Rapport de Stage

Super-Résolution d'Images via Réseau Auto-encodeur

Grégoire Doat

Sous la tutelle de M. Yann Traonmilin

Mai – Juin 2024

Introduction

Dans ce rapport de stage, on s'intéresse au problème de super-résolution. C'est-à-dire la reconstruction d'image $\boldsymbol{x}$, disons de taille (n, m), à partir d'une version sous-échantillonnée $\boldsymbol{y}$ de taille (p, q) et se pose ainsi[1] :

Étant donnée une image $\boldsymbol{x_0} \in \mathbb{R}^{n\times m}$ et sa version sous-échantillonnée $\boldsymbol{y_0} \in \mathbb{R}^{p\times q}$ on cherche à minimiser la fonctionnelle :

$$F : \begin{array}{ccc} \mathbb{R}^{n\times m} & \longrightarrow & \mathbb{R} \\ \boldsymbol{x} & \longmapsto & \frac{1}{2}\left\|A\boldsymbol{x} - \boldsymbol{y_0}\right\|_2^2 \end{array} \tag{1}$$

que l'on suppose convexe et où A représente le sous-échantillonnage, aussi appelé opérateur de mesure puisque l'on peut l'interpréter comme une mesure incomplètement d'une "véritable" image.

Par nature, l'application A n'est pas injective ce qui fait rentrer le problème de super-résolution dans la classe des problèmes inverses mal posés.

Une façon de retrouver (au moins partiellement) l'injectivité de A consiste à introduire une paramétrisation ϕ de l'espace des images $\boldsymbol{x}$ telle que $\phi(\theta) = \boldsymbol{x}$ et à chercher les bons paramètres θ plutôt que $\boldsymbol{x}$. L'intérêt étant que si le nombre de paramètres est suffisamment petit par rapport à la taille de l'espace des mesures, on peut espérer que le problème soit mieux posé en cherchant à inverser $A\phi$.
Le contre coût de cette méthode est qu'elle à toutes les chances d'impacter la convexité du problème. Se pose donc la question du comportement de la fonctionnelle $F \circ \phi$ au voisinage de ses minimums.
C'est précisément ce qu'ont étudié Y. Traonmilin, J.-F. Aujol et A. Leclaire dans leur article *The basins of attraction of the global minimizers of non-convex inverse problems with low-dimensional models in infinite dimension* de 2022 [2]. Ils donnent une caractérisation d'abord générale des bassins d'attractions, c'est-à-dire l'ensemble des paramètres θ partant desquelles la descente de gradient convergerait vers un minimum globale de $F \circ \phi$. Puis appliquent leur résultat à des problèmes inverses classiques : reconstruction de matrice de bas rang, Off-the-grid sparse spike recovery, estimation de mélange de gaussiennes.

Le point commun de ces trois problèmes est que l'espace des objets à reconstruire possède une paramétrisation connue et exploitable, ce qui n'est pas le cas de l'ensemble des images. En toute généralité, l'ensemble Σ des images de taille (n, m) est $\mathbb{R}^{n\times m}$, mais en le restreignant à un type d'images (ne serait-ce qu'en excluant les images de bruit blanc), sa structure est beaucoup moins claire et ses paramétrisations encore moins. C'est là qu'intervient le machine learning.

[1] ce n'est pas nécessairement la seule façon de poser le problème cela dit

Le but du stage est d'étudier dans quelle mesure les travaux de Traonmilin *et al.* s'appliquent au problème de super-résolution en prenant comme paramétrisation le décodeur d'un auto-encodeur entraîné sur un jeu de données (dans notre cas MNIST).
À côté de cela, P. Peng, S. Jalali et X. Yuan ont étudié la descente de gradient projeté pour résoudre le même problème de super-résolution en prenant comme projection un auto-encodeur appris. On tentera, dans un second temps, de reproduire leurs résultats pour les comparer à ceux de la descente de gradient depuis l'espace latent.

I. Cadre théorique

1.1. Formalisation du problème

Soit A un opérateur de mesure linéaire à valeur de $\mathbb{R}^{n\times m}$ dans $\mathbb{R}^{p\times q}$. Il se compose d'un sous-échantillonnage (projection) S et pour éviter les problèmes d'aliasing, d'un éventuel filtre passe-bas $C_{\boldsymbol{h}}$. $C_{\boldsymbol{h}}$ est donc une convolution par un filtre que l'on va noter $\boldsymbol{h}$ (d'où la notation) de sorte que A s'écrive :

$$\forall \boldsymbol{x} \in \mathbb{R}^{n\times m}, \qquad A\boldsymbol{x} = S(\boldsymbol{h} * \boldsymbol{x}) = SC_{\boldsymbol{h}}\boldsymbol{x}$$

Ci-dessous deux représentations de S pour $(n, m) = (3, 4)$ et $(p, q) = (2, 2)$: sur la figure *2*, S ne garde d'un pixel sur 4 de l'image $\boldsymbol{x}$, ce qui correspond à appliquer la matrice de la figure *1* au vecteur $\boldsymbol{x}$ aplati.

Pour effectuer la convolution avec $\boldsymbol{h}$, l'algorithme passe l'espace de fréquences, le changeant en simple produit et c'est dans cet espace que sont ajusté les paramètres des filtres :
Le filtre gaussien à pour paramètre l'écart-type σ de $\hat{\boldsymbol{h}}$. Pour le filtre porte, le paramètre a ajuste la taille de la fenêtre $[-a, a] \times [-a, a]$ dans l'espace des fréquences. La figure **??** en donne quelques exemples, plus de détails en annexes B.

Problématique — On considère $\boldsymbol{x_0} \in \mathbb{R}^{n\times m}$ une image que l'on cherche à reconstruire et $\boldsymbol{y_0} := A(\boldsymbol{x_0})$ sa version sous-échantillonnée. Une bonne reconstruction de $\boldsymbol{x_0}$ étant un minimiseur :

$$\boldsymbol{x^*} \in \underset{\boldsymbol{x}\in\mathbb{R}^{n\times m}}{\operatorname{argmin}} F(\boldsymbol{x}) \tag{2}$$

$$\left(\begin{array}{cc|cc} 1&0&0&0\\ 0&0&1&0 \end{array}\right.$$
$$\left.\begin{array}{c|cc} \multicolumn{3}{c}{\mathbb{O}} \\ \hline 0&0&1&0\\ 0&0&0&0 \end{array}\right.$$
$$\left.\begin{array}{cc|cc} \multicolumn{2}{c|}{\mathbb{O}} & \multicolumn{2}{c}{} \\ \hline 0&0&0&0\\ 0&0&0&1 \end{array}\right)$$

fig. 1 — Opérateur S, point de vue matriciel

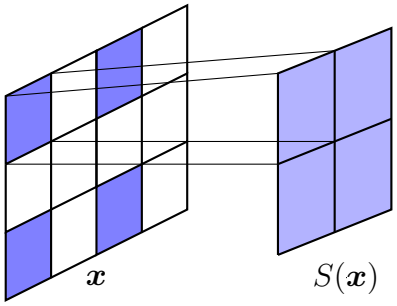

fig. 2 — Opérateur S, point de vue image

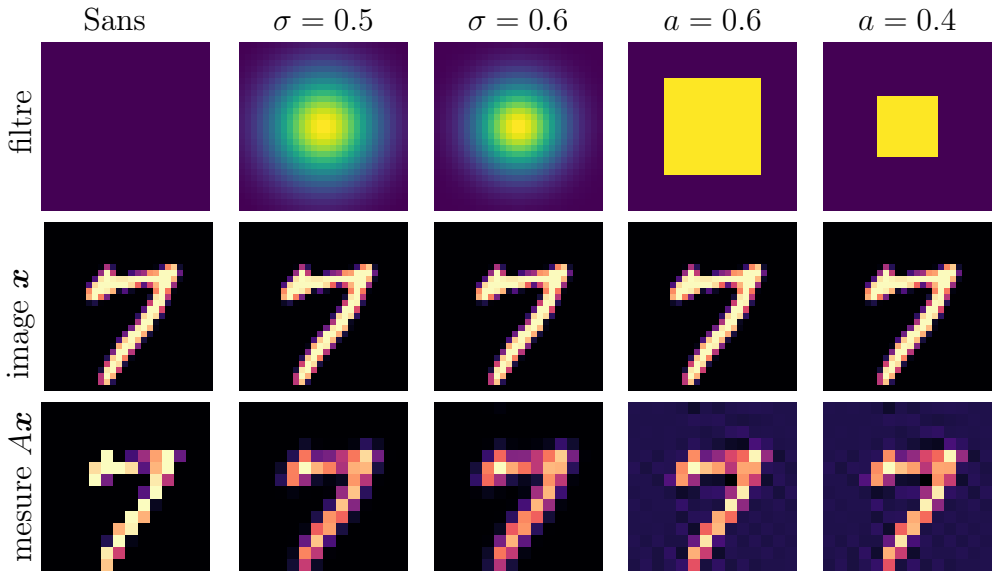

fig. 3 — Quelques exemples d'applications de A à une même image en fonction des filtres.

Si pour le bien de l'étude $\boldsymbol{x_0}$ sera connu, il va de soit que le problème ne nécessite pas de le connaître *a priori*.
Aussi, la linéarité de A assure la convexité de F mais, bien évidemment, son caractère hautement non injectif (surtout pour $pq \lll nm$) fait qu'il existe tout un sous-espace affine de minimum globaux. D'où l'intérêt des deux méthodes étudiées pour "choisir" ce minimum.

Les auto-encodeurs utilisés pour paramétrer l'espace des images sont des réseaux MLP que l'on notera indifféremment f et dont f_E et f_D sont les parties encodage et décodage :

$$f = f_E \circ f_D$$

Conformément à la figure 4, ils se composent tous d'une couche cachée de taille 1 500 pour les parties encodeurs et décodeurs et la seule chose qui les différencie est la taille de leur espace latent qui varient entre 100 et 800. Leur entraînement s'est fait sur le jeu de données MNIST et c'est donc uniquement sur ces images que seront présentés les résultats. Plus de détails sur les performances des auto-encodeurs en annexe A.

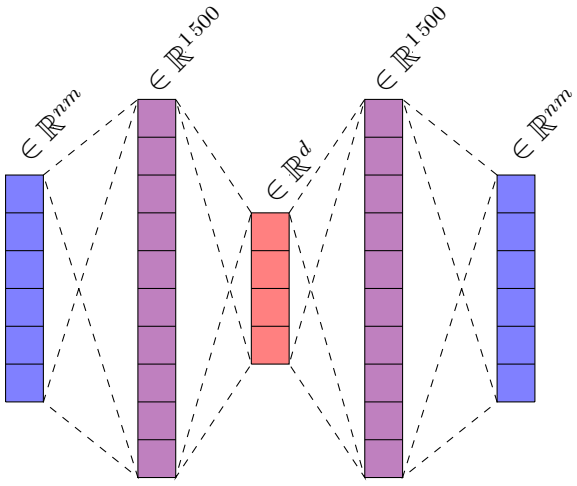

fig. 4 — Schéma des auto-encodeurs f avec $d = 100,\ 200,\ 400,\ 800$

1.2. Cadre théorique de l'article

Comme expliqué plus tôt, tout l'objet de l'article [2] est d'étudier dans quelle mesure il est possible de faire une descente de gradient sur une paramétrisation (non-convexe) d'un modèle plutôt que sur le modèle lui-même.
Pour cela, on note $\Sigma \subset \mathbb{R}^{n\times m}$ le modèle de basse-dimension et $\phi : \mathbb{R}^d \longrightarrow \mathbb{R}^{n\times m}$ une paramétrisation de ce dernier avec :

$$\Sigma \subset \phi\big(\mathbb{R}^d\big) \qquad\qquad \Theta := \phi^{-1}(\Sigma)$$

La paramétrisation ϕ, doit vérifier certaines hypothèses de régularité pour pouvoir appliquer les résultats de l'article. Dans notre cas, $\phi = f_D$ dont les fonctions d'activations sont toutes des sigmoïdes. Elle est donc C^∞ et pour cette raison on ne s'attardera pas outre mesure sur les questions de régularité de ϕ.

À noter qu'on ne suppose ni que $\phi(\mathbb{R}^d) = \Sigma$ ni que $\Theta = \mathbb{R}^d$ et ce ne sera pas le cas pour nous. Aussi, le formalisme proposé par Traonmilin et al. est légèrement différent de celui donné en section précédente. Il y est considéré un espace de mesure $\mathscr{D}$ et les signaux (dans notre cas les images) sont pris dans $\mathscr{D}^*$, de sorte que la mesure par $\alpha \in \mathscr{D}$ de $x \in \mathscr{D}^*$ soit représentéé par le crochet de dualité $\langle \alpha, x\rangle$. Cela à son intérêt surtout en dimension infinie, mais autrement, cette écriture est tout à fait équivalente à celle donnée en section 1.1.

Le problème est donc de trouver les paramètres θ^* minimisant $g := A\phi$ sur Θ, ou dans notre cas :

$$\theta^* \in \underset{\theta\in\Theta}{\operatorname{argmin}} \frac{1}{2}\big\|Af_D(\theta) - \boldsymbol{y_0}\big\|_2^2$$

Minimum que l'on cherche à obtenir par descente de gradient. On considérera donc dans toute la suite, (θ_n) la descente de gradient à pas fixe, $\forall \theta_0 \in \mathbb{R}^d$:

$$\forall n \in \mathbb{N}^*, \qquad \theta_{n+1} = \theta_n - \tau\nabla g(\theta_n)$$

Sachant que ϕ est n'est pas convexe, on introduit la définition suivante :

θ_0 : initialisation
x : image à reconstruire
τ : pas de descente
N : nombre d'itération
$\hat{x} \longleftarrow f(x)$

for $n \in [\![1, N-1]\!]$ **do**
  $\theta_{n+1} \longleftarrow \theta_n - \tau\nabla g(\theta_u n)$
  $x_{n+1} \longleftarrow f_D(\theta_{n+1})$
end for
return $(x_n)_{0\leqslant n\leqslant N}$

fig. 5 — Algorithme de descente depuis l'espace des paramètres

Définition 1 — Une partie $\Lambda \subset \Theta$ de $\mathbb{R}^d$ sera un *bassin d'attraction* de minimum global $\theta^* \in \Lambda$ si en tout point $\theta_0 \in \Lambda$, alors existe un pas de descente τ_0 tel que :

$$\forall \tau \in]0, \tau_0], \qquad \lim_{n\longrightarrow+\infty} g\big(\theta_n\big) = g(\theta^*)$$

Avec cette définition, (θ_n) peut tout à fait converger vers un autre minimum que θ^*. Pour en tenir compte en même temps que la potentielle non-injectivité de ϕ, ils introduisent les définitions suivantes :

DÉFINITION 2 — On note $d(\theta, \theta^*)$ la distance de θ au plus proche paramètre de même image de θ^* et $p(\theta, \theta^*)$ la projection (potentiellement multivaluée) de θ sur ce même paramètre, e.i. sur $\phi^{-1} \circ \phi(\{\theta^*\})$: $\forall \theta \in \mathbb{R}^d$:

$$d(\theta, \theta^*) := \min_{\substack{\tilde{\theta} \in \Theta \\ \phi(\tilde{\theta}) = \phi(\theta^*)}} \left\| \tilde{\theta} - \theta \right\|_2 \qquad p(\theta, \theta^*) := \operatorname*{argmin}_{\substack{\tilde{\theta} \in \Theta \\ \phi(\tilde{\theta}) = \phi(\theta^*)}} \left\| \tilde{\theta} - \theta \right\|_2 \subset \mathbb{R}^d$$

Avec, sont définis les bassins $\Lambda_\beta(\theta^*)$ de la forme :

$$\forall \beta \in \mathbb{R}^{+*}, \qquad \Lambda_\beta(\theta^*) := \left\{ \theta \in \mathbb{R}^d \mid d(\theta, \theta^*) < \beta \right\}$$

Contrairement aux exemples traités dans l'article, la paramétrisation par l'espace latent ne donne pas de formule vraiment exploitable pour $\phi = f_D$. En revanche, f_D faisant parti d'un auto-encodeur, on peut supposer que f_E est la réciproque de f_D au moins sur Θ :

$$f_D|_\Theta{}^{-1} = f_E|_\Sigma \qquad \text{avec} \qquad \Theta = f_E(\Sigma)$$

En particulier, cela permet d'avoir l'injectivité de f_D sur Θ. Ce dont il découle que $d(\theta, \theta^*) = \|\theta - \theta^*\|_2$ est une distance (ce qui n'est pas le cas en général), que $\Lambda_\beta(\theta^*)$ une boule ouverte et que $p(\theta, \theta^*) = \theta^*$.

Cette hypothèse est quand-même très forte, elle sera détaillé dans la section 2.4.

Enfin, dans toute la suite Σ sera supposé être un cône, c'est-à-dire que pour tout $\lambda > 0$, $\lambda\Sigma = \Sigma$ et sont introduits les deux ensembles suivants :

DÉFINITION 3 — On appelle *ensemble des sécantes de* Σ l'ensemble :

$$\mathscr{S}(\Sigma) := \Sigma - \Sigma = \left\{ x - y \in \mathbb{R}^{n \times m} \mid x, y \in \Sigma \right\}$$

En notant $\partial_u \phi(\theta)$ la dérivée directionnelle de ϕ en θ et dans la direction u, on définit l'*ensemble généralisé des sécantes de* Σ, l'ensemble noté :

$$\overline{\mathscr{S}(\Sigma)} := \mathscr{S}(\Sigma) \cup \left\{ \partial_u \phi(\theta) \mid \forall (\theta, u) \in \Theta \times \mathbb{R}^d \right\}$$

Pour nous, $\phi = f_D$ est C^∞, donc la dérivée directionnelle $\partial_u \phi(u)$ est équivalente à la différentielle de ϕ au point θ valué en u, $D_\theta \phi(u)$.

L'auto-encodeur f n'a été entraîné que sur des images dont les pixels prennent valeur dans $[0, 1]$ et il n'est capable de reconstruire que ce type d'image. Et ce, pour la simple et

bonne raison que ses fonctions d'activation sont des sigmoïdes y compris celles de sortie. Il est donc strictement impossible qu'une image $f_D(\theta)$ prenne des valeurs au-delà de 1, ce qui va à l'encontre de l'hypothèse que Σ soit un cône.
Si ce choix de fonction d'activation a été fait c'était à l'origine pour reprendre les codes de l'article [1] sur la descente projetée (voir section **??**), mais en toute vraisemblance utiliser des fonctions ReLu ou autre ne devrait pas trop changer les résultats obtenus. La perte de régularité entraînée ne devrait pas non plus être un problème en pratique et pourrait rendre l'hypothèse un peu plus raisonnable.

1.3. Les résultats de Traonmilin *et al.*

Le théorème de Traonmilin *et al.* , dans le cas sans bruit, s'énonce ainsi :

THÉORÈME 1 (sans bruit) — Soit $\theta^* \in \Theta$ un minimiseur global de g. Si on a les hypothèses suivantes :

- l'opérateur de mesure A est continue et vérifie la *propriété d'isométrie restreinte* (RIP) de constante $\gamma \in]0,1]$:

$$\forall v \in \overline{\mathscr{S}(\Sigma)}, \qquad (1-\gamma)\|v\|^2 \leqslant \|Av\|^2 \leqslant (1+\gamma)\|v\|^2 \tag{3}$$

- Si $\Lambda_{2\beta}(\theta^*)$, tel que défini en 2, est inclus dans Θ, ou autrement dit :

$$\phi\big(\Lambda_{2\beta}\big) \subset \Sigma \tag{4}$$

- Si la *technical assumption* 1 (donnée plus bas) est vérifiée sur $\Lambda_{2\beta}$ avec la constante $C > 0$ et qu'il existe $\tilde{\theta} \in p(\theta, \theta^*)$ tel qu'on ait l'inégalité :

$$\forall z \in [\theta, \tilde{\theta}], \qquad \frac{(1-\gamma)\big\|\partial_{\tilde{\theta}-\theta}\phi(z)\big\|_2^{\,2}}{\sqrt{1+\gamma}\big\|A\partial^2_{\tilde{\theta}-\theta}\phi(z)\big\|_2} \geqslant C\beta \tag{5}$$

Si A et ϕ vérifient de plus la *framework assumption* 2, alors l'ensemble $\Lambda_\beta(\theta^*)$ est un bassin d'attraction.

COROLLAIRE 1.1 — En particulier, s'il existe une constante $\beta_1 > 0$ vérifiant la *technical assumption* 1 et telle que $p(\,\cdot\,, \theta^*)$ soit mono-valuée sur $\Lambda_{\beta_1}(\theta^*)$:

$$\forall \theta \in \Lambda_{\beta_1}(\theta^*), \qquad p(\theta, \theta^*) = \big\{\tilde{\theta}\big\}$$

Alors $\Lambda_{\min(\beta_1,\beta_2)}(\theta^*)$ est un bassin d'attraction, où β_2 est donné par la formule :

$$\beta_2 := \frac{1-\gamma}{C\sqrt{1+\gamma}} \inf_{\theta \in \Lambda_{\beta_1}(\theta^*)} \inf_{z \in [\theta,\tilde{\theta}]} \frac{\big\|\partial_{\tilde{\theta}-\theta}\phi(z)\big\|_2^{\,2}}{\big\|A\partial^2_{\tilde{\theta}-\theta}\phi(z)\big\|_2}$$

Ce théorème (et son corollaire) donne une estimation de la marge que l'on a autour de θ^* pour initialiser la descente de gradient depuis l'espace des paramètres de sorte à s'assurer sa convergence. En particulier, l'inégalité 5 donne, sachant γ et C, à quel point le "rayon" du bassin β peut être grand. Estimation qui est donc valable sachant les deux hypothèses supplémentaires :

HYPOTHÈSE 1 (Technical assumption) — La *technical assumption* est vérifiée sur $\Lambda_{2\beta}(\theta^*)$ avec la constante $C > 0$ si elle donne un contrôle de d sur $\|\cdot\|_2$:

$$\forall \theta \in \Lambda_{2\beta}(\theta^*), \qquad \left\|\theta - \theta^*\right\|_2 \leqslant C d\big(\theta, \theta^*\big)$$

et si les deux premières dérivées (directionnelles) de $A\phi$ sont uniformément bornées respectivement sur $\phi^{-1} \circ \phi(\{\theta^*\})$ et $\Lambda_{2\beta}$:

$$\sup_{\theta \in \phi^{-1}\circ\phi(\{\theta^*\})} \sup_{\|u\|_2=1} \left\|A\partial_u\phi(\theta)\right\|_2 < +\infty \qquad \sup_{\theta \in \Lambda_{2\beta}(\theta^*)} \sup_{\|u\|_2,\|v\|_2=1} \left\|A\partial_u\partial_v\phi(\theta)\right\|_2 < +\infty$$

Comme dit plus haut, supposer $\phi = f_D$ inversible sur Θ impose $d\big(\theta, \theta^*\big) = \left\|\theta - \theta^*\right\|_2$. Ainsi la première inégalité est vérifiée pour tout $C \geqslant 1$ et les majorations uniformes ne sont pas un problème non plus puisque f_D est C^∞. Sachant que c'est l'inégalité 5 qui nous intéresse, on va plutôt avoir tendance à prendre $C = 1$.

HYPOTHÈSE 2 (Framework assumption) — La paramétrisation ϕ doit être deux fois (Gateaux) différentiable et il faut que pour toute suite $|h_n| \xrightarrow[n\longrightarrow+\infty]{} 0$, $\left\|\frac{\phi(\theta+|h_n|v)-\phi(\theta)}{|h_n|}\right\|_2$ converge indépendant de $(|h_n|)$ et ceux, pour tout (θ, v) tel que $\partial_v\phi(\theta) \in \overline{\mathscr{T}(\Sigma)}$.

Dans l'article [2], la mesure de proximité des images est faite à travers une norme $\|\cdot\|_{\mathscr{H}}$ associé un espace hilbertien $\mathscr{H}$ contenant Σ. Cette hypothèse permet alors de s'assurer que cette norme s'étende à $\overline{\mathscr{T}(\Sigma)}$. Mais encore, une fois, pour nous $\mathscr{H} = \mathbb{R}^{n\times m}$ et il n'y a pas de risque à ce sujet. Aussi, le fait que $\phi = f_E$ soit différentiable rend immédiat la convergence de la suite $\left\|\frac{\phi(\theta+|h_n|v)-\phi(\theta)}{|h_n|}\right\|_2$ indépendamment de $\big(|h_n|\big)_n$.

Revenons maintenant aux autres hypothèses du théorème.
D'abord, dans la RIP (3), c'est surtout l'inégalité de gauche qui est importante puisqu'elle garantit que A soit inversible sur $\overline{\mathscr{T}(\Sigma)}$.
Comme Σ n'est pas proprement définie, le mieux qu'on puisse faire c'est de vérifier si c'est au moins le cas pour tous les $\boldsymbol{x} - \boldsymbol{y}$ avec $\boldsymbol{x}$, $\boldsymbol{y}$ du jeu de données (MNIST donc). Pour se faire, on les équivalences :

$$\begin{aligned}(1-\gamma)\|v\|^2 \leqslant \|Av\|^2 \leqslant (1+\gamma)\|v\|^2 \iff \frac{\|Av\|^2}{\|v\|^2} - 1 \leqslant \gamma \\ \iff \left|\frac{\|Av\|^2}{\|v\|^2} - 1\right| \leqslant \gamma\end{aligned}$$

En calculant le rapport $\left|\frac{\|Av\|^2}{\|v\|^2} - 1\right|$, on peut donc estimer une borne pour γ. Seulement, avec les moyens disponibles et notre code, passer en revue les $60\,000 \times (60\,000+1)$ couples d'images du set de test demanderait quelque 11h de calcul. On se contentera donc, faute de mieux, de faire les calculs sur Σ, ce qui nous donnes les histogrammes *6* et *7* :

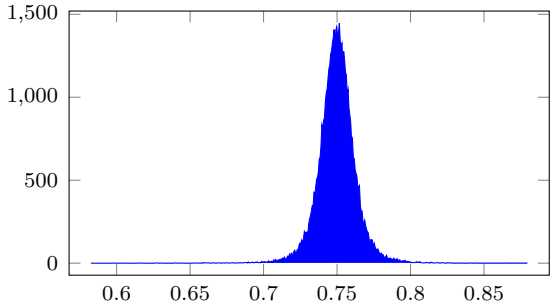

fig. 6 — Histogramme des valeurs de $\left|\|Av\|_2{}^2/\|v\|_2{}^2 - 1\right|$ sans filtre passe-bas sur le set de test

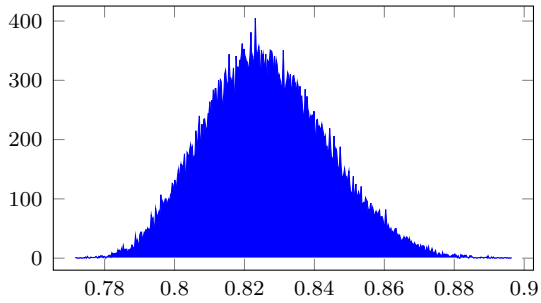

fig. 7 — Histogramme des valeurs de $\left|\|Av\|_2{}^2/\|v\|_2{}^2 - 1\right|$ avec filtre gaussien ($\sigma = 0,6$) sur le set de test

On y voit que dans les deux cas, on peut espérer que γ soit au mieux aux alentours des 0.9. Ce serait suffisant pour vérifier la RIP mais sachant qu'on veut le restreindre le moins possible β dans l'inégalité (5), cela reste contraignant. D'autant plus qu'avoir $C = 1$ n'aide pas beaucoup.

Enfin, l'hypothèse d'inclusion $\phi\big(\Lambda_{2\beta}(\theta^*)\big) \subset \Sigma$, et plus généralement à chaque occurrence de $\Lambda_{2\beta}$, le facteur 2 est là pour s'assurer une marge de manoeuvre dans les démonstrations. Ils peuvent sans trop de difficulté être remplacés par des bassins de la forme $\Lambda_{\beta+\varepsilon}(\theta^*)$ et pour cette raison, l'hypothèse (4) sera supposée vraie pour ε suffisamment petit.

Il reste une dernière grosse inconnue, à savoir l'infimum : $\inf_{\theta\in\Lambda_{\beta_1}(\theta^*)} \inf_{z\in[\theta,\tilde{\theta}]} \frac{\left\|\partial_{\tilde{\theta}-\theta}\phi(z)\right\|_2{}^2}{\left\|A\partial^2_{\tilde{\theta}-\theta}\phi(z)\right\|_2}$
La seule chose vraiment importante à en dire est que si le numérateur s'annule, alors la descente de gradient ne pourra pas converger à coût sûr. Si cela arrive, c'est l'injectivité de ϕ qui est perdue et on peut lier cela avec la taille d de l'espace latent : Plus la différence $np - d$ est grande, plus on risque de perdre l'injectivité de f_D sur Θ. C'est quelque chose qui sera étudié dans la section 2.3.

II. Descente de gradient depuis l'espace latent

À partir de maintenant, les vecteurs de l'espace latent seront noté $\boldsymbol{u}$ et en particulier $(\boldsymbol{u_n})$ sera la suite obtenue par descente de gradient d'initialisation $\boldsymbol{u_0}$. On pose aussi $(n, m) = (28, 28)$, $(p, q) = (14, 14)$ et $d = 100$. Tout les résultats présentés sont disponibles sur le GitHub et reproductibles via les codes `main_code.py` et `LGD.py`.

2.1. Premier résultats, différentes initialisations

Si la partie théorique n'est pas très prometteuse, en pratique la descente depuis l'espace latent (LGD) est pourtant très efficace et surtout robuste. Pour estimer les tailles des bassins d'attractions, la LGD est faite sur la même image cible avec différentes initialisations.

La descente se fait à pas fixe[2] et après avoir ajusté les pas en fonction de A, on obtient les résultats des figures *8* et *9* ci-dessous.
Sur chaque figure, à gauche se trouve l'image que l'on veut reconstruire (Target) et à côté différentes initialisations décodées $f_D(\boldsymbol{u_0})$, avec en dessous le résultat de la LGD. Le graphique de droite donne l'évolution du "Loss", *e.i.* la fonctionnelle g et à droite l'évolution du PSNR entre l'image $f_E(\boldsymbol{u_n})$ à l'itération et l'image cible.

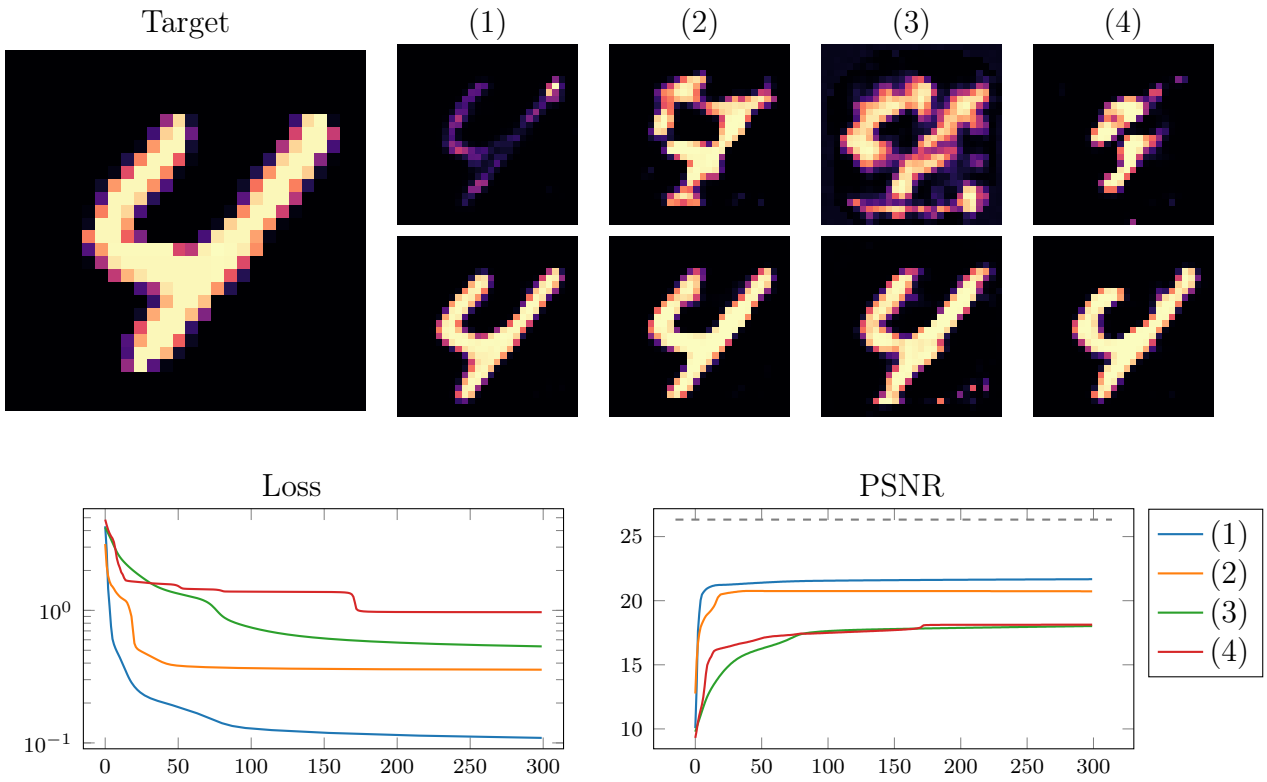

fig. 8 — En première lignes différentes initialisations avec en dessous le résultat de la LGD après 300 itérations – sans filtre passe-bas

[2] *Il y a aussi un pas adaptatif par backtracking dans le code mais il ne marche pas bien, plus de détails dans le* `main_code.py`

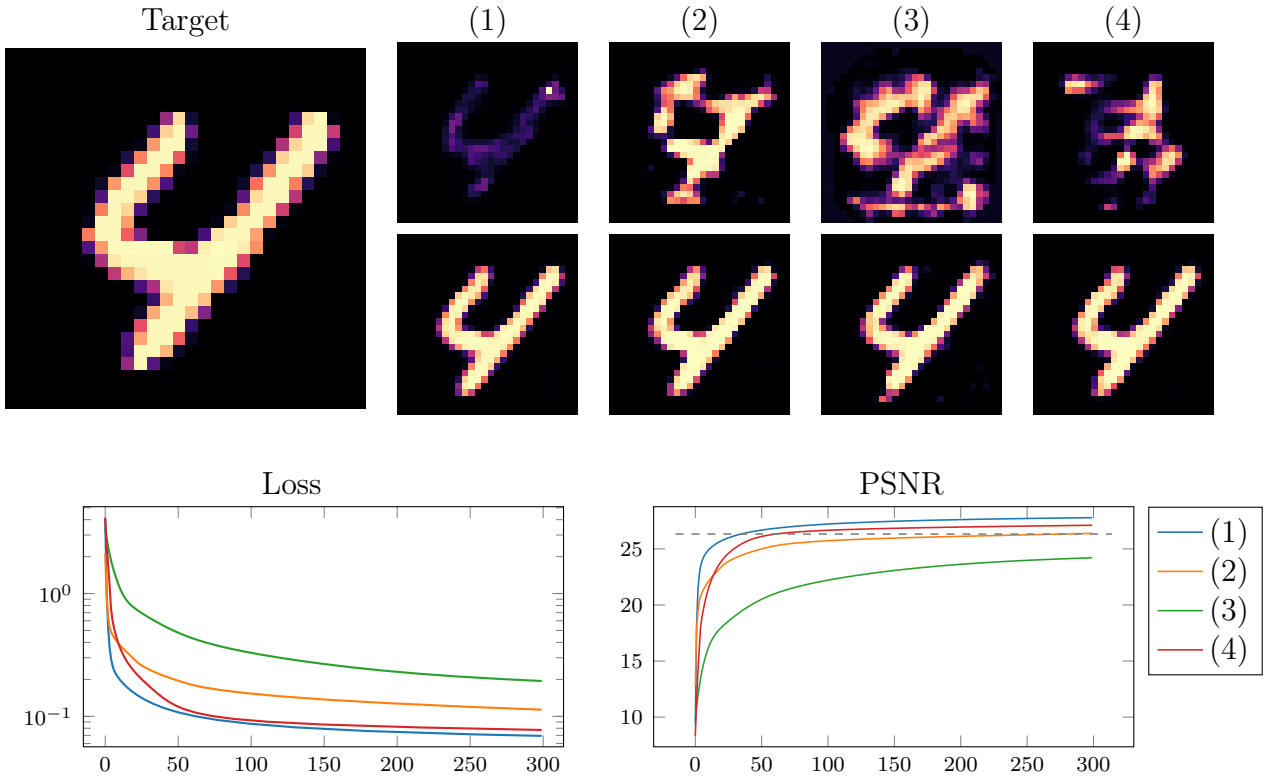

fig. 9 — En premières lignes différentes initialisations avec en dessous le résultat de la LGD après 300 itérations – avec passe-bas gaussien ($\sigma = 0.5$)

Il va de soi que le calcul de PSNR est à but purement indicatif et n'intervient pas dans la LGD elle-même. Les ligne en pointillées représentent $\mathrm{PSNR}\big(\boldsymbol{x_0}, f(\boldsymbol{x_0})\big)$, le PSNR entre la cible $\boldsymbol{x_0}$ est sa version auto-encodée. Dans la première figure *8* il n'y pas de filtre passe-bas (*e.i.* $A = S$) et dans la seconde *9*, $C_{\boldsymbol{h}}$ est un filtre gaussien de paramètre $\sigma = 0.6$ (voir section 1.1. pour un rappel sur σ).
Les différentes initialisations sont les suivantes :

(1) $\boldsymbol{u_0} = f_E({}^tA\boldsymbol{y_0})$, la backprojection encodée de $\boldsymbol{y_0}$

(2) $\boldsymbol{u_0} = f_E(\boldsymbol{x_0}) + e$, l'image cible encodée puis bruité (bruit uniforme sur $]-0.5, 0.5[$)

(3) idem, avec un bruit gaussien d'écart-type 0.5

(4) $\boldsymbol{u_0} = \boldsymbol{u}_{\text{rand}}$, vecteur aléatoire de l'espace latent

La première initialisation est la plus naturelle puisqu'elle utilise ${}^tA$, ce qui se rapproche le plus d'un inverse pour A (voire *fig. 32, 33 & 34* en annexe B pour les visuels) et de même pour f_E par rapport à f_D.
Les initialisations (2) et (3) sont au voisinage de la cible pour voir à quel point il est possible de s'éloigner de $\boldsymbol{x_0}$ tout en conservant la convergence de l'algorithme.

Le premier constat est que la méthode fonctionne et ceux même en partant d'un vecteur aléatoire. Aussi, même s'ils sont meilleurs avec un passe-bas, les résultats restent satisfaisants sans filtre. La structure très simple du jeu de données aide probablement

sur ce point et il serait intéressant de voir si les différences avec/sans filtre sont plus marquées sur un jeu de données plus complexe comme Fashion-MNIST.
De plus, les courbes de loss de la figure *9* n'ont pas toutes finies de descendre (*e.i.* ne sont pas plates à $x = 300$) et c'est une remarque que l'on pourra faire sur beaucoup des graphes à suivre dès qu'il y a un filtre passe-bas. Il aurait fallu pousser le nombre d'itérations mais l'algorithme étant lent, on s'est arrêté en bout de 300 par manque de moyen. Cela dit, dans la plupart des cas ce n'est pas un problème au vu de la qualité des reconstructions.
Pour les cas sans passe-bas, les courbes ont tendance à passer par des plateaux, suggérant que la loss puisse encore diminuer en ajoutant de l'inertie à la descente.

Autre chose remarquable, le résultat (4) est meilleur que le (3) alors que l'initialisation du premier est plus éloigné de la cible que le second. Cela est dû à la façon dont est tiré $\boldsymbol{u}_{\text{rand}}$. Pour rester cohérent avec la structure des images, et le réseau f (composé de sigmoïde), les coefficients de $\boldsymbol{u}_{\text{rand}}$ sont tirés suivant une loi uniforme sur $[0, 1]$ et l'auto encodeur gère très mal les vecteurs à valeurs trop loin de l'intervalle $[0, 1]$.
Les figures *11* et *12* page suivante, mettent cela en évidence. Chaque ligne correspond à une descente avec en première ligne l'initialisation (décodée), en deuxième la reconstruction dont la cible est en troisième ligne. Les coefficients des initialisations sont tirés suivant une loi normale d'écart-type 2. En comparaison, les initialisations de la figure *10* ci-dessous ont été tirées suivant une loi uniforme sur $[0, 1]$.

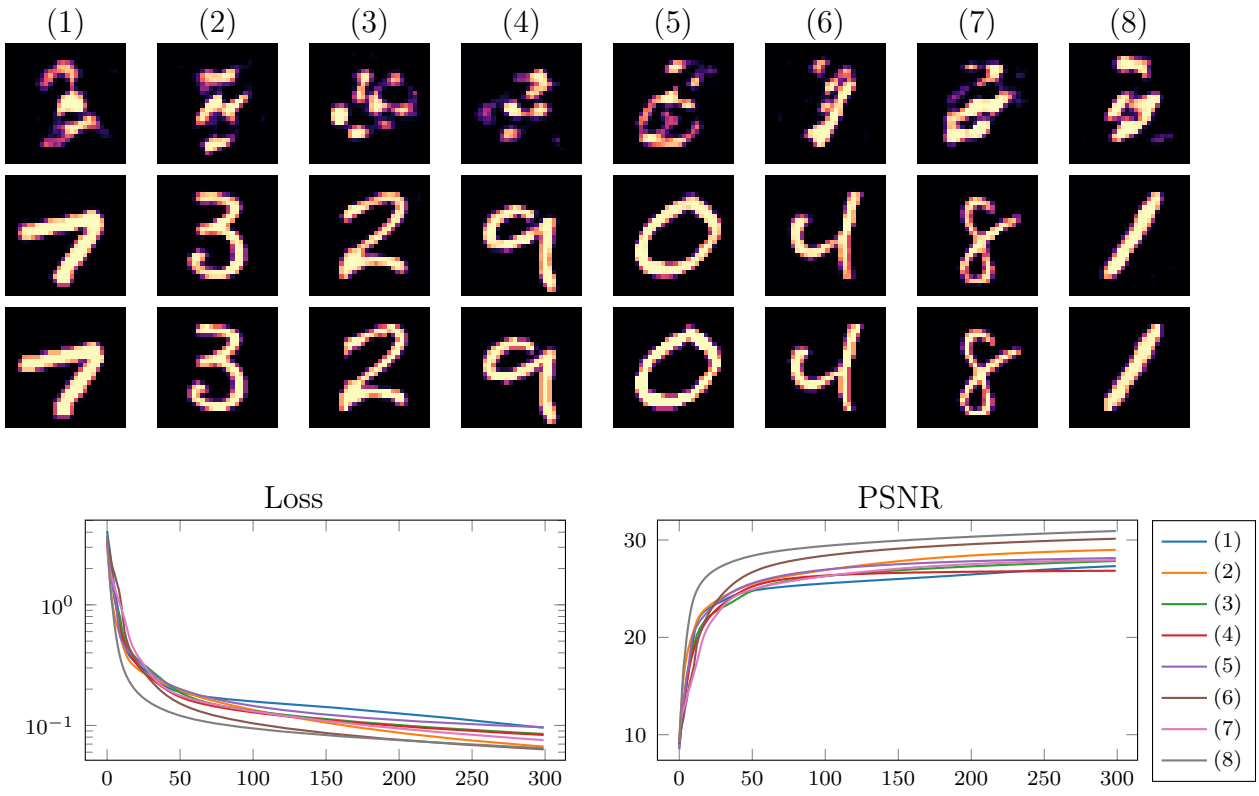

fig. 10 — Résultats de LGD toutes initialiser par un vecteur tiré suivant une lui uniforme sur $[0, 1]$ — avec passe-bas gaussien ($\sigma = 0.6$)

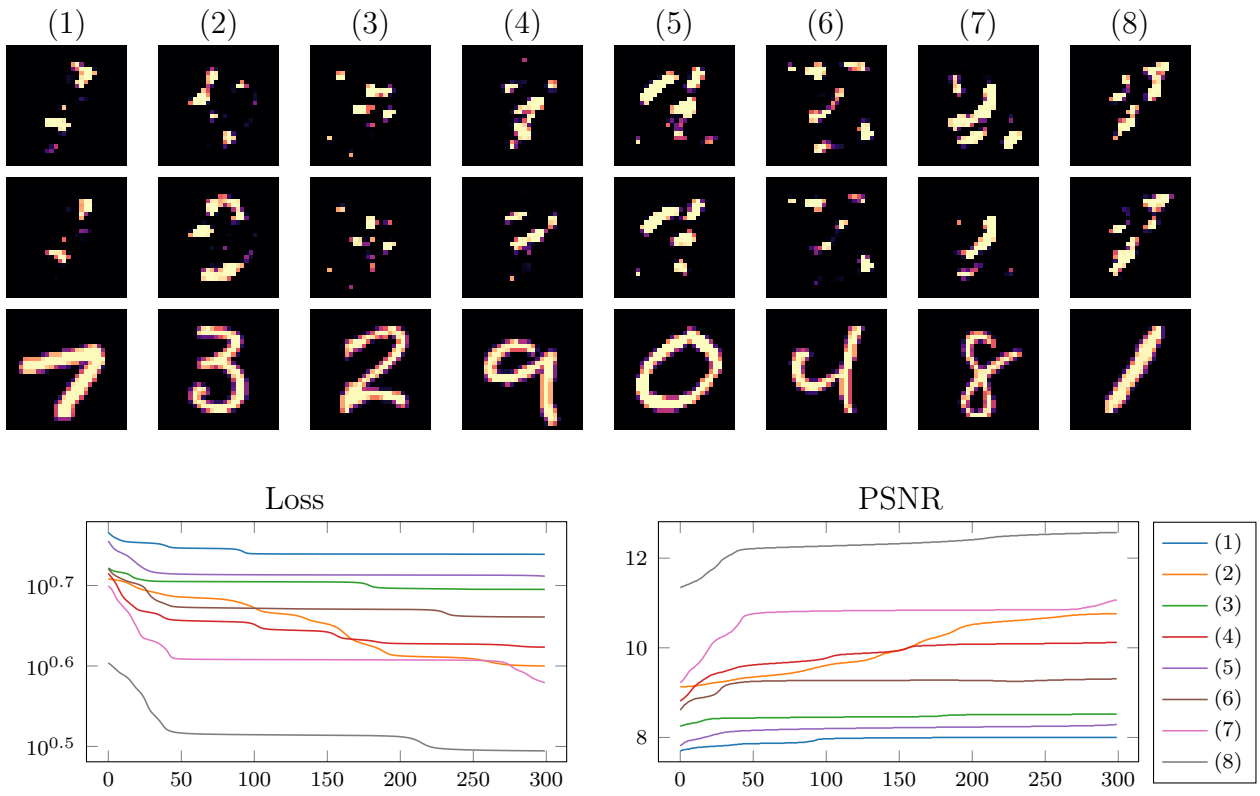

fig. 11 — Plusieurs résultats de LGD — sans passe-bas

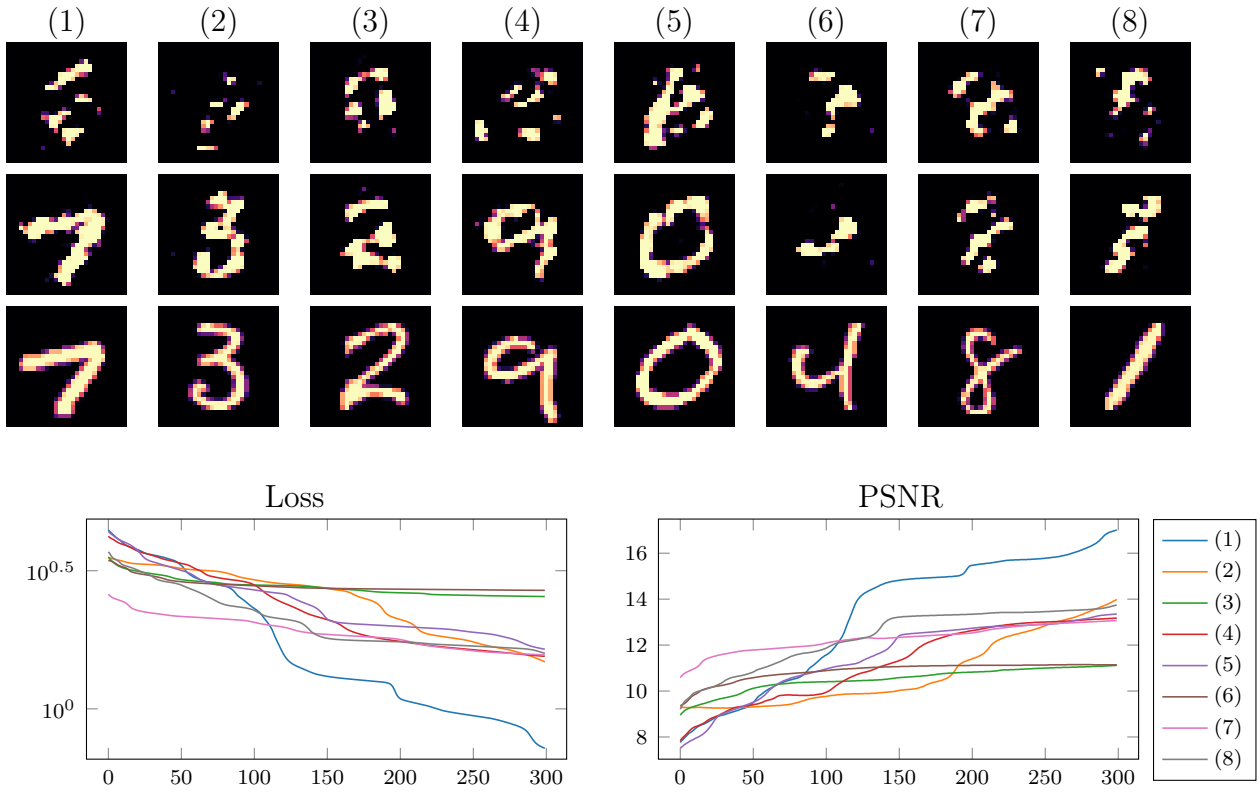

fig. 12 — Plusieurs résultats de LGD — avec passe-bas gaussien ($\sigma = 0.6$)

Clairement, les courbes sont beaucoup plus chaotiques et on pourrait supposer que cela vient de l'apprentissage de f qui aurait "écarté" les parties de l'espace latent où f_D n'est pas injective en dehors de $[0,1]^d$.

2.2. Variation de la qualité de mesure (e.i. $p \times q$)

Pour tester les limites de la méthode par rapport au rapports n/p et m/q, la LGD à été effectuée avec les valeurs suivantes (les pas ont été ajustés en fonction des cas) :

(1) $(p, q) = (14, 14)$

(2) $(p, q) = (14, 7)$

(3) $(p, q) = (7, 14)$

(4) $(p, q) = (7, 7)$

(5) $(p, q) = (5, 5)$

Les résultats, *fig. 13 & 14*, restent concluant malgré le haut niveau de compression. La première ligne correspond aux initialisations (par backprojection), la seconde au sous-échantillonnage $A\boldsymbol{x_0}$ de la cible $\boldsymbol{x_0}$ et la troisième à la reconstruction par LGD. Avec la diminution de p et q, l'impact du filtre se fait sentir, sans passe-bas l'algorithme à beaucoup plus de mal à reconstruire l'information perdue. Du point de vue des courbes, on voit aussi que la filtre passe-bas à tendance à bien plus les lisser, ce qui sous-entend une potentielle meilleure convexité de g.

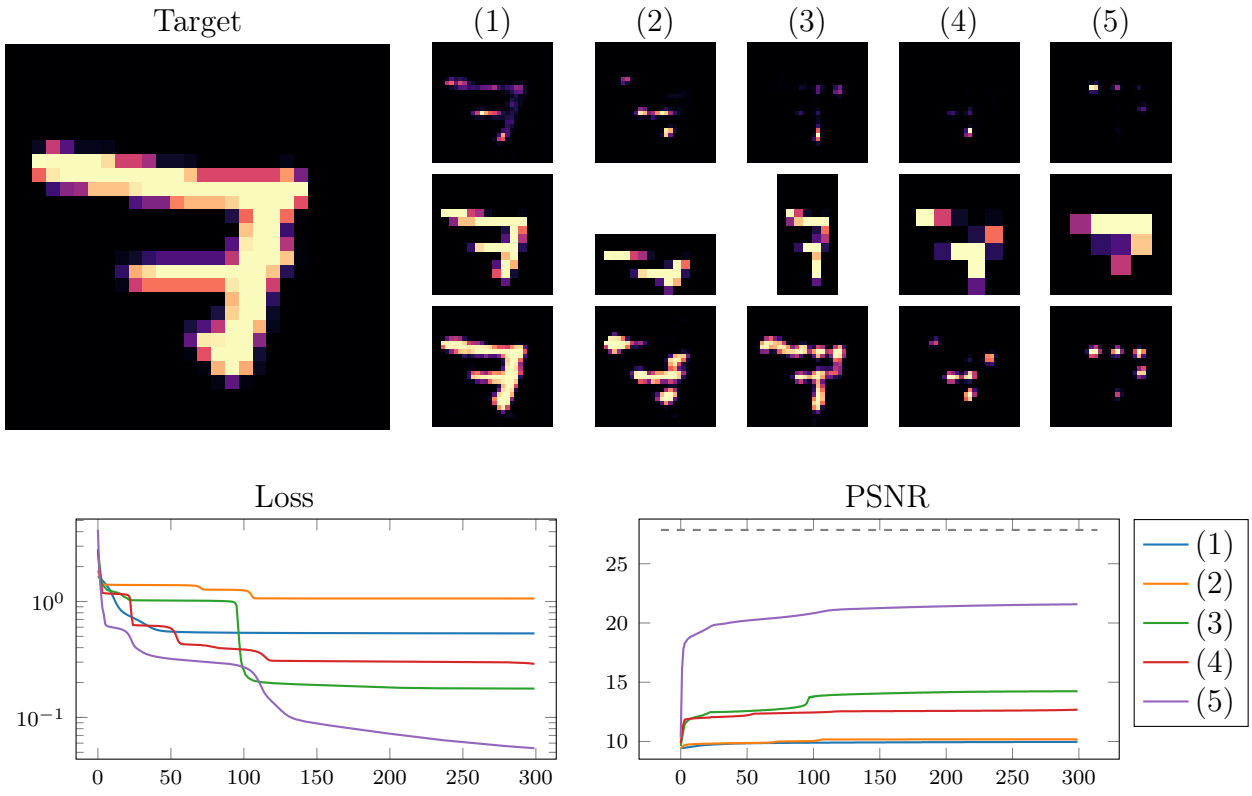

fig. 13 — Résultats de la LGD avec différents niveau de compression — sans passe-bas

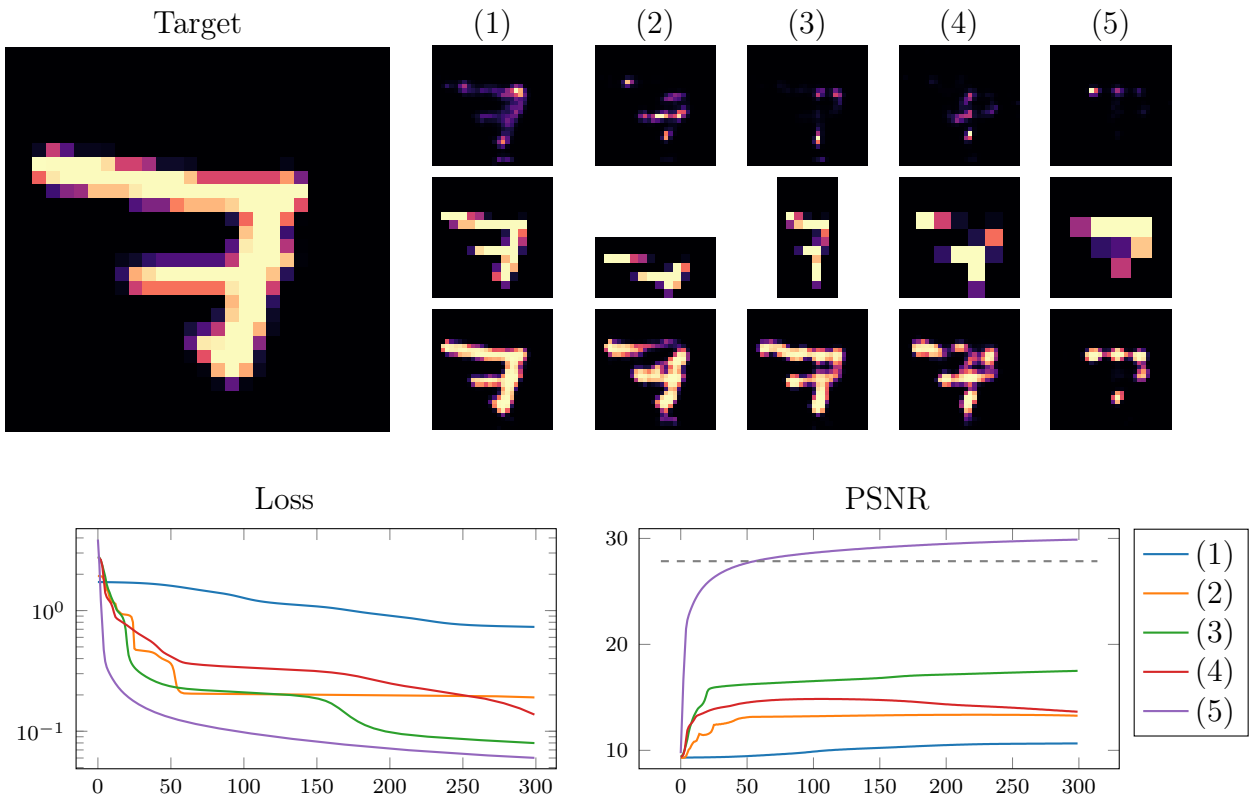

fig. 14 — Idem, cette fois avec un passe-bas gaussien ($\sigma = 0.5$)

2.3. Variation de la taille de l'espace latent

Toujours dans l'optique de voir les limites de la méthode, trois autres auto-encodeurs ont été entraînés avec différentes tailles d'espace latent. Pour chacun — *fig. 15* à *22* — la LGD est effectuée avec différentes initialisations (toujours en première ligne avec la prédiction en dessous). En fonction des cas, la loss fait parfois des zig-zags, ce qui indique que le pas est trop grand. Mais même sachant cela, les résultats sont étonnamment bons. En particulier pour les deux dernières figures, où $d = 800$ ce qui est plus que la taille des images, à savoir $28 \times 28 = 784$. À noter tout de même qu'à mesure que d augmente les PSNR reste de plus en plus loin du PSNR de référence ($\text{PSNR}(\boldsymbol{x_0}, f(\boldsymbol{x_0}))$), qui lui reste globalement stable d'un auto-encodeur à l'autre.
On y voit aussi qu'à mesure que d augmente, les cas sans passe-bas présentent de moins en moins de plateaux, là où avec, on commence à voir apparaître des sauts, peut-être également dû au pas trop grand.

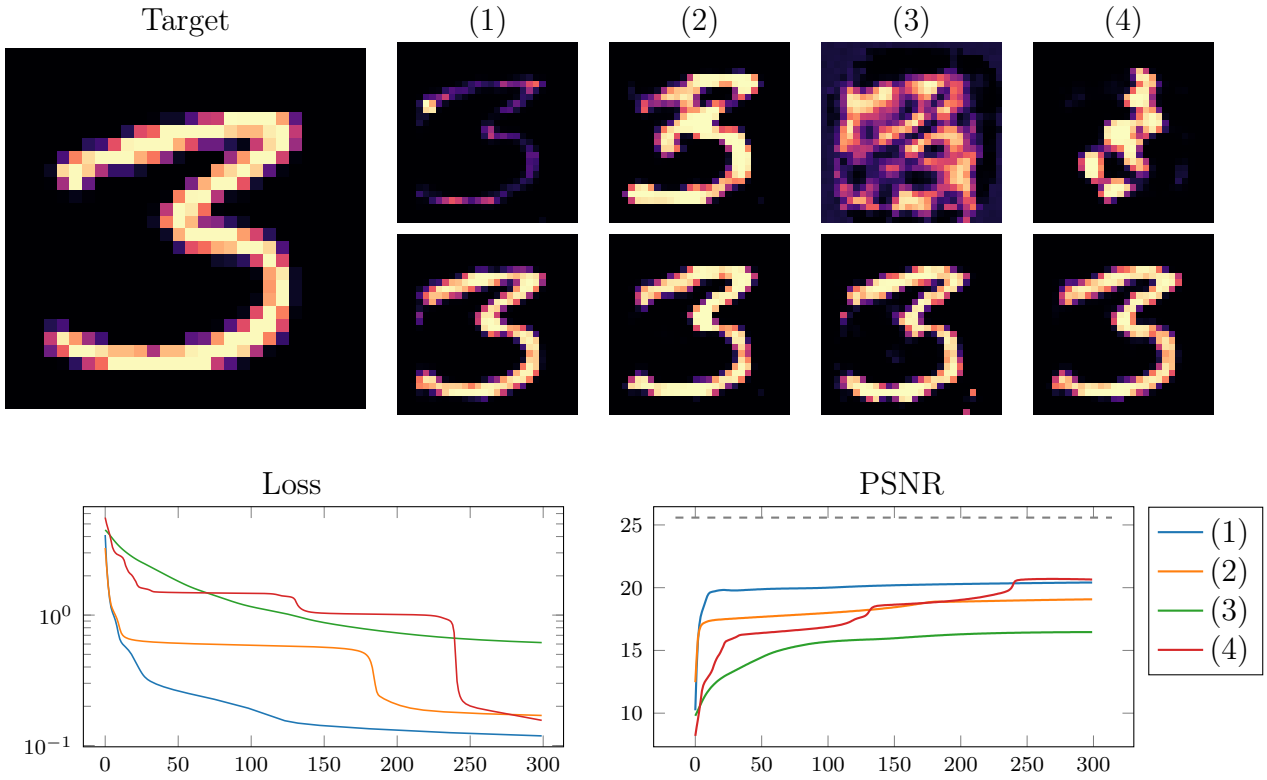

fig. 15 — $d = 100$, sans passe-base

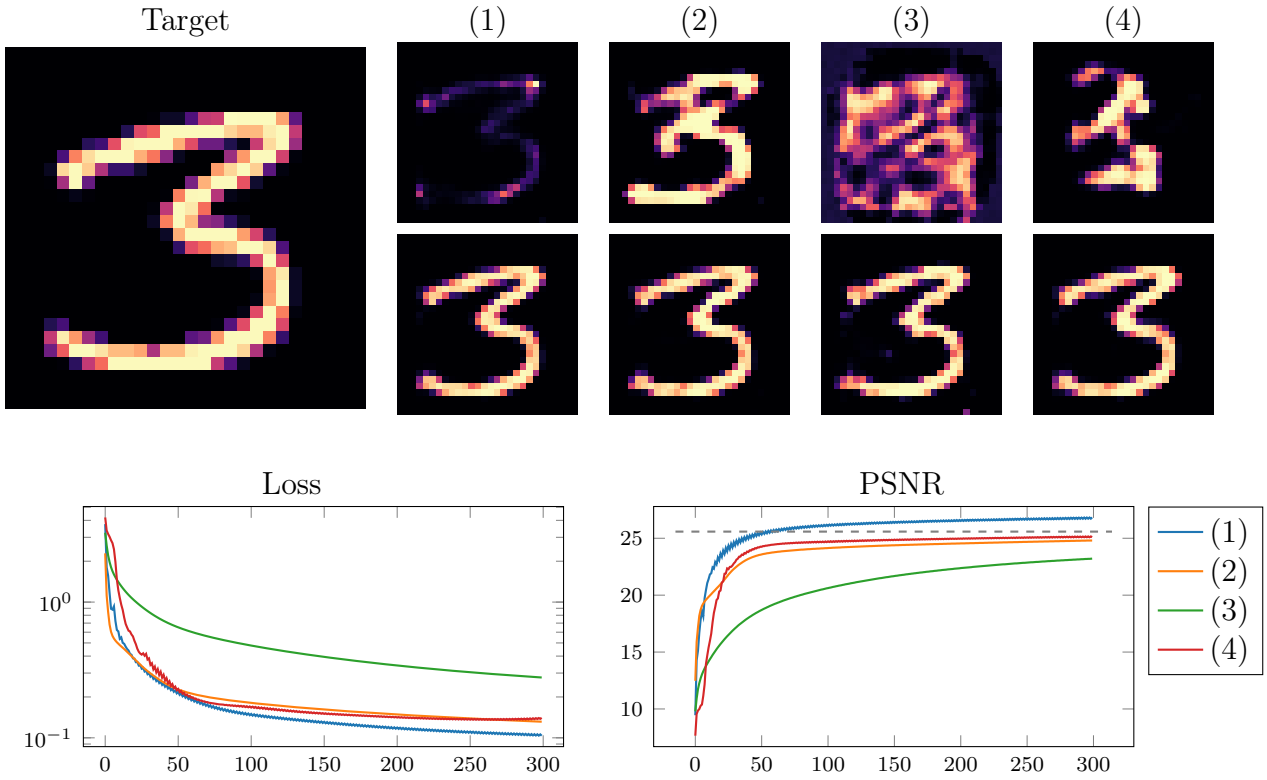

fig. 16 — $d = 100$, avec passe-base gaussien ($\sigma = 0.6$)

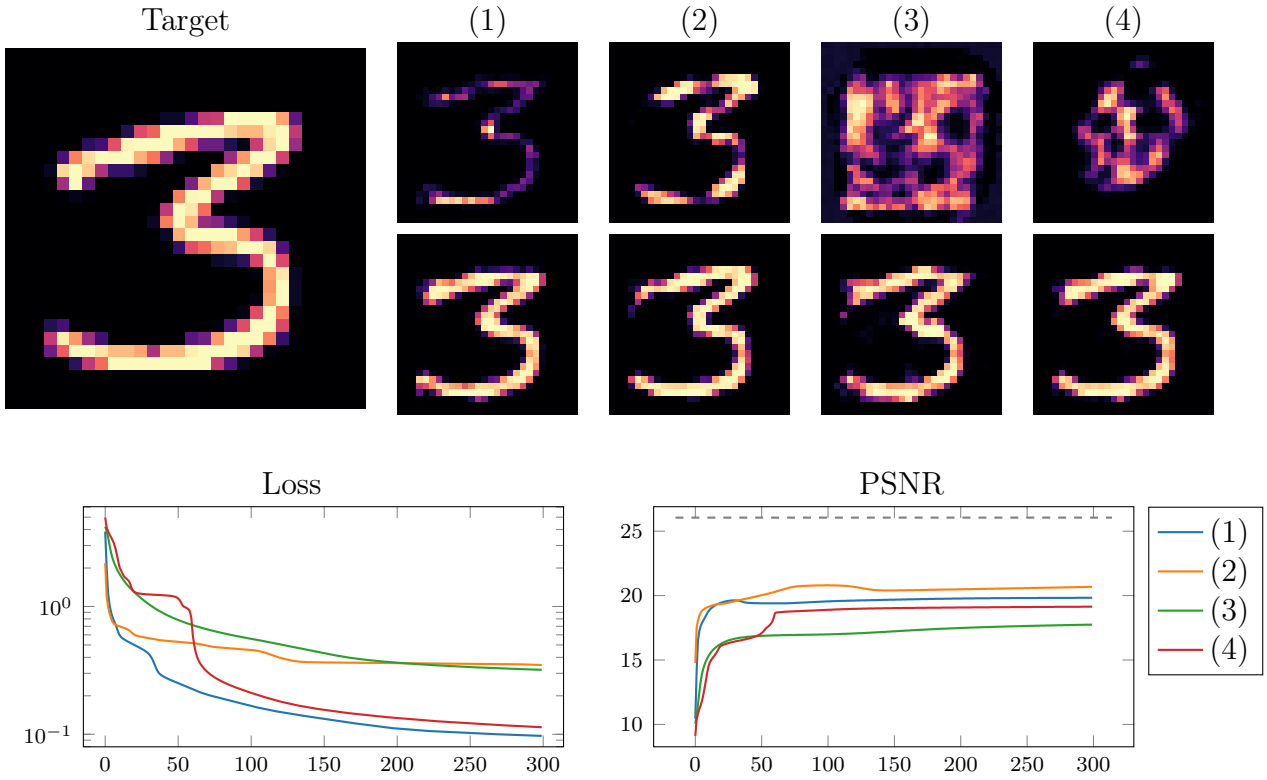

fig. 17 — $d = 200$, sans passe-base

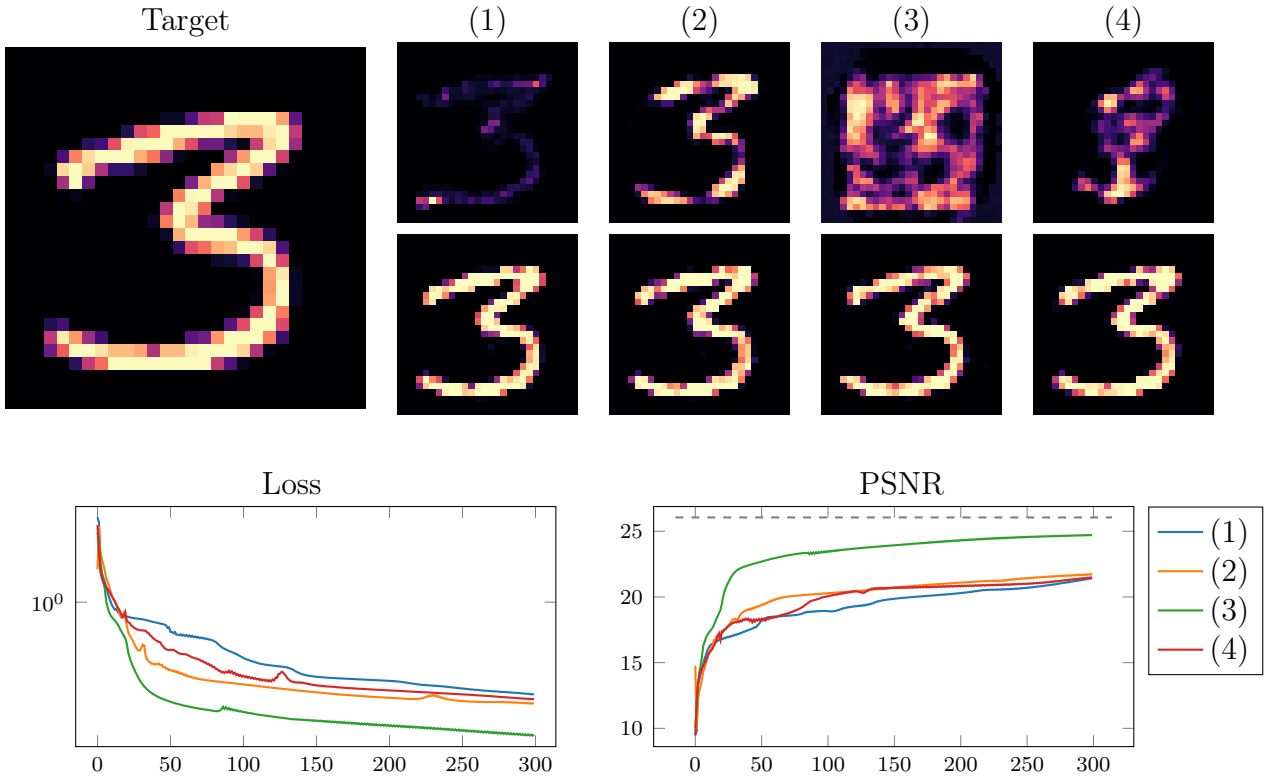

fig. 18 — $d = 200$, avec passe-base gaussien ($\sigma = 0.6$)

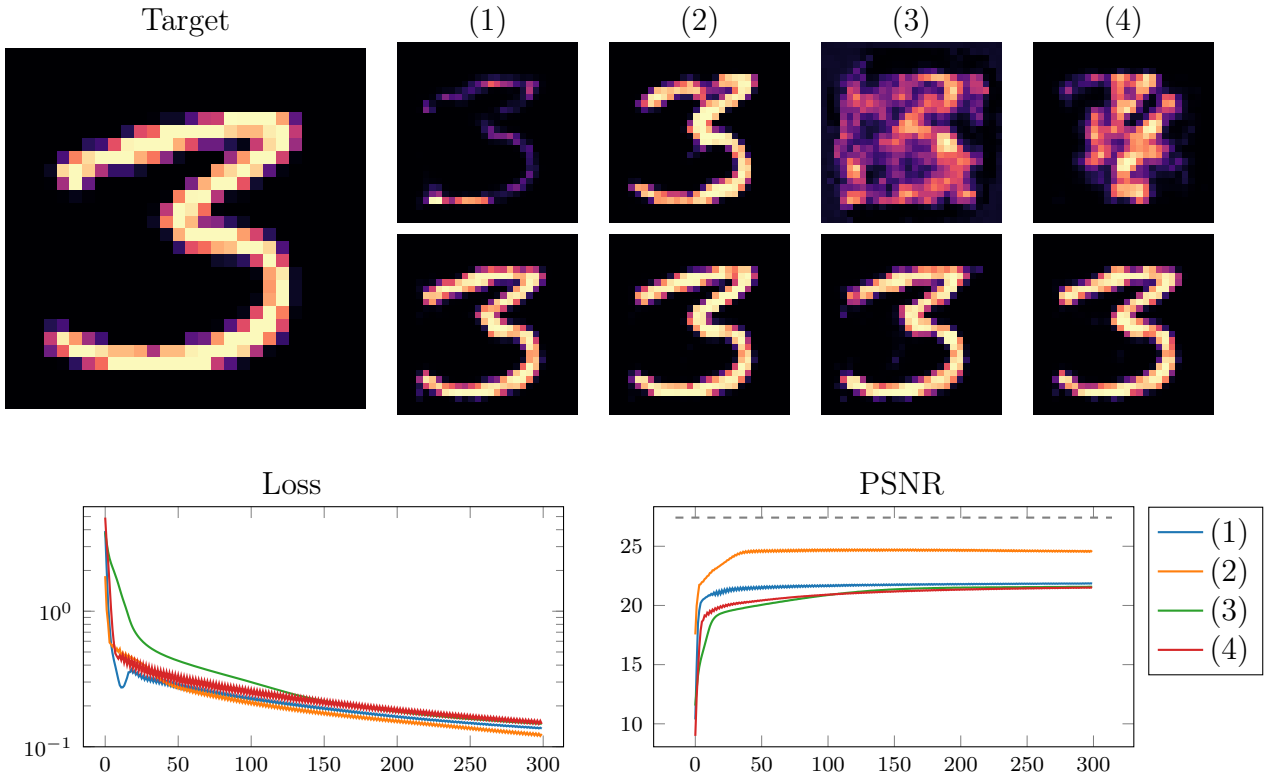

fig. 19 — $d = 400$, sans passe-base

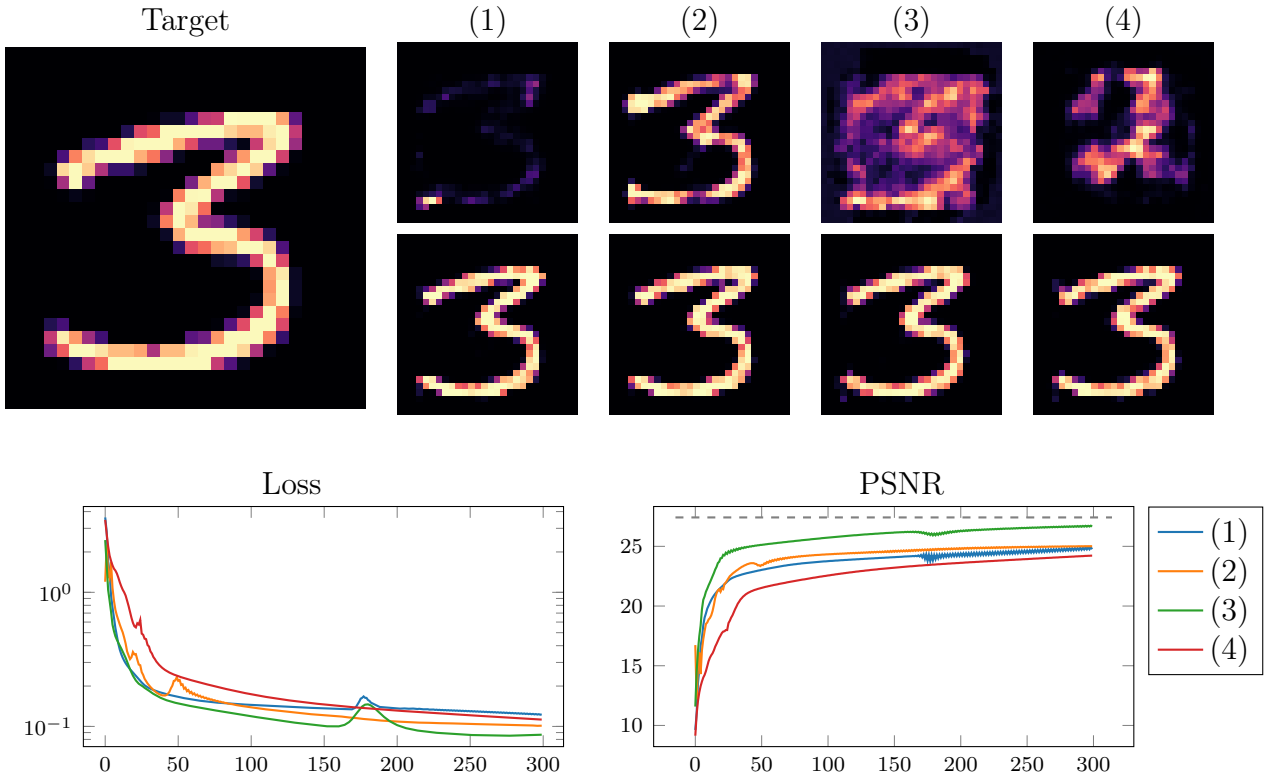

fig. 20 — $d = 400$, avec passe-base gaussien ($\sigma = 0.6$)

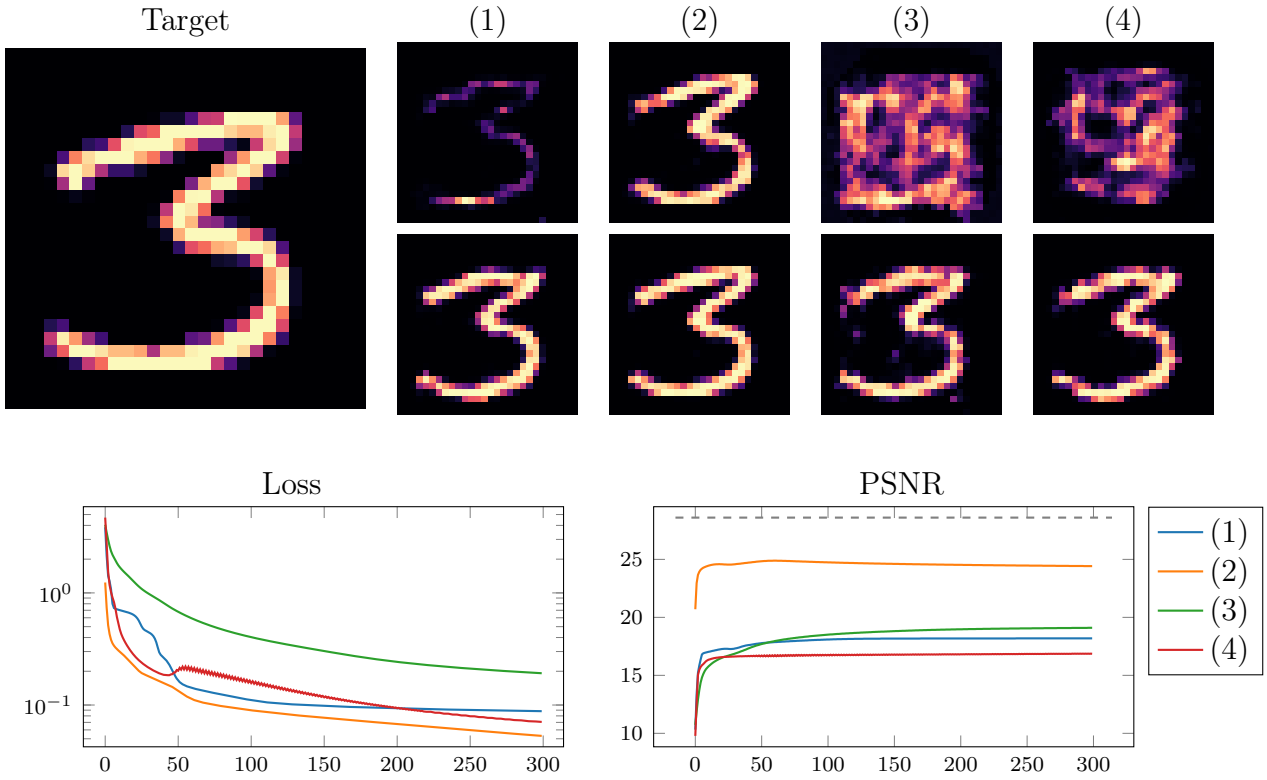

fig. 21 — $d = 800$, sans passe-base

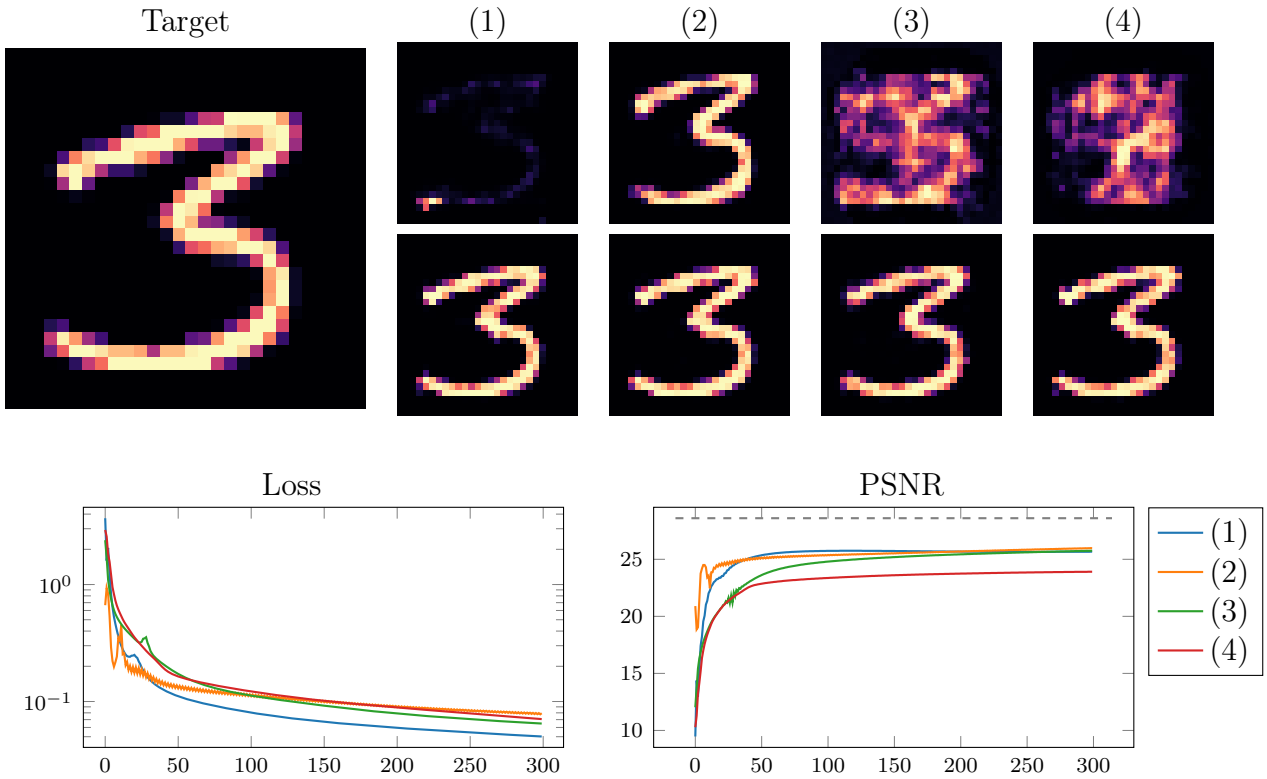

fig. 22 — $d = 800$, avec passe-base gaussien ($\sigma = 0.6$)

2.4. Comparaison avec la descente de gradient projetée

La descente de gradient projeté (PGD) par auto-encodeur, comme son nom l'indique, consiste en une descente de gradient classique à cela près qu'à chaque itération, le vecteur obtenu est projeté par une fonction$p$ (voir *fig. 23* ci-contre).
Ce rapport étant surtout porté sur la descente de gradient depuis l'espace latent de p, on ne détaillera pas les arguments mathématiques qui supportent cette méthode mais on peut tout de même essayer de se convaincre de l'intérêt de la PGD.
Comme expliqué plus haut, le problème est convexe ce qui assure la convergence de la descente de gradient. La difficulté étant de tomber sur un "bon" argmin. C'est là que composer par p peut aider à diriger la descente vers le bon minimum. Si on parle de descente *projetée* c'est aussi parce que, en appliquant p, on s'assure qu'à chaque itération l'image en reconstruction $\boldsymbol{x}_n$ appartient au modèle de basse dimension Σ.

x_0 : initialisation
x : image à reconstruire
ρ : pas de descente
N : nombre d'itération
$y \longleftarrow A(x)$

for $n \in [\![1, N-1]\!]$ **do**
$\nabla F(x_n) \longleftarrow {}^tA\big(Ax_n - y_0\big)$
$x_{n+1} \longleftarrow p\big(x_n - \rho\nabla F(x_n)\big)$
end for
return $(x_n)_{0\leqslant n\leqslant N}$

fig. 23 — Algorithme de PGD

Là encore, il n'y a pas de définition claire pour Σ pour les images et donc pas de moyen de construire simplement une projection. C'est là qu'un intervient le réseau f. L'auto-encodeur étant entraîné sur les données, f doit pouvoir faire office de projection pour des images qui ne sont pas trop éloignées de Σ. Cette méthode est tirée de l'article [1] de P. Peng, S. Jalali and X. Yuan et on a pu reproduire leurs résultats.

Comme précédemment, dans les figures *24* à *29*, plusieurs PGD sont effectuées, un coup sans filtre passe-bas, un coup avec. Les deux premières sont initialisées par back-projeciton (*e.i.* par ${}^tA\boldsymbol{y_0}$), les deux suivantes par des vecteurs aléatoires tirés suivant une loi uniforme sur $[0, 1]$ et les deux dernières suivant une loi normale d'écart-type 2.
Dans tous les cas, il est clair que la descente projetée est moins efficace que la LGD. Par contre, elle converge beaucoup plus vite : en seulement une quinzaine d'itérations la loss ne descend plus mais commence à remonter. Cette remontée est très probablement dû au fait que le gradient est devenu quasiment nul et que l'algorithme revient à appliquer f en boucle. Comme l'auto-encodeur c'est pas équivalent à l'application identité, on constate une dérive.
De plus, si la PGD est moins efficace que la LGD, les figures *28* et *29* montre qu'elle est en revanche plus robuste à l'initialisation par des vecteurs en dehors de $[0, 1]^{n\times m}$.

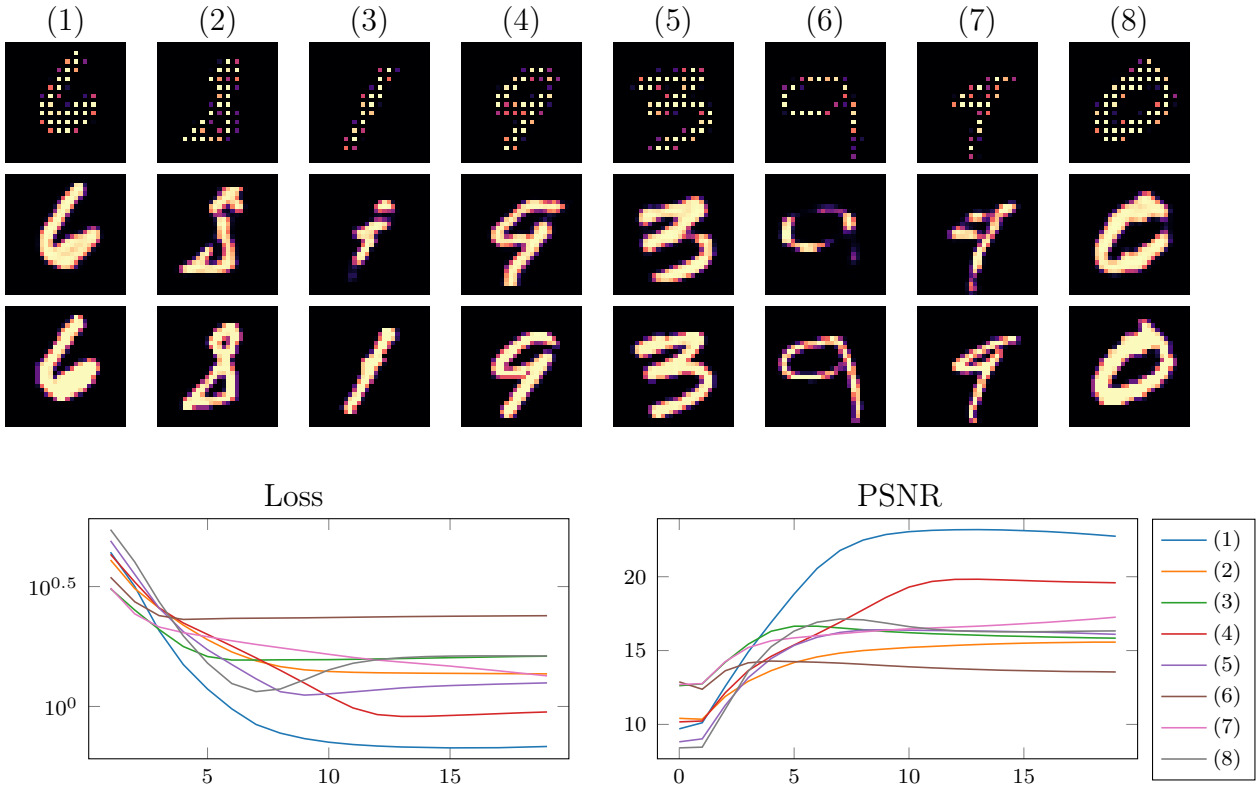

fig. 24 — PGD initialisée par backprojection — sans passe-bas

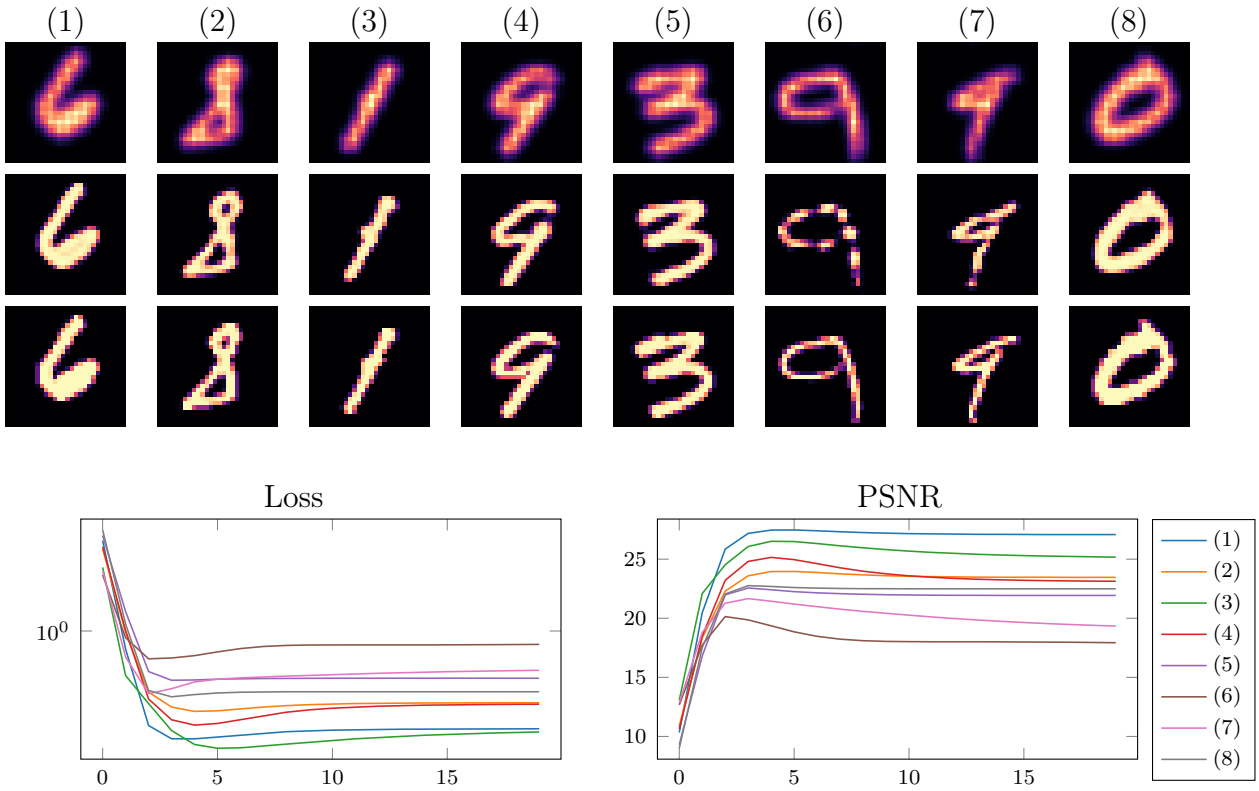

fig. 25 — PGD initialisée par backprojection — avec passe-bas gaussien ($\sigma = 0.6$)

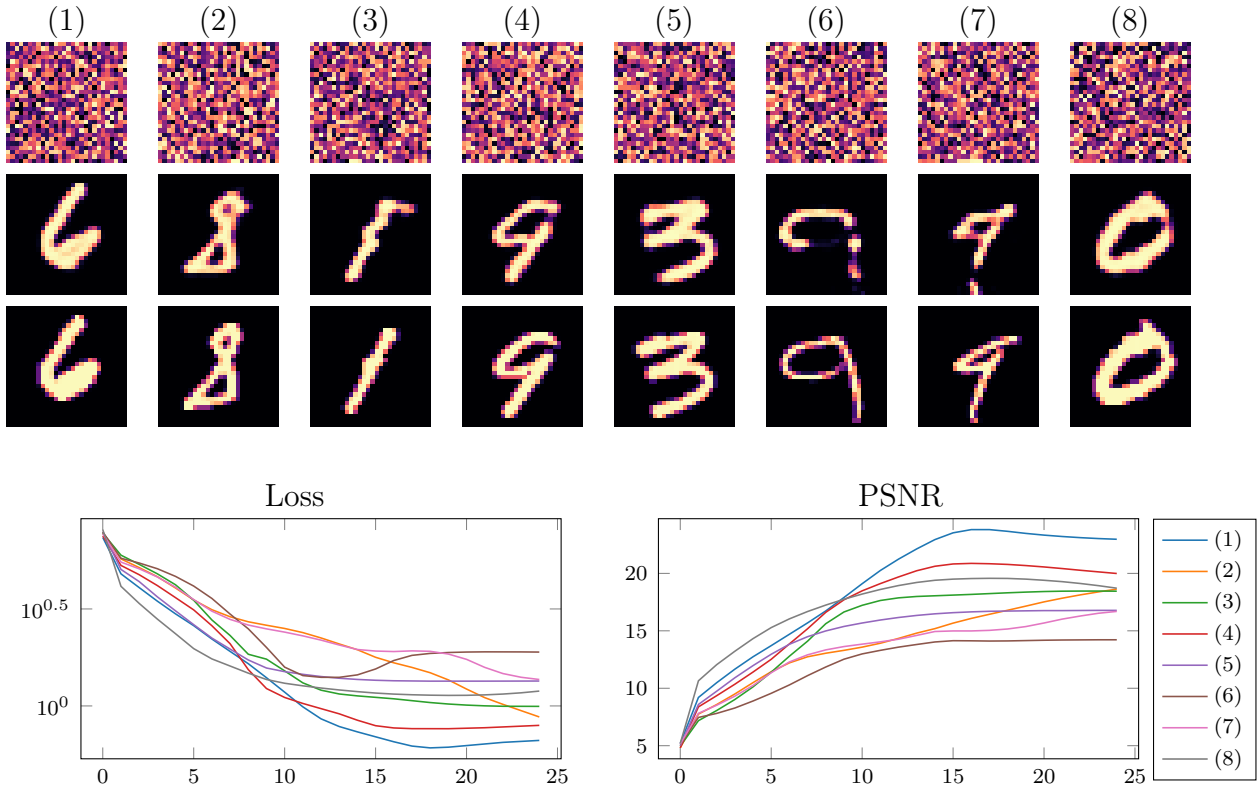

fig. 26 — PGD initialisée suivant une loi uniforme — sans passe-bas

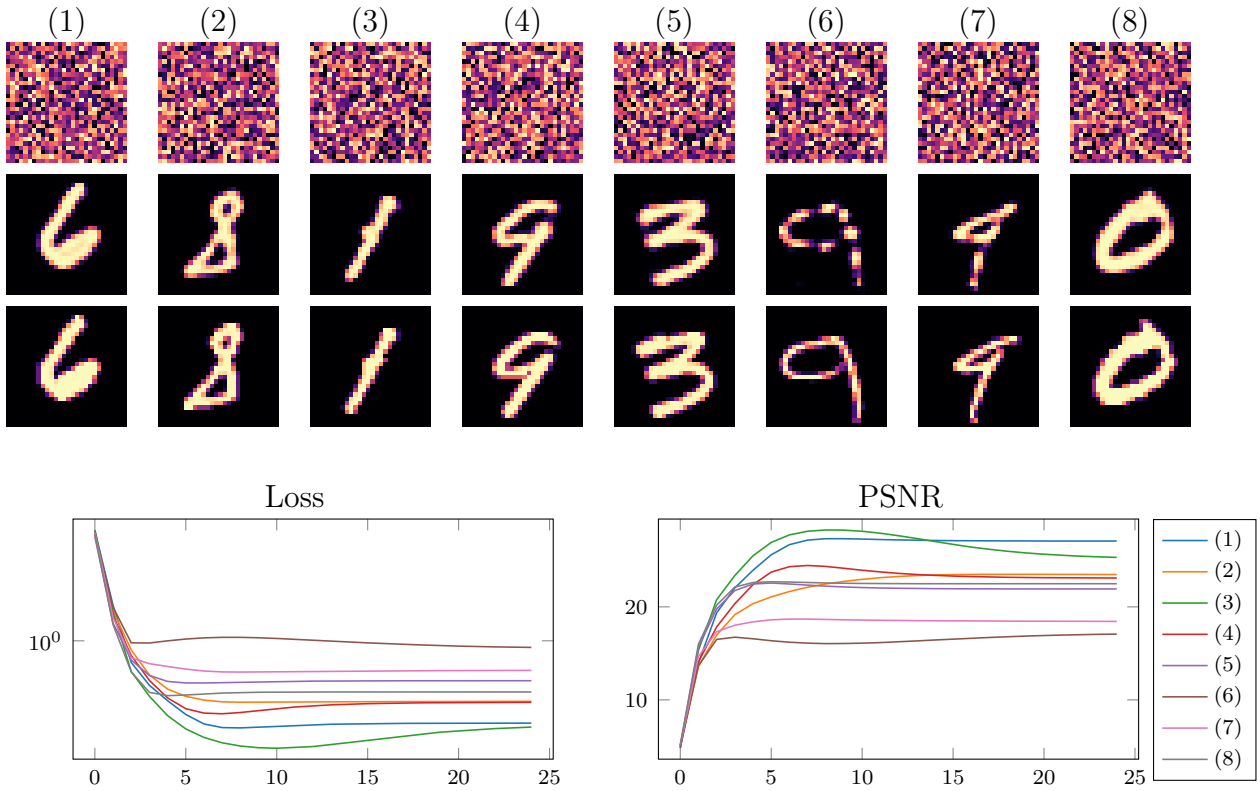

fig. 27 — PGD initialisée suivant une loi uniforme — avec passe-bas gaussien ($\sigma = 0.6$)

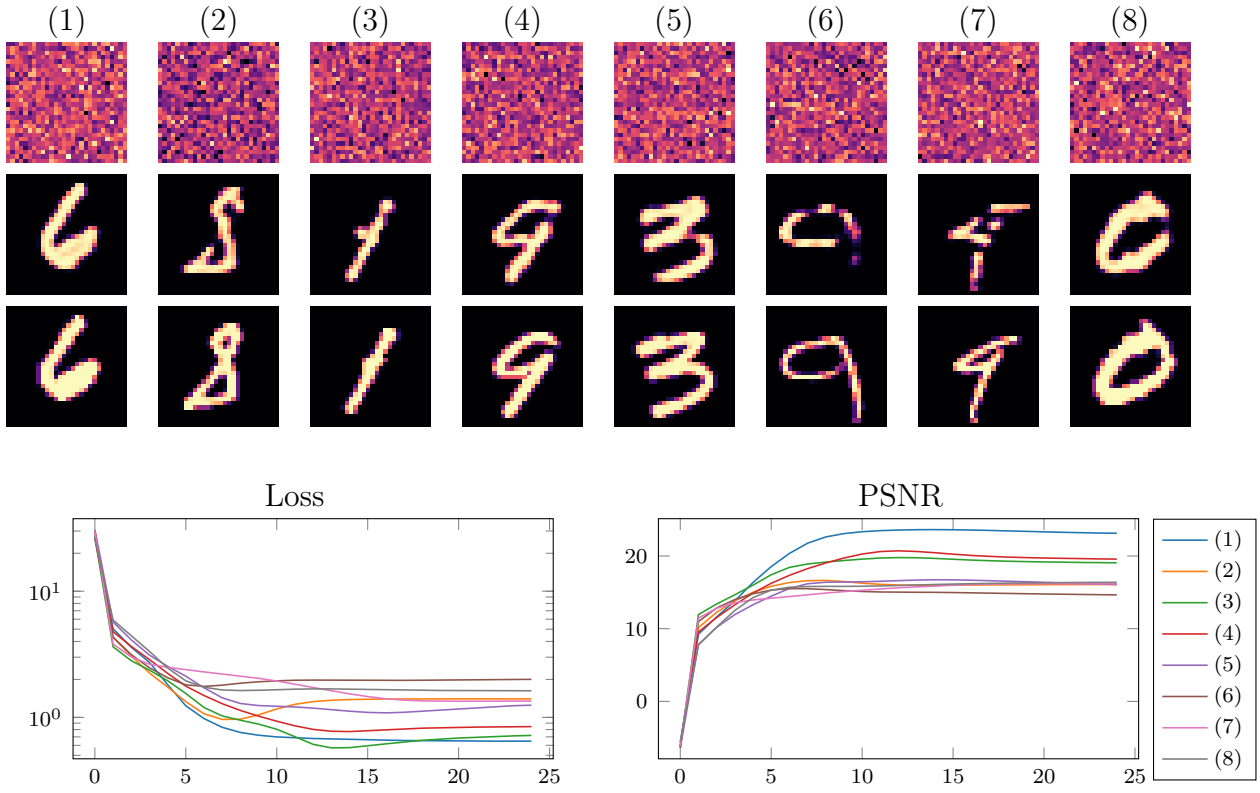

fig. 28 — PGD initialisée suivant une loi normale — sans passe-bas

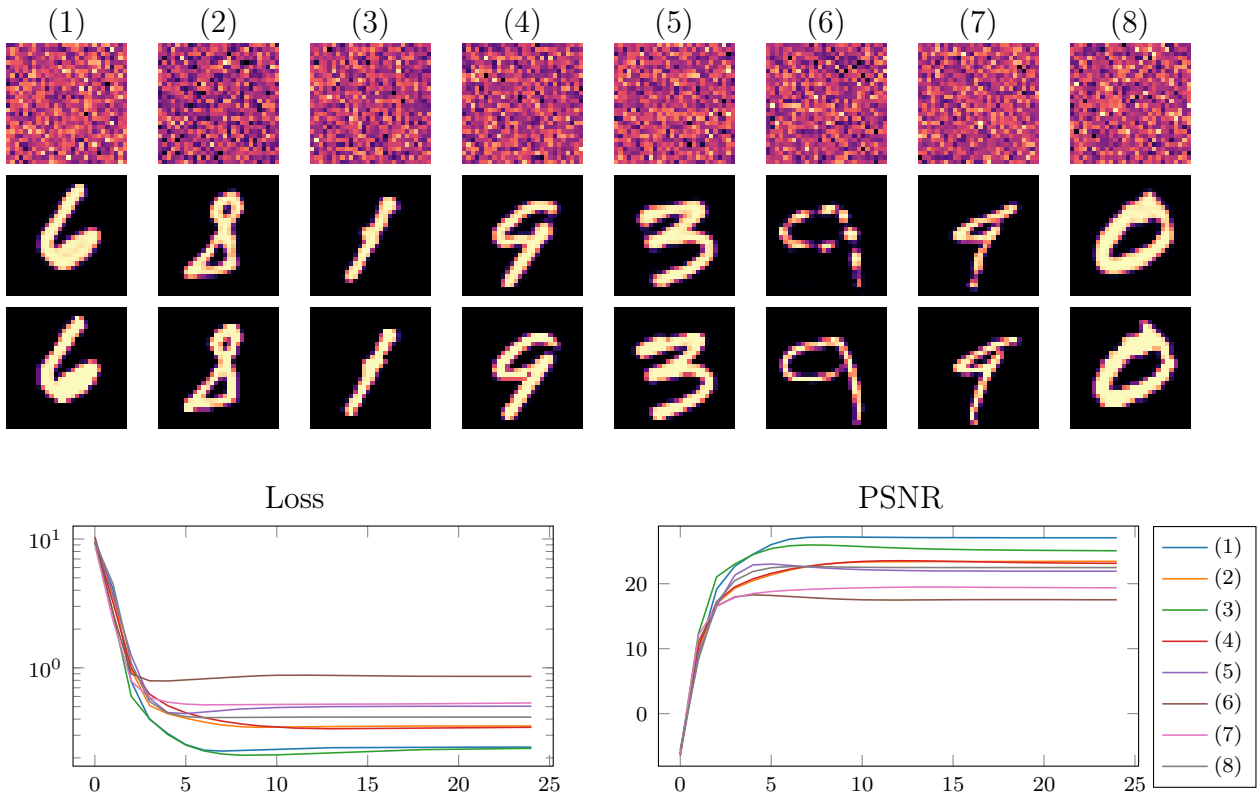

fig. 29 — PGD initialisée suivant une loi normale — avec passe-bas gaussien ($\sigma = 0.6$)

III. Conclusion

La LGD est étonnamment robuste, alors que la théorie n'était pas encourageante. Reste à savoir si elle le reste sur un jeu de données plus complexe et si c'est le cas, c'est vraisemblablement qu'il doit être possible d'améliorer ces résultats théorique, au moins pour les décodeurs appris en paramétrisation.

Aussi, le choix a été fait de supposer f_D inversible sur Θ pour exprimer le fait que f_D vienne qu'un auto-encodeur. L'apport sur la théorie de cette hypothèse n'est pas énorme et sachant la dérive observée dans les derniers résultats (section 2.4.), ce n'est pas nécessairement un choix très pertinent. Peut-être qu'une formalisation moins dure, pas exemple en terme de voisinage, serait plus intéressante.
Enfin, le filtre porte n'a pas du tout été utilisé dans ce rapport, il serait intéressant de voir dans quelle mesure la descente depuis l'espace latent est robuste au phénomène de Gibbs, quitte à améliorer le code en ajoutant de l'inertie et éventuellement un pas adaptatif fonctionnel.

Annexes

Annexe A — Auto-encodeur

Comme expliqué en début de rapport, section 1.1., les auto-encodeurs utilisés, sont des réseaux MLP comptant une couche cachée de taille 1500 pour les parties encodage et décodage, et d'espace latent de taille respective 100, 200, 400 et 800. Pour reprendre ce qui a été fait par Peng *et al.* , toutes les fonctions d'activations sont des sigmoïdes, ce qui, rétrospectivement, n'était pas le plus judicieux (*cf.* section 1.2.).
Leur entraînement s'est fait sur le jeu de données MNIST sur 60 epochs avec des batchs de tailles 100. Il est effectué par le code `training autoencoder.py` disponible sur le GitHub et dont voici un résumé des performances :

fig. 30 — Reconstruction par auto-encodeur des images Input en fonction de la taille de l'espace latent dudit auto-encodeur et avec les PSNR associés

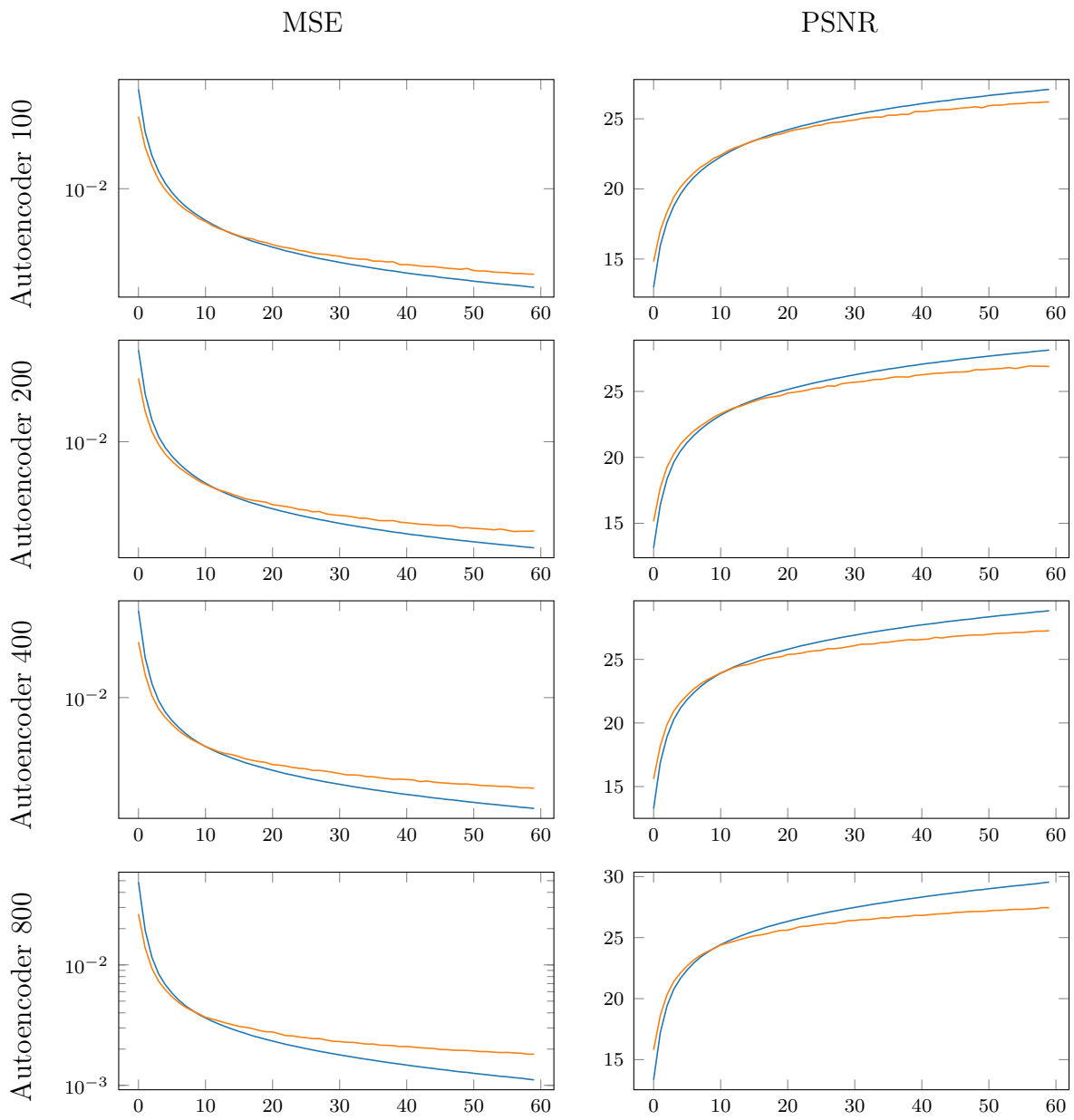

fig. 31 — Évolution de la MSE et du PSNR par batch des auto-encodeurs au cours des epochs. En bleu sur le set d'entraînement et en orange sur celui de test

Annexe B — Calcul pratique de F et de son gradient

NOTATION — Comme A est linéaire, dans toute la suite elle sera assimilé à matrice la représentant. Là où l'opérateur A prend pour entrée un image $\boldsymbol{x} \in \mathbb{R}^{n \times m}$, sa représentation matricielle elle prend un vecteur $x \in \mathbb{R}^{nm}$. Par soucis de lisibilité on notera dans toute la suite les images sous forme de matrice en gras, leur version vectorielle en fin.

Le fait de supposer A linéaire permet de calculer explicitement le gradient de F à moindre coût :

$$\forall x \in \mathbb{R}^{nm}, \qquad \nabla F(x) = {}^tA\big(Ax - y\big) \tag{6}$$

Si la formule (6) est mathématiquement très simple, dès que n et m seront un peu trop grands, la taille de A va très vite ralentir tout algorithme de descente et prendre énormément de place en mémoire.

Pour éviter d'avoir à construire à explicitement on passe par la transformée de Fourier. On note $\mathscr{F}$ (resp. $\mathscr{F}^{-1}$) la matrice associée à la transformée (resp. inverse) de Fourier discrète. En notant de plus $\hat{\boldsymbol{x}} = \mathscr{F}\boldsymbol{x}$ et $\odot$ le produit terme à terme de vecteur/matrice, A s'écrit alors :

$$\begin{aligned}\forall \boldsymbol{x} \in \mathbb{R}^{n\times m}, \qquad A(\boldsymbol{x}) = S \circ C_h(\boldsymbol{x}) = S(\boldsymbol{h} * \boldsymbol{x}) &= S \circ \mathscr{F}^{-1}\mathscr{F}(\boldsymbol{h} * \boldsymbol{x}) \\ &= S \circ \mathscr{F}^{-1}\big(\hat{\boldsymbol{h}} \odot \hat{\boldsymbol{x}}\big)\end{aligned}$$

Notons D_h l'application/matrice associée au produit $\hat{h} \odot \cdot$. Comme la transformée de Fourier discrète et D_h sont symétriques[3], le gradient de F se réécrire comme :

$$\begin{aligned}\forall x \in \mathbb{R}^{nm}, \qquad \nabla F(x) = {}^tA\big(Ax - y\big) &= {}^t\Big(S\mathscr{F}^{-1}D_h\mathscr{F}\Big)\big(S\mathscr{F}^{-1}D_h\mathscr{F}x - y\big) \\ &= {}^t\mathscr{F}\,{}^tD_h\,{}^t\mathscr{F}^{-1}\,{}^tS\big(S\mathscr{F}^{-1}D_h\mathscr{F}x - y\big) \\ &= \mathscr{F}D_h\mathscr{F}^{-1}\,{}^tS\big(S\mathscr{F}^{-1}D_h\mathscr{F}x - y\big)\end{aligned}$$

Sous cette forme et même s'il n'en n'a pas l'air, le gradient est bien plus simple à calculer et rapide à calculer.

D'abord, la projection S est très peu coûteuse en temps de calcul puisqu'elle consiste à ne garder qu'un certain nombre de pixel de l'image. Du point de vu python, il n'y a pas besoin de construire S, simplement de faire l'opération :

$$\forall x \in \mathbb{R}^{n\times m}, \qquad S(x) := \texttt{Sx = x[::n//p, ::m//q]}$$

Inversement, la transposé ${}^tS$ à une image $y \in \mathbb{R}^{p\times q}$ revient à construire une image ${}^tSy = \texttt{tSy}$ de taille (n, m) remplie de 0 puis de faire la modification :

$$\forall (i,j) \in [\![1,p]\!] \times [\![1,q]\!], \qquad \texttt{tSy[i*n//p, j*p//q] = y[i,j]}$$

Il n'y a pas besoin non plus de construire D_h, le produit $\hat{\boldsymbol{h}} \odot \hat{\boldsymbol{x}}$ suffit et le passage dans/hors de l'espace des fréquences est fait par `fft`. Ainsi, il n'y a jamais besoin d'aplatir les images et dans ce cas, on a bien la symétrie de $\mathscr{F}$ dans le sens où si, avec les conventions de notations d'Einstein, $\hat{\boldsymbol{x}}^{kl} = \mathscr{F}_{ij}^{kl}\boldsymbol{x}^{ij}$, alors $\mathscr{F}$ vérifie :

$$\forall i,j,k,l, \qquad \mathscr{F}_{ij}^{kl} = \mathscr{F}_{kj}^{il} = \mathscr{F}_{il}^{kj}$$

[3]comme x a été aplatie, $\mathscr{F}$ n'est pas diagonale à proprement parler, nous y revenons plus loin

Enfin, comme son nom l'indique, il est plus naturel de construire le filtre *passe-bas* directement dans l'espace de Fourier, et ce qui est fait dans le code. Les figures *33* et *34* montrent les applications successives de A et ${}^tA$ à une image. La figure *34* en particulier, met bien en évidence l'effet de Gibbs qui apparaît lorsque les fréquences sont brutalement coupées.

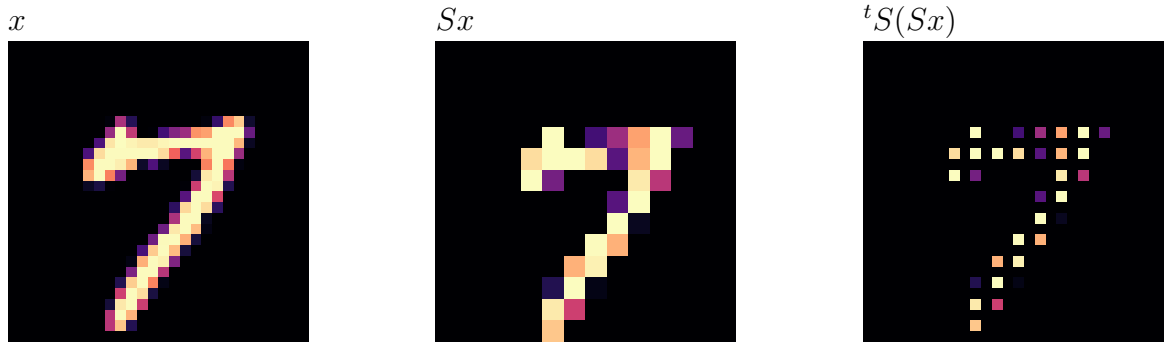

fig. 32 — Application successive de S et ${}^tS$ à une image

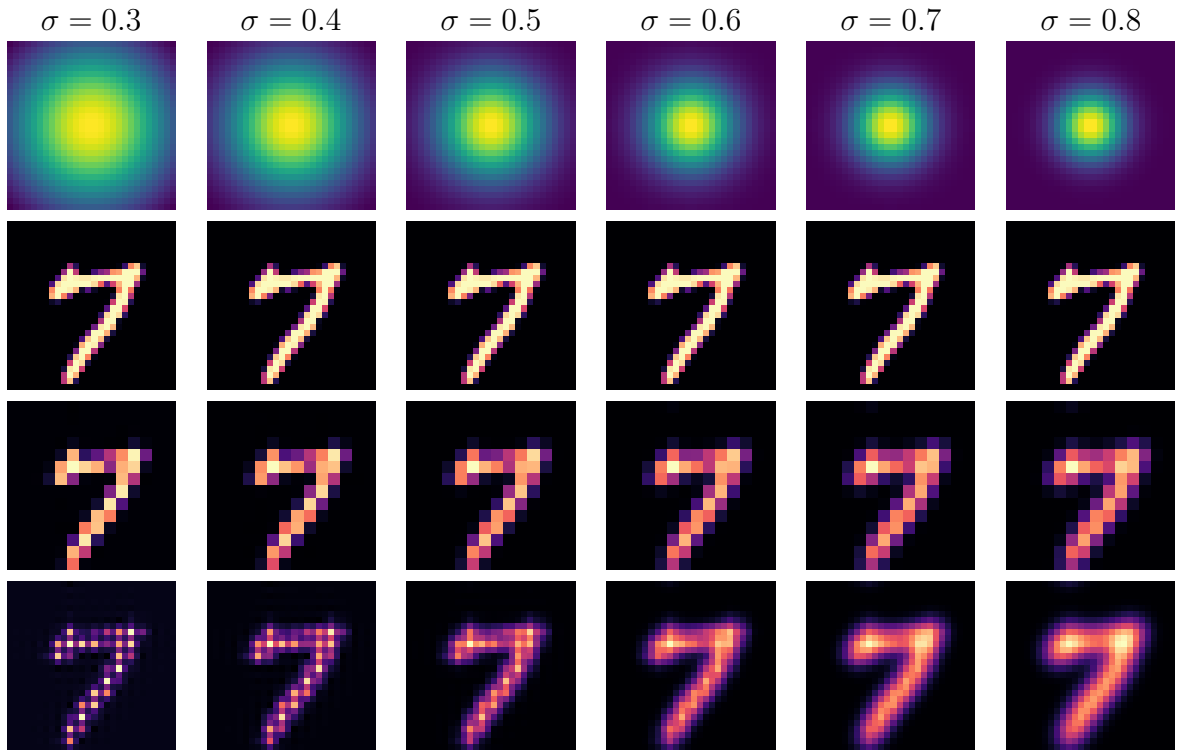

fig. 33 — En première ligne la transformée de filtres $\hat{\boldsymbol{h}}$ gaussien d'écart type σ, en deuxième l'image $\boldsymbol{x}$ avec en dessous le produits $A(\boldsymbol{x})$ puis $tA(A\boldsymbol{x})$.

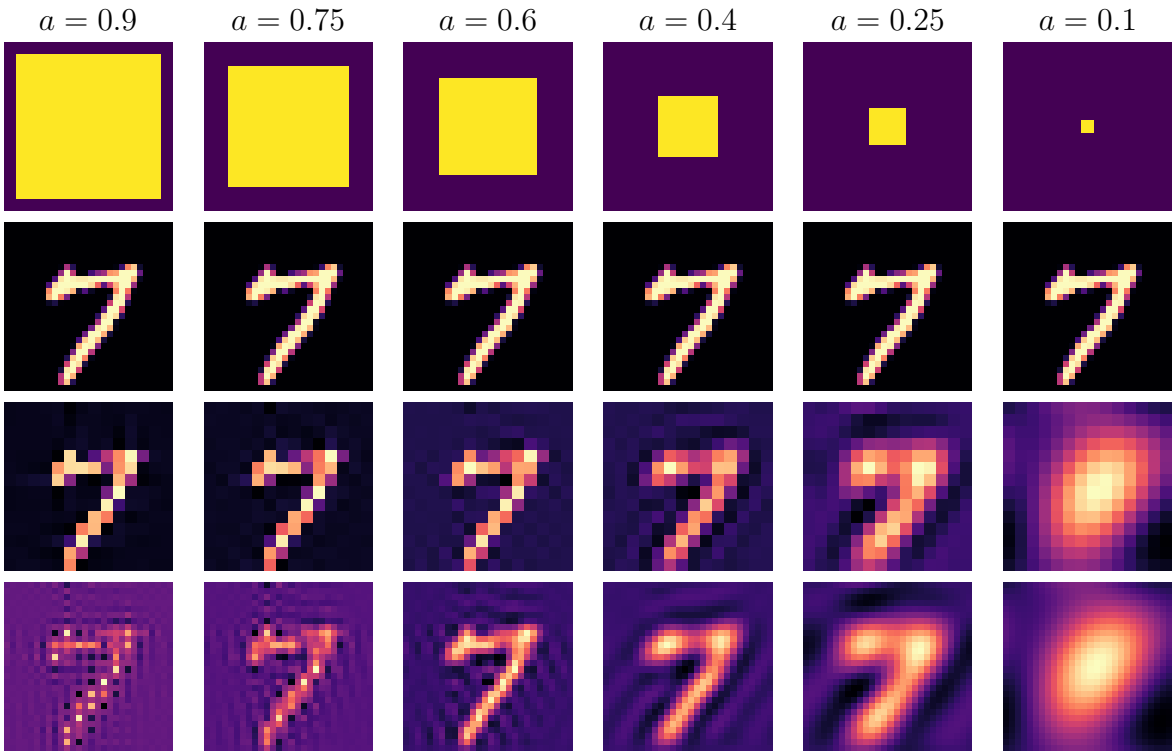

fig. 34 — Idem avec cette fois $\hat{\boldsymbol{h}}$ qui est l'indicatrice de $[-a, a] \times [-a, a]$

Table des Figures

Références

[1] P. Peng, S. Jalali, and X. Yuan, *Solving inverse problems via auto-encoders*, Dec. 2019. Issue: arXiv:1901.05045 arXiv:1901.05045 [cs, math].

[2] Y. Traonmilin, J.-F. Aujol, and A. Leclaire, *The basins of attraction of the global minimizers of non-convex inverse problems with low-dimensional models in infinite dimension*, Information and Inference, (2022). Publisher: Oxford University Press (OUP).