DVDnet : Un réseau profond rapide pour le débruitage vidéo

Matias TASSANO[1,2], Julie DELON[1], Thomas VEIT[2]

[1]Laboratoire MAP5
Université Paris Descartes, France

[2]GoPro Technology France
matias.tassano@parisdescartes.fr

Résumé – Dans cet article, nous proposons un algorithme de débruitage vidéo basé sur une architecture de réseau convolutif. Alors que ces réseaux ont déjà montré toute leur efficacité pour le débruitage d'image, ils ne permettaient pas encore de rivaliser avec les méthodes plus classiques de l'état de l'art (comme celles utilisant des patchs) pour le débruitage vidéo. Notre approche permet de dépasser ces limites tout en assurant des temps de calcul nettement plus courts que les méthodes de débruitage vidéo classiques. Contrairement à d'autres approches par réseaux de neurones existants, notre algorithme présente plusieurs propriétés souhaitables telles qu'une faible empreinte mémoire, et la capacité à gérer une large gamme de niveaux de bruit avec un seul réseau, ce qui le rend très attractif pour les applications pratiques. Nous comparons notre méthode avec différents algorithmes de l'état de l'art, à la fois visuellement et pour des mesures de qualité. Les expériences montrent que notre algorithme obtient des performances supérieures ou identiques à d'autres méthodes de l'état de l'art, pour un coût de calcul nettement plus faible. Différents exemples de vidéos, les codes et l'architecture utilisés sont disponibles à l'adresse `https://github.com/m-tassano/dvdnet`.

Abstract – In this paper, we propose a state-of-the-art video denoising algorithm based on a convolutional neural network architecture. While neural network based approaches are nowadays state-of-the-art in image denoising, these methods have been unsuccessful for video denoising as their performance cannot compete with the performance of patch-based methods. However, our approach outperforms other patch-based competitors with significantly lower computing times. In contrast to other existing neural network denoisers, our algorithm exhibits several desirable properties such as a small memory footprint, and the ability to handle a wide range of noise levels with a single network model. The combination between its denoising performance and lower computational load makes this algorithm attractive for practical denoising applications. We compare our method with different state-of-art algorithms, both visually and with respect to objective quality metrics. The experiments show that our algorithm compares favorably to other state-of-art methods. Video examples, code and models are publicly available at `https://github.com/m-tassano/dvdnet`.

1 Introduction

Dans cet article, nous proposons une méthode de débruitage vidéo par réseau convolutif, nommée DVDnet, pour *Deep Video Denoising network*. L'algorithme permet d'obtenir des résultats comparables ou meilleurs que ceux des méthodes de l'état de l'art, tout en offrant des temps d'exécution rapides. Les vidéos débruitées par notre approche présentent une cohérence temporelle remarquable, un très faible scintillement (ou *flickering*), un bruit fortement réduit et une bonne préservation des détails.

1.1 Débruitage d'Image

Comparé au débruitage d'image, le débruitage vidéo apparaît comme un domaine largement sous-exploré. Récemment, de nouvelles méthodes de débruitage d'image basées sur des techniques d'apprentissage profond, telles que celles proposés par Schmidt et Roth dans [1] et Chen et Pock dans [2], ont attiré l'attention en raison de leurs performances exceptionnelles. Des méthodes comme celles-ci atteignent des performances de débruitage comparables à celles d'algorithmes par patchs bien connus tels que BM3D [3] ou Non-local Bayes (NLB [4]). Toutefois, elles reposent sur des formes spécifiques de modèles a priori sur les images. De plus, elles reposent sur de nombreux paramètres réglés à la main. Dans [5], un perceptron multicouche est appliqué avec succès pour le débruitage d'image. Un inconvénient important de tous ces algorithmes est qu'un modèle spécifique doit être entraîné pour chaque niveau de bruit.

Une autre approche récente, et déjà populaire pour le débruitage, consiste à utiliser uniquement des réseaux convolutionnels (*convolutional neural networks* ou CNN). C'est le cas des algorithmes DnCNN [6], et FFDNet [7]. Leurs performances se comparent favorablement à celles d'autres algorithmes de débruitage d'images de l'état de l'art, tant sur le plan quantitatif que sur le plan visuel. Ces méthodes sont composées d'une succession de couches convolutives avec des fonctions d'activation non linéaires. Ce type d'architecture a été appliqué au débruitage et au démosaïquage conjoint des images RGB et des images raw par Gharbi et al. [8]. Contrairement à d'autres méthodes par apprentissage profond, l'une des caractéristiques remarquables de ces méthodes est la possibilité de débruiter plusieurs niveaux de bruit en entraînant un seul modèle. Parmi ces méthodes, la méthode DnCNN proposée par Zhang et al. dans [6] a pour caractéristique qu'elle implémente un apprentissage résiduel [9], estimant le bruit existant dans l'image d'entrée plutôt que directement l'image débruitée. Dans leur article suivant [7], les mêmes auteurs proposent l'algorithme FFDNet, fortement inspiré de DnCNN et présentant des performances encore plus impressionnantes pour de forts niveaux de bruit.

1.2 Débruitage de Vidéo

En ce qui concerne le débruitage vidéo, l'une des rares méthodes utilisant des réseaux de neurones est celle proposée en 2016 par Chen et ses co-auteurs [10]. Cependant, leur algorithme ne fonctionne que sur des images en niveaux de gris et ne donne pas de résultats satisfaisants, probablement en raison des difficultés liées à l'entraînement des réseaux neuronaux récurrents. Récemment, Vogels et al. proposent dans [11] une architecture capable de débruiter des séquences d'animation dont le rendu a été fait par *ray tracing*. En matière de débruitage vidéo, les méthodes par patchs restent jusqu'à aujourd'hui les plus compétitives. Parmi les plus efficaces, on peut citer l'extension de l'algorithme BM3D au débruitage vidéo, V-BM4D [12], et l'algorithme Video non-local Bayes (VNLB [13]). Aujourd'hui, VNLB est reconnu le meilleur algorithme de débruitage vidéo en termes de qualité des résultats, de loin devant V-BM4D. Néanmoins, cet algorithme a un charge de calcul trop lourde, prenant plusieurs minutes pour débruiter une seule image. Comme nous le verrons, DVDnet est nettement plus rapide, tout en atteignant une qualité de résultat comparable.

2 Notre Approche

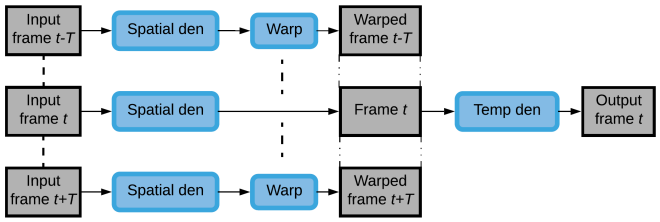

FIGURE 1 – Diagramme simplifié de l'architecture de la méthode proposée.

La cohérence temporelle et l'absence de scintillement sont deux aspects vitaux de la qualité perçue d'une vidéo. La plupart des algorithmes récents en matière de débruitage vidéo sont des extensions de leurs homologues en matière de débruitage d'image (V-BM4D pour BM3D, ou VNLB pour NLB). Il y a principalement deux facteurs qui renforcent la cohérence temporelle des résultats : l'extension de voisinages spatiaux à des voisinages spatio-temporels, et le recours à l'estimation de mouvement. La première implique que lors du débruitage d'un pixel donné, l'algorithme recherche des pixels similaires non seulement dans la même image, mais aussi dans des images voisines temporellement. De même, l'utilisation de l'estimation et/ou de la compensation de mouvement permet d'améliorer les performances de débruitage vidéo [13, 12]. Nous incorporons ces deux éléments dans notre algorithme, ainsi que différents aspects venant d'autres architectures de débruitage pertinentes basées sur CNN [7, 8, 11].

La fig. 1 affiche un diagramme simplifié de l'architecture de notre méthode. Lors du débruitage d'une frame donnée, les $2T$ frames voisines sont prises comme donnés d'entrée. Le processus de débruitage est divisé en deux étapes. Tout d'abord, les $2T+1$ frames sont débruitées individuellement avec un débruiteur spatial. Bien qu'à ce stade, la qualité d'image de chaque image individuelle soit relativement bonne, la séquence obtenue manque fortement de cohérence temporelle. Dans un deuxième temps, nous réalignons les $2T$ frames voisines sur la frame centrale en estimant le flot optique. Enfin, les $2T+1$ frames ainsi alignées sont concaténées et constituent l'entrée de notre bloc de débruitage temporel. Nous ajoutons également une carte de bruit en entrée aux débruiteurs spatiaux et temporels. L'inclusion de la carte de bruit en entrée permet le traitement des bruits spatialement variables [14]. Contrairement à d'autres algorithmes de débruitage, notre débruiteur ne prend aucun autre paramètre en entrée que la séquence d'images et l'estimation du bruit d'entrée.

Les expériences présentées dans cet article se restreignent au cas du bruit blanc gaussien additif (AWGN). Néanmoins, cet algorithme peut être facilement généralisé à d'autres types de bruit, par exemple au bruit Poissonien. Soit $\mathbf{I}$ une image sans bruit, et $\tilde{\mathbf{I}}$ sa version corrompue par un bruit blanc gaussien de moyenne nulle $\mathbf{N}$ et d'écart type σ :

$$\tilde{\mathbf{I}} = \mathbf{I} + \mathbf{N} \,. \tag{1}$$

2.1 Blocs de Débruitage Spatial et Temporel

Les caractéristiques des blocs de débruitage spatiaux et temporel résultent d'un compromis entre performance et rapidité d'exécution. Les deux blocs sont implémentés comme des réseaux de type *feed-forward*, comme indiqué dans la fig. 2. Le réseau de débruitage spatial s'inspire des architectures de [7, 8], tandis que le réseau de débruitage temporel emprunte certains éléments de [11].

Les blocs de débruitage spatial et temporel sont composés respectivement de $D_{spa} = 12$, et $D_{temp}. = 6$ couches convolutionnelles. Le nombre de *feature maps* est fixé à $W = 96$. Les sorties des couches convolutives sont suivies par les fonctions d'activation ponctuelles *ReLU* [15] $ReLU(\cdot) = \max(\cdot, 0)$. Au moment de l'entraînement, les couches de normalisation par batch (*BN* [16]) sont placées entre les couches convolutives et la fonction *ReLU*. Au moment de l'évaluation, les couches de normalisation par batch sont supprimées, et remplacées par une couche affine qui applique la normalisation apprise. La taille spatiale des noyaux convolutifs est de 3×3, et le *stride* est fixé à 1. Dans les deux blocs, les images entrées sont d'abord transformées en 4 images sous-échantillonnées d'un facteur 2 dans chaque dimension. Le principal avantage d'effectuer le débruitage sur ces images plus petites est la réduction importante des temps de calcul et des besoins en mémoire [7, 14]. L'image de pleine résolution est recréée à la fin grâce aux 4 images restaurées, comme dans [17]. Les deux blocs comportent des connexions résiduelles [9].

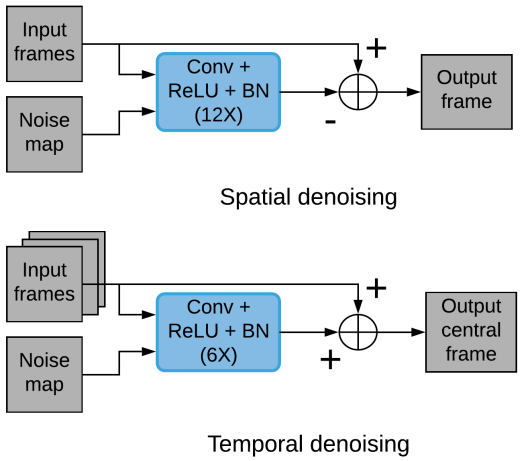

FIGURE 2 – Diagramme simplifié de l'architecture des débruiteurs spatial et temporel. Note : les sorties des débruiteurs spatiaux sont les entréss du débruiteur temporel, après l'alignement, voir fig. 1

3 Détails de l'entraînement

Les réseaux de débruitage spatial et temporel sont entraînés séparément, le débruitage spatial étant entrainé en premier car ses sorties sont utilisées pour entrainer le débruitage temporel. Les deux blocs sont entraînés à l'aide de patchs. Dans le cas du débruitage spatial, l'ensemble des données d'apprentissage est composé de paires de patchs entrées-sorties $\left\{\left((\tilde{\mathbf{I}}^j, \mathbf{M}^j),\, \mathbf{I}^j\right)\right\}_{j=0}^{m_s}$ générés par addition de bruit blanc gaussien d'écart type $\sigma \in [0,\, 55]$ aux patchs d'origine $\mathbf{I}^j$. Les cartes de bruit correspondantes sont notées $\mathbf{M}^j$ (elles sont constantes avec tous leurs éléments égaux à σ). Un total de $m_s = 1024000$ patchs sont extraits de la base de données *Waterloo Exploration Database* [18]. Des patchs 50×50 sont extraits au hasard d'images échantillonnées aléatoirement de l'ensemble des données d'entraînement. L'apprentissage résiduel est utilisé : le réseau sort une estimation du bruit d'entrée $\mathcal{F}_{spa}(\,\tilde{\mathbf{I}}; \theta_{spa}\,) = \hat{\mathbf{N}}$, alors que l'image débruitée est calculée en soustrayant le bruit de sortie à l'entrée bruitée

$$\hat{\mathbf{I}}(\,\tilde{\mathbf{I}}; \theta_{spa}\,) = \tilde{\mathbf{I}} - \mathcal{F}_{spa}(\,\tilde{\mathbf{I}}; \theta_{spa}\,)\,. \tag{2}$$

La fonction de perte du débruitage spatial s'écrit

$$\mathcal{L}_{spa}(\theta_{spa}) = \frac{1}{2m_s} \sum_{j=1}^{m_s} \left\| \hat{\mathbf{I}}^j(\,\tilde{\mathbf{I}}^j; \theta_{spa}\,) - \mathbf{I}^j \right\|^2 , \tag{3}$$

où θ_{spa} est la collection de tous les paramètres.

Comme pour le débruitage temporel, l'ensemble des données d'apprentissage est constitué de paires d'entrée-sortie

$$P_t^j = \left\{\left((\,({}^w\hat{\mathbf{I}}_{t-T}^j, \ldots, \hat{\mathbf{I}}_t^j, \ldots, {}^w\hat{\mathbf{I}}_{t+T}^j), \mathbf{M}^j\,),\, \mathbf{I}_t^j\right)\right\}_{j=0}^{m_t} ,$$

où $({}^w\hat{\mathbf{I}}_{t-T}^j, \ldots, \hat{\mathbf{I}}_t^j, \ldots, {}^w\hat{\mathbf{I}}_{t+T}^j)$ est une collection de $2T+1$ patchs spatiaux extraits au même emplacement dans des images contigues temporellement. Ces patchs sont générés en ajoutant du bruit blanc gaussien d'écart-type $\sigma \in [0,\, 55]$ et en les débruitant à l'aide du débruitage spatial. Ensuite, les $2T$ patchs contigus au patch de référence central $\mathbf{I}_t^j$ sont alignés géométriquement par rapport à ce dernier par estimation du mouvement. Le patch ${}^w\hat{\mathbf{I}}_l^j$ est donc obtenu à partir de $(\hat{\mathbf{I}}_l^j,\, \hat{\mathbf{I}}_t^j)$. Pour estimer le mouvement et aligner les images, nous utilisons l'algorithme DeepFlow [19]. La carte de bruit $\mathbf{M}^j$ est la même que celle utilisée dans la phase de débruitage spatial. Un total de $m_t = 450000$ échantillons d'entraînement sont extraits de l'ensemble d'entraînement de la base de données DAVIS [20]. La taille spatiale des patchs est de 44×44, alors que la fenêtre temporelle est de $2T + 1 = 5$. La fonction de perte pour le débruitage temporel est la suivante

$$\mathcal{L}_{temp}(\theta_{temp}) = \frac{1}{2m_t} \sum_{j=1}^{m_t} \left\| \hat{\mathbf{I}}_{temp,\,t}^j - \mathbf{I}_t^j \right\|^2 , \tag{4}$$

où $\hat{\mathbf{I}}_{temp,\,t}^j = \mathcal{F}_{temp}(P_t^j; \theta_{temp})$.

Dans les deux cas, l'algorithme ADAM [21] est appliqué pour minimiser la fonction de perte, avec tous ses hyper-paramètres mis à leurs valeurs par défaut. Le nombre d'époques est fixé à 80, et la taille du mini-batch est de 128. Dans les deux cas, le pas de descente commence à $1\mathrm{e}{-3}$ pour les premières 50 époques, puis passe à $1\mathrm{e}{-4}$ pour les 10 suivantes, et finalement passe à $1\mathrm{e}{-6}$ pour le reste de l'entraînement. Les données sont augmentées cinq fois par l'introduction de différents changements d'échelles et retournements aléatoires. Pendant les 60 premières époques, une étape d'orthogonalisation des noyaux convolutifs est appliquée comme un moyen de régularisation, comme suggéré dans [7, 14].

4 Results

Deux ensembles de tests différents ont été utilisés pour l'évaluation comparative de notre méthode : l' ensemble DAVIS, et l'ensemble Set8. Ce dernier est composé de 4 de séquences de couleurs de la collection *Derf's Test Media collection* [1] et de 4 séquences de couleurs capturées avec un appareil GoPro. L'ensemble DAVIS contient 30 séquences couleur de résolution 854×480. Les séquences de Set8 ont été réduites à une résolution de 960×540. Dans tous les cas, les séquences sont limitées à un maximum de 85 frames. Par souci d'équité, nous utilisons le même algorithme d'estimation du mouvement (DeepFlow) pour DVDnet et VNLB.

En général, DVDnet produit des séquences présentant une cohérence temporelle remarquable, avec très peu de scintillement, en particulier dans les zones constantes, où les algorithmes basés sur des patchs gardent souvent un bruit résiduel basse fréquence. Un exemple peut être observé dans la fig. 3 (il est préférable de le visualiser au format numérique). Le bruit basse fréquence temporellement décorrélé dans les zones plates apparaît comme particulièrement gênant visuellement. D'autres exemples vidéo peuvent être trouvés sur le site web `https://github.com/m-tassano/dvdnet`.

La table 1 montre une comparaison de $PSNR$ sur les deux ensembles de données couleur (les valeurs affichées sont la moyenne de toutes les séquences du jeu de test, le PNSR d'une séquence est calculé comme la moyenne des PSNR de ses images). On peut observer que pour des valeurs de bruit plus faibles, VNLB est le plus performant. DVDnet a tendance à trop débruiter en présence de faible bruit. Cependant, pour des valeurs de bruit plus élevées, DVDnet surpasse VNLB.

TABLE 1 – Comparaison de $PSNR$ sur différents testsets.

DAVIS testset	DVDnet	VNLB	V-BM4D
$\sigma = 10$	38.13	**38.85**	37.58
$\sigma = 20$	**35.70**	35.68	33.88
$\sigma = 30$	**34.08**	33.73	31.65
$\sigma = 40$	**32.86**	32.32	30.05
$\sigma = 50$	**31.85**	31.13	28.80
Set8 testset	**DVDnet**	**VNLB**	**V-BM4D**
$\sigma = 10$	36.08	**37.26**	36.05
$\sigma = 20$	33.49	**33.72**	32.19
$\sigma = 30$	**31.79**	31.74	30.00
$\sigma = 40$	**30.55**	30.39	28.48
$\sigma = 50$	**29.56**	29.24	27.33

4.1 Running times

Notre méthode permet d'obtenir des temps d'inférence rapides, grâce à son architecture simple. Elle prend moins de $8s$ pour débruiter une image couleur de 960×540, ce qui est environ 20 fois plus rapide que V-BM4D, et environ 50 fois plus rapide que VNLB. Même sur CPU, DVDnet est environ un ordre de grandeur plus rapide que ces méthodes. Sur les $8s$ nécessaires pour débruiter une image, $6s$ sont consacrés à la compensation du mouvement des images temporelles voisines. La table 2 compare les temps de calcul des différents algorithmes.

1. `https://media.xiph.org/video/derf`

(b) Bruité $\sigma = 50$ (c) V-BM4D (d) VNLB (e) DVDnet

FIGURE 3 – *Comparaison des résultats*. De gauche à droite : image bruitée (PSNR$_{seq} = 14.15dB$), résultat de V-BM4D (PSNR$_{seq} = 24.91dB$), résultat de VNLB (PSNR$_{seq} = 26.34dB$), résultat de DVDnet (PSNR$_{seq} = 26.62dB$). On remarquera la clarté du texte débruité et l'absence de bruit résiduel basse fréquence et de bruit chromatique dans le résultat de DVDnet. Il est préférable de visualiser ces résultats au format pdf.

TABLE 2 – *Comparaison des temps de calcul.* Temps nécessaire pour débruiter une image couleur de résolution 960×540. Note : les valeurs affichées pour VNLB n'incluent pas le temps nécessaire pour estimer le mouvement.

Méthode	V-BM4D	VNLB	DVDnet (CPU)	DVDnet (GPU)
Temps (s)	156	420	19	8

5 Conclusions

Dans cet article, nous avons présenté DVDnet, un algorithme de débruitage vidéo améliorant l'état de l'art. Les résultats de débruitage de DVDnet présentent une cohérence temporelle remarquable, un très faible scintillement et une excellente préservation des détails. L'algorithme permet d'obtenir des temps d'exécution qui sont au moins un ordre de grandeur plus rapides que ceux des algorithmes concurrents. Bien que les résultats présentés dans cet article soient valables pour le bruit gaussien, notre méthode pourrait être étendue à d'autres types de bruit.

Références

[1] U. Schmidt and S. Roth, "Shrinkage fields for effective image restoration," 2014, number 8, pp. 2774–2781.

[2] Y. Chen and T. Pock, "Trainable Nonlinear Reaction Diffusion : A Flexible Framework for Fast and Effective Image Restoration," *IEEE TPAMI*, vol. 39, no. 6, pp. 1256–1272, 2017.

[3] K Dabov, A Foi, and V Katkovnik, "Image denoising by sparse 3D transformation-domain collaborative filtering," *IEEE TIP*, vol. 16, no. 8, pp. 1–16, 2007.

[4] M. Lebrun, A. Buades, and J. M. Morel, "A Nonlocal Bayesian Image Denoising Algorithm," *SIIMIS*, vol. 6, no. 3, pp. 1665–1688, 2013.

[5] H.C. Burger, C.J. Schuler, and S. Harmeling, "Image denoising : Can plain neural networks compete with BM3D ?," 2012, pp. 2392–2399.

[6] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser : Residual learning of deep CNN for image denoising," *IEEE TIP*, vol. 26, no. 7, pp. 3142–3155, 2017.

[7] K. Zhang, W. Zuo, and L. Zhang, "FFDNet : Toward a Fast and Flexible Solution for CNN based Image Denoising," *IEEE TIP*, vol. 27, no. 9, pp. 4608–4622, 2018.

[8] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand, "Deep joint demosaicking and denoising," *ACM TOG*, vol. 35, no. 6, pp. 1–12, 2016.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016, pp. 770–778.

[10] X. Chen, L. Song, and X. Yang, "Deep RNNs for video denoising," in *Applications of Digital Image Processing XXXIX*. SPIE, 2016, vol. 9971, p. 99711T.

[11] T. Vogels, F. Rousselle, B. McWilliams, G. Röthlin, A. Harvill, D. Adler, M. Meyer, and J. Novák, "Denoising with kernel prediction and asymmetric loss functions," *ACM TOG*, vol. 37, no. 4, pp. 124, 2018.

[12] M. Maggioni, G. Boracchi, A. Foi, and K. Egiazarian, "Video denoising, deblocking, and enhancement through separable 4-D nonlocal spatiotemporal transforms," *IEEE TIP*, vol. 21, no. 9, pp. 3952–3966, 2012.

[13] P. Arias and J.-M. Morel, "Video denoising via empirical Bayesian estimation of space-time patches," *JMIV*, vol. 60, no. 1, pp. 70–93, 2018.

[14] M. Tassano, J. Delon, and T. Veit, "An Analysis and Implementation of the FFDNet Image Denoising Method," *IPOL*, vol. 9, pp. 1–25, 2019.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *NIPS*, pp. 1–9, 2012.

[16] S. Ioffe and C. Szegedy, "Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *ICML*, 2015, pp. 448–456.

[17] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," in *CVPR*, 2016, pp. 1874–1883.

[18] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, and L. Zhang, "Waterloo Exploration Database : New Challenges for Image Quality Assessment Models," *IEEE TIP*, vol. 26, no. 2, pp. 1004–1016, 2017.

[19] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow : Large displacement optical flow with deep matching," in *ICCV*, Dec. 2013.

[20] A. Khoreva, A. Rohrbach, and B. Schiele, "Video object segmentation with language referring expressions," in *ACCV*, 2018.

[21] D.P. Kingma and J.L. Ba, "ADAM : a Method for Stochastic Optimization," *ICLR*, pp. 1–15, 2015.