Réseaux de Neurones Légers Multi-Échelle pour la Super-Résolution Stylisée d'Images

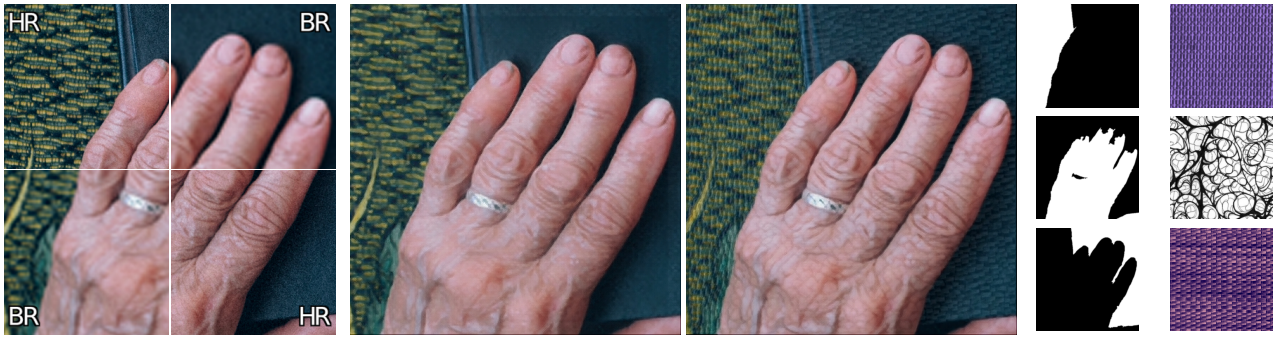
Thibault Durand, Julien Rabin, David Tschumperlé

Réseaux de Neurones Légers Multi-Échelle pour la Super-Résolution Stylisée d'Images*

Thibault Durand Julien Rabin David Tschumperlé

Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France *

{Thibault.Durand, Julien.Rabin, David.Tschumperle}@unicaen.fr



HR & BR (interp. bicubique) Résultat de l'étape-SR (×4) Résultat de l'étape-ST Masques β_i Styles Y_i

FIGURE 1 – *Notre méthode de super-résolution (SR) repose sur deux étapes : une image Basse Résolution (BR) est tout d'abord sur-échantillonnée par interpolation bicubique, puis envoyée à un réseau convolutif multi-échelle léger (Étape-SR ; 120*K *paramètres). Au sein de ce même réseau, le résultat peut être stylisé localement (Étape-ST ; 35*K *paramètres) à l'aide de masques (β_i) choisis par l'utilisateur et de « styles » Y_i pré-entrainés. Zoomer sur l'image pour mieux visualiser les détails. D'autres exemples de résultats sont disponibles sur la page [1].*

Résumé

Les algorithmes de Super-Résolution (SR) pour le traitement d'images ont beaucoup progressé ces dernières années. L'utilisation de réseaux convolutifs profonds fournit actuellement les résultats les plus impressionnants. Cependant, très peu de ces techniques sont utilisées en pratique dans des logiciels d'infographie, du fait de pré-requis techniques importants (taille des réseaux en mémoire, calculs nécessitant des GPUs), mais aussi parce que l'utilisateur n'a que très peu de contrôle sur le résultat généré. Pour pallier ces problèmes, nous proposons une approche de super-résolution en deux temps, baptisée « Super-Résolution stylisée », qui repose sur une architecture multi-échelle composée de branches convolutives parallèles et indépendantes pour la génération d'images. Notre contribution est ainsi double, avec la description : *i.* d'un réseau *léger* inspiré des méthodes de filtrage classiques de traitement d'images, facilement utilisable sur des systèmes sans GPU et à capacité mémoire limitée ; *ii.* d'un réseau polyvalent et personnalisable, autorisant un contrôle fin de la stylisation du rendu.

* *Ce travail a été financé par la Région Normandie dans le cadre d'une thèse de doctorat 100% RIN (Réseau d'Intérêt Normand). Une version en anglais de ce document [2] a été acceptée pour publication dans les actes de IEEE ICIP 2021.*

Mots-Clés : *Super-résolution d'images ; Transfert de Style ; Réseaux de neurones légers et interactifs ; Synthèse de texture.*

1 Introduction

L'objectif de la Super-Résolution (SR) est de restaurer une image Haute Résolution (HR) à partir de sa version Basse Résolution (BR). Le passage de HR à BR est généralement modélisé par une dégradation (par exemple un flou gaussien) suivi d'un sous-échantillonnage, qui retirent et modifient principalement les hautes et moyennes fréquences de l'image. Dans ce travail, nous considérons le problème de la SR (avec un facteur de sur-échantillonnage noté $\times k$), tel qu'illustré en figure 1. Il s'agit alors non seulement de restaurer la géométrie de l'image (telle que les contours de la main, devenus flous ou aliasés) mais également de reconstruire les textures qui sont fortement dégradées, voire perdues (grain de la couverture du livre).

Single-Image SR (SI-SR) Cette catégorie d'approches consiste à entraîner un modèle à restaurer l'image HR à partir d'une image BR seule. SRCNN [3] a été le premier réseau de neurones profond purement convolutif proposé dans la littérature pour cette tâche. Pendant l'entraînement, des images d'une base de données sont artificiellement dégradées pour créer des paires (BR,HR) et le PSNR moyen (rapport signal à bruit) des images restaurées par rapport aux images HR est utilisé comme critère d'évaluation du réseau. Depuis, cette stratégie a été reprise avec différentes architectures plus larges et plus profondes afin d'améliorer ce PSNR. Parmi ces architectures, certaines utilisent directement l'image BR (voir par exemple [4, 5]), tandis que d'autres, utilisent une interpolation bicubique ou bilinéaire de BR [6, 7, 8].
Ces méthodes sont toutes entraînées à partir de pénalités calculées pixel par pixel, utilisant soit l'erreur quadratique moyenne (MSE) ou l'erreur moyenne absolue (Norme L1). Le critère PSNR permet de quantifier et de comparer les résultats de manière certes objective, mais ne prend pas en compte la perception humaine. Par conséquent, comme montré dans [9], les textures perdues ou fortement dégradées ne sont pas reconstruites. Pour pallier ce problème, d'autres critères d'optimisation « perceptuels » ont été proposés, inspirées des méthodes de synthèse d'images [10] et de textures [11], générant des résultats visuellement plus satisfaisants, bien qu'ayant un PSNR plus bas. Alors que les méthodes utilisant des pénalités purement pixelliques se contentent de reconstruire des caractéristiques principalement géométriques, celles reposant sur des critères perceptuels exploitent des caractéristiques sémantiques pour synthétiser des textures réalistes. Par exemple, EnhanceNet [12] s'appuie sur le critère d'optimisation de Gatys [11, 13] utilisant le réseau VGG [14] pré-entrainé pour la classification d'ImageNet. A contrario, SRGAN [9] utilise un réseau adversaire qui est simultanément entrainé [10] pour reconstruire, voire halluciner des détails.

Reference-Based SR (Ref-SR) Ces techniques visent à transférer des caractéristiques d'une image haute résolution (dites de référence) à l'image reconstruite. Certaines méthodes combinent réseaux de neurones avec mise en correspondance de patchs, comme [15], pour améliorer les performances, à la condition que les images soient de la même scène. Plus récemment, cette mise en correspondance est faite par le réseau grâce aux caractéristiques « perceptuelles » de VGG [14], soit en l'incorporant au modèle [16], soit en apprenant un réseau descripteur à partir de VGG (TTSR [17]). À nouveau, les performances sont accrues lorsque l'image de référence provient de la même scène que la BR.

Limitations Les méthodes SI-SR comme Ref-SR souffrent en pratique des mêmes limitations. Pour commencer, le nombre de paramètres utilisés est de plus en plus important, dépassant par exemple plusieurs dizaines de millions lorsque des encodeurs sont associés au réseau pour extraire des descripteurs sémantiques, ce qui nécessite une capacité de stockage et de calcul non négligeable pour une intégration dans un logiciel grand-public de traitement d'images. De plus, puisque profonds et larges, ces réseaux sont parfois difficiles à entraîner et à interpréter. Surtout, ils ne laissent pas ou peu de contrôle à l'utilisateur. Il s'agit d'une problématique assez peu étudiée dans la littérature en vision par ordinateur, qui s'attache souvent à proposer une solution entièrement automatique, a contrario du processus utilisé en infographie qui laisse la part belle au processus créatif de l'artiste. Une exception notable est la méthode de [18] pour la colorisation d'images qui permet à l'utilisateur de choisir parmi différentes solutions proposées. À ce titre, la SR en tant que synthèse d'image est un problème inverse mal posé où différentes solutions sont envisageables. Dans ce contexte, les méthodes de Ref-SR [16] comme TTSR [17] proposent un premier pas dans cette direction. Toutefois, une fois l'image de référence sélectionnée, ces approches restent complètement automatiques et ne proposent pas de contrôle local, comme démontré expérimentalement dans ce travail.

Contributions Premièrement, nous présentons dans la section 2.1 une architecture *légère* et *multi-échelle* pour la Super-Résolution, inspirée des techniques de décomposition en espace échelle pour le traitement d'images. Comme démontré par [19] pour la synthèse de textures, un réseau très léger peut être utilisé pour synthétiser des textures. Notre architecture est composée de branches parallèles indépendantes, chacune spécialisée dans une bande de fréquences donnée. Ce premier réseau compact permet de reconstruire efficacement les caractéristiques géométriques de l'image (Étape SR en figure 1). Deuxièmement, nous proposons dans la section 2.2 de nouvelles branches parallèles à ce réseau appelées « branches de style ». Ces dernières permettent à l'utilisateur de raffiner le résultat produit, en générant les détails hautes fréquences désirés à la manière d'un transfert de style (Étape ST de la figure 1).

2 Réseau multi-échelle pour la super-résolution stylisée

Dans cette partie, nous présentons notre **R**éseau de **N**eurones **L**éger **M**ulti-**E**chelle pour la **S**uper-**R**ésolution **S**tylisée **RLME-SR**, qui combine des branches de Super Résolution (étape-SR, présentée en § 2.1) avec des branches de stylisation (étape-ST, présentée en § 2.2). Une représentation de l'architecture est représentée Fig. 2.

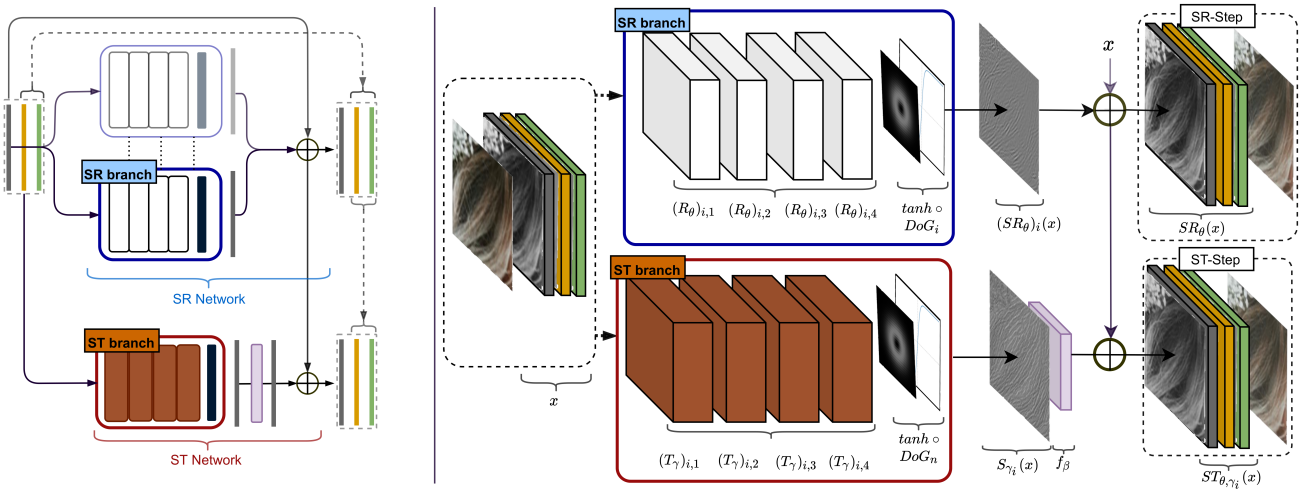


FIGURE 2 – *Schéma de l'architecture du Réseau (à gauche) composé de branches parallèles (SR et ST) (détaillées à droite), respectivement pour la super-résolution et pour la stylisation. La sortie est générée à partir d'une représentation multi-échelle des détails construite grâce à des Différences de Gaussiennes (DoG).*

Nous adoptons les notations suivantes : $\mathbf{X}$ (resp. $\mathbf{x}$) $\in \mathbb{R}^{K\times N\times N\times 3}$ désigne un tenseur de K images couleurs HR (resp. BR) de taille $N \times N$. Nous utilisons les mêmes notations pour l'entraînement et l'évaluation. La k-ième image BR, notée $x_k \in \mathbb{R}^{N\times N\times 3}$ est encodée dans l'espace de couleur YCbCr. L'image BR x_k a été sur-échantillonnée à la taille de l'image HR par interpolation bicubique avant d'être traitée par le réseau.

2.1 Réseau de Neurones Multi-Échelle (étape-SR)

Le réseau principal, appelé Réseau-SR, est représenté sur la partie supérieure de la Fig. 2. Il a pour but de permettre une bonne reconstruction de la géométrie de l'image avec une empreinte mémoire contenue.

Architecture. Le réseau SR se base sur une décomposition multi-échelle. Il est constitué de n branches parallèles, indépendantes, dont les sorties sont linéairement combinées à l'image BR pour générer la sortie. Chaque branche i est spécialisée à une échelle pré-déterminée grâce à l'utilisation finale d'une Différence de Gaussiennes (DoG_i) après une série de convolutions. Les écarts-types des noyaux gaussiens permettent de contrôler le domaine fréquentiel de chaque contribution, la DoG_i de la branche i agissant comme un filtre passe-bande dont les fréquences de coupures sont définies par σ_{i-1} et σ_i. Afin d'assurer l'indépendance des contributions de chaque branche, ces paramètres sont définis selon une loi géométrique :

$$\sigma_i = \sigma_0 \; q^i \; \forall \; i \in \{1, \ldots, n-1\}$$

avec $\sigma_0 = 1$ et $q = 1,6$.

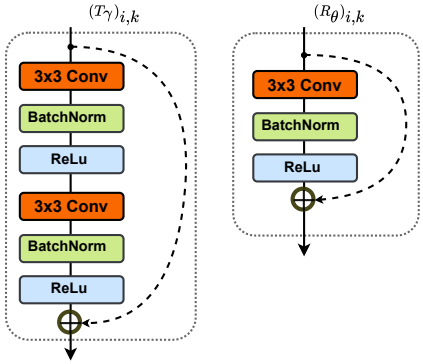


FIGURE 3 – $R_{\theta_{i,k}}$ *(resp.* $T_{\gamma_{j,k}}$*) est le* k*-ième module de convolutions de la branche* i *(resp.* j*) du réseau-SR (resp. réseau-ST).*

Cette approche s'inspire des filtres Laplacien normalisés utilisés en détection multi-échelle, dont les DoG correspondent à l'approximation robuste au premier ordre exploité par SIFT [20]. On dénote θ les paramètres entraînables du modèle et $R_{\theta_{i,k}}$ le k-ième module de convolutions ($1 \leq k \leq 4$) de la branche i. Dans le but de réduire le nombre de paramètres de chaque branche, nous nous sommes basés sur une architecture ResNet [21]. Chacun de ces modules est construit à partir d'une couche de convolution 3×3 de 18 à 33 canaux internes selon la branche, les branches hautes fréquences ayant plus de paramètres que les branches basses fréquences. Ces convolutions sont suivies d'une normalisation par batch (BatchNorm) et d'une non-linéarité ($ReLU$), comme résumé en figure 3. Dans notre configuration, la sortie de chaque branche peut

s'écrire : $(\mathrm{SR}_\theta)_i(x_k) = [tanh \circ DoG_i \circ R_{\theta_{i,4}} \circ R_{\theta_{i,3}} \circ R_{\theta_{i,2}} \circ R_{\theta_{i,1}}](x_k)$. La DoG de chaque branche est suivie d'une non linéarité (tangente hyperbolique) et d'une BatchNorm, indispensable à la bonne convergence du modèle. Finalement, la sortie du réseau SR dans l'espace de couleur YCbCr est construite à partir de la somme des sorties des branches parallèles, ajoutée à l'interpolation bicubique BR fournie en entrée. La concaténation avec les canaux de couleurs issus de l'entrée donne la sortie finale qui peut s'écrire, pour le pixel t :

$$\mathrm{SR}_\theta(x_k)(t) = x_k(t) + \left[\sum_{i=1}^{n}(\mathrm{SR}_\theta)_i(x_k)(t);\, 0;\, 0\right] \in \mathbb{R}^3.$$

En utilisant $n = 6$ branches, notre réseau SR contient approximativement 120K paramètres.

Entraînement du réseau SR Nous utilisons comme critère d'optimisation la combinaison linéaire de la MSE avec une pénalité de type perceptuelle pour entraîner le réseau SR, puisqu'il est bien connu qu'utiliser la MSE seule favorise la génération d'images sans texture [12]. Le réseau SR est ainsi entraîné en optimisant le critère $\min_\theta \mathcal{L}_{\mathrm{SR}}(\mathbf{X}, \mathrm{SR}_\theta(\mathbf{x}))$, avec pour fonction objectif :

$$\mathcal{L}_{\mathrm{SR}}(\mathbf{X}, \mathbf{Y}) = \sum_{k=1}^{K} \|X_k - Y_k\|^2 + \lambda_{\mathrm{SR}} \mathcal{L}_{\mathrm{Perc}}(X_k, Y_k). \tag{1}$$

où $\|.\|$ correspond à la norme de Frobenius, et $\mathcal{L}_{\mathrm{Perc}}(x, y) = \sum_{\ell \in L_{\mathrm{Perc}}} \|\phi_\ell(x) - \phi_\ell(y)\|^2$ définit la pénalité perceptuelle, où $\phi_\ell(.)$ correspond aux *réponses* normalisées de la ℓ-ième couche du VGG-16 [14]. Nous utilisons le réseau VGG-16 fourni par la bibliothèque Python TensorFlow de l'API Keras. Pour capturer les détails des images à différentes échelles, nous considérons différentes couches du VGG-16 : $\mathrm{L}_{\mathrm{Perc}} = \{5, 9, 13\}$. Le paramètre λ_{SR} est fixé à 0.05.

2.2 Branches de Style (Étape-ST)

Afin de générer les détails manquants en hautes fréquences, tout en permettant le contrôle par l'utilisateur des détails générés par le réseau, nous adoptons une stratégie de transfert de style par l'exemple. Ce principe est assez similaire à l'approche proposée dans [22] où les textures sélectionnées par l'utilisateur sont transférées par copie locale de patchs. Ici, nous proposons d'intégrer en parallèle du réseau SR des branches de style permettant de transférer des textures hautes fréquences depuis l'image de référence sur l'image intermédiaire (SR). La Figure 2 illustre ce principe (avec une seule branche par soucis de lisibilité). Chaque branche de style est entraînée indépendamment afin d'ajouter des détails cohérents à l'image issue du réseau SR, entraîné au préalable.

Architecture. Dénotant γ les paramètres entraînables des branches de style, $T_{\gamma_{i,k}}$ fait référence à un module résiduel (le numéro k de la branche i) constitué de 2 couches de convolutions 3×3 sur 20 canaux, comme décrit dans [23] et illustré en Figure 3. Notez ainsi que chaque branche de style possède ainsi une capacité doublée par rapport aux branches SR puisqu'elles doivent créer des structures absentes dans l'image d'entrée. Le nombre de paramètres γ par branche de style est inférieur à 35K.
Chaque branche utilise à nouveau 4 de ces modules T. La sortie de la i-ième branche de style peut s'écrire : $S_{\gamma_i}(x_k) = [tanh \circ DoG_{ST} \circ T_{\gamma_{i,4}} \circ T_{\gamma_{i,3}} \circ T_{\gamma_{i,2}} \circ T_{\gamma_{i,1}}](x_k)$. Le filtrage avec une DoG à la fin de la branche permet de ne générer les détails qu'à l'échelle souhaitée. Ce filtre DoG_{ST} est un passe-haut implémenté à l'aide d'une DoG entre un noyau d'écart type σ_{ST} et un noyau d'écart type nul $\sigma_{ST,0} = 0$.

Stylisation interactive À l'inverse d'un transfert de style entièrement automatique à la manière de [23, 19], nous souhaitons pouvoir contrôler l'ajout des détails injectés dans le résultat final. Pour cela un module de BatchNorm f_β est utilisé, où les paramètres affines sont fixés, de manière à imposer une espérance nulle et un écart-type égal à σ_{x_k}, l'écart-type de l'image d'entrée x_k, modulé par un masque β

$$f_\beta(y_k)(t) = \beta(t)\sigma_{x_k} \frac{y_k - \overline{y_k}}{\sigma_{y_k}}$$

où $\beta(t)$ est la valeur du masque au pixel t défini par l'utilisateur, comme illustrée en Figure 1. Pendant l'entraînement, β est égal à 1.
Avec m branches de style, la sortie du réseau RLME pour le pixel t s'écrit ainsi :

$$\mathrm{ST}_{\theta,\gamma}(x_k)(t) = \mathrm{SR}_\theta(x_k)(t) + \left[\sum_{i=1}^{m} f_\beta(S_{\gamma_i}(x_k)(t)); 0; 0\right]$$

Entraînement de la branche ST Séparément, pour chaque image de référence Y_i, nous optimisons par *batch* de 6 images le critère $\min_\gamma \mathcal{L}_{\mathrm{ST}}(\mathbf{X}, Y_i, \mathrm{ST}_{\theta,\gamma}(\mathbf{x}))$, où la fonction objectif $\mathcal{L}_{ST}$ s'écrit

$$\mathcal{L}_{\mathrm{ST}}(\mathbf{X}, Y_i, \mathbf{Z}) = \sum_{k=1}^{K} \lambda_{ST} \mathcal{L}_{\mathrm{Perc}}(X_k, Z_k) + \mathcal{L}_{\mathrm{Tex}}(Y_i, Z_k).$$

où la distance perceptuelles entre textures $\mathcal{L}_{\mathrm{Tex}}$ [11] est construite à partir de la matrice de Gram G

$$\mathcal{L}_{\mathrm{Tex}}(x, y) = \sum_{\ell \in L_{\mathrm{Tex}}} \|G(\phi_\ell(x)) - G(\phi_\ell(y))\|^2. \tag{2}$$

Pour stabiliser l'apprentissage et le rendre invariant au contraste de l'image de référence choisie, les canaux de luminance de Y_i et X_k sont normalisés (moyenne nulle

et écart-type de 0.2). De cette manière, le paramètre $\lambda_{ST} = 0.5$ est fixé pour chaque branche de style. Afin de privilégier la capture des détails à petites échelles, et pour préserver les caractéristiques à grandes échelles de la sortie du réseau SR, nous considérons les couches du VGG-16 suivantes : $L_{Tex} = \{2, 5, 7, 9\}$ avec $L_{Perc} = \{7\}$.

Filtrage des artefacts en damier Comme précisé auparavant, l'image d'entrée est interpolée à la resolution désirée, et ce afin d'éviter les artefacts très hautes fréquences en damier qui sont généralement obtenus avec les filtres convolutifs transposés [24]. Ces artefacts sont bien absents lorsque seule la MSE est utilisée pour l'entraînement. Toutefois, en raison de l'utilisation de modules de sous-échantillonnage (*pooling*) dans VGG, le réseau entraîné avec la pénalité perceptuelle (1) génère ce type d'artefacts en damier, toujours notables dans les méthodes utilisant ce type d'approche (voir par exemple [23, 19]), et ce même en remplaçant les modules de *max-pooling* par des *average pooling*.
Pour supprimer totalement ces artefacts très identifiables, mais laissés intacts par les filtres DoG passe-haut dans les dernières branches (DoG_6 dans la dernière branche de SR ainsi que DoG_{ST} dans les branches de style), nous utilisons un filtre médian de taille 2×2 à la fin de ces branches, y compris pendant l'entraînement.

3 Experiences

Données et configuration d'apprentissage Pour évaluer et comparer notre approche, nous entraînons et évaluons les réseaux sur les données *DIV2K* [25] (interpolation bicubique, $\times 4$ SR). Les paires d'images (BR,HR) ne sont fournies que pour les données d'entraînement et de validation. Nous utilisons les 150 dernières images (parmi 800) des paires d'images d'entraînement comme données d'évaluation. Pendant l'entraînement, des patchs carrés (254x254) sont extraits depuis les images d'entraînement `001` à `650`. Plus de 20K patchs (de variance minimum) sont utilisés pour l'entraînement.

Évaluation du réseau SR Même si l'enjeu principal de notre approche est d'offrir un contrôle facile d'utilisation et intuitif pour la super résolution stylisée, nous proposons tout de même une évaluation comparative des performances du réseau SR (pour $\lambda_{SR} = 0$) par le PSNR moyen. Bien que le PSNR soit un critère imparfait [9], elle reste une mesure de référence.
La Table 1 montre les gains moyens en PSNR entre l'image générée et l'approximation bicubique pour différentes méthodes. L'évaluation est faîte sur le canal de luminance des 150 images de test issues de *DIV2K* [25] comme décrit ci-dessus, ainsi que sur 3 autres jeux de données (*Set5*, *Set14*, and *BSD100*). Une bordure de 15 pixels correspondant à la taille typique du support de notre *DoG* aux noyaux les plus larges est retirée pour éviter des effets de bords.

Model	# Param.	DIV2K	Set5	Set14	Bsd100
RLME	120K	0.86	2.28	0.98	0.59
SRCNN [3]	440K	0.72	1.95	0.81	0.54
EDSR [8]	1517K	1.54	4.71	1.86	1.24
SRGAN [9]	1554K	-0.50	2.00	-0.19	-0.48
RDN [26]	2205K	1.59	4.76	1.83	1.29

TABLE 1 – *Comparaison des gains moyens en PSNR (gains) sur différents jeux de données pour la SR* $4\times$*, par rapport à l'approximation bicubique.*

Avec bien moins de paramètres que les autres méthodes, l'architecture proposée obtient un niveau de performance intéressant, et permet de constituer une base pour ajouter les textures dans l'étape de stylisation.
La Figure 4 montre que, tout comme SRCNN [3] (en large configuration) et EDSR [8], RLME permet une bonne reconstruction des textures simples (comme les lignes ou les bordures) compte tenu du nombre de paramètres utilisés. En toute logique, le réseau SR (RLME) seul n'est cependant pas capable de générer les textures perdues, contrairement aux réseaux très profonds et utilisant des pénalités adverses comme SRGAN [9], qui a approximativement 13 fois plus de paramètres que notre méthode.

FIGURE 4 – *Comparaisons visuelles de résultats pour la SR* $4\times$ *sur 1 image de test de* Div2K *[25].*

Ainsi, notre méthode laisse l'utilisateur choisir le type de texture à introduire dans l'image via différentes images de style, comme illustré en Figure 1 (ST-step) et discuté dans le prochain paragraphe.

Stylisation à partir de l'étape-SR (étape-ST). Comme décrit dans § 2.2, les branches de style sont entraînées une par une, après avoir entraîné le réseau SR. En pratique, pour tous les styles présentés, on utilise $\sigma_{ST} = 2.0$ ce qui autorise la stylisation des hautes mais aussi moyennes fréquences de l'image SR. Ainsi, comme illustré dans les Figures. 1, 5, 6 et 8, l'utilisateur peut, via un masque ou un pinceau, ajouter localement les textures choisies, et ajuster leur intensité.

TTSR [17], avec plus de 9M de paramètres, est utilisé comme référence pour la Ref-SR. Remarquons (Fig. 5) que TTSR ne permet pas d'appliquer la texture uniformément sur la zone désirée, contrairement à notre approche.

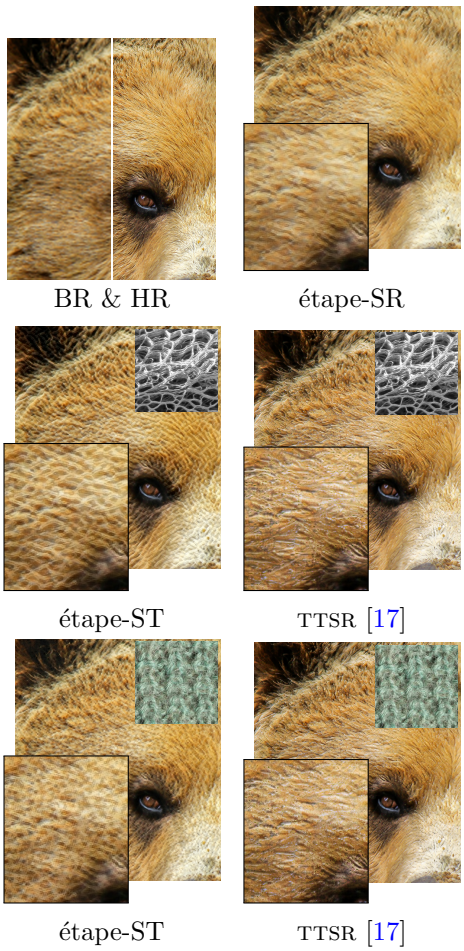


FIGURE 5 – *Comparaison de résultats pour l'Etape-ST 4× sur 1 image de test de* Div2K *[25] avec* TTSR *et ce pour différents styles Y_i. Plus de comparaisons sont présentées sur le site [1].*

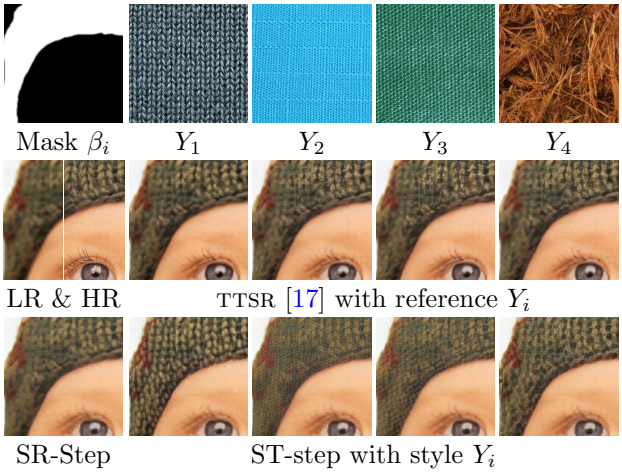


FIGURE 6 – Comparaison de résultats pour l'Etape-ST 4× sur l'image `baby` (Set5 dataset) avec TTSR et ce pour différents styles Y_i.

Stylisation à partir d'un autre modèle de SR Les branches de style peuvent être entraînées et utilisées à partir de n'importe quel réseau de SR. Nous avons ainsi entraîné nos branches de style comme décrit dans § 2.2 avec l'architecture représentée dans la Figure 7. Seuls les quelques 35K paramètres des branches de style sont entraînables.

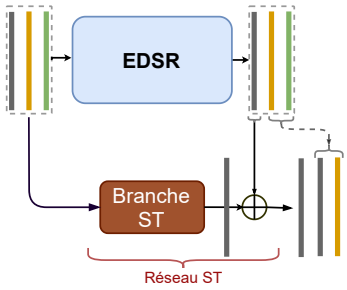


FIGURE 7 – *Schéma de l'architecture du Réseau ST entrainé à partir d'un modèle quelconque de SR (ici,* EDSR *[8]).*

La Figure 8 présente des résultats issus de branches de style entraînées à partir du modèle EDSR [8] et appliquées à une image de *Div2K* [25]. Disposant de beaucoup de paramètres, EDSR seul donne déjà des résultats satisfaisants (*ligne 2*). Il est ainsi intéressant de constater que nos branches de style, très légères, permettent d'améliorer des modèles très volumineux comme EDSR. De plus, bien que ce dernier ne dispose pas des caractéristiques des styles présentés (ou ne les a pas sélectionné automatiquement) dans le processus de reconstruction, les textures imposées par nos branches restent visuellement plausibles.

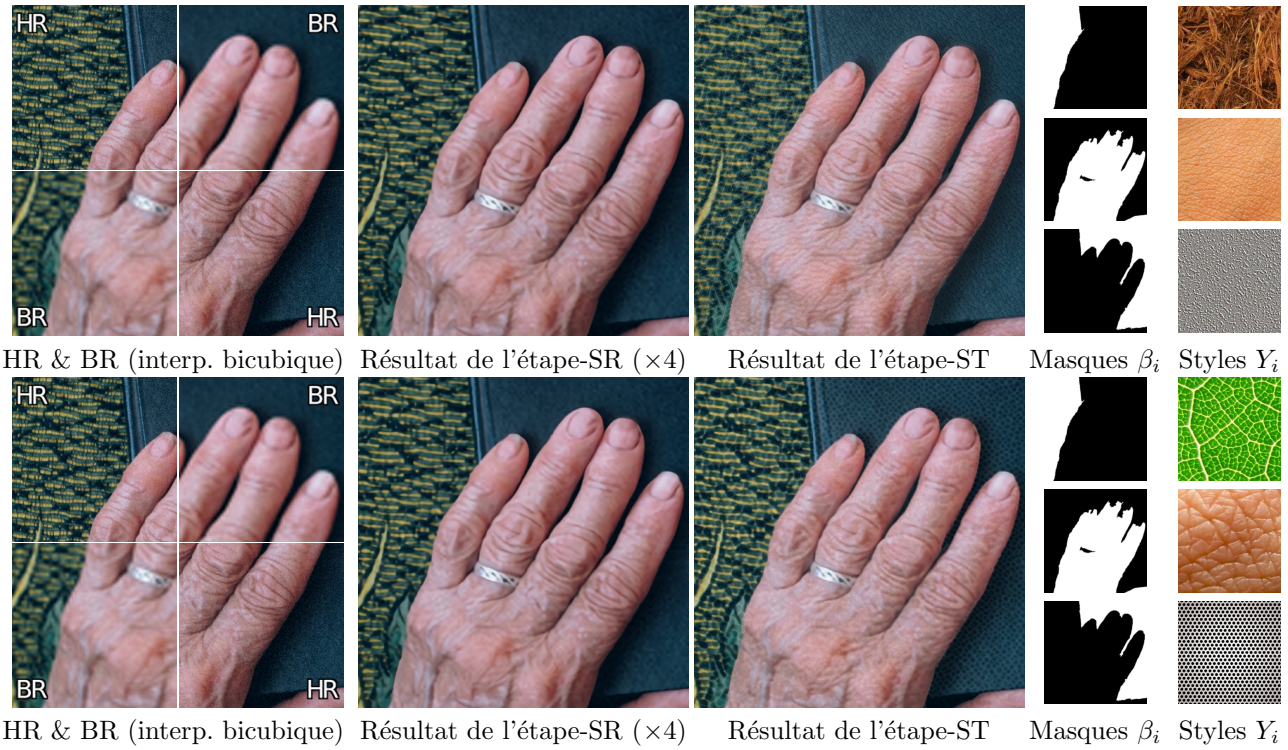


FIGURE 8 – *Différents résultats pour l'étape-ST* 4× *sur 1 image de test de* Div2K *[25] à partir du modèle* EDSR *[8] et ce pour différents styles* Y_i.

4 Conclusion

Nous avons présenté une architecture de réseau légère pour la super résolution stylisée, composée de branches de super-résolution parallèles et indépendantes, chacune spécialisée dans une bande fréquentielle. Les branches de style, entraînées indépendamment et après le réseau de super-résolution, permettent la génération de détails et de textures. Avec peu de paramètres, notre méthode offre une nouvelle manière d'envisager la super résolution, proposant un contrôle local à l'utilisateur, contrairement aux réseaux très profonds et automatisés. L'architecture multi-échelle rend la méthode facile à entraîner, à interpréter et à enrichir via l'ajout de branches. Entre autres, d'autres branches légères, locales et modulables sont en cours d'études, concernant d'autres tâches en traitement d'images.

Références

[1] https://durand192.users.greyc.fr/SMS-SR/.

[2] "Shallow multi-scale network for stylized super-resolution," .

[3] C. Dong, C. Loy, K. He, and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," *IEEE TPAMI*, vol. 38, 2016.

[4] 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 *Proceedings of CVPR*, Las Vegas, NV, USA, June 2016, pp. 1874–1883, IEEE.

[5] C. Dong, C. C. Loy, and X. Tang, "Accelerating the Super-Resolution Convolutional Neural Network," in *Proceedings of ECCV*, Aug. 2016.

[6] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate Image Super-Resolution Using Very Deep Convolutional Networks," in *Proceedings of CVPR*, 2016, pp. 1646–1654.

[7] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-Recursive Convolutional Network for Image Super-Resolution," in *Proceedings of CVPR*, Las Vegas, NV, USA, June 2016, pp. 1637–1645, IEEE.

[8] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," in *Proceedings of CVPR Workshops*, July 2017.

[9] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *NIPS*, vol. 27, pp. 2672–2680, 2014.

[11] L. Gatys, A. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," *Advances in neural information processing systems*, vol. 28, pp. 262–270, 2015.

[12] M. Sajjadi, B. Scholkopf, and M. Hirsch, "Enhancenet : Single image super-resolution through automated texture synthesis," in *Proceedings of ICCV*, 2017, pp. 4491–4500.

[13] L. Gatys, A. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv :1508.06576*, 2015.

[14] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv :1409.1556 [cs]*, Apr. 2015, arXiv : 1409.1556.

[15] H. Zheng, M. Ji, L. Han, Z. Xu, H. Wang, Y. Liu, and Lu Fang, "Learning Cross-scale Correspondence and Patch-based Synthesis for Reference-based Super-Resolution," in *Proceedings of BMCV*, London, UK, 2017, p. 138.

[16] Z. Zhang, Z. Wang, Z. Lin, and H. Qi, "Image super-resolution by neural texture transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[17] F Yang, H. Yang, J. Fu, H. Lu, and B. Guo, "Learning Texture Transformer Network for Image Super-Resolution," in *Proceedings of CVPR*, Seattle, WA, USA, June 2020, pp. 5790–5799, IEEE.

[18] R. Zhang, P. Isola, and A. A. Efros, "Colorful Image Colorization," in *Proceedings of ECCV*, 2016, vol. 9907, pp. 649–666.

[19] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky, "Texture networks : Feed-forward synthesis of textures and stylized images.," in *ICML*, 2016, vol. 1, p. 4.

[20] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[22] Y. HaCohen, R. Fattal, and D. Lischinski, "Image upsampling via texture hallucination," in *2010 IEEE International Conference on Computational Photography (ICCP)*. IEEE, 2010, pp. 1–8.

[23] J. Johnson, A. Alahi, and Li Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," in *Proceedings of ECCV*, vol. 9906, pp. 694–711. 2016.

[24] Augustus Odena, Vincent Dumoulin, and Chris Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016.

[25] R. Timofte, S. Gu, J. Wu, L. Van Gool, L. Zhang, M.-H. Yang, M. Haris, et al., "Ntire 2018 challenge on single image super-resolution : Methods and results," in *Proceedings of CVPR Workshops*, June 2018.

[26] Y. Zhang, Y. Tian, Yu Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proceedings of CVPR*, 2018, pp. 2472–2481.