Reconnaissance d'actions et architectures à deux voies

Guillaume Lorre, Jaonary Rabarisoa, Astrid Orcesi, Samia Ainouz, Stéphane Canu

Reconnaissance d'actions et architectures à deux voies

Guillaume LORRE *[1], Jaonary RABARISOA[1], Astrid ORCESI[1], Samia AINOUZ[2] et Stéphane CANU[2]

[1]CEA, LIST, Laboratoire Vision et Ingénierie des Contenus, Gif-sur-Yvette, France
[2]Normandie Univ, INSA Rouen, UNIROUEN, UNIHAVRE, LITIS, France

Résumé

Cet article propose une étude comparative des méthodes d'apprentissage profond pour la reconnaissance d'actions à partir de séquences vidéo. Nous comparons différentes adaptations des réseaux de neurones convolutionnel (CNN) comme le règlage fin (*fine tuning*), l'apprentissage par transfert (*transfert learning*) et l'architecture de reconnaissance. Nous portons ensuite une plus grande attention à l'architecture à deux voies (*2-stream*) dont on compare les critères pratiques de réussite. Nos résultats expérimentaux montrent que pour ce problème, l'architecture à deux voies est préférable avec un haut taux de dropout.

Mots-clef : Reconnaissance d'actions, Apprentissage profond, Architecture 2 stream.

1 Introduction

La reconnaissance d'actions est une tâche essentielle dans le domaine de la vision par ordinateur. Il s'agit d'associer une classe d'actions à une séquence vidéo. La reconnaissance se distingue de la détection qui consiste à donner la localisation spatio-temporelle des actions considérées. Elle peut être appliquée à la vidéo-surveillance (détections d'anomalies ou de comportements violents) ou dans le cadre d'applications *smart home* (assistance aux personnes agées). La majorité des méthodes récentes utilisent l'apprentissage supervisé de réseaux de neurones profonds. Ceux-ci prennent en entrée une vidéo brute et en calculent des représentations de plus en plus haut niveau. La difficulté principale reste la conception de modélisations spatio-temporelles efficaces. Pour ajouter une dimension temporelle, le plus naturel est d'utiliser la convolution 3D ou de combiner (par fusion ou réseau récurrent) les informations spatiales à différents instants de la vidéo. Une autre direction est d'utiliser la représentation du mouvement par le flot optique. C'est le cas essentiellement de l'architecture *2-stream* qui est actuellement la référence de l'état de l'art. Dans la section 2, nous présenterons les algorithmes qui n'utilisent pas le flot optique. Nous détaillerons les différentes variantes de l'architecture *2-stream* en section 3. Les expériences menées sur cette architecture sont décrites en section 4.

*guillaume.lorre@cea.fr

2 Reconnaissance d'actions

Dans cette partie, nous présentons 3 catégories de méthodes qui permettent de prendre en compte la temporalité : la convolution 3D, la fusion d'information spatiale à différents instants et l'utilisation des réseaux récurrents.

2.1 Convolutions 3D

Les convolutions 3D permettent de calculer des caractéristiques spatio-temporelles et sont donc intéressantes pour la reconnaissance d'actions. [TBF+14] propose un réseau avec 8 couches de convolution dont les noyaux sont de taille $3 \times 3 \times 3$. Cependant, uniquement 16 frames sont prises en compte pour que la complexité soit acceptable. La fenêtre temporelle de la modélisation est donc très réduite. [VLS16] utilise 60 frames en entrée de son réseau ce qui contraint la résolution spatiale des images. [SJYS15] factorise les noyaux 3D en un noyau spatial 2D suivi d'un noyau temporel 1D. Ceci réduit fortement le nombre de paramètres mais aussi l'expressivité du noyau. Les limitations principales de ces méthodes sont la complexité importante des convolutions 3D, la prise en compte d'une faible durée temporelle et enfin l'impossibilité d'utiliser de larges bases de données d'images (en pré-

entraînement par exemple). [CZ17] résout ce dernier problème en proposant de dupliquer les poids sur la dimension temporelle des CNN 2D. Il propose une nouvelle base d'apprentissage kinetics [KCS+17] de taille importante où il est possible d'apprendre des CNN 3D beaucoup plus profonds. Le module Inception qu'il propose comme [DFS+17] permet d'apprendre des relations sur des temporalités différentes.

2.2 Méthodes de fusion

Une autre solution consiste à combiner des représentations spatiales au niveau temporel. Il s'agit d'utiliser un CNN avec des convolutions 2D auquel on ajoute des méthodes d'aggrégation sur la dimension temporelle (*max pooling*, *mean pooling*, *strided convolutions 1D*, ...). [KTS+14] et [NHV+15] ont testé ces différentes méthodes de fusion ainsi que différentes localisations de celles-ci dans le réseau (voir figure 1) : 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 temporelle de plus en plus longue aux couches les plus hautes (*slow fusion*). [KTS+14] obtient des meilleurs résultats avec la *slow fusion* pour les convolutions 1D et [NHV+15] avec la *late fusion* pour le *max pooling*.

[DSG16] aggrège les cartes de caractéristiques issues d'un CNN par une multiplication élément par élément. Il utilise par la suite un encodage bilinéaire (avec une réduction de dimension par l'algorithme *Tensor Sketch*) afin de mieux capturer les interactions des caractéristiques. Cette méthode permet d'avoir une représentation compacte et fortement discriminante. D'autres méthodes d'encoding comme VLAD ou les vecteurs de Fisher ont aussi été utilisées mais elles donnent des résultats inférieurs.

[WGGH17] propose une opération non locale qui prend en compte des interactions entre toutes les positions de la carte des caractéristiques d'un CNN. Ceci permet de calculer des dépendances long-terme sans passer par une longue série d'opérations locales (comme les convolutions ou les cellules récurrentes). En notant y_i la position i de la sortie, x_i la position i de l'entrée, f une fonction de relation entre 2 caractéritiques et g une fonction de représentation,

$$y_i = \frac{1}{C(x)} \sum_j f(x_i, x_j) g(x_j)$$

Ces méthodes ont une complexité faible et permettent de prendre en compte une longueur de vidéo importante. Cependant, elles séparent le traitement spatial et temporel. Certaines peuvent aussi être utilisées sur des CNN 3D.

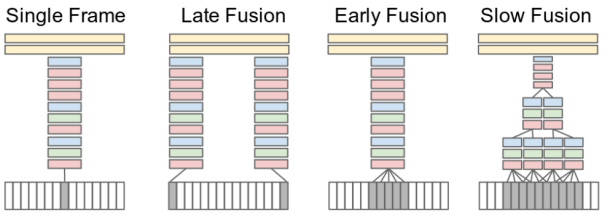

FIGURE 1 – Différentes méthodes de fusion temporelle de l'information spatiale dans des réseaux de convolution (repris de [KTS+14]).

2.3 Réseaux récurrents

Dans [DHG+14] et [NHV+15], un réseau récurrent encode la séquence de représentations (issues d'un CNN) des frames de la vidéo. Ceci est naturel considérant la vidéo comme une série temporelle et permet de prendre en entrée un nombre variable de frames. Les LSTM [HS97] grâce à leurs 2 cellules (cachée et mémoire) et les différentes portes pour intéragir avec elles réduisent le problème des gradients évanescents et apprenent des relations temporelles sur un plus long terme. [BYPC15] remarque que 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. Il propose un réseau composé de couches de convolutions et de plusieurs couches récurrentes (Gated Recurrent Units : GRU) sur des cartes de caractéristiques. 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.

3 Réseau 2-stream

L'architecture *2 stream* [SZ14] sépare la modélisation spatiale et temporelle (voir figure 2). La méthode est composée de 2 CNN (branche spatiale et temporelle), l'un prenant en entrée des images et l'autre un empilement de flots optiques (le plus souvent 10 et donc un nombre de 20 canaux). La prédiction finale est obtenue par moyenne des probabilités de sortie des 2 réseaux.

Le flot optique est une représentation classique du mouvement en vision par ordinateur. Il représente le

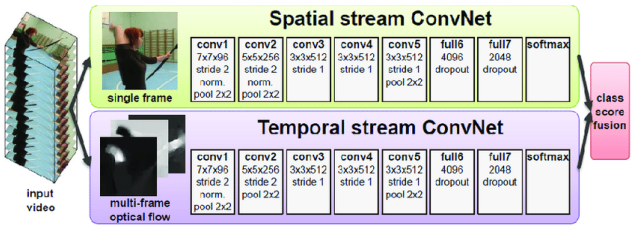

FIGURE 2 – Architecture *2 stream* (repris de [SZ14]), composée de 2 branches prenant en entrée 2 modalités différentes : les images et les flots optiques

vecteur de déplacement pour chaque pixel entre 2 images consécutives. On note u(x,y) et v(x,y) sa composante horizontale et verticale. Il est souvent calculé à partir de l'hypohèse de constance de luminosité.

$$I_{t+1}(x + u(x, y), y + v(x, y)) = I_t(x, y)$$

Une contrainte de régularité pour que le flot optique soit cohérent localement (notemment au niveau des zones plates) est aussi utilisée.

$$\min \quad (\frac{\delta u}{\delta x})^2 + (\frac{\delta u}{\delta y})^2 + (\frac{\delta v}{\delta x})^2 + (\frac{\delta v}{\delta y})^2$$

Récemment, il est calculé par des réseaux de neurone de convolution/déconvolution prenant en entrée 2 images et générant le flot optique [FDI+15]. Le réseau est appris de manière supervisée sur des données synthétiques (FlyingChairs [IMS+17] par exemple).

[SZ14] a montré que l'empilement de frames donne des résultats inférieurs aux flots optiques. Un empilement des flots par trajectoire (les flots sont référencés par rapport à la première frame [1] ainsi que des flots bidirectionnels ont été testés, le dernier améliore légèrement les résultats.

Les principaux problèmes de cette méthode sont le manque de modélisation temporelle sur le long terme, la non-communication entre les réseaux spatial et temporel et enfin le besoin d'un flot optique pré-calculé. Nous allons voir différentes réponses à ces problèmes dans les paragraphes suivants.

Certaines méthodes vues dans la section 2 sont applicables pour palier au problème de la modélisation court-terme. [WXW+16] propose de découper la vidéo en K segments et de calculer un score issu du consensus entre ceux-ci. Pour celà, il tire aléatoirement une frame et un empilement de flots sur chaque segment, les fait passer par le réseau *2-stream* et retourne la moyenne des scores des K segments. [MCKA17] effectue une aggrégation sur chaque segment (max-pooling) et encode la séquence des caractéristiques en sortie du *2-stream* pour chaque segment par un LSTM. Il propose également un CNN temporel à 2 couches pour combiner l'information temporelle.

[FPZ16] a étudié la fusion entre la branche spatiale et la branche temporelle. Les fusions *early* , *slow* et *late* ont été testées, avec différentes opérations telles que maximum, somme, bilinéaire et convolution. Les meilleurs résultats sont obtenus avec une fusion *late*, les opérations maximum, somme ayant des scores similaires. Dans [FPW16], des "skip" connections sont ajoutées entre les 2 réseaux. En notant x_l^s et x_l^t la l^{me} couche du réseau spatial et temporel respectivement, la relation suivante est utilisée :

$$x_{l+1}^s = f(x_l^s) + F(x_l^s + f(x_l^t), W_l^s)$$

Ainsi, l'information de la branche temporelle est transférée au réseau spatial, sans pour autant le perturber de manière trop importante ;

[ZLNH17] considère que le pré-calcul et stockage des flots optiques sont très limitants pour les approches *2-stream*. Leur méthode "end to end" n'utilise pas de flot optique pré-calculé. Le Motion Net prend en entrée une succession de frames et génère une représentation semblable au flot optique (qui respecte l'hypohèse de constance de la luminosité et de régluarisation). La régularisation a une forme semblable à celle présentée précédemment et la fonction de coût s'écrit sous la forme suivante (avec ρ la pénalité généralisée de Charbonnier).

$$L = \frac{1}{N} \sum_{i,j}^{N} \rho(I_1(i, j) - I_2(i + u(i, j), j + v(i, j)))$$

La représentation obtenue peut être plus adaptée à la reconnaissance d'actions car elle est aussi apprise en fonction de l'erreur de classification (rétropopagée jusqu'au Motion Net).

4 Expériences

Dans cette partie, nous nous concentrons sur l'apprentissage des méthodes *2 stream* car elles ont des

1. $p_1 = (u, v)$ et $p_k = p_{k-1} + f_{k-1}(p_{k-1})$ avec p la somme des flots sur la trajectoire (empilement), (u,v) le flot de la première frame et f_k le flot sur la frame k

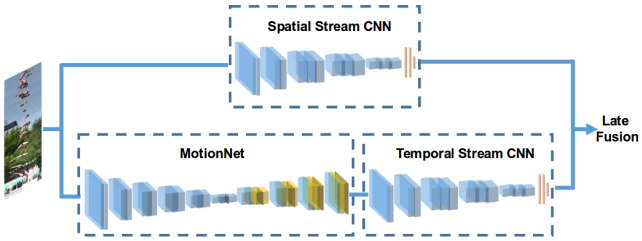

FIGURE 3 – Architecture *2-stream* caché (repris de [ZLNH17]) qui estime le flot (*Motion Net*) passé en entrée de la branche temporelle

résultats parmi les meilleures de l'état de l'art et sont très générales. En effet, elles peuvent être associées à de nombreuses autres méthodes, comme les convolutions 3D ou les réseaux récurrents. Nous proposons une étude des différentes architectures et paramètres utiles à l'apprentissage de ces réseaux.

4.1 Dataset

Pour l'évaluation, nous avons utilisé le jeu de données ucf-101 [SZS12]. Il est composé de 13320 vidéos réalistes provenant de Youtube (27 heures en tout), présentant un mouvement de caméra et un arrière-plan brut. Chaque vidéo montre une action parmi les 101 classes d'actions considérées, comme par exemple jouer d'un instrument de musique, pratiquer un sport, un loisir ou un travail. Ces actions sont assez courtes (quelques secondes pour la plupart). Chaque classe contient environ 5 fragments de 25 vidéos. Il permet de se comparer à la plupart des travaux de l'état de l'art en restant de taille raisonnable à l'encontre d'autres jeux de données plus volumineux comme charades [SVW+16] ou kinetics [KCS+17].

4.2 Détails d'implémentation

Dans le modèle *2-stream* de base pris pour nos expériences et détaillé ci-dessous, nous apprenons chaque branche séparemment. Nous extrayons la couche *global pool* du CNN qui est utilisé en entrée d'une dernière couche entièrement connectée. La fonction d'activation softmax est appliquée à cette dernière couche et la fonction de coût utilisée est la cross-entropie. Le réseau est régularisé par une pénalité L2 sur les poids et un dropout sur la dernière couche.

4.2.1 Entraînement

L'image ou empilement de flots est tiré aléatoirement dans la vidéo. Il subit les transformations suivantes pour faire de l'augmentation de données : *flip* aléatoire horizontal, *cropping* aléatoire et RGB *jittering* (uniquement pour les images). Les valeurs des images sont ramenées entre -1 et 1 et les flots optiques sont centrés spatialement. Le réseau est appris par descente de gradient stochastique avec un *momentum* de 0.9.

4.2.2 Prédiction

Pour la prédiction, 20 frames également espacées sont tirées de la vidéo et recadrées en leur centre. Le score final est la moyenne des scores pour chaque frame.

4.3 Comparaison des méthodes

Nous avons testé différentes architectures pour les CNN (alexnet [KSH12], resnet-50 [HZRS15], resnet-101 [HZRS15] et inception-resnet [SIV16]) ainsi que différents paramètres (*dropout*, *batch normalisation*, ...) sur chaque branche du réseau séparément. Nous avons aussi testé différentes stratégies de finetuning.

Dans la suite de l'article, nous décrivons l'infuence de ces différents paramètres et ainsi que les résultats obtenus pour l'architecture complète.

Modèle	**Score**
Alexnet	81.9
Resnet 50	82.2
Resnet 101	**84.8**
Inception resnet	83.5

TABLE 1 – Comparaison de différentes architectures de réseaux convolutionnels sur la branche temporelle

Modèle	**Score**
Resnet 101	82.5
Inception resnet	**84.2**

TABLE 2 – Comparaison du finetuning de différentes architectures sur la branche spatiale

Les tables 1 et 2 nous montrent 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 du à une trop grande profondeur par rapport au nombre de données disponibles.

Stratégie	Score
Transfert	77.5
Finetuning	**82.5**
Transfert + Finetuning	80.3

TABLE 3 – Etude des résultats de transfert et finetuning pour la branche spatiale avec *resnet-101* comme réseau

D'après la table 3, mettre à jour l'ensemble du réseau améliore l'apprentissage par rapport à un apprentissage de la dernière couche uniquement (après un pré-entraînement sur ImageNet). Apprendre la dernière couche puis effectuer le finetuning ne permet pas d'atteindre les résultats du finetuning seul.

Un *dropout* élevé permet une meilleure régularisation, un sur-apprentissage plus faible et donne donc de meilleurs scores (voir table 4).

Enfin, la cross-modalité [2] permet d'obtenir des résultats légèrement meilleurs. Cependant, le consensus entre segments (voir section 3) et le fait de fixer la *batch normalisation* n'améliorent pas les résultats du modèle.

D'après la table 6, la somme des probabilités sur les 2 branches permet d'obtenir des scores bien supérieurs que sur les branches séparemment ce qui montre leur complémentarité.

Les critères principaux de l'apprentissage *2-stream* sont donc, d'après nos expériences, l'utilisation de réseaux profonds, de pre-training et finetuning complet ainsi qu'une forte régularisation (*dropout* élevé et pénalisation L2).

5 Conclusion et perspectives

En conclusion, nous avons comparé différents réseaux de l'état de l'art pour la reconnaissance d'activités, avec aussi bien des modélisations court-terme (Convolutions 3D, flots optiques) que long-terme (Fusion et réseaux récurrents). Nous avons par la suite étudié l'apprentissage des réseaux *2-stream* à travers nos expériences. Nous en concluons que la profondeur des réseaux associée à une régularisation importante sont les facteurs clés de leur apprentissage.

L'overfitting étant un des critères importants (sur des datasets de la taille de ucf-101), le pré-apprentissage non supervisé paraît être une solution intéressante pour améliorer la généralisation du modèle. Nous avons pour perspective d'utiliser des modèles génératifs comme les GAN [GPAM+14] ou les VAE [KW13] entraînés pour prédire les frames ou les flots optiques futurs comme initialisation du réseau *2-stream*. Il est par exemple possible d'ajouter un réseau décodeur à la fin de chaque branche pour leur prédiction.

2. Initialisation du réseau sur les flots par un réseau pré-entraîné sur Imagenet en modifiant la 1ère couche de convolution

Modèle	dropout	Score
Inception-resnet	0.5	84.2
Inception-resnet	0.8	**84.7**

TABLE 4 – Etude de l'influence du taux de *dropout* pour la branche spatiale

Branche	Amélioration	Score
Temporel	Cross-modalité	**85.1**
Temporel	Consensus + Cross-modalité	84.6
Spatial	sans BN	83.7

TABLE 5 – Scores de différentes modifications pour améliorer la généralisation sur le réseau resnet-101 - BN : *Batch Normalization*

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

TABLE 6 – Résultats *2-stream*

Références

[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.

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

[DFS+17] Ali Diba, Mohsen Fayyaz, Vivek Sharma, Amir Hossein Karami, Mohammad Mahdi Arzani, Rahman Yousef-

zadeh, and Luc Van Gool. Temporal 3d convnets : New architecture and transfer learning for video classification. *CoRR*, abs/1711.08200, 2017.

[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.

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

[FPW16] Christoph Feichtenhofer, Axel Pinz, and Richard P. Wildes. Spatiotemporal residual networks for video action recognition. *CoRR*, abs/1611.02155, 2016.

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

[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.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8) :1735–1780, 1997.

[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.

[KCS+17] 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.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[KTS+14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, June 2014.

[KW13] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 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.

[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.

[SIV16] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[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.

[SVW+16] Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood

in homes : Crowdsourcing data collection for activity understanding. *CoRR*, abs/1604.01753, 2016.

[SZ14] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199, 2014.

[SZS12] 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.

[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.

[VLS16] Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. *CoRR*, abs/1604.04494, 2016.

[WGGH17] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *CoRR*, abs/1711.07971, 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.

[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.