Discussion

Nous remarquons ici, une amélioration de l'accuracy pour deux jeux de données traitant des problèmes similaires: la reconnaissance d'action squelettiques pour des niveaux différents (la main et le corps), pour des dimensions différentes (2d et 3d) et pour des dimensions de convolutions variantes.

L'apport de la normalisation semble plus marqué sur SHREC que sur JHMDB. Une explication serait que le niveau d'abstraction et de précision n'est pas le même entre ces jeux de données. Le nombre de keypoints dans SHREC est supérieur à JHMDB et par conséquent l'organisation de ceux-ci pourrait apporter plus d'information.

En choisissant un modèle complexe pour la convolution 1d et un modèle trivial pour la convolution 2d, nous émettons l'hypothèse que cette transformation peut-être bénéfique quelque soit le niveau de complexité du modèle. Cependant, celle-ci modifie significativement la taille de l'entrée du réseau et par conséquent la taille en paramètres des approches augmentera. Cette augmentation n'est que peu visible sur DD-Net car il n'y a pas de layer dense. Pour Lenet-5 la taille du modèle augmente du simple au double, et ce à cause de la partie fully connected de l'architecture. Il conviendra donc d'éviter le plus possible l'utilisation de layer dense pour cette approche ou de travailler sur des réseaux de taille réduite.

Finalement, cette normalisation nécessite une connaissance à priori de l'organisation de la structure des données en entrée (coude: keypoint 1, tête: keypoint 2, ...). Dans le cadre de la thèse, cela ne pose pas de contraintes spécifiques. Cependant, il sera par exemple impossible de réaliser ce travail sur des jeux de données dont la structure des données n'est pas précisée.

3.2 Autoencodeur semi-supervisé pour l'action recognition

3.2.1 Introduction

Les auto encodeurs étant une composition de transformation non linéaires en esperant trouver une représentation dans l'espace adaptée au format des données et conservant un maximum de sémantique de celui-ci.

Je me suis intéréssé à cette question de représentation pour plusieurs raisons:

- Tout est une question d'embedding et de normalisation en machine learning. Le role premier des couches cachées est de transformer cet embedding en espérant qu'il soit porteur d'information. Le jour où l'on trouve un moyen de normaliser et de représenter les données de manière plus adéquate, n'importe quel classifieur pourra obtenir

de bons résultats pour peu que les données en entrée soient porteuses d'information. Le jour où l'on arrivera à obtenir une représentation utilisable plus rapidement au lieu de stacker des layers au sein d'un réseau de neurones, d'autres problématiques d'apprentissage apparaîtront mais le gain serait intéressant: (Selon le théorème d'approximation universelle, n'importe quel Shallow Network peut supposément approximer une fonction donnée [Cybenko, 1989, Scarselli and Tsoi, 1998]).

- En s'intéressant à la question de la représentation des données, on évite un apprentissage de cette représentation à "l'aveugle": en optimisant un réseau précis sur un jeu de données précis et en testant un maximum d'hyper-paramètres on est assurés de faire un bon résultat. Plus la représentation des données est fiable, plus l'architecture utilisée pour la classification peut-être triviale. On peut donc potentiellement réduire la taille de notre réseau et donc par définition réduire le temps d'inférence de celui-ci, ce qui peut être intéressant dans le cadre d'un sujet en temps réel.

- Peu de travaux de recherche en reconnaissance d'actions se focalisent sur la représentation des données, la majorité des travaux actuels translatent la connaissance de domaines connexes tels que la computer vision ou le text-mining et s'inspirent des méthodes de l'état de l'art dans ces domaines. Il n'existe à ce jour à ma connaissance aucune publication sur l'utilisation des auto-encodeurs pour la reconnaissance d'actions squelettiques.

L'idée étant d'entraîner un auto-encodeur avec une fonction de coût modifiée en rajoutant une contrainte sur la séparation linéaire des classes dans l'espace latent (LDA / QDA / FDA). Il a été montré que la représentation optimale interne d'un MLP s'obtient en faisant une analogie avec une analyse discriminante non linéaire [Webb and Lowe, 1990]. Parallèlement, [Brigato and Iocchi, 2020] montrent que l'ajout d'une fonction de coût non-supervisée lors d'un problème de classification avec peu de données permet une meilleure généralisation des réseaux.

Les approches factorielles citées plus haut s'apparentent à la projection de données dans l'espace comme celle d'une ACP mais tandis que l'ACP maximise la variance du jeu de données, celles-ci se focalisent sur la maximisation de la séparabilité des classes dans l'espace, ce qui en fait des méthodes supervisées. On obtiendrait donc un Auto-encodeur "semi-supervisé" dans le sens où celui-ci se focalise sur deux informations complètement différentes dans les données, l'une de manière non-supervisée, l'autre de manière supervisée:

- La structure inhérente des données capturée de manière non supervisée grâce à la reconstruction de l'Auto-encodeur et sa capacité d'abstraction. On conserverait

alors une partie de l'information importante et discriminante du jeu de données. (feature extraction)

- La séparabilité des classes dans l'espace grâce à l'analyse discriminante linéaire. Ce qui permettrait de réduire le nombre de layers.

Nous pouvons imaginer deux utilisations à cette approche:

- Récupérer cet espace latent combinant les informations "data driven" grâce à l'apprentissage non-supervisé de l'auto encodeur classique et la première ébauche de séparabilité linéaire grâce à la seconde partie de la fonction de coût. Envoyer l'espace latent à un réseau de neurones très peu profond pour la classification. Finetuner le réseau de bout en bout et voir ce que cela apporte quand à la capacité de discrimination de l'approche.

- Générer des données en réalisant une génération d'instances au niveau de l'espace latent grâce à des approches type SMOTE [Chawla et al., 2002] ou ADASYN [He et al., 2008] et utiliser la partie décodeur pour faire de la data génération.

3.2.2 Formalisation

La pipeline complète de notre approche est présentée en figure 3.6. Nous définissons le problème comme ceci:

$$\min_{\theta_1,\theta_2}, \|\mathbf{X} - g_{\theta_2}\left(f_{\theta 1}(\mathbf{X})\right)\|^2$$

Fonction de reconstruction habituelle d'un auto-encodeur avec θ_1, θ_2 les paramètres des blocs encodeur et décodeur de l'AE.

$$\min_{\theta_1,\theta_2,\mathbf{S}} \|\mathbf{X} - g_{\theta_2}\left(f_{\theta_1}(\mathbf{X})\right)\|^2 + \lambda \|f_{\theta_1}(\mathbf{X}) - \mathbf{S}_{f_{\theta 1}}(\mathbf{X})\|^2$$

Ajout d'une variation dans la fonction de coût: avec S la matrice de projection des individus dans l'espace latent obtenue avec une analyse discriminante linéaire (LDA) et λ un régularisateur.

Une fois, l'entraînement de l'auto-encodeur réalisé, récupération des poids de la partie encodeur: $f_{\theta 1}$, ajout d'une fonction classifieur softmax:

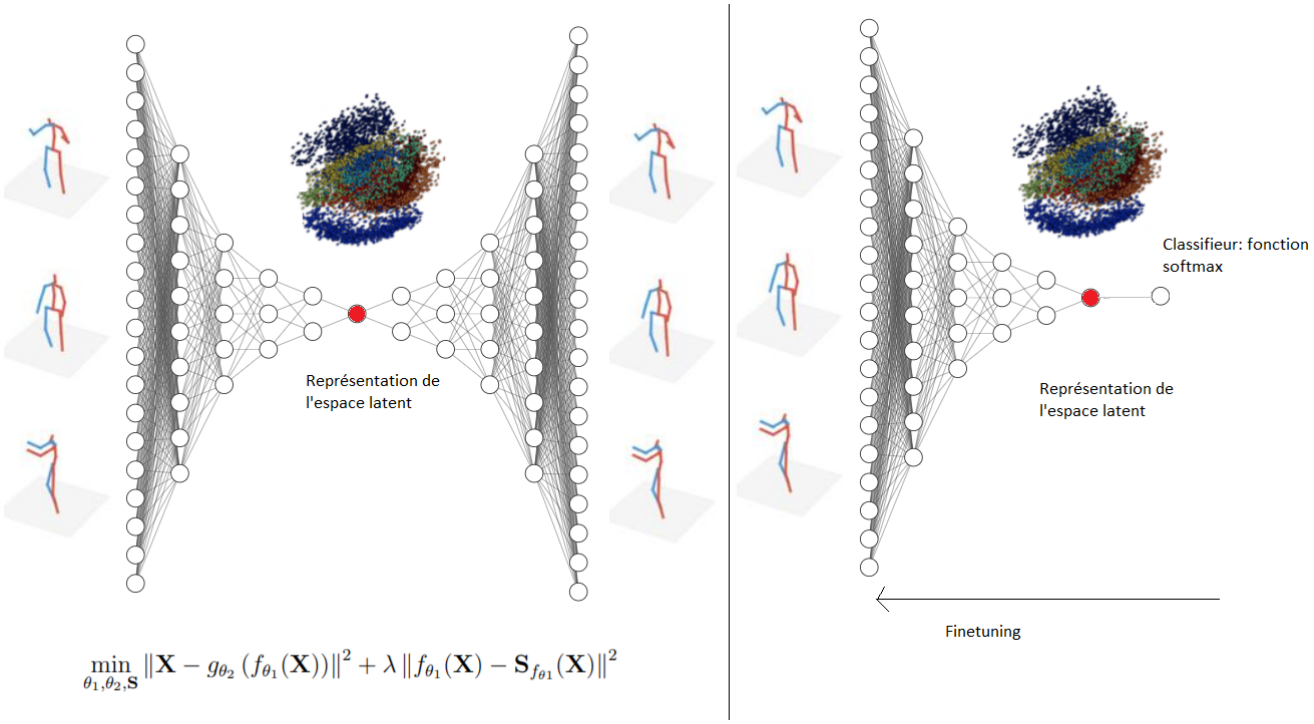

Figure 3.6: Pipeline de l'approche: (1) nous entraînons un auto-encodeur pour reconstruire une séquence représentant une action en y ajoutant une contrainte spécifique à la séparabilité des classes dans l'espace latent. (2) extraction des poids de la partie encodeur jusqu'au bottleneck représenté en rouge et ajout d'une softmax, transformant alors la partie encodeur en un réseau pré-entraîné sur les données pour la classification des actions.

Nous souhaitons dans un premier temps évaluer l'apport de cette représentation pour la classification.

3.2.3 Tests et évaluation

Evaluation datasets

Par soucis d'économie de temps, j'ai réalisé l'évaluation de cette approche sur les mêmes jeux de données que ceux présentés en 3.1.3, à savoir SHREC [De Smedt et al., 2017] et JHMDB [Jhuang et al., 2013].

Détails d'implémentation

Afin d'évaluer la qualité de représentation, nous avons choisi de prendre une architecture d'autoencodeur la plus facile possible: semblable à celle d'un perceptron multicouches. Naturellement, il est possible de réaliser le même travail sur des auto-encodeurs convolutifs ou séquentiels afin de capturer la séquentialité de l'action ou de minimiser la taille du

réseau et ainsi potentiellement améliorer l'accuracy et le temps d'inférence.

L'architecture est composée d'une partie encodeur et d'une partie décodeur. La partie encodeur est définie telle quelle:

- Un couche d'entrée de dimension 2112 représentant les valeurs d'une action complète pour 32 pas de temps, 22 keypoints et 3 dimensions.

- 5 layers dense d'extractions de features dont la dimension diminue d'une puissance de deux pour chaque layer (512, 256, 128, 64, 32).

- Un bottleneck de dimension du nombre de classe du jeu de données moins un: le nombre maximum d'axes obtenus par les analyses factorielles correspond au nombre de classe moins un. Pour SHREC 14 celui-ci aura 13 valeurs pour SHREC 28 27 valeurs et pour JHMDB 20 valeurs.

La partie décodeur est symétrique à la partie encodeur à l'exception du bottleneck qui est unique.
Pour chaque layers, nous ajoutons une couche de dropout de valeur *0.1* et de la batchnormalisation préalablement au dropout. L'entraînement est réalisé avec l'optimiseur Adam [Kingma and Ba, 2014] suivant les recommandations du papier initial pour la première et la seconde étape.

L'optimisation du paramètre λ est actuellement sujette à discussions, nous avons pour l'instant fait varier les valeurs de lambda de manière empirique afin de suivre l'impact que la normalisation LDA avait sur le résultat final.

Résultats

Méthode	**Nb Params**	**SHREC 14**	**SHREC 28**	**FPS (RTX 2080 ti)**
LDA sur l'input	-	33.0%	27.6%	
LDA sur le bottleneck AE classique ($\lambda = 0$)	-	37.9%	42.8%	
LDA sur le bottleneck AE modifié ($\lambda = 5$)	-	43.5%	35.6%	
MLP Sans initialisation	1.2M	91.2%	85.2%	20 000
MLP Avec initialisation AE classique ($\lambda = 0$)	1.2M	91.5%	85.9%	-
MLP Avec initialisation AE Modifié ($\lambda = 1$)	1.2M	91.9%	87.6%	-
MLP Avec initialisation AE Modifié ($\lambda = 2.5$)	1.2M	92.4%	86.9%	-
MLP Avec initialisation AE Modifié ($\lambda = 5$)	1.2M	**92.5%**	**87.1**%	-
MLP Avec initialisation AE Modifié ($\lambda = 7.5$)	1.2M	91.9%	86.4%	-
MLP Avec initialisation AE Modifié ($\lambda = 10$)	1.2M	90.9%	85.2%	-

Table 3.5: Résulats obtenus grâce à l'initialisation des poids avec un AE modifié sur SHREC [De Smedt et al., 2017]

Methode	Nb Params	Average accuracy of 3 splits)	Classifications par secondes
Chained Net (ICCV17)	17.50 M	56.8%	33 (1080 Ti)
EHPI (ITSC19)	1.22 M	65.5%	29 (1080 Ti)
PoTion (CVPR18)	4.87 M	67.9%	100 (1080 Ti)
DD-Net	1.82M	77.2%	2,200 (1080 Ti)
Sans initialisation ($\lambda = 0$)	0.67M	65.2%	-
Avec initialisation AE classique ($\lambda = 0$)	0.67M	66.4%	-
Avec initialisation AE modifié ($\lambda = 1$)	-	66.2%	-
Avec initialisation AE modifié ($\lambda = 2.5$)	-	68.3%	-
Avec initialisation AE modifié ($\lambda = 5$)	-	67.9%	-
Avec initialisation AE modifié ($\lambda = 7.5$)	-	66.5%	-

Table 3.6: Résulats obtenus grâce à l'initialisation des poids avec un AE modifié sur JHMDB.

Discussions et visualisations

Nous avons souhaité dans un premier temps vérifier que le jeu de données n'était pas suffisamment trivial pour qu'une simple LDA sur l'entrée X suffise à obtenir des résultats cohérents et ce sans utiliser l'apprentissage profond. Selon le tableau 3.6, nous remarquons qu'une simple LDA sur l'entrée n'est pas suffisante pour obtenir des résultats corrects. Cependant il nous a semblé intéressant de regarder si l'utilisation des données projetées dans l'espace latent après entraînement d'un auto-encodeur classique et de notre encodeur modifié apportait de l'information avant finetuning. Après une LDA sur les données projetées jusqu'au bottleneck, nous réalisons qu'un auto-encodeur classique et que notre auto-encodeur modifié permettent à une simple LDA de trouver plus de sens aux données. D'après la figure 3.7 ci-dessous, nous remarquons également que la projection des données vers l'espace latent dans le cadre de notre auto-encodeur permet d'obtenir une séparation plus visible dans l'espace des centroïdes de chaque classe.

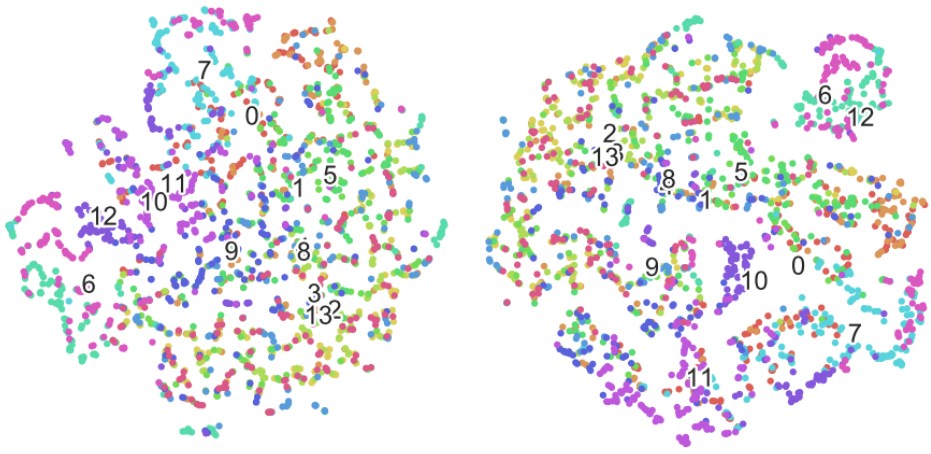

Figure 3.7: Visualisation des espaces latents via T-Sne: à gauche auto-encodeur classique, à droite auto-encodeur modifié ($\lambda = 10$)

Toujours selon le tableau 3.6, nous avons donc voulu comparer la qualité de représentation pour la classification entre un auto-encodeur à fonction de coût classique ($\lambda = 0$)

et à fonction de coût modifiée après finetuning. Nous remarquons ici que la normalisation LDA permet d'obtenir des résultats sensiblement meilleurs aux résultats obtenus par un auto-encodeur général sur les deux jeux de données proposés.

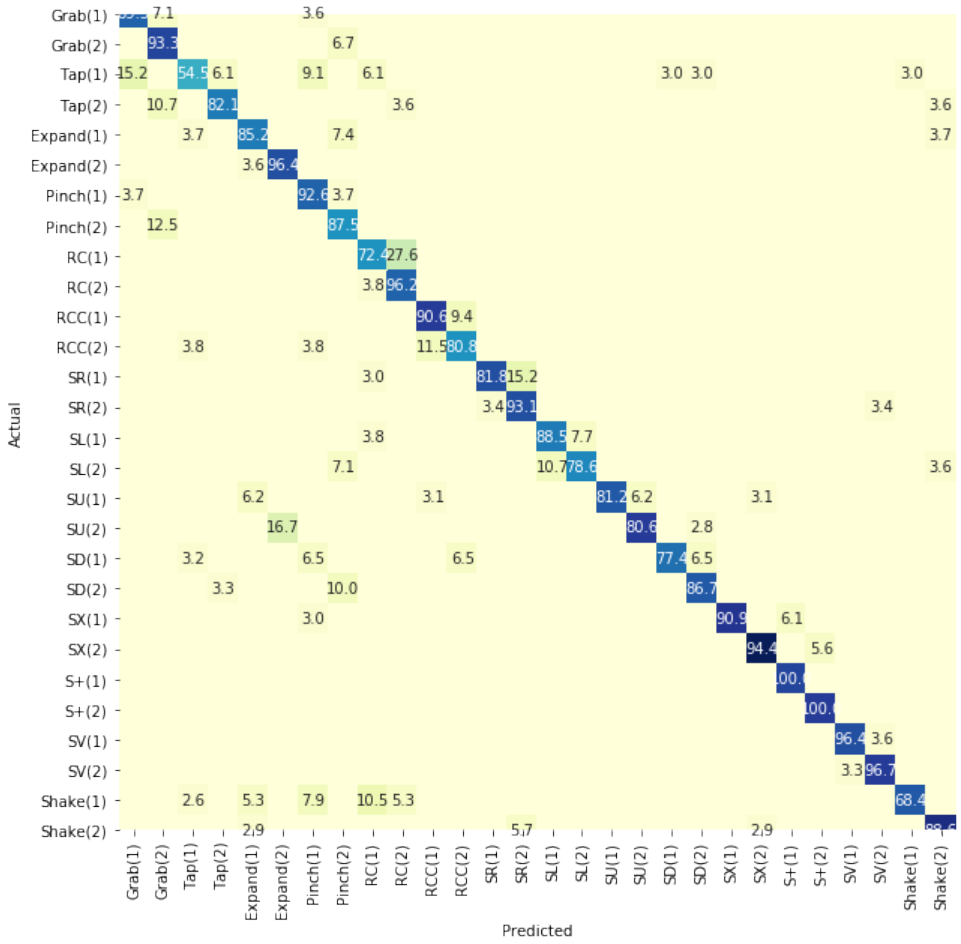

Figure 3.8: Matrice de confusion obtenue sur SHREC 28 avec un auto-encodeur modifié ($\lambda = 5$).

Nous avons dans un dernier temps, souhaité comparer l'approche AE-LDA à l'état de l'art pour ces deux jeux de données et ainsi voir si un modèle trivial se focalisant sur la représentation de ses données avec une régularisation non supervisée pouvait concourir avec des approches plus complexes comme les réseaux de neurones à état ou réseau de neurones convolutifs: la figure 3.9 et le tableau 3.6 nous permettent de constater que notre méthode dont les résultats sur SHREC 14 et SHREC 28 sont respectivement 92.5% et 87.1% et de 68.3% sur JHMDB peut jusqu'à une certaine limite concourir avec des approches plus complexes. Il convient cependant de rappeler qu'en se fiant aux résultats de [Yang et al., 2019], notre méthode est jusqu'à 4 fois plus rapide que les méthodes de l'état de l'art (20 000 classifications par secondes).

Model	Accuracy (14)	Accuracy (28)
Oreifej and Liu (2013)	78.5	74
Devanne et al. (2014)	79.6	62
De Smedt et al. (2017)	82.9	71.9
Ohn-Bar and Trivedi (2013)	83.9	76.5
Weng et al. (2018a)	85.8	80.2
De Smedt et al. (2016) (SoCJ + HoHD + HoW	88.2	81.9
Caputo et al. (2018)	89.5	-
Boulahia et al. (2017)	90.5	80.5
Hou et al. (2018) (Res-TCN)	91.1	87.3
Devineau et al. (2018)	91.3	84.4
Chen et al. (2019)	91.3	86.6
Nguyen et al. (2019b)	92.38	86.31
Li et al. (2019b) (HG-GCN)	92.8	88.3
Hou et al. (2018) (STA-R	93.6	90.7
Maghoumi and LaViol	94.5	91.4
Yang et al. (2	94.6	91.9
Avola et al. (201	97.62	91.43

Figure 3.9: Resultats de l'état de l'art sur SHREC 14/28, image provenant des recherches approfondies sur le jeu de données par Guillaume Devineau.

Pour conclure, nous avons montré ici, qu'il est plus qu'intéressant de se focaliser sur la question de la représentation des données pour la reconnaissance d'action squelettiques.

3.2.4 Perspectives

Le réseau présenté ci-dessus étant trivial, il est possible d'envisager plusieurs axes d'amélioration:

- Essayer plusieurs structures d'auto-encodeur plus travaillées qu'un simple MLP: AE convolutif, AE séquentiel, utiliser des couches d'attention...
- Cumuler la structuration des keypoints présentée en 3.1 à cette approche pour un auto-encodeur convolutif et évaluer la pertinence de cumuler les deux approches.
- L'évaluation de la capacité de génération de l'auto-encodeur n'a pas encore été évaluée.

Cependant, nous ne sommes pas complètement convaincus de la capacité de ces deux jeux de données à représenter la totalité des difficultés qui existent pour la prédiction d'intention de piétons. Ainsi, nous avons pour l'instant préféré laisser cette approche en suspens jusqu'à la mise en place d'une pipeline sur des jeux de données de piétons supposément plus complexes.

Chapter 4

Démarche proposée au cours de la thèse

4.1 Bases de données

Nous avons identifié deux jeux de données sur lesquels nous baserons nos travaux: Joint Attention for Autonomous Driving(JAAD) [Kotseruba et al., 2016, Rasouli et al., 2017] et Pedestrian Intention Estimation (PIE) Dataset [Rasouli et al., 2019].

Ces jeux de données couvrent un large éventail d'apparences de piétons pour des conditions environnementales diverses: hiver, printemps, été, automne, fortes pluies, brouillard, pour des niveaux de luminosité et de contraste différents en fonction de la position du soleil dans la journée.

En plus de proposer des conditions environnementales diverses, ces jeux de données proposent des séquences pour des niveaux d'occlusions variants (0%, 35%, 44%). Les différentes situations d'occlusion de piéton dans une action entraînent des cas particuliers devant être pris en compte dans l'élaboration d'un modèle afin que celui-ci soit robuste.

A ce jour, ces deux jeux de données sont les deux références principales dans la recherche pour la prédiction d'intention des piétons: les autres jeux de données étant trop peu fournis ou ne reflétant pas la complexité de la tache.

Cependant, il n'existe à ce jour pas d'annotations opensource pour l'estimation de pose pour ces jeux de données. La squelettisation des piétons étant une étape fondamentale de notre approche, il sera nécessaire de réaliser une estimation de pose grâce à des approches de l'état de l'art pour chacune des séquences des jeux de données afin d'obtenir une version « squelettisée » pour répondre à la problématique.