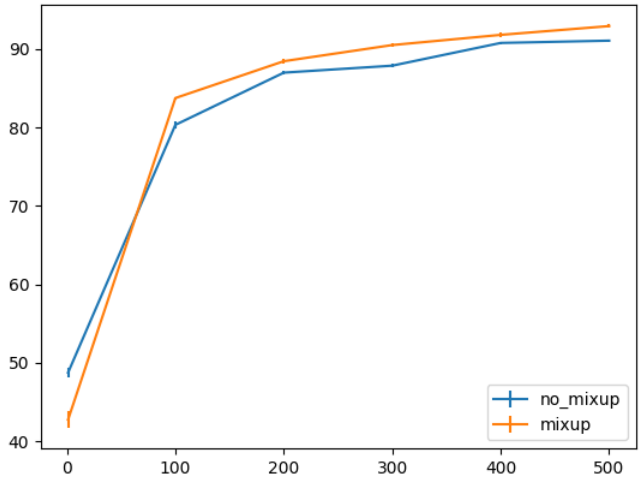
mettent d'apprendre de meilleures représentations. Nous remarquons aussi l'importance de la normalisation des représentations z comme expliqué dans l'article SimCLR.

Comparaison avec et sans mixup

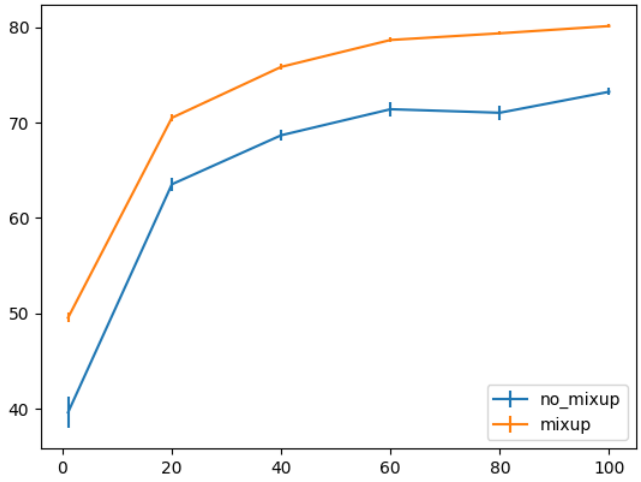
L'utilisation du mixup apporte un gain en performance significatif notamment sur les datasets Imagewang et STL10. Plus de 7 p. sont gagnés sur le dataset Imagewang et un peu moins de 7 p. sur le dataset STL10. La figure 6.5 montre l'évolution des scores en classification linéaire en fonction du temps d'apprentissage non-supervisé (compté en époques). On remarque notamment que l'utilisation du mixup permet d'apprendre bien plus rapidemment que sans. Les performances en classification linéaire sont tout le temps meilleures avec mixup que sans ce qui montre encore plus l'intérêt de la méthode.

DATASET	MIXUP	TRAIN ACCURACY	TEST ACCURACY
IMAGENETTE	FALSE	89.44 ± 0.25	91.02 ± 0.16
IMAGENETTE	TRUE	$\mathbf{92.13 \pm 0.28}$	$\mathbf{92.73 \pm 0.17}$
IMAGEWANG	FALSE	70.11 ± 0.32	56.36 ± 1.28
IMAGEWANG	TRUE	$\mathbf{76.61 \pm 0.24}$	$\mathbf{63.80 \pm 0.69}$
STL10	FALSE	X	73.34 ± 0.37
STL10	TRUE	X	$\mathbf{80.13 \pm 0.17}$

TABLEAU 6.2 – Comparaison des résultats de classification linéaire pour la reconnaissance d'images pour les pré-apprentissages avec et sans utilisation du mixup



(a) Imagenette



(b) STL10

FIGURE 6.5 – Comparaison du score avec et sans mixup en fonction du temps de pré-apprentissage.

Etude de la formulation du mixup

DATASET	TYPE MIXUP	TRAIN ACCURACY	TEST ACCURACY
IMAGENETTE	1	**92.13 ± 0.28**	**92.73 ± 0.17**
IMAGENETTE	2	89.66 ± 0.21	91.51 ± 0.14
IMAGENETTE	3	89.04 ± 0.31	91.66 ± 0.20
IMAGEWANG	1	**76.61 ± 0.24**	**63.80 ± 0.69**
IMAGEWANG	2	64.14 ± 0.40	43.36 ± 0.65
IMAGEWANG	3	69.88 ± 0.34	58.22 ± 0.68

TABLEAU 6.3 – Comparaison des résultats de classification linéaire avec les pré-apprentissages basés sur différentes formulations du mixup. 1 correspond à la formulation principale. 2 à celle avec les exemples négatifs uniquement. 3 à celle avec le mixup sur la requête.

Les résultats pour les différentes formulations de la méthode de mixup sont présentés dans le tableau 6.3. Même si les performances sont un peu meilleures lorsque le mixup est utilisé pour les différentes formulations, la formulation principale est bien meilleure. Elle a notamment plus de 1 p. d'écart avec les autres sur Imagenette et plus de 5 p. d'écart sur Imagewang.

Etude des paramètres

Les paramètres de la formulation principale du *mixup* sont étudiés ici. Le tableau 6.4 montre que les performances sont meilleures lorsque le nombre de *mixup* positif est important. En quelque sorte, plus on ajoute d'exemples positifs mixés,

N_p	TRAIN ACCURACY	TEST ACCURACY
1	72.77 ± 0.35	60.91 ± 0.79
2	74.27 ± 0.29	62.21 ± 0.67
5	75.59 ± 0.36	**64.12 ± 1.19**
10	**76.61 ± 0.24**	63.80 ± 0.69

TABLEAU 6.4 – Comparaison des scores de classification linéaire sur Imagewang pour un pré-apprentissage avec différentes valeurs de N_p, le nombre de mixup positif.

meilleures sont les représentations. Cela montre donc particulièrement bien l'apport des exemples positifs mixés et donc de notre méthode.

6.6 Résultats sur les vidéos

Les résultats de la méthode pour l'apprentissage de représentations sur les vidéos sont présentés ici. L'apport de la méthode est mis en évidence sur différentes tâches, la reconnaissance d'actions (par classification linéaire, finetuning ou K plus proches voisins) ainsi que la recherche de vidéos. Des visualisations (t-SNE et de recherche de vidéo) sont proposées pour mieux concevoir la qualité des représentations ainsi que l'apport de la méthode.

6.6.1 Résultats quantitatifs

Dans cette section, les performances de reconnaissance d'actions et autres métriques sont présentées pour évaluer la qualité des représentations obtenues par la méthode SimCLR sur la vidéo. Nous nous intéressons particulièrement à mettre en évidence les différences de performances entre la méthode avec et sans *mixup*.

RÉSEAU	PRETRAIN	FINETUNING	MIXUP	UCF-101	HMDB51
RESNET3D	UCF-101	NON	NON	54.3	26.0
RESNET3D	UCF-101	NON	OUI	57.3	29.3
RESNET(2+1)D	UCF-101	NON	NON	52.2	25.7
RESNET(2+1)D	UCF-101	NON	OUI	**57.4**	25.4
X	CBT	NON	X	54.0	29.5
X	MEMDPC	NON	X	54.1	**30.5**

TABLEAU 6.5 – Comparaison des performances de reconnaissance d'actions entre les préentraînement utilisant SimCLR avec et sans mixup.

Le tableau 6.5 montre que la méthode SimCLR sur la vidéo permet d'obtenir des résultats cohérents avec l'état de l'art. On remarque aussi une amélioration significative en utilisant la méthode de mixup en classification linéaire sur les 2 bases de données. Cette amélioration permet par exemple de gagner 3 p. sur la base de données UCF-101 avec le réseau resnet3D.

DATASET	UCF-101				HMDB51			
MÉTHODE | K	1	5	10	20	1	5	10	20
3D	38.2	37.6	37.7	38.8	14.4	16.1	17.3	17.0
3D MIXUP	**48.1**	**48.1**	**48.4**	**48.5**	16.7	**19.2**	**21.1**	20.4
(2+1)D	40.9	40.4	40.1	39.2	14.2	17.1	17.7	18.4
(2+1)D MIXUP	47.9	47.7	47.8	48.2	**17.3**	18.6	20.7	**20.8**

TABLEAU 6.6 – Résultats de reconnaissance d'actions à partir des K plus proches voisins.

Les résultats de reconnaissance d'actions en utilisant l'algorithme des k plus proches voisins sur les représentations obtenues à partir de l'apprentissage non-supervisé sont présentés dans le tableau 6.6. Pour chaque valeur de k, la méthode avec *mixup* a des performances largement supérieures à la méthode sans *mixup* ce qui montre encore plus clairement l'apport important de l'utilisation du *mixup* pour obtenir des représentations de bonne qualité.

DATASET	UCF-101				HMDB51			
MÉTHODE | K	1	5	10	20	1	5	10	20
3D	43.8	60.1	68.1	75.3	16.1	37.8	50.9	65.2
3D MIXUP	48.7	64.2	**71.9**	**77.6**	17.1	38.2	52.2	67.3
(2+1)D	42.2	61.0	69.5	77.7	15.5	37.0	49.9	63.9
(2+1)D MIXUP	**49.1**	**64.3**	71.2	**77.6**	**18.5**	**38.9**	**53.6**	**67.9**
MEMDPC [HXZ20A]	20.2	40.4	52.4	64.7	7.7	25.7	40.6	57.7
CLIP ORDER [AME18]	14.1	30.3	40.0	51.1	7.6	22.9	34.4	48.8
SPEEDNET [BEL+20]	13.0	28.1	37.5	49.5				
TEMP TRANS [JMF20]	26.1	48.5	59.1	69.6				

TABLEAU 6.7 – Résultats de recherche de vidéos pour différents nombre de vidéos retournées en utilisant les représentations issues de la méthode SimCLR avec et sans *mixup*. Les dernières lignes du tableau correspondent aux méthodes de l'état de l'art.

Finalement, on étudie l'utilité des représentations pour la tâche de recherche de vidéo dans le tableau 6.7. L'amélioration basée sur le *mixup* permet de faire des recherches de meilleure qualité. Les performances sont en effet meilleures pour les différentes quantités de vidéo retournées avec le plus souvent 3 à 4 points de plus.

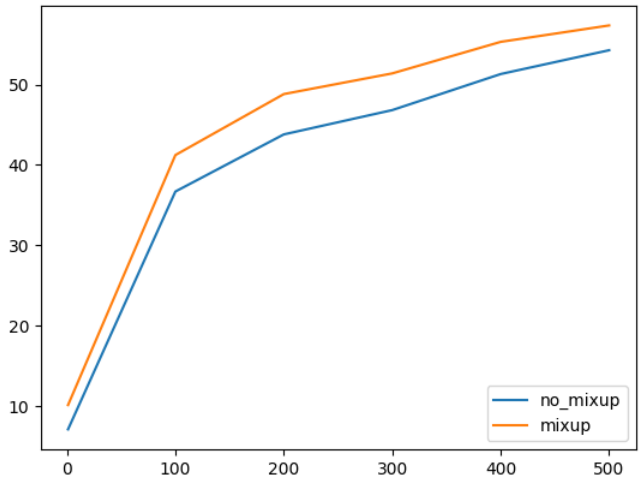


FIGURE 6.6 – Comparaison du score avec et sans *mixup* en fonction du temps de pré-apprentissage sur UCF-101.

Les résultats sont aussi bien au-dessus des performances des méthodes de l'état de l'art.

Il est aussi intéressant de regarder l'évolution des scores de classification linéaire en fonction du temps d'apprentissage non-supervisé. On remarque sur le graphique 6.6 que les scores en classification linéaire augmentent avec le temps d'apprentissage non-supervisé, ce qui est attendu. Les performances de la méthode avec *mixup* sont tout le temps au-desus de celles de la méthode sans. Ainsi, l'utilisation du *mixup* permet d'avoir des meilleurs résultats pour n'importe quel temps d'apprentissage non-supervisé.

Finalement, nous nous intéressons à l'évolution des performances en fonction d'un paramètre important de la méthode : le nombre de *mixup* positif. Le nombre de *mixup* positif est défini comme le nombre d'exemples positifs mixés divisé par la taille du *batch*. La figure 6.7 montre que la qualité de la représentation obtenue s'améliore si l'on augmente le nombre de *mixup* positif. En effet, les performances en reconnaissance d'actions avec les k plus proches voisins et en recherche de vidéos s'améliorent. L'utilisation du *mixup* sur les exemples positifs qui est un point important de notre amélioration a donc une influence positive sur les performances.

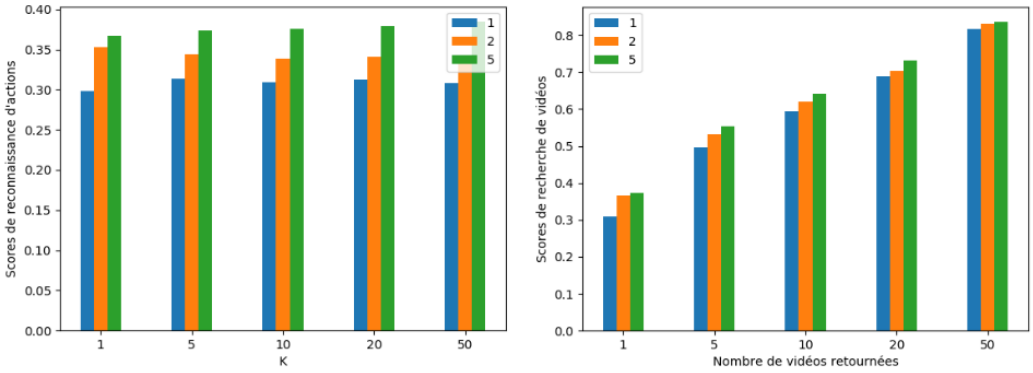


FIGURE 6.7 – Comparaison des scores de classification avec les K plus proches voisins et de recherche de vidéo pour différentes valeurs du nombre de *mixup* pour les exemples positifs.

6.6.2 Résultats qualitatifs

Cette partie propose différentes visualisations sur les représentations obtenues pour donner une idée au lecteur de la qualité des représentations obtenues par la méthode avec et sans *mixup*.

Les visualisations t-SNE présentées en 6.8 montrent que les vidéos sont plutôt bien regroupées en *cluster* de la même classe. C'est par exemple le cas pour les classes BasketballDunk, Soccer Penalty et Sky Diving dans les t-SNE de la première ligne. On peut aussi remarquer que les classes sont mieux séparées en utilisant les représentations provenant de la méthode avec *mixup*.

La figure 6.9 propose quelques résultats de recherche de vidéos en utilisant la méthode avec et sans *mixup*. On peut remarquer que les vidéos retournées sont assez similaires à la vidéo requête. La première recherche montre des vidéos de sport de lancer. Les erreurs pour le SimCLR avec *mixup* sont entre le lancer de poids et le lancer de disques. Les vidéos retournées avec les représentations utilisant le *mixup* sont plus en adéquation avec la requête. En effet, il n'a que 3 erreurs sur la première requête et 2 sur la seconde contre 8 et 5 pour la méthode sans *mixup*. D'autres résultats de recherche de vidéo sont présentés en annexe 8.2.

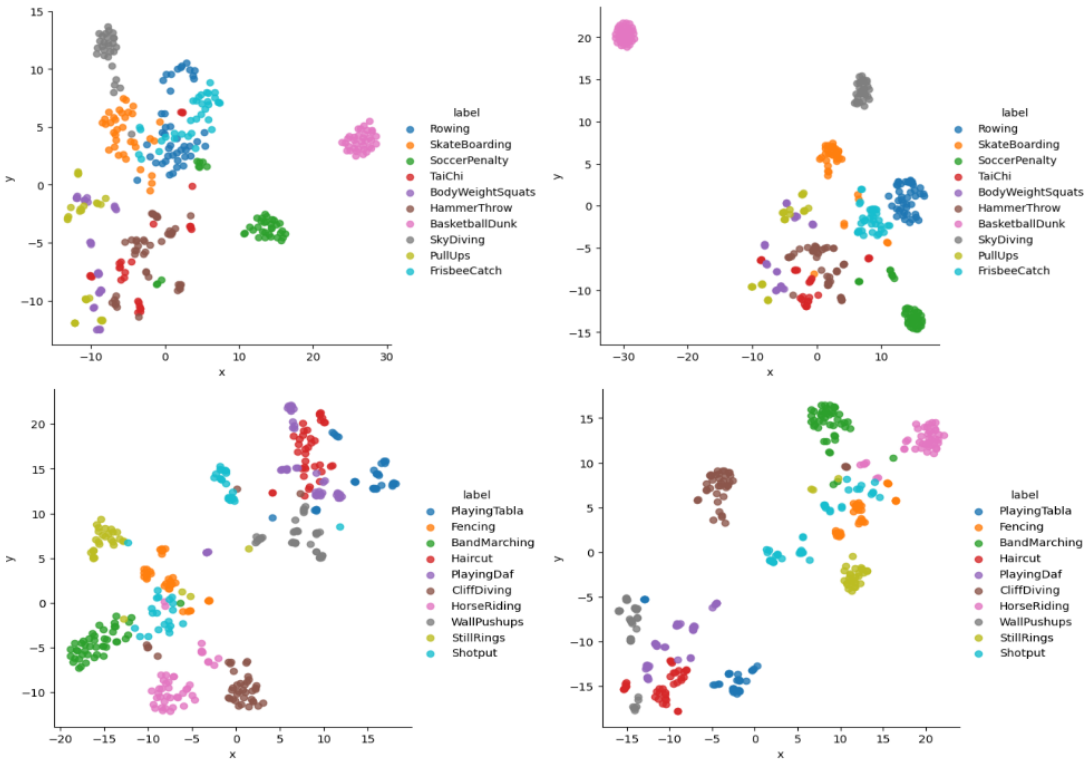


FIGURE 6.8 – Visualisations du tsne des représentations obtenues avec SimCLR sans *mixup* (à gauche) et avec (à droite).

FIGURE 6.9 – Comparaison des résultats de recherche de vidéo entre SimCLR sans *mixup* (à gauche) et avec (à droite). 5 images sont présentées pour chaque vidéo. Pour les deux versions, les deux mêmes requêtes sont utilisées (vidéo en haut). Les vidéos retournées sont les vidéos en dessous, ordonnées par similarité. Elles sont entourées de vert si correctes (même classe que la requête) et de rouge si incorrectes.

6.7 Conclusion

Dans ce chapitre, nous avons proposé une amélioration des méthodes contrastives comme SimCLR. Elle se base sur une augmentation des exemples positifs et négatifs par une opération d'interpolation. L'interpolation est effectuée sur les représentations haut niveau afin de limiter le temps de calcul supplémentaire. De plus, [VLB+19] montre que l'interpolation au niveau des représentations a de bonnes propriétés et permet d'améliorer significativement la généralisation en classification supervisé. Nous avons comparé la formulation proposée avec différentes formulations et montré sa supériorité pour l'apprentissage de représentations d'images. Nous avons montré l'apport en comparant par rapport à la méthode SimCLR classique sur l'apprentissage de représentations sur les images et sur les vidéos. Les méthodes sont évaluées principalement sur la classification linéaire après pré-apprentissage. La méthode proposée a des résultats au niveau de l'état de l'art pour l'apprentissage de représentations de vidéos. Nous l'évaluons aussi sur la reconnaissance d'actions par l'algorithme des k plus proches voisins et la tâche de recherche de vidéo. Les résultats obtenus sont concluants.

Différentes améliorations à partir de cette méthode sont envisageables. Il serait intéressant d'étudier les exemples les plus intéressants à utiliser dans l'interpolation pour les exemples positifs et pour les exemples négatifs. Une question intéressante est s'il est plus intéressant d'interpoler des exemples proches de la requête pour générer les exemples positifs. Même question pour les exemples négatifs. Finalement, ce n'est qu'une manière de générer des exemples supplémentaires. A la vue de l'apport de la méthode, il serait intéressant d'étudier d'autres méthodes de génération d'exemples qui pourraient s'avérer plus efficaces.

Chapitre 7

Conclusion et perspectives

7.1 Conclusion

Dans cette thèse, nous nous sommes intéressés à l'apprentissage non-supervisé de représentations pour l'analyse de séquences vidéos. Le but était donc de proposer ou d'améliorer des méthodes d'apprentissage non-supervisé pour pré-entraîner un réseau de neurones afin d'améliorer les performances en reconnaissance d'actions. Nous nous sommes plus particulièrement intéressés aux méthodes contrastives qui se sont récemment imposées dans l'état de l'art. Leur principale force est qu'elles effectuent une comparaison de similarité au niveau des représentations et qu'elles permettent de séparer les exemples en se basant sur des exemples négatifs. Les représentations apprises sont ensuite utilisées pour apprendre à reconnaître des actions par classification linéaire ou par K plus proches voisins par exemple. Les deux premières approches montrent comment utiliser l'information temporelle dans les vidéos pour l'apprentissage non-supervisé.

La première méthode montre comment adapter CPC pour prédire le futur de la vidéo. Voici les différentes conclusions que nous tirons des expériences menées. La formulation autorégressive est bien plus performante que celle non-autorégressive qui effectue beaucoup moins de prédictions. Il est aussi important d'utiliser des réseaux autorégressifs expressifs (composés de convolutions 3D par exemple) prenant en entrée des cartes de caractéristiques (pas de *global pooling* utilisé). Pour rendre la tâche plus difficile, il est nécessaire d'effectuer des augmentations importantes sur les entrées et de prendre en compte des exemples négatifs difficiles (provenant de la même vidéo mais à un moment différent). Différentes modalités et réseaux ont été étudiés et les résultats surpassent dans la plupart des cas ceux de l'état de l'art.

Certaines de ces constatations sont appliquées dans les méthodes suivantes.

La deuxième contribution propose un modèle plus souple qui agrège l'information de contexte et effectue les prédictions avec un modèle *transformer*. Il est montré qu'il est possible d'utiliser *BERT* et *XLNet* en combinaison avec la fonction de coût *InfoNCE* sur des séquences d'image de chiffres. La variante avec *BERT* est par la suite appliquée à la vidéo. Nous nous intéressons plus particulièrement à la stratégie de masquage. Nous montrons notamment que la stratégie bidirectionnelle est la plus performante et qu'il est donc important de prendre en compte la structure temporelle de la vidéo. Les résultats proposés sont compétitifs par rapport à l'état de l'art.

Un des points faibles des méthodes contrastives est le besoin d'un nombre important d'exemples négatifs donnant lieu à l'utilisation de *batchs* très importants. Les expériences ne sont donc pas réalisables sur des accélérateurs classiques. Les méthodes contrastives nécessitent aussi un temps d'apprentissage très long. Pour y remédier, nous proposons d'augmenter le nombre d'exemples par une opération de mixup sur les représentations en sortie du CNN. Cela permet d'augmenter le nombre d'exemples positifs et négatifs avec un surplus de calcul faible (la plus grosse partie des calculs est faite au niveau du CNN). Nous montrons l'apport de notre méthode en la comparant avec la méthode SimCLR et avec d'autres formulations du mixup. Les expériences sont réalisées sur des images et sur des vidéos. Les résultats obtenus sur les vidéos sont compétitifs avec l'état de l'art.

Nous avons donc proposé et mis en place des approches pour mieux prendre en compte l'aspect spatio-temporel des vidéos dans les méthodes d'apprentissage de représentations non-supervisées et proposé une méthode pour augmenter le nombre d'exemples sans augmenter le coût de calcul. Ces approches permettent d'obtenir des représentations haut niveau à partir desquelles la reconnaissance d'actions est plus facile à apprendre. Nous montrons notamment des résultats compétitifs par rapport aux autres méthodes non-supervisées de l'état de l'art en classification linéaire.

7.2 Perspectives

Les travaux réalisés dans cette thèse, laissent place à de nombreuses perspectives :

— La plupart des méthodes ont présenté des résultats sur différentes modalités

(images, flots optiques et différence d'images). Il serait intéressant de combiner ces modalités lors de l'apprentissage non supervisé. Par exemple, dans la méthode CPC pour la vidéo, le modèle pourrait prédire les représentations des images suivantes ainsi que des flots optiques suivants. Les résultats sur les flots optiques étant bien meilleurs que sur les images, cela permettrait de transférer l'information de mouvement contenue dans les flots optiques au réseau appris sur les images. On pourrait par la suite utiliser uniquement le réseau sur les images en espérant de meilleures performances.

— Considérant de la même manière les biens meilleures performances obtenues grâce aux flots optiques, il serait intéressant d'intégrer le calcul des flots optiques au réseau. Le précalcul des flots optiques ne serait donc pas nécessaire. Cette amélioration pourrait être apportée sur les différentes approches que nous avons proposées en intégrant par exemple un réseau de type *TVNet*.

— Les deux premières approches effectuent des prédictions temporelles. Cependant, les vidéos sont une donnée spatio-temporelle, il serait donc adéquat d'effectuer des prédictions à la fois spatiales et temporelles. En effet, les prédictions spatiales ont montré leur utilité pour l'apprentissage de représentations d'images mais elles peuvent aussi être utiles pour les vidéos. Les prédictions se font aussi uniquement à une échelle fixe, une prédiction multi-échelle serait donc intéressante à envisager.

— Nous avons proposé d'utiliser le mixup pour augmenter le nombre d'exemples négatifs mais ce n'est qu'une façon de faire. D'autres manières peuvent être étudiées et comparées.

— Finalement, un défaut de la méthode SimCLR par rapport aux autres est qu'elle ne contient pas d'exemples négatifs difficiles. Ainsi, la position temporelle des segments n'est pas prise en compte. De manière plus générale, la transformation effectuée pourrait être prise en compte. Il s'agirait alors de combiner la méthode SimCLR avec les méthodes de prédiction de transformations.

Finalement, les méthodes non-supervisées proposées peuvent être évaluées sur de nombreuses tâches et pas uniquement sur la reconnaissance d'actions. Il serait par exemple possible d'effectuer de la détection temporelle ou spatio-temporelle d'actions, de la détection d'interactions, du *video captioning* ou encore de la détection d'anomalies dans les vidéos.

Bibliographie

[AKK+19] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. *CoRR*, abs/1902.09229, 2019.

[AME18] Unaiza Ahsan, Rishi Madhok, and Irfan A. Essa. Video jigsaw : Unsupervised learning of spatiotemporal context for video action recognition. *CoRR*, abs/1808.07507, 2018.

[AMT+19] Humam Alwassel, Dhruv Mahajan, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering. *arXiv preprint arXiv :1911.12667*, 2019.

[BCL+20] A. Benzine, F. Chabot, B. Luvison, Q. Pham, and C. Achard. Pandanet : Anchor-based single-shot multi-person 3d pose estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6855–6864, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society.

[BDS18] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096, 2018.

[BEL+20] Sagie Benaim, Ariel Ephrat, Oran Lang, Inbar Mosseri, William T Freeman, Michael Rubinstein, Michal Irani, and Tali Dekel. Speednet : Learning the speediness in videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9922–9931, 2020.

[Ben12] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In Isabelle Guyon, Gideon Dror, Vincent Le-

maire, Graham Taylor, and Daniel Silver, editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 17–36, Bellevue, Washington, USA, 02 Jul 2012. PMLR.

[BFE+17] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic variational video prediction. *CoRR*, abs/1710.11252, 2017.

[BGS+05] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1395–1402 Vol. 2, 2005.

[BHB19] Philip Bachman, R. Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *CoRR*, abs/1906.00910, 2019.

[BJ17] Piotr Bojanowski and Armand Joulin. Unsupervised learning by predicting noise. *arXiv preprint arXiv :1704.05310*, 2017.

[BLPA19] Abdallah Benzine, Bertrand Luvison, Quoc Cuong Pham, and Catherine Achard. Deep, Robust and Single Shot 3D Multi-Person Human Pose Estimation from Monocular Images. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 584–588, Taipei, Taiwan, September 2019. The Institute of Electrical and Electronics Engineers Signal Processing Society, IEEE. ISBN : 978-1-5386-6249-6.

[BLPL06] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, page 153–160, Cambridge, MA, USA, 2006. MIT Press.

[BRB+18] Ishmael Belghazi, Sai Rajeswar, Aristide Baratin, R. Devon Hjelm, and Aaron C. Courville. MINE : mutual information neural estimation. *CoRR*, abs/1801.04062, 2018.

[BT16] Adrian Bulat and Georgios Tzimiropoulos. Human pose estimation

via convolutional part heatmap regression. *CoRR*, abs/1609.01743, 2016.

[BWMT18] Fabien Baradel, Christian Wolf, Julien Mille, and Graham W. Taylor. Glimpse clouds : Human activity recognition from unstructured feature points. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[BYPC15] Nicolas Ballas, Li Yao, Chris Pal, and Aaron C. Courville. Delving deeper into convolutional networks for learning video representations. *CoRR*, abs/1511.06432, 2015.

[CBFAB94] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Proceedings of 1st International Conference on Image Processing*, volume 2, pages 168–172 vol.2, 1994.

[CBJD18] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. *CoRR*, abs/1807.05520, 2018.

[CBMJ19] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Leveraging large-scale uncurated data for unsupervised pre-training of visual features. *CoRR*, abs/1905.01278, 2019.

[CdSB+19] Carlos Caetano, Jessica Sena de Souza, François Brémond, Jefersson A. dos Santos, and William Robson Schwartz. Skelemotion : A new representation of skeleton joint sequences based on motion information for 3d action recognition. *CoRR*, abs/1907.13025, 2019.

[Cha17] Florian Chabot. *Analyse fine 2D/3D de véhicules par réseaux de neurones profonds*. Theses, Université Clermont Auvergne, June 2017.

[CHEGCN15] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet : A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[CKL+18] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Multi-fiber networks for video recognition. *CoRR*, abs/1807.11195, 2018.

[CKNH20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv :2002.05709*, 2020.

[CMM+20] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv :2006.09882*, 2020.

[CNBH+18] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600, 2018.

[CNHZ19] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset, 2019.

[CNL11] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[Cru15] Michel Crucianu. Mexaction2 : action detection and localization dataset, 07 2015.

[CZ17] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750, 2017.

[CZM+19] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment : Learning augmentation policies from data. 2019.

[CZSL20] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. Randaugment : Practical automated data augmentation with a reduced search space. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and

H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18613–18624. Curran Associates, Inc., 2020.

[DB17] Emily Denton and Vighnesh Birodkar. Unsupervised learning of disentangled representations from video. *CoRR*, abs/1705.10915, 2017.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT : Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[DDF+20] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision, 2020.

[DGE15] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. *CoRR*, abs/1505.05192, 2015.

[DHG+14] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389, 2014.

[DKD16] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016.

[DS19] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. *CoRR*, abs/1907.02544, 2019.

[DSG16] Ali Diba, Vivek Sharma, and Luc Van Gool. Deep temporal linear encoding networks. *CoRR*, abs/1611.06678, 2016.

[DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 1 :886–893, 2005.

[EBC+10] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(19) :625–660, 2010.

[EMB+09] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 153–160, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[FBGG16] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. *CoRR*, abs/1611.06646, 2016.

[FDI+15] 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.

[FFMH18] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. *CoRR*, abs/1812.03982, 2018.

[FGL16] Chelsea Finn, Ian J. Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *CoRR*, abs/1605.07157, 2016.

[FHG+18] Lijie Fan, Wen-bing Huang, Chuang Gan, Stefano Ermon, Boqing Gong, and Junzhou Huang. End-to-end learning of motion representation for video understanding. *CoRR*, abs/1804.00413, 2018.

[FPZ16] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. *CoRR*, abs/1604.06573, 2016.

[GH10] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation : A new estimation principle for unnormalized statistical models. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[GKM+17] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fründ, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. *CoRR*, abs/1706.04261, 2017.

[GM19] Sina Ghassemi and Enrico Magli. Convolutional neural networks for on-board cloud screening. *Remote Sensing*, 11(12), 2019.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[GRG+17] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell. ActionVLAD : Learning spatio-temporal aggregation for action classification. In *CVPR*, 2017.

[GSK18] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *CoRR*, abs/1803.07728, 2018.

[GSR+18] Chunhui Gu, Chen Sun, David A. Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. Ava : A video dataset of spatio-temporally localized atomic visual actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[HEGN15] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet : A large-scale video benchmark for human activity understanding. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–970, 2015.

[HFLM+18] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv :1808.06670*, 2018.

[HFW+20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[HG20] Matthew Hutchinson and Vijay Gadepally. Video action understanding : A tutorial. *arXiv preprint arXiv :2010.06647*, 2020.

[HKS17] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. *CoRR*, abs/1708.07632, 2017.

[HKS18] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[HR11] Yanping Huang and Rajesh PN Rao. Predictive coding. *Wiley Interdisciplinary Reviews : Cognitive Science*, 2(5) :580–593, 2011.

[HSF+19] Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aäron van den Oord. Data-efficient image recognition with contrastive predictive coding. *CoRR*, abs/1905.09272, 2019.

[HXZ19] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[HXZ20a] Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning. *arXiv preprint arXiv :2008.01065*, 2020.

[HXZ20b] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning. *arXiv preprint arXiv :2010.09709*, 2020.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[IMS+17] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0 : Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[IZJ+17] H. Idrees, A. R. Zamir, Y. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155 :1–23, 2017.

[JG15] Dinesh Jayaraman and Kristen Grauman. Slow and steady feature analysis : Higher order temporal coherence in video. *CoRR*, abs/1506.04714, 2015.

[JGZ+13] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J. Black. Towards understanding action recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[JLRZ+13] Y.-G. Jiang, J. Liu, A. Roshan Zamir, I. Laptev, M. Piccardi, M. Shah, and R. Sukthankar. THUMOS challenge : Action recognition with a large number of classes. `/ICCV13-Action-Workshop/`, 2013.

[JMF20] Simon Jenni, Givi Meishvili, and Paolo Favaro. Video representation learning by recognizing temporal transformations. *arXiv preprint arXiv :2007.10730*, 2020.

[JT18] Longlong Jing and Yingli Tian. Self-supervised spatiotemporal feature learning by video geometric transformations. *CoRR*, abs/1811.11387, 2018.

[KCK18] Dahun Kim, Donghyeon Cho, and In So Kweon. Self-supervised video representation learning with space-time cubic puzzles. *CoRR*, abs/1811.09795, 2018.

[KCS+17a] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017.

[KCS+17b] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset, 2017.

[KHPG17] Bernhard Kratzwald, Zhiwu Huang, Danda Pani Paudel, and Luc Van Gool. Towards an understanding of our world by ganing videos in the wild. *CoRR*, abs/1711.11453, 2017.

[KJG+11a] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB : a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[KJG+11b] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb : A large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563, 2011.

[KSZQ20] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8) :5455–5516, 2020.

[KTDG16] Till Kroeger, Radu Timofte, Dengxin Dai, and Luc Van Gool. Fast optical flow using dense inverse search. *CoRR*, abs/1603.03590, 2016.

[KTS+14a] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural net-

works. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, June 2014.

[KTS+14b] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv :1312.6114*, 2013.

[KZS15] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. 2015.

[LC10] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[LGH18] Ji Lin, Chuang Gan, and Song Han. Temporal shift module for efficient video understanding. *CoRR*, abs/1811.08383, 2018.

[LHSY17] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. *CoRR*, abs/1708.01246, 2017.

[LL03] Laptev and Lindeberg. Space-time interest points. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 432–439 vol.1, 2003.

[LL04] I. Laptev and Tony Lindeberg. Velocity adaptation of space-time interest points. volume 2, pages 52– 56 Vol.1, 09 2004.

[LLDX17] Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P. Xing. Dual motion GAN for future-flow embedded video prediction. *CoRR*, abs/1708.00284, 2017.

[LLR18] Yin Li, Miao Liu, and James M. Rehg. In the eye of beholder : Joint learning of gaze and actions in first person video. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2) :91–110, November 2004.

[LP07] I. Laptev and P. Perez. Retrieving actions in movies. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007.

[LPH+17] Zelun Luo, Boya Peng, De-An Huang, Alexandre Alahi, and Li Fei-Fei. Unsupervised learning of long-term motion dynamics for videos. *CoRR*, abs/1701.01821, 2017.

[LSP+19] J. Liu, A. Shahroudy, M. L. Perez, G. Wang, L. Duan, and A. Kot Chichung. Ntu rgb+d 120 : A large-scale benchmark for 3d human activity understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2019.

[LSXW16] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-temporal LSTM with trust gates for 3d human action recognition. *CoRR*, abs/1607.07043, 2016.

[LTR+20] Ang Li, Meghana Thotakuri, David A. Ross, João Carreira, Alexander Vostrikov, and Andrew Zisserman. The ava-kinetics localized human actions video dataset, 2020.

[LZX+20] Junnan Li, Pan Zhou, Caiming Xiong, Richard Socher, and Steven CH Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv :2005.04966*, 2020.

[LZXP17] Chao Li, Qiaoyong Zhong, Di Xie, and Shiliang Pu. Skeleton-based action recognition with convolutional neural networks. *CoRR*, abs/1704.07595, 2017.

[MAS+20] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-End Learning of Visual Representations from Uncurated Instructional Videos. In *CVPR*, 2020.

[MAZ+18] Mathew Monfort, Alex Andonian, Bolei Zhou, Kandan Ramakrishnan, Sarah Adel Bargal, Tom Yan, Lisa Brown, Quanfu Fan, Dan Gutfruend, Carl Vondrick, and Aude Oliva. Moments in time dataset : one million videos for event understanding, 2018.

[MBBM19] Joanna Materzynska, Guillaume Berger, Ingo Bax, and Roland Memisevic. The jester dataset : A large-scale video dataset of human gestures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[MBG+18] Farzaneh Mahdisoltani, Guillaume Berger, Waseem Gharbieh, David Fleet, and Roland Memisevic. On the effectiveness of task granularity for transfer learning, 2018.

[MC18] Zhuang Ma and Michael Collins. Noise contrastive estimation and negative sampling for conditional models : Consistency and statistical efficiency. *CoRR*, abs/1809.01812, 2018.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781*, 2013.

[MCKA17] Chih-Yao Ma, Min-Hung Chen, Zsolt Kira, and Ghassan AlRegib. TS-LSTM and temporal-inception : Exploiting spatiotemporal dynamics for activity recognition. *CoRR*, abs/1703.10667, 2017.

[MCL15] Michaël Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *CoRR*, abs/1511.05440, 2015.

[MCL18] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. Posefix : Model-agnostic general human pose refinement network. *CoRR*, abs/1812.03595, 2018.

[MLS09] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2929–2936, 2009.

[MM20] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

[Moo96] T.K. Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6) :47–60, 1996.

[MRA+19] Mathew Monfort, Kandan Ramakrishnan, Alex Andonian, Barry A McNamara, Alex Lascelles, Bowen Pan, Quanfu Fan, Dan Gutfreund, Rogerio Feris, and Aude Oliva. Multi-moments in time : Learning and interpreting models for multi-action video understanding, 2019.

[MS20] David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information. volume 108 of *Proceedings of Machine Learning Research*, pages 875–884, Online, 26–28 Aug 2020. PMLR.

[MZH16] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Unsupervised learning using sequential verification for action recognition. *CoRR*, abs/1603.08561, 2016.

[ND16] Alejandro Newell and Jia Deng. Associative embedding : End-to-end learning for joint detection and grouping. *CoRR*, abs/1611.05424, 2016.

[NF16] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *CoRR*, abs/1603.09246, 2016.

[NHV+15] Joe Yue-Hei Ng, Matthew J. Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets : Deep networks for video classification. *CoRR*, abs/1503.08909, 2015.

[NYD16] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. *CoRR*, abs/1603.06937, 2016.

[OHP+11] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C. Chen, J. T. Lee, S. Mukherjee, J. K. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, and M. Desai. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR 2011*, pages 3153–3160, 2011.

[PGD+16] Deepak Pathak, Ross B. Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. *CoRR*, abs/1612.06370, 2016.

[PHC15] Viorica Patraucean, Ankur Handa, and Roberto Cipolla. Spatio-temporal video autoencoder with differentiable memory. *CoRR*, abs/1511.06309, 2015.

[PIT+15] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V. Gehler, and Bernt Schiele. Deepcut : Joint subset partition and labeling for multi person pose estimation. *CoRR*, abs/1511.06645, 2015.

[PKD+16] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders : Feature learning by inpainting. *CoRR*, abs/1604.07379, 2016.

[PR12] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2847–2854, 2012.

[PR18] A. J. Piergiovanni and Michael S. Ryoo. Representation flow for action recognition. *CoRR*, abs/1810.01455, 2018.

[PR20] AJ Piergiovanni and Michael S. Ryoo. Avid dataset : Anonymized videos from diverse countries, 2020.

[PSCZ14] Tomas Pfister, Karen Simonyan, James Charles, and Andrew Zisserman. Deep convolutional neural networks for efficient pose estimation in gesture videos. In Daniel Cremers, Ian D. Reid, Hideo Saito, and Ming-Hsuan Yang, editors, *ACCV (1)*, volume 9003 of *Lecture Notes in Computer Science*, pages 538–552. Springer, 2014.

[PSK+19] Huy Hieu Pham, Houssam Salmane, Louahdi Khoudour, Alain Crouzil, Pablo Zegers, and Sergio A. Velastin. Spatio–temporal image representation of 3d skeletal movements for view-invariant action recognition with deep convolutional neural networks. *Sensors*, 19(8), 2019.

[QZCT19] Guo-Jun Qi, Liheng Zhang, Chang Wen Chen, and Qi Tian. AVT : unsupervised learning of transformation equivariant representations by autoencoding variational transformations. *CoRR*, abs/1903.10863, 2019.

[RDS+14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2014.

[RLS+18] Fitsum A. Reda, Guilin Liu, Kevin J. Shih, Robert Kirby, Jon Barker, David Tarjan, Andrew Tao, and Bryan Catanzaro. Sdcnet : Video prediction using spatially-displaced convolution. *CoRR*, abs/1811.00684, 2018.

[RN18] A. Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[SBMS19] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Contrastive bidirectional transformer for temporal representation learning. *CoRR*, abs/1906.05743, 2019.

[SGS+18] Gunnar A. Sigurdsson, Abhinav Gupta, Cordelia Schmid, Ali Farhadi, and Karteek Alahari. Charades-ego : A large-scale dataset of paired third and first person videos, 2018.

[SJYS15] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. *CoRR*, abs/1510.00562, 2015.

[SKO+17] Shuyang Sun, Zhanghui Kuang, Wanli Ouyang, Lu Sheng, and Wei Zhang. Optical flow guided feature : A fast and robust motion representation for video action recognition. *CoRR*, abs/1711.11152, 2017.

[SLC04] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions : a local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 32–36 Vol.3, 2004.

[SMH07] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 791–798, New York, NY, USA, 2007. Association for Computing Machinery.

[SPMLF13] Javier Sánchez Pérez, Enric Meinhardt-Llopis, and Gabriele Facciolo. TV-L1 Optical Flow Estimation. *Image Processing On Line*, 3 :137–150, 2013.

[SVI+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.

[SXLW19] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. *CoRR*, abs/1902.09212, 2019.

[SZ14] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199, 2014.

[SZCL19] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with directed graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[SZS12a] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101 : A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.

[SZS12b] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101 : A dataset of 101 human actions classes from videos in the wild, 2012.

[TBF+14] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. C3D : generic features for video analysis. *CoRR*, abs/1412.0767, 2014.

[TDR+20] M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, and M. Lucic. On mutual information maximization for representation learning. In *8th International Conference on Learning Representations (ICLR)*, April 2020.

[TKI19] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *CoRR*, abs/1906.05849, 2019.

[TLYK17] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan : Decomposing motion and content for video generation. *CoRR*, abs/1707.04993, 2017.

[TS13] Alexander Toshev and Christian Szegedy. Deeppose : Human pose estimation via deep neural networks. *CoRR*, abs/1312.4659, 2013.

[TWT+18] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[TWTF19] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. *CoRR*, abs/1904.02811, 2019.

[vdOKE+16] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[vdOLV18] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

[VLB+19] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup : Better representations by interpolating hidden states. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6438–6447. PMLR, 09–15 Jun 2019.

[VLS16] Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. *CoRR*, abs/1604.04494, 2016.

[VPT15] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating the future by watching unlabeled video. *CoRR*, abs/1504.08023, 2015.

[VPT16] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *CoRR*, abs/1609.02612, 2016.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[VT17] C. Vondrick and A. Torralba. Generating the future with adversarial transformers. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2992–3000, July 2017.

[WG15] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. *CoRR*, abs/1505.00687, 2015.

[WGGH17] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *CoRR*, abs/1711.07971, 2017.

[WKSCL11] Heng Wang, Alexander Kläser, Cordelia Schmid, and Liu Cheng-Lin. Action Recognition by Dense Trajectories. In *CVPR 2011 - IEEE Conference on Computer Vision & Pattern Recognition*, pages 3169–3176, Colorado Springs, United States, June 2011. IEEE.

[WKSL13] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 103(1) :60–79, May 2013.

[WLZF18] D. Wei, J. Lim, A. Zisserman, and W. T. Freeman. Learning and using the arrow of time. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8052–8060, 2018.

[WRKS16] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. *CoRR*, abs/1602.00134, 2016.

[WW17] Hongsong Wang and Liang Wang. Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks. *CoRR*, abs/1704.02581, 2017.

[WXW+16] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks : Towards good practices for deep action recognition. *CoRR*, abs/1608.00859, 2016.

[WXYL18] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination. *CoRR*, abs/1805.01978, 2018.

[XNL+18] Jingwei Xu, Bingbing Ni, Zefan Li, Shuo Cheng, and Xiaokang Yang. Structure preserving video prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[XXZ+19] D. Xu, J. Xiao, Z. Zhao, J. Shao, D. Xie, and Y. Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10326–10335, 2019.

[YDY+19] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet : Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[YRJ+18] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts : Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 126(2) :375–389, 2018.

[YXL18] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. *CoRR*, abs/1801.07455, 2018.

[ZCDL17] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup : Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017.

[ZF13] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[ZIE16a] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. *CoRR*, abs/1603.08511, 2016.

[ZIE16b] Richard Zhang, Phillip Isola, and Alexei A. Efros. Split-brain autoencoders : Unsupervised learning by cross-channel prediction. *CoRR*, abs/1611.09842, 2016.

[ZLNH17] Yi Zhu, Zhen-Zhong Lan, Shawn D. Newsam, and Alexander G. Hauptmann. Hidden two-stream convolutional networks for action recognition. *CoRR*, abs/1704.00389, 2017.

[ZLX+16] Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, and Xiaohui Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks. *CoRR*, abs/1603.07772, 2016.

[ZMGL15] Junbo Jake Zhao, Michaël Mathieu, Ross Goroshin, and Yann LeCun. Stacked what-where auto-encoders. *CoRR*, abs/1506.02351, 2015.

[ZQWL19] Liheng Zhang, Guo-Jun Qi, Liqiang Wang, and Jiebo Luo. AET vs. AED : unsupervised representation learning by auto-encoding transformations rather than data. *CoRR*, abs/1901.04596, 2019.

[ZTTY17] Hang Zhao, Antonio Torralba, Lorenzo Torresani, and Zhicheng Yan. Hacs : Human action clips and segments dataset for recognition and temporal localization, 2017.

[ZXT20] Xikun Zhang, Chang Xu, and Dacheng Tao. Context aware graph convolution for skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[ZZH+17] Na Zhao, Hanwang Zhang, Richang Hong, Meng Wang, and Tat-Seng Chua. Videowhisper : Toward discriminative unsupervised video feature learning with attention-based recurrent neural networks. *IEEE Transactions on Multimedia*, 19 :2080–2092, 2017.

[ZZY19] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. *CoRR*, abs/1903.12355, 2019.

Chapitre 8

Annexes

8.1 Résultats complémentaires CPC Vidéo

8.1.1 Visualisations t-SNE pour différentes modalités

Des visualisations t-SNE sont présentées pour les représentations apprises par la méthode CPC avec différentes modalités d'entrée (image, différence d'images et flots optiques).

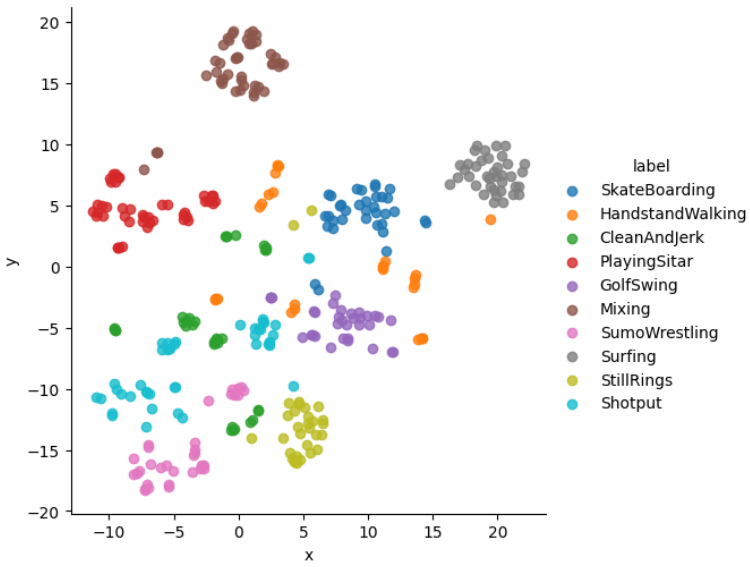


FIGURE 8.1 – t-SNE pour les images

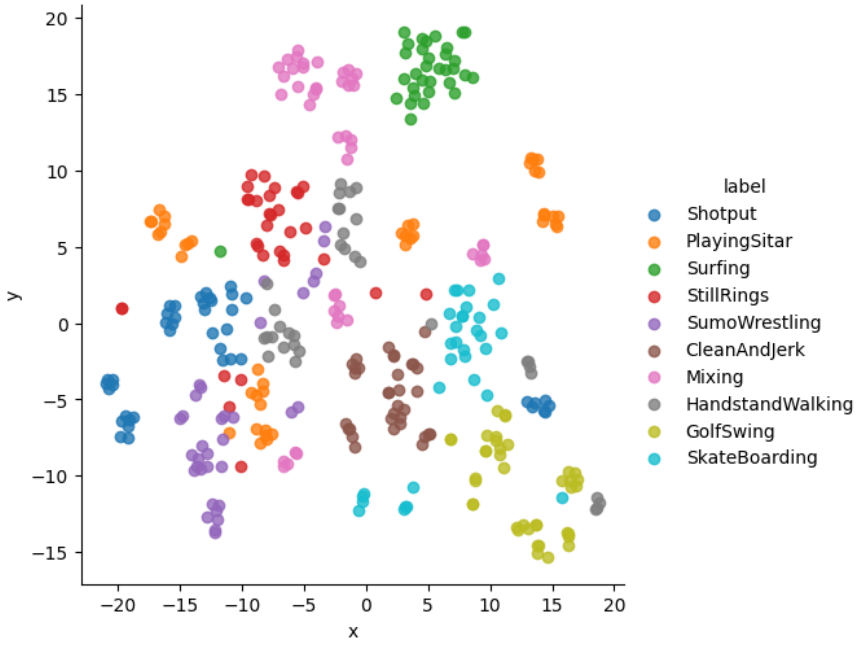


FIGURE 8.2 – t-SNE pour les différences d'images

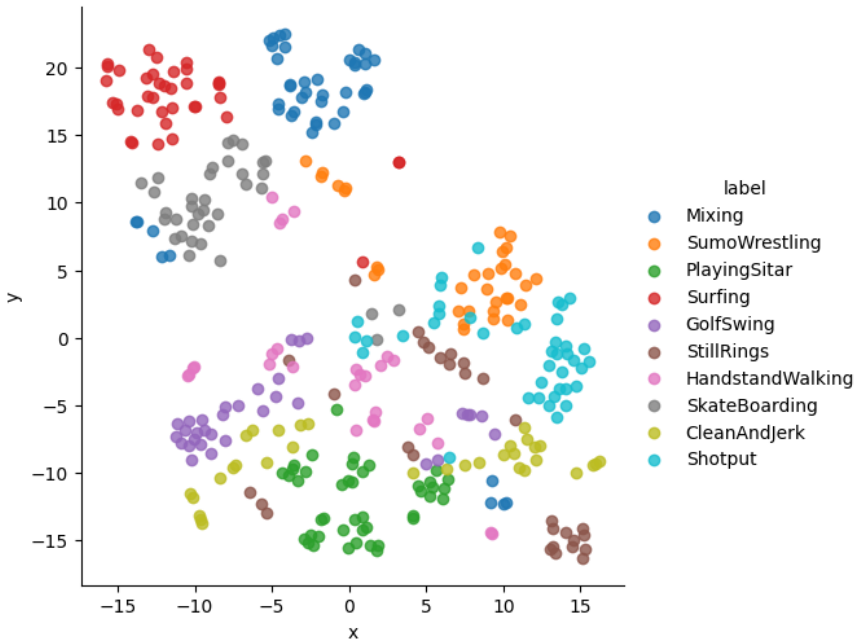


FIGURE 8.3 – t-SNE pour les flots optiques

8.1.2 Evolution des t-SNE en fonction du temps d'apprentissage non-supervisé

Les visualisations t-SNE issues des représentations provenant de différents temps d'entraînement non-supervisé sont présentés. Nous remarquons que la qualité des représentations s'améliorent car les classes sont de mieux en mieux séparées.

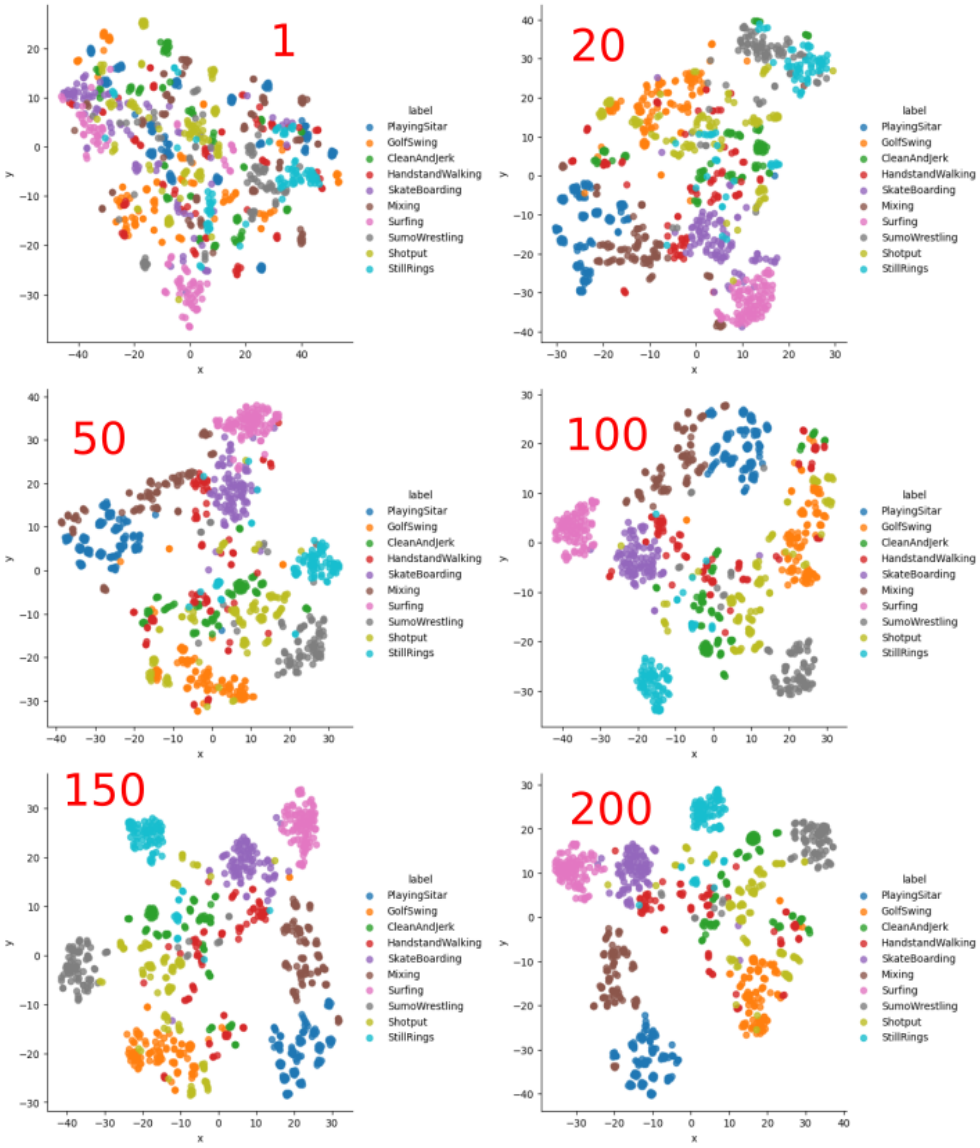
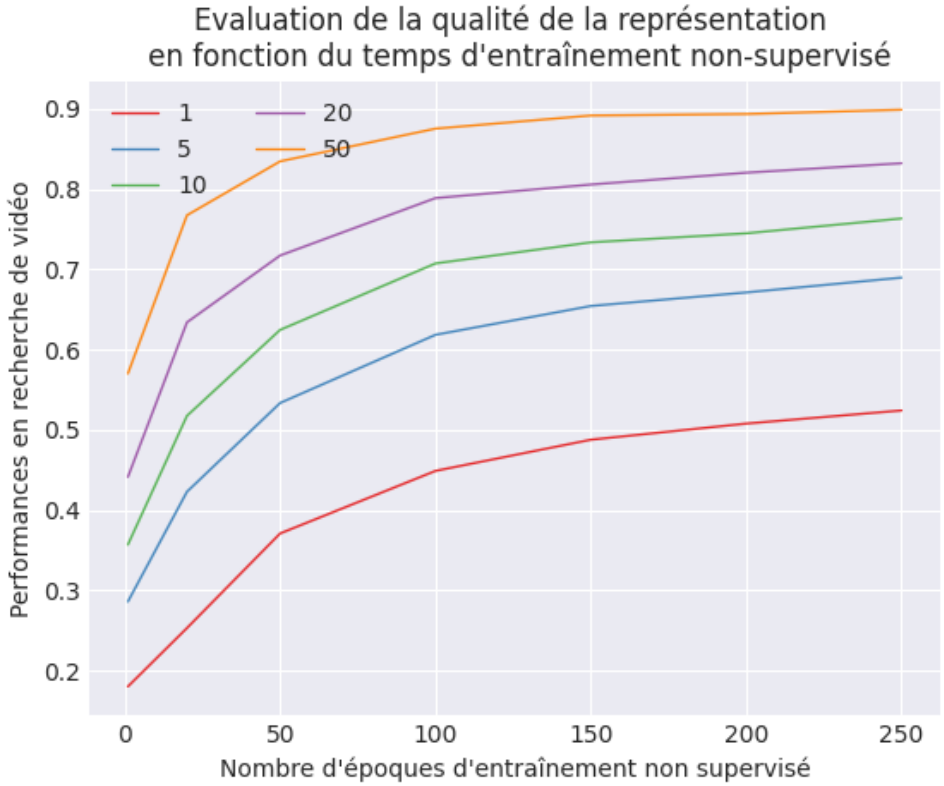


FIGURE 8.4 – Evolution des t-SNE en fonction du nombre d'époques de pré-entraînement (indiqué en rouge)

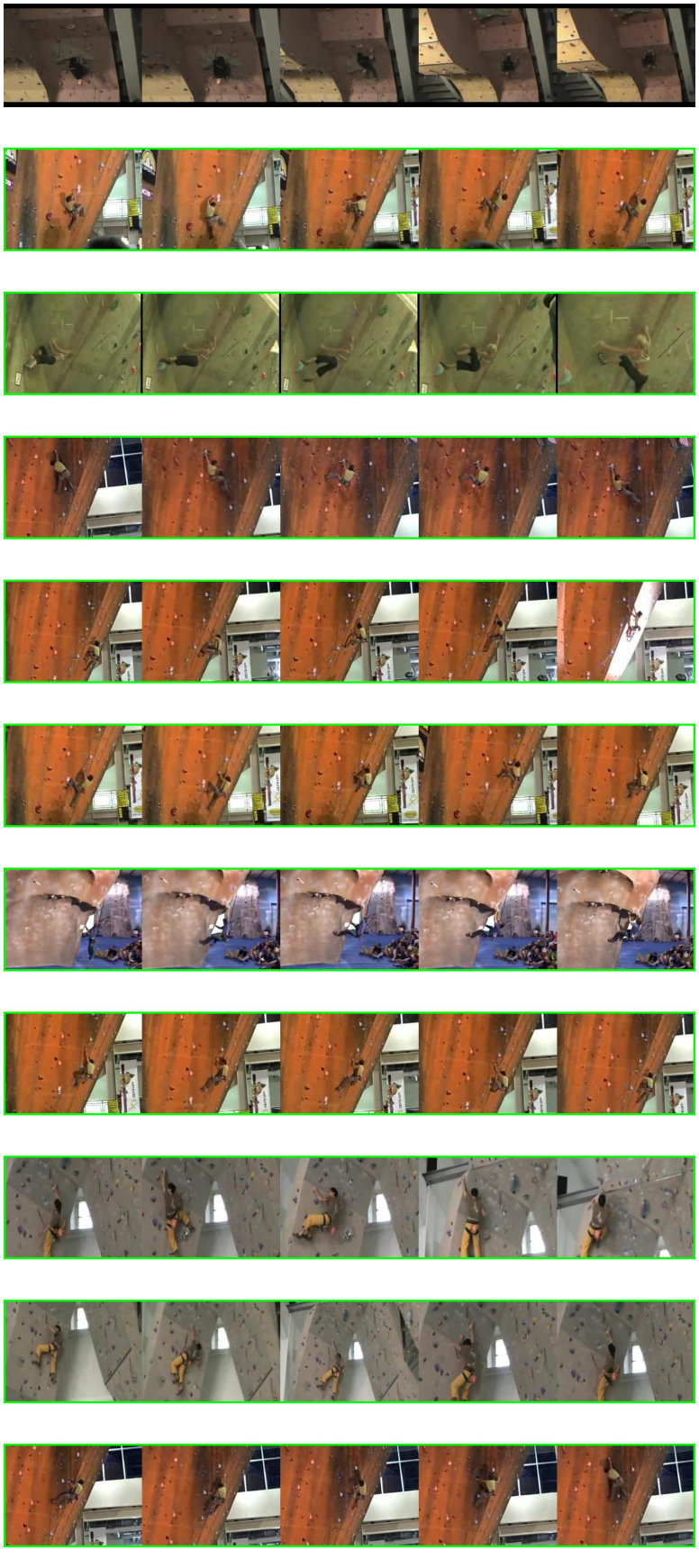
8.1.3 Evolution des scores de recherche de vidéos

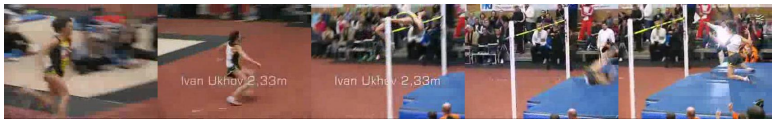
Ce graphique montre l'évolution des performances en recherche de vidéo (pour différents nombres de vidéos retournées [1,5,10,20,50] en fonction du temps d'apprentissage non-supervisé. Ces performances augmentent ce qui montre bien l'amélioration des représentations plus l'apprentissage non-supervisé avance.

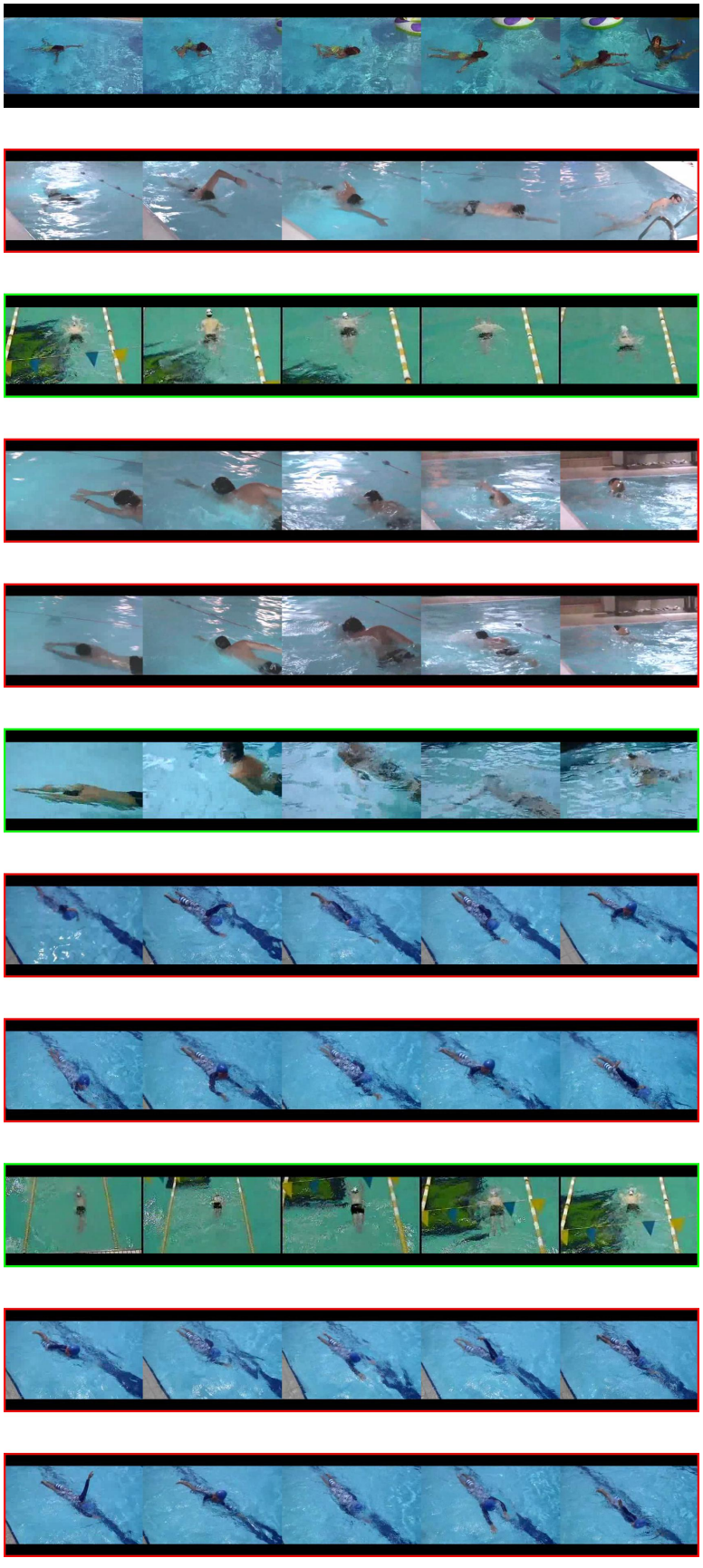


8.1.4 Résultats de recherche de vidéos

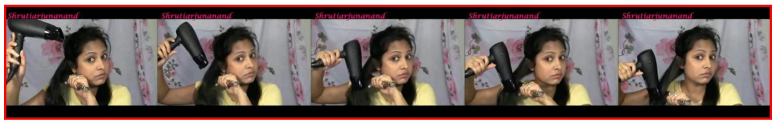
Les résultats de recherche de vidéo ci-dessous sont obtenus avec la méthode CPC Vidéo sur les images et le resnet3D. 5 images par vidéo sont présentées. La vidéo requête est tout en haut et les vidéos retournées classées par similarité en dessous. Les vidéos entourées en vert sont correctes (elles ont la même classe que la vidéo requête) et celles eontourées en rouge sont incorrectes.




Ivan Ukhov 2,33m
Ivan Ukhov 2,33m



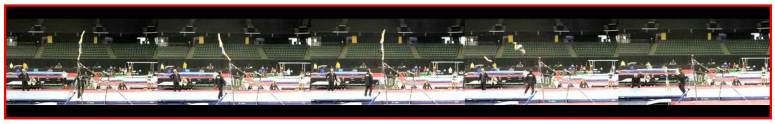






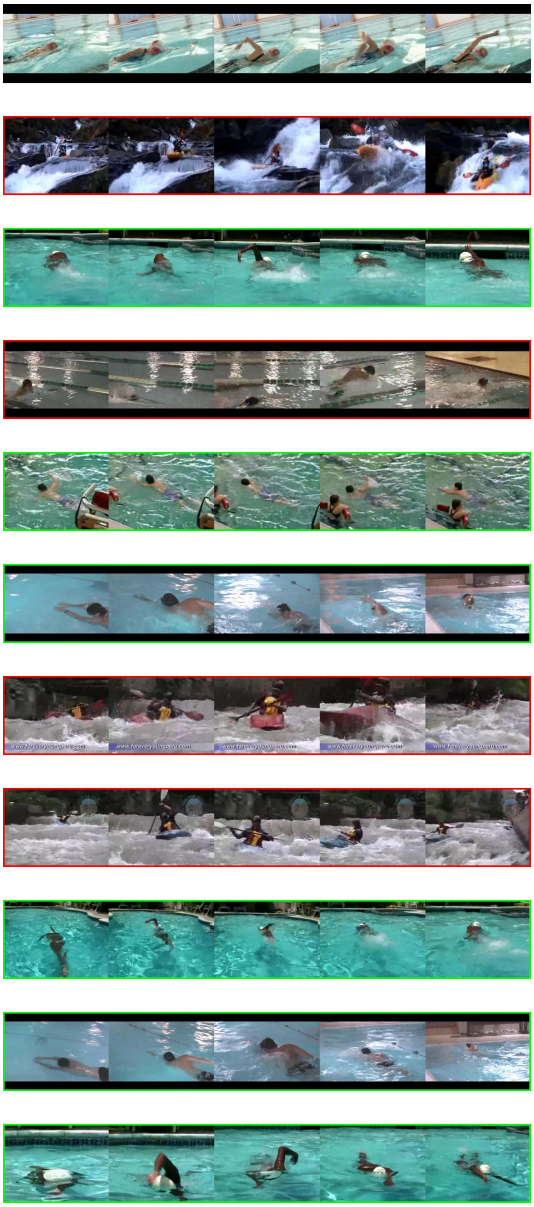




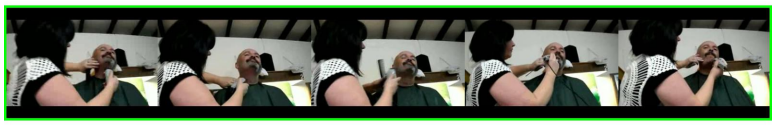
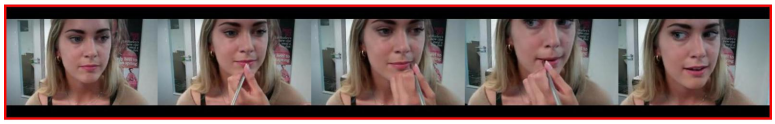


8.2 Résultats complémentaires BERT Video

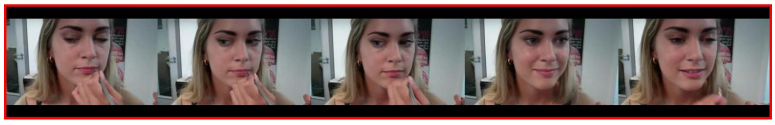
Les résultats de recherche de vidéo ci-dessous sont obtenus avec un pré-apprentissage avec BERT NCE utilisant un masquage passé/futur et avec comme modalité d'entrée les flots optiques.

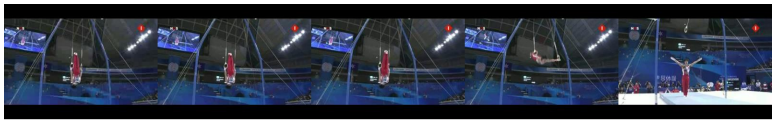


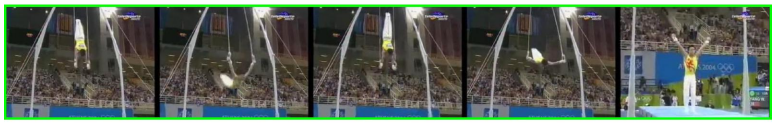


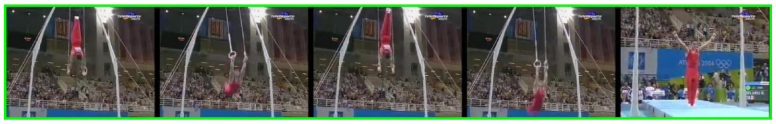



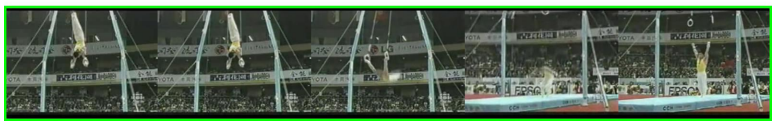


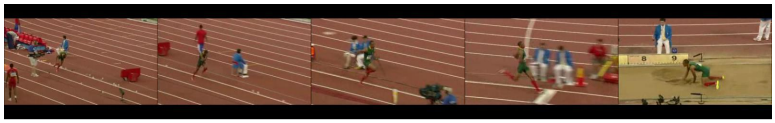


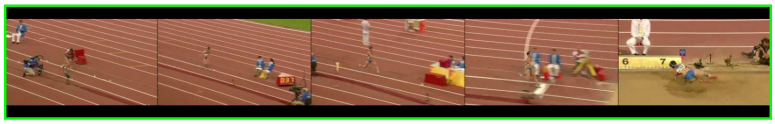



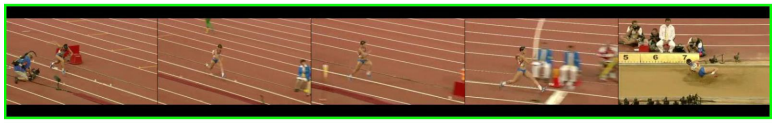


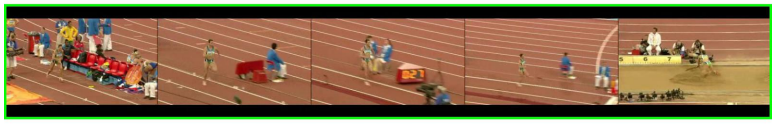


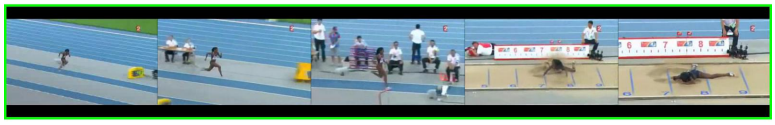


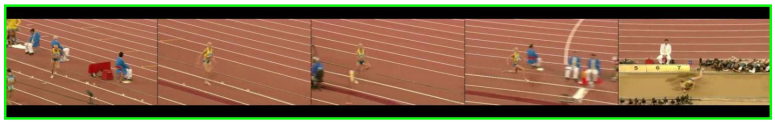


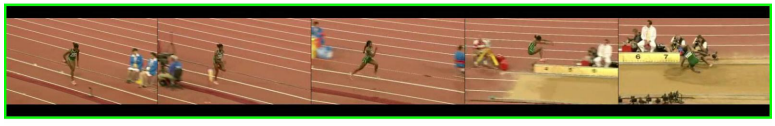


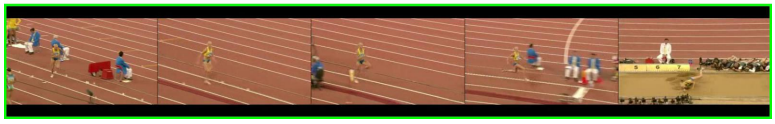


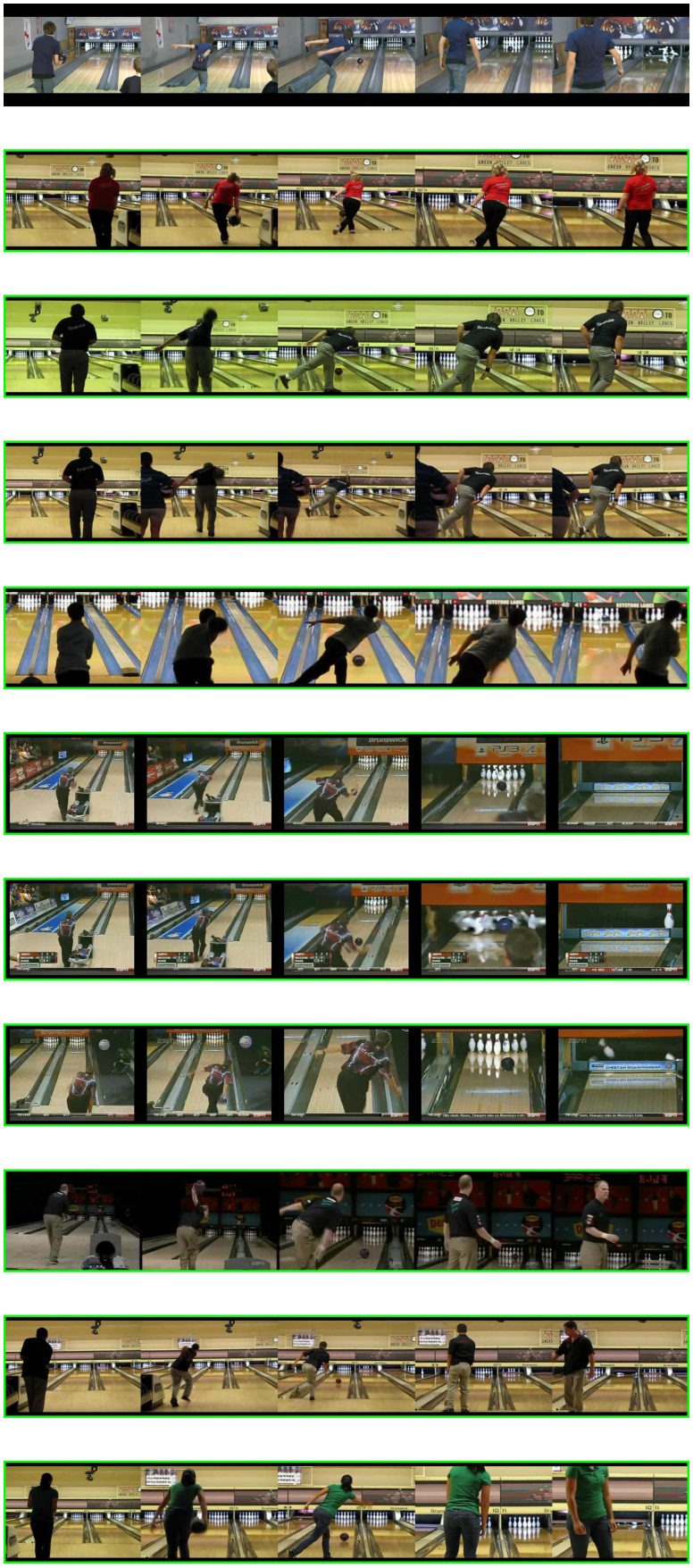






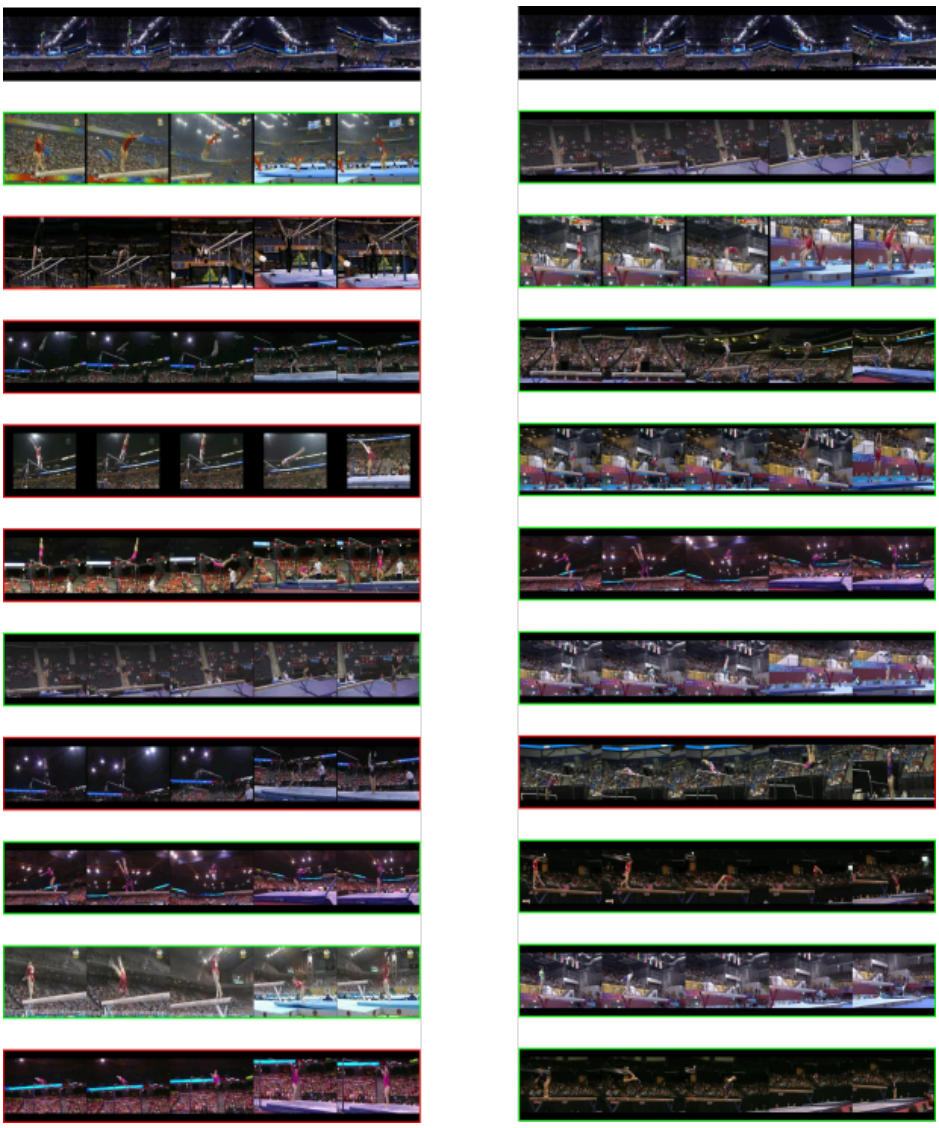


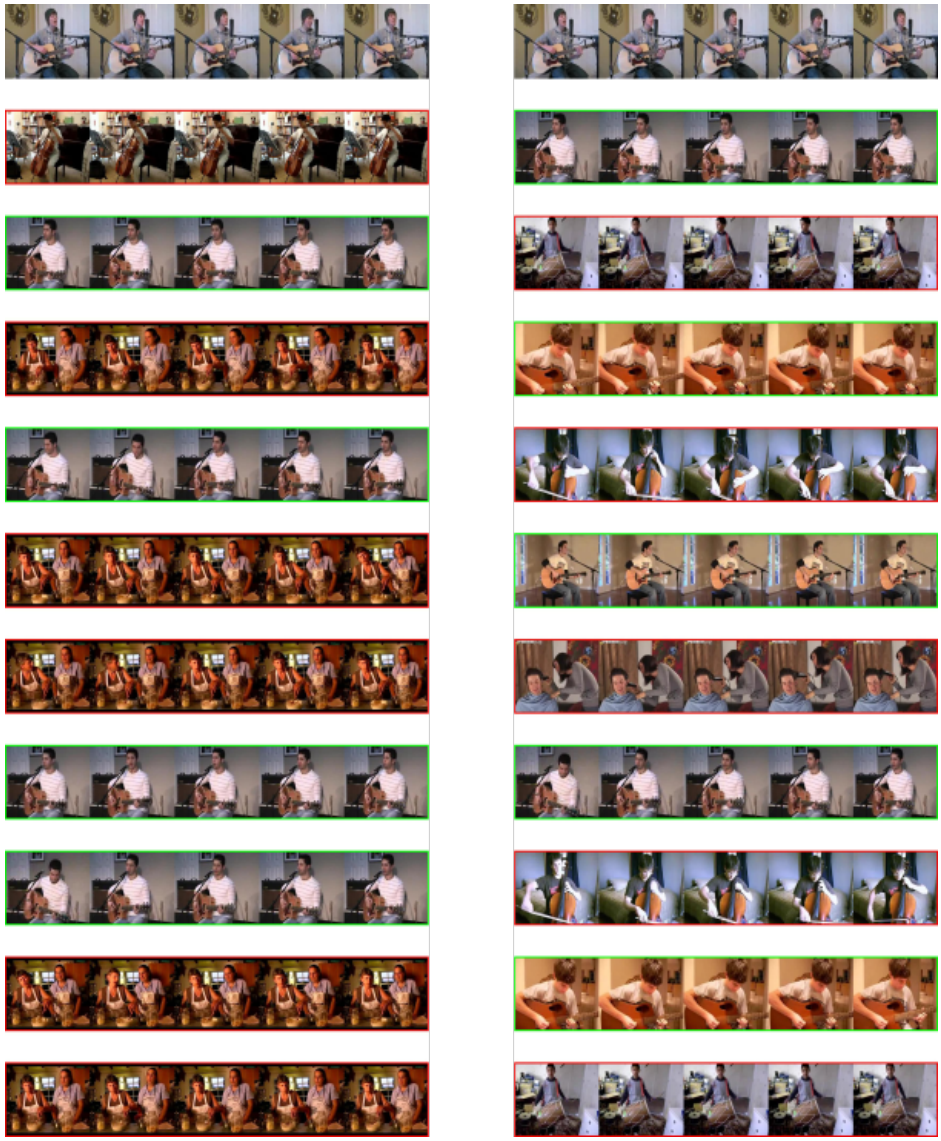




8.3 Résultats complémentaires SimCLR mixup

Les résultats de recherche de vidéos sont obtenus avec un pré-apprentissage SimCLR sur UCF-101 avec un réseau resnet3D. Les résultats à gauche sont sans mixup et les résultats à droite avec mixup.





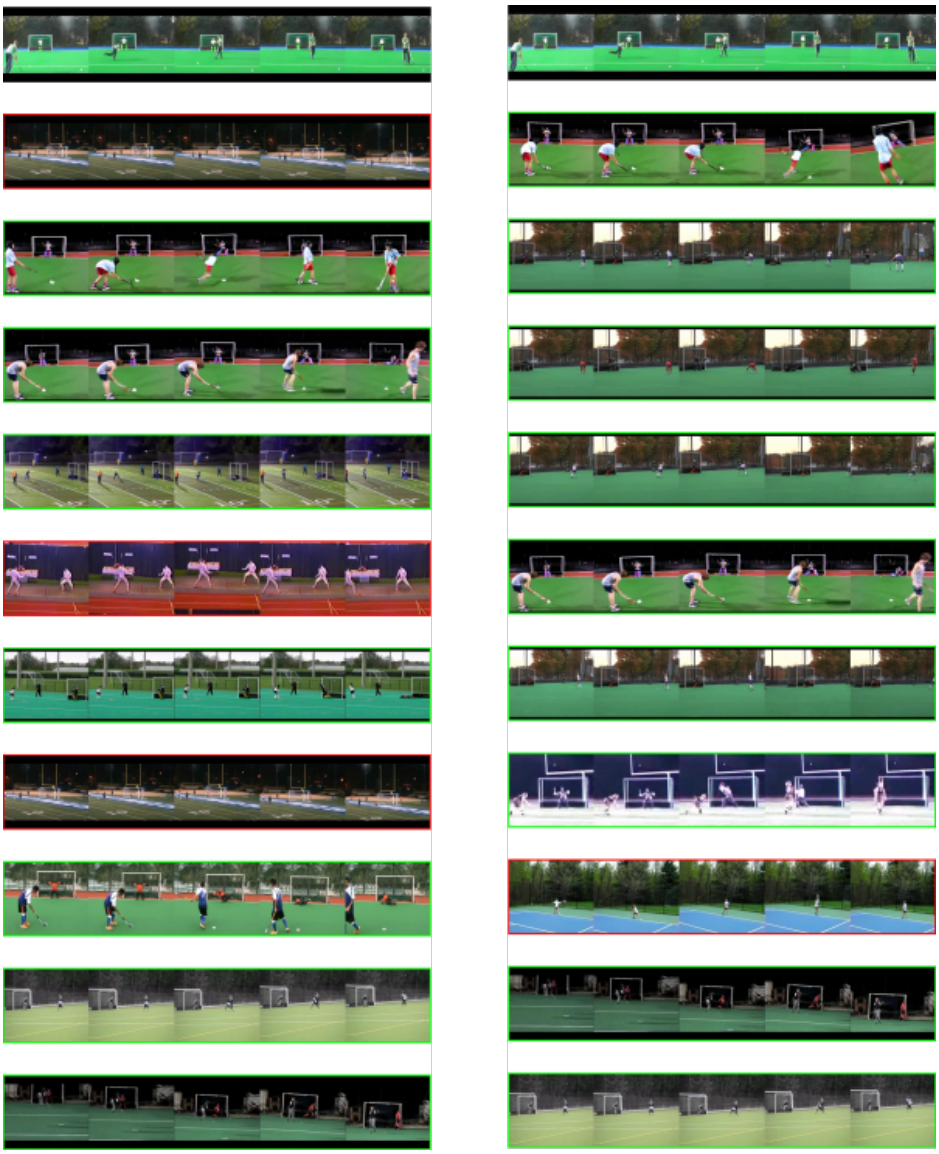


Table des figures

1.1 Schéma d'un réseau de neurones multicouche (2 couches cachées de largeur 3 sont représentées). 3

1.2 Visualisation des activations d'un réseau de neurone convolutionnel (*Inception* [SVI+16]) pour différentes couches. 4

1.3 Illustration des opérations classiques utilisées dans les réseaux convolutionnels (repris de [Cha17]) 5

1.4 Schéma d'une connexion résiduelle (repris de [HZRS15]). Le groupe est composé de 2 couches et l'entrée est additionnée à la sortie. 6

1.5 Architecture d'un Resnet18 (repris de [GM19]). Il est composé d'une convolution suivie de 4 groupes de 2 blocks résiduels. A l'entrée de chaque groupe, la dimension spatiale est réduite par 2 dans les 2 dimensions. 7

1.6 Architecture d'un réseau récurrent LSTM. 8

1.7 Schéma représentant le principe de pré-apprentissage supervisé de réseaux de neurones. 9

1.8 Schéma de la méthode de pré-apprentissage GPT. U est l'ensemble des mots précédents utilisés pour prédire le mot courant (w_6 ici). *Masked transformer* est le réseau de neurone à apprendre. 13

1.9 Schéma d'un modèle auto-encodeur. L'erreur de reconstruction entre l'entrée x et la sortie r : $||x-r||^2$ est minimisée. Une régularisation est appliquée sur le code z en sortie de l'encodeur. 14

1.10 Schéma général d'un *GAN*. Le réseau discriminateur D doit différencier les exemples provenant des données de ceux créés par le générateur G. 15

2.1 Schéma général des différentes familles de méthodes d'apprentissage non-supervisé. Les prédictions sont notées en rouge et les valeurs à prédire en vert. (1) : méthodes génératives, (2) : méthodes auto-supervisées sur les transformations, (3) : méthodes basées sur les pseudo-labels, (4) : méthodes contrastives basées sur l'information mutuelle. 20

2.2 Schéma regroupant les différentes méthodes d'apprentissage auto-supervisé pour les images. 26

2.3 Schéma regroupant les différentes méthodes d'apprentissage auto-supervisé pour les vidéos. 27

2.4 Schéma général des méthodes basées sur les pseudo-labels. Les étapes 1 et 2 sont alternées. La génération des pseudo-labels dépend de la méthode, c'est par exemple un clustering avec k-moyennes pour la méthode Deep Cluster. 29

2.5 Schéma de la méthode CPC appliquée à l'image en suivant l'article [HSF+19]. 40

2.6 Visualisation des reconstructions des 3 dernières lignes de l'image. La 1ère ligne d'images montre la vérité terrain et la 2ème ligne les prédictions de CPC. 2 lignes sont à chaque fois ignorées pour les prédictions, ainsi la 4ème ligne est prédite uniquement à partir de la 1ère, la 5ème à partir des 2 premières lignes et ainsi de suite. Les 2 premières colonnes proviennent du dataset Imagenette et les 2 autres du dataset Imagewoof. 41

2.7 Visualisation des résultats de recherche d'images à partir des représentations obtenues par l'apprentissage de CPC sur Imagenet. L'image en haut à gauche est la requête est les autres images sont les plus proches. Une similarité cosinus sur les représentations est utilisée. . . 42

3.1 Illustration du problème de reconnaissance d'actions avec différents exemples associés à leur classe. 46

3.2 Différentes localisations de la fusion (provenant de [NHV+15]). 48

3.3 Schéma du principe du flot optique. On considère ici le mouvement (flèches rouges) d'un carré entre 2 images consécutives I_t et I_{t+1}. u et v sont les 2 composantes du flot optique. 50

3.4 Schema de l'architecture *2 stream*. Le réseau bleu constitue la branche spatiale et le réseau rouge la branche temporelle. La sortie est la probabilité pour les différentes classes. 51

3.5 Exemples de vidéos de UCF-101 en haut et HMDB51 en bas pour les classes High Jump et Shoot Bow. Une séquence d'image pour chaque vidéo est présentée. 55

4.1 Schema illustrant la méthode *Contrastive Predictive Coding* pour les vidéos. Les différents segments vidéos sont encodés par un CNN. Les cartes de caractéristiques passées en sortie sont aggrégées par un modèle autorégressif. Le contexte c_t permet de prédire les représentations des segments futurs 61

4.2 Représentation des principaux points de la méthode CPC sur la vidéo. (1) Utilisation de réseaux autorégressifs basés sur les convolutions 3D masquées. (2) Utilisation d'augmentations différentes sur les segments. (3) Utilisation d'exemples négatifs difficiles. (4) Prédiction à partir de tous les instants de la vidéo et de tous les segments restants. 64

4.3 Schéma de différents modèles d'aggrégation/autorégressifs. Pour les 2 premiers, avec des convolutions causales, le modèle devient autorégressif. 64

4.4 Schema de la variante bidirectionnelle de la méthode CPC pour les vidéos. Deux modèles autorégressifs sont utilisés pour effectuer des prédictions dans les deux sens : vers le futur et vers le passé. 67

4.5 Schéma de la méthode d'évaluation de l'apprentissage de représentations sur les vidéos. La partie CNN est la partie pré-entraînée. 70

4.6 Visualisation des différentes modalités utilisées pour cette méthode (images, flots optiques et différence d'images) 71

4.7 Précision non-supervisée (sur la classification entre l'exemple positif et les exemples négatifs) en fonction du pas de prédiction (k lorsque l'on prédit z_{t+k} à partir de c_t). Pour la courbe en bleu, des exemples négatifs difficiles sont utilisés ainsi que des transformations différentes entre les segments et non pour la courbe en orange. 75

4.8 Performances de reconnaissance d'actions avec les flots optiques pour différentes quantités de données labélisées utilisées. Les résultats avec et sans pré-entraînement sont comparés. Le premier diagramme montre les résultats en classification linéaire et le deuxième en finetuning 76

4.9 Visualisations t-SNE des représentations obtenues à partir des flots optiques sur le test de UCF-101. Seulement 10 classes sur les 101 sont présentées pour une meilleure visualisation. La figure gauche montre la visualisation pour un réseau initialisé aléatoirement et la figure droite avec notre pré-entraînement. 83

4.10 Vidéos les plus similaires à la requête en utilisant la distance cosinus sur les représentations apprises de manière non supervisée. Les flots optiques sont utilisés et nous considérons les représentations comme la moyenne des valeurs z_t sur 10 segments. La requête est en rouge et les résultats sont en vert. 85

5.1 Schéma représentant la méthode BERT associé à la fonction de coût InfoNCE. Dans cet exemple, l'élément x_t est masqué ($m_t = 0$) et va être prédit à partir des autres éléments. La prédiction p_t est claculée à partir de la sortie o_t du transformer. L'exemple positif est alors z_t et les exemples négatifs sont les autres éléments de la séquence (comme h_1 et h_T par exemple) mais aussi du *batch* 93

5.2 Schéma représentant la méthode XLNet associée à la fonction de coût InfoNCE. La séquence est permutée aléatoirement. L'élément suivant est prédit à partir des éléments précédents dans l'ordre de la permutation. Un encoding positionnel est ajouté et l'architecture à 2 voies de XLNet est utilisée. La sortie $o_{\sigma^{-1}(t)}$ est utilisée pour effectuer la prédiction $p_{\sigma^{-1}(t)}$. La fonction InfoNCE est utilisée pour comparer la prédiction avec $z_{\sigma^{-1}(t)}$ et les exemples négatifs. 95

5.3 Visualisations t-SNE des représentations obtenues par BERT et XLNet comparées au supervisé et à un réseau initialisé aléatoirement 97

5.4 Schéma représentant les différentes stratégies de masquage pour la vidéo 98

5.5 Visualisations du tsne des représentations obtenues avec BERT en utilisant les flots optiques et un masquage bidirectionnel 103

5.6 Visualisations des résultats de recherche de vidéos. Les 2 premières vidéos en haut sont les requêtes (5 images de la vidéo sont représentées) et celles en dessous sont les vidéos retournées. Les vidéos retournées entourées en rouge sont incorrectes (elles n'appartiennent pas à la même classe que la requête) et les vertes sont correctes. 105

6.1 Schéma des augmentations utilisées sur les vidéos pour générer la requête et la clé dans SimCLR. Un crop temporel puis un crop spatial et d'autres augmentations (*color jittering* par exemple) sont appliquées. . 110

6.2 Schéma de la méthode de mixup sur les entrées. Le réseau f a pour entrée l'interpolation de deux images et doit prédire l'interpolation des deux vérités terrain. 111

6.3 Schéma de la méthode de mixup sur les représentations intermédiaires. Le réseau de classification est découpé en deux. L'interpolation se fait au niveau d'une représentation intermédiaire (à la fin de la première partie de réseau de classification). Le but est encore de prédire l'interpolation des vérités terrain. 112

6.4 Schéma de la méthode SimCLR avec utilisation du mixup pour augmenter les exemples positifs et négatifs. La clé est interpolée au niveau des représentations avec d'autres exemples. La fonction de coût InfoNCE est appliquée entre la clé et ces exemples mixés. Les exemples négatifs sont aussi issus de l'opération de *mixup*. La fonction de coût est ensuite pondérée par λ. 113

6.5 Comparaison du score avec et sans mixup en fonction du temps de pré-apprentissage. 119

6.6 Comparaison du score avec et sans *mixup* en fonction du temps de pré-apprentissage sur UCF-101. 123

6.7 Comparaison des scores de classification avec les K plus proches voisins et de recherche de vidéo pour différentes valeurs du nombre de *mixup* pour les exemples positifs. 124

6.8 Visualisations du tsne des représentations obtenues avec SimCLR sans *mixup* (à gauche) et avec (à droite). 125

6.9 Comparaison des résultats de recherche de vidéo entre SimCLR sans *mixup* (à gauche) et avec (à droite). 5 images sont présentées pour chaque vidéo. Pour les deux versions, les deux mêmes requêtes sont utilisées (vidéo en haut). Les vidéos retournées sont les vidéos en dessous, ordonnées par similarité. Elles sont entourées de vert si correctes (même classe que la requête) et de rouge si incorrectes. 126

8.1 t-SNE pour les images XXIII
8.2 t-SNE pour les différences d'images XXIV
8.3 t-SNE pour les flots optiques XXIV
8.4 Evolution des t-SNE en fonction du nombre d'époques de pré-entraînement (indiqué en rouge) XXV

Liste des tableaux

1.1 Comparaison des scores de reconnaissance sur différentes bases de données d'images entre sans initialisation, classification linéaire et *finetuning*. Le pré-apprentissage est effectué sur *Imagenet*. Les résultats sont repris de [CKNH20]. 9
1.2 Exemples de relations haut niveau obtenues à partir des plongements lexicaux. La représentation du mot de droite est translatée par la relation et le mot sélectionné est le plus proche trouvé. 12

2.1 Résultats de pré-entraînement par la méthode CPC sur les datasets Imagenette, Imagewoof et Imagewang. 42
2.2 Comparaison de différents types de transfert sur la base Imagewoof. Patch indique que la moyenne des représentations sur chaque patch est utilisée et non la représentation sur toute l'image. 43

3.1 Tableau des principales bases de données de reconnaissance d'actions avec différentes informations comme l'année, le nombre de citations, le nombre d'actions et d'exemples. Les acteurs sont séparés entre humains (H) et non humains (N). (C) indique une annotation globale, (T) une annotation temporelle et (S) une annotation spatio-temporelle avec des boîtes englobantes. Adapté de la table 2 de [HG20]. 56
3.2 Résultats des méthodes de l'état en reconnaissance d'actions sur UCF-101 et HMDB51. 57
3.3 Comparatif des résultats de reconnaissance d'actions sur UCF-101 et HMDB51 avec et sans pré-entraînement sur Kinetics 57
3.4 Comparaison de différentes architectures de réseaux convolutionnels sur les branches temporelle et spatiale 58
3.5 Résultats de la méthode *2-stream* 58

4.1 Performance de pré-entraînement sur les différents splits de UCF-101 en utilisant les flots optique. La première partie montre les résultats de classification linéaire et la seconde les résultats de finetuning (délimitées par des lignes doubles). Ces parties sont séparées entre nos résultats et ceux de l'état de l'art. Les résultats sur les 3 splits de UCF-101 et sur la moyenne des 3 splits de HMDB51 sont détaillés comme dans [WLZF18]. La dernière partie montre les résultats des méthodes utilisant les images comme modalité d'entrée. 76

4.2 Evaluation du pré-entraînement sur différentes modalités et en transférant sur différentes bases de données. Les résultats de l'état de l'art sont présentés à la fin de chaque partie de la modalité. 77

4.3 Evaluation du pré-entraînement en combinant plusieurs modalités . . 78

4.4 Résultats de classification linéaire sur UCF-101 après pré-entraînement sur UCF-101 avec le modèle sans autorégressivité. Différents modèles d'aggrégation sont comparés ainsi que sans pré-entraînement. Différents pourcentages de données annotés sont utilisés 79

4.5 Résultats du modèle bidirectionnel en classification linéaire sur les flots optiques 79

4.6 Résultats de reconnaissance d'actions suite au pré-entraînement de CPC en utilisant les améliorations présentées dans la partie 4.2.4 et les réseaux convolutionnels 3D 80

4.7 Résultats de reconnaissance d'actions en utilisant les k plus proches voisins sur les représentations obtenues par apprentissage non-supervisé. 81

4.8 Etude d'ablation du modèle : La 1ère partie correspond aux résultats du modèle de référence. La 2ème à l'ablation sur le nombre de segments futurs prédits (pour une prédiction, seulement z_{t+1} est prédit à partir de c_t). La 3ème compare les différents modèles autoregressifs. Finalement, les 2 dernières lignes montrent les résultats sans transformations différentes sur les entrées et sans exemples négatifs difficiles. La précision en classification linéaire utilisant la représentation pré-apprise sur UCF-101 avec les flots optiques est présentée (sur le split 2). 82

4.9 Scores de recherche de vidéos basés sur les représentations obtenues par pré-apprentissage non-supervisé. 84

5.1 Resultats de classification linéaire après un pré-entraînement avec BERT et XLNet comparé à l'apprentissage supervisé 96
5.2 Performances de reconnaissance d'actions par classification linéaire sur les représentations obtenues par notre méthode 100
5.3 Résultats de reconnaissance utilisant les k plus proches voisins à partir des représentations non-supervisées. 101
5.4 Résultats de recherche de vidéos se basant sur la similarité cosinus sur les représentations 102

6.1 Etude des différents paramètres de la méthode SimCLR sur les images : normalisation, type de transformation. Les résultats de classification linéaire sont présentés. 118
6.2 Comparaison des résultats de classification linéaire pour la reconnaissance d'images pour les pré-apprentissages avec et sans utilisation du mixup 118
6.3 Comparaison des résultats de classification linéaire avec les pré-apprentissages basés sur différentes formulations du mixup. 1 correspond à la formulation principale. 2 à celle avec les exemples négatifs uniquement. 3 à celle avec le mixup sur la requête. 120
6.4 Comparaison des scores de classification linéaire sur Imagewang pour un pré-apprentissage avec différentes valeurs de N_p, le nombre de mixup positif. 120
6.5 Comparaison des performances de reconnaissance d'actions entre les pré-entraînement utilisant SimCLR avec et sans mixup. 121
6.6 Résultats de reconnaissance d'actions à partir des K plus proches voisins. 122
6.7 Résultats de recherche de vidéos pour différents nombre de vidéos retournées en utilisant les représentations issues de la méthode SimCLR avec et sans *mixup*. Les dernières lignes du tableau correspondent aux méthodes de l'état de l'art. 122

Résumé

Les travaux effectués dans cette thèse s'intéressent au développement de méthodes d'apprentissage non-supervisé de représentations de vidéos. Le but est donc de pré-entraîner un réseau de neurones sans utiliser de données annotées qui sont difficiles à obtenir en grand nombre. Les méthodes sont ensuite évaluées sur la tâche de reconnaissance d'actions, en classification linéaire par exemple. Nous nous sommes plus particulièrement intéressés aux méthodes contrastives qui utilisent une similarité dans l'espace des représentations. Cette similarité doit être maximisée avec l'exemple positif et minimisée avec les exemples négatifs. Nous avons tout d'abord développé deux méthodes qui prennent en compte l'aspect temporel des vidéos comme supervision. Le but est de prédire certains segments vidéos à partir des autres. La première approche se base sur la méthode CPC [vdOLV18]. Nous montrons qu'il est possible de l'appliquer au domaine de la vidéo et montrons l'avantage de la formulation autoregressive avec des modèles expressifs prenant en entrée des cartes de caractéristique ainsi que l'utilisation d'exemples négatifs difficiles. La deuxième se base sur BERT [DCLT19], une méthode d'apprentissage de représentations dans le domaine du NLP. Nous montrons qu'elle est utilisable avec la fonction de coût InfoNCE et l'importance de la stratégie de masquage. Finalement, nous nous attaquons aux problèmes du nombre d'exemples négatifs et du temps d'apprentissage des méthodes contrastives. On propose d'augmenter les exemples positifs et négatifs de SimCLR [CKNH20] au niveau des représentations par interpolation linéaire. Nous avons montré un gain sur les images et les vidéos par rapport à la méthode SimCLR classique. Ces différentes méthodes nous permettent d'obtenir des résultats équivalents ou meilleurs que l'état de l'art.

Abstract

The work in this thesis proposes new unsupervised representation learning methods on videos. The goal is to pretrain a neural network without using annotated examples, which are difficult to collect in large quantities. The methods are then evaluated on the task of action recognition. Our work focuses on contrastive learning methods which uses a similarity in the representation space. This similarity is maximized with the positive example and minimized with negative ones. The first two methods take into account the temporality of the videos as a supervision. The first one is based on CPC [vdOLV18]. Our work shows that it is useful on videos and the advantages of using the autoregressive formulation with powerful models taking in input feature maps and also the advantage of using difficult negative examples. The second one is based on BERT [DCLT19], a NLP pretraining method. We show that it can be used in combination with InfoNCE loss and the importance of the masking strategy choice. Finally, we tackle the problem of the important number of negative examples and the long training time. We increased the number of negative and positive examples of SimCLR [CKNH20] at the representation level by linear interpolation. This proposition showed a gain compared with the SimCLR baseline on image and video data. These different pretraining methods obtain equivalent or better results than state of the art ones.