

RAPPORT DE STAGE

SUPER-RÉSOLUTION D'IMAGES VIA RÉSEAU AUTOENCODEUR

Grégoire DOAT

Sous la tutelle de M. Yann TRAONMILIN

Mai – Juin 2024

Introduction	1
I. Cadre théorique	2
1.1. Formalisation du problème	2
1.2. Cadre théorique de l'article	4
1.3. Les résultats de Traonmilin <i>et al.</i>	6
II. Descente de gradient depuis l'espace latent	9
2.1. Premier résultats, différentes initialisations	9
2.2. Variation de la qualité de mesure (e.i. $\mathbf{p} \times \mathbf{q}$)	12
2.3. Variation de la taille de l'espace latent.	14
III. Comparaison avec la descente de gradient projetée	20
3.1. Le principe et les résultats	20
3.2. Comparaison avec la LGD.	20
IV. Conclusion	22
Annexes	23
Annexe A Auto-encodeur	23
Annexe B Calcul pratique de \mathbf{F} et de son gradient	24

Introduction

Dans ce rapport de stage on s'intéresse au problème de super-résolution. C'est-à-dire la reconstruction d'image \mathbf{x} , disons de taille (n, m) , à partir d'une version sous-échantillonnée \mathbf{y} de taille (p, q) et se pose ainsi¹ :

Étant donnée une image $\mathbf{x}_0 \in \mathbb{R}^{n \times m}$ et sa version sous-échantillonnée $\mathbf{y}_0 \in \mathbb{R}^{p \times q}$ on cherche à minimiser la fonctionnelle :

$$F : \begin{array}{ccc} \mathbb{R}^{n \times m} & \longrightarrow & \mathbb{R} \\ \mathbf{x} & \longmapsto & \frac{1}{2} \|A\mathbf{x} - \mathbf{y}_0\|_2^2 \end{array} \quad (1)$$

que l'on suppose convexe et où A représente le sous-échantillonnage, aussi appelée 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, elle ne conserve pas toute l'information originale. Cela 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 \mathbf{x} telle que $\phi(\theta) = \mathbf{x}$ et à chercher les bons paramètres θ plutôt que \mathbf{x} . L'intérêt étant que si le nombre de paramètre 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 a toutes les chances d'impacter la convexité du problème. Ce pose donc la question du comportement de la fonctionnelle $F \circ \phi$ au voisinage de ses minimums.

C'est précisément ce qu'on étudie 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 [?]. 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 global de $F \circ \phi$. Puis applique 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 ses 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.

¹ce 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'applique 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 leur 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 d'un éventuel filtre passe-bas C_h pour éviter les problèmes d'aliasing. C_h est donc une convolution par un filtre que l'on va noter h (d'où la notation) de sorte que A s'écrive :

$$\forall \mathbf{x} \in \mathbb{R}^{n \times m}, \quad A\mathbf{x} = S(h * \mathbf{x}) = SC_h\mathbf{x}$$

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 \mathbf{x} , ce qui correspond à appliquer la matrice de la figure 1 au vecteur \mathbf{x} aplati.

Pour affectuer la convolution avec h , l'algorithme passe l'espace de fréquences, le changeant en simple produit et c'est dans cet espace que son ajuster pas les paramètres des deux filtres proposés :

Le filtre gaussien à pour paramètre l'écart-type σ de \hat{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 quelque exemple, plus de détail en annexes B.

PROBLÉMATIQUE — On considère $\mathbf{x}_0 \in \mathbb{R}^{n \times m}$ une image que l'on cherche à reconstruire et $\mathbf{y}_0 := A(\mathbf{x}_0)$ sa version sous-échantillonnée. Une bonne reconstruction de \mathbf{x}_0 étant un minimiseur :

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

$$\left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & & \\ 0 & 0 & 1 & 0 & & \\ \hline & & & & & \\ \hline & & & & & \end{array} \right) \begin{array}{c} \mathbb{O} \\ \\ \mathbb{O} \end{array} \begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}$$

fig. 1 — Opérateur S , point de vue matriciel

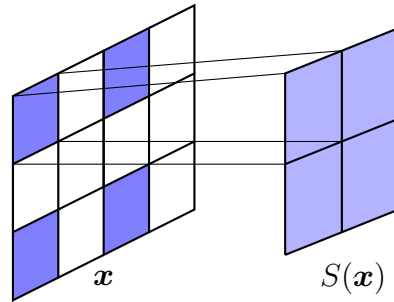


fig. 2 — Opérateur S , point de vue image

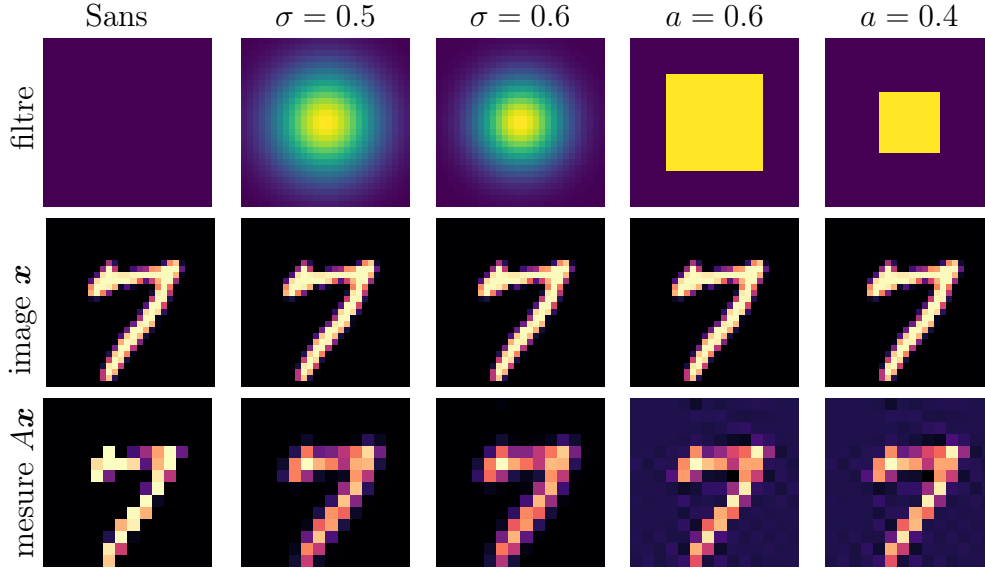


fig. 3 — Quelques exemples d’application de A à un même image en fonction des filtres.

Si pour le bien de l’étude \mathbf{x}_0 sera connu, il va de soit que le problème 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 $pq \ll nm$) 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 cachées de taille 1 500 pour les parties encodeurs et décodeurs et la seule choses qui les différencies est la taille de leur espace latent qui variant entre 100 et 800. Leur entraînement s’est fait sur le jeu donnés MNIST et c’est donc uniquement sur ces images que seront présenté les résultats. Plus de détail sur les performances des auto-encodeurs en annexe A.

TRANSITION ?

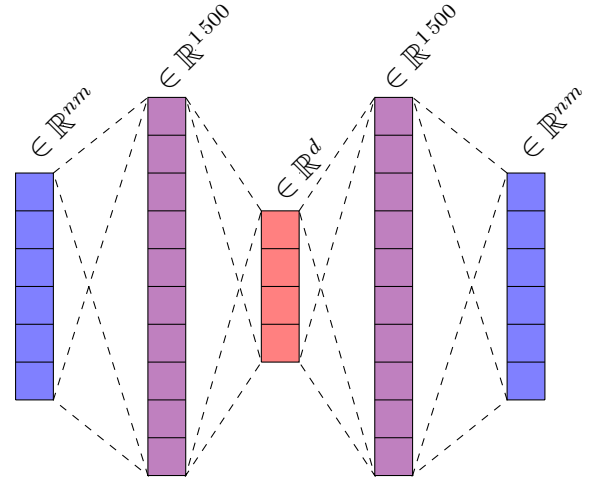


fig. 4 — Schéma des auto-encodeurs f avec $d = 100, 200, 400, 800$

1.2. Cadre théorique de l'article

Comme expliqué plus tôt, tout l'objet de l'article [?] 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 $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^{n \times m}$ une paramétrisation de ce dernier avec :

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

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 régularités 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 \mathcal{D} et les signaux (dans notre cas les images) sont pris dans \mathcal{D}^* , de sorte que la mesure par $\alpha \in \mathcal{D}$ de $x \in \mathcal{D}^*$ soit représenté 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 $g := Af_D$ sur Θ , ou dans notre cas :

$$\theta^* \in \operatorname{argmin}_{\theta \in \Theta} \frac{1}{2} \|Af_D(\theta) - \mathbf{y}_0\|_2^2$$

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

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

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

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

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

```

 $\theta_0$  : initialisation
 $x$  : image à reconstruire
 $\tau$  : pas de descente
 $N$  : nombre d'itération
 $\hat{x} \leftarrow f(x)$ 

for  $n \in \llbracket 1, N - 1 \rrbracket$  do
     $\theta_{n+1} \leftarrow \theta_n - \tau \nabla g(\theta_n)$ 
     $x_{n+1} \leftarrow f_D(\theta_{n+1})$ 
end for
return  $(x_n)_{0 \leq n \leq N}$ 

```

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

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 $\phi^{-1} \circ \phi(\{\theta^*\}) : \forall \theta \in \mathbb{R}^d :$

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

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

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

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 :

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

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 générale), que $\Lambda_\beta(\theta^*)$ une boule ouverte et que $p(\theta, \theta^*) = \theta^*$.

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 introduites les deux ensembles suivants :

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

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

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 :

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

Pour nous, $\phi = f_D$ est C^∞ , donc la dérivée directionnelle $\partial_u \phi(u)$ est équivalente à la différentiel 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 ceux, pour la simple

et bonne raison que ses fonctions d'activations sont des sigmoïdes y compris les celles sorties. 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 à été fait c'était à l'origine pour reprendre les codes de l'article [?] sur la descente projeté (voir section III.), mais en toute vraisemblance utiliser des fonctions ReLu ou autre ne devrait pas trop changer les résultats obtenus. La perte de régularité entraîné 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 globale 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 $\gamma \in]0, 1]$:

$$\forall v \in \overline{\mathcal{P}(\Sigma)}, \quad (1 - \gamma)\|v\|^2 \leq \|Av\|^2 \leq (1 + \gamma)\|v\|^2 \quad (3)$$

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

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

- Si la *technical assumption* 1 (donné plus bas) est vérifier 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 :

$$\forall z \in [\theta, \tilde{\theta}], \quad \frac{(1 - \gamma)\|\partial_{\tilde{\theta}-\theta}\phi(z)\|_2^2}{\sqrt{1 + \gamma}\|A\partial_{\tilde{\theta}-\theta}^2\phi(z)\|_2} \geq C\beta \quad (5)$$

Si A et ϕ vérifie 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 $\Lambda_{\beta_1}(\theta^*)$:

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

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

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

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 $\|\cdot\|_2$:

$$\forall \theta \in \Lambda_{2\beta}(\theta^*), \quad \|\theta - \theta^*\|_2 \leq Cd(\theta, \theta^*)$$

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

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

Comme dit plus haut, supposer $\phi = f_D$ inversible sur Θ impose $d(\theta, \theta^*) = \|\theta - \theta^*\|_2$. Ainsi la première inégalité est vérifiée pour tout $C \geq 1$ et les majorations uniformes ne sont pas un problème non plus puisque que 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 $|h_n| \xrightarrow[n \rightarrow +\infty]{} 0$, $\left\| \frac{\phi(\theta + |h_n|v) - \phi(\theta)}{|h_n|} \right\|_2$ converge indépendamment de $(|h_n|)$ et ceux, pour tout (θ, v) tel que $\partial_v \phi(\theta) \in \overline{\mathcal{S}(\Sigma)}$.

Dans l'article [?], la mesure de proximité des images est fait à travers une norme $\|\cdot\|_{\mathcal{H}}$ associé un espace hilbertien \mathcal{H} contenant Σ . Cette hypothèse permet alors de s'assurer que cette norme s'étende à $\overline{\mathcal{S}(\Sigma)}$. Mais encore, une fois, pour nous $\mathcal{H} = \mathbb{R}^{n \times m}$ 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 $(|h_n|)_n$.

Revenons maintenant sur les autres hypothèses du théorèmes. D'abord, dans la RIP (3), c'est surtout l'inégalité de gauche qui est important puisqu'elle garantit que A soit inversible sur $\overline{\mathcal{S}(\Sigma)}$. 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 tout les $\mathbf{x} - \mathbf{y}$ avec \mathbf{x}, \mathbf{y} du jeu de données (MNIST donc). Pour se faire, on les équivalences :

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

En calculant le rapport $\left| \frac{\|Av\|_2^2}{\|v\|_2^2} - 1 \right|$, 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 donne les histogrammes 6 et 7 :

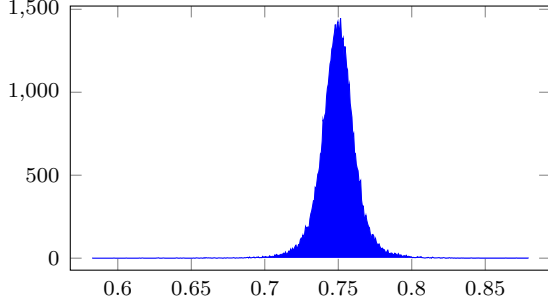


fig. 6 — Histogramme des valeurs de $\left| \frac{\|Av\|_2^2}{\|v\|_2^2} - 1 \right|$ sans filtre passe-bas sur le set de test

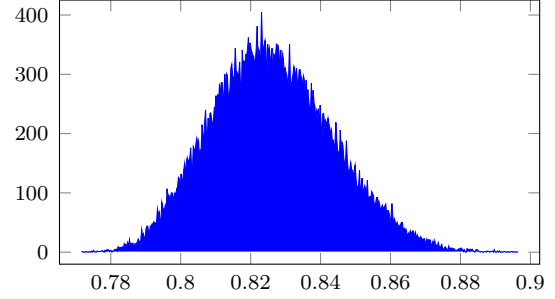


fig. 7 — Histogramme des valeurs de $\left| \frac{\|Av\|_2^2}{\|v\|_2^2} - 1 \right|$ avec filtre gaussien ($\sigma = 0,6$) sur le set de test

On y voit que dans les deux cas, on peut espérer que γ soit au mieux au alentour 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 $\phi(\Lambda_{2\beta}(\theta^*)) \subset \Sigma$, 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 remplacé par des bassins de la forme $\Lambda_{\beta+\varepsilon}$ et pour cette raison, l'hypothèse (4) sera supposer vraie pour ε suffisamment petit.

Il reste une dernière grosse inconnue, à savoir l'infimum :
$$\inf_{\theta \in \Lambda_{\beta_1}(\theta^*)} \inf_{z \in [\theta, \tilde{\theta}]} \frac{\|\partial_{\tilde{\theta}-\theta} \phi(z)\|_2^2}{\|A \partial_{\tilde{\theta}-\theta}^2 \phi(z)\|_2}$$

Comme l'auto-encodeur utilisé est composé de sigmoïde, je sais pas quoi dire

II. Descente de gradient depuis l'espace latent

À partir de maintenant, les vecteurs de l'espace latent seront noté \mathbf{u} et en particulier (\mathbf{u}_n) sera la suite obtenue par descente gradient d'initialisation \mathbf{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 fixe² 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 décodé $f_D(\mathbf{u}_0)$, 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 $f_E(\mathbf{u}_n)$ à l'itération et l'image cible.

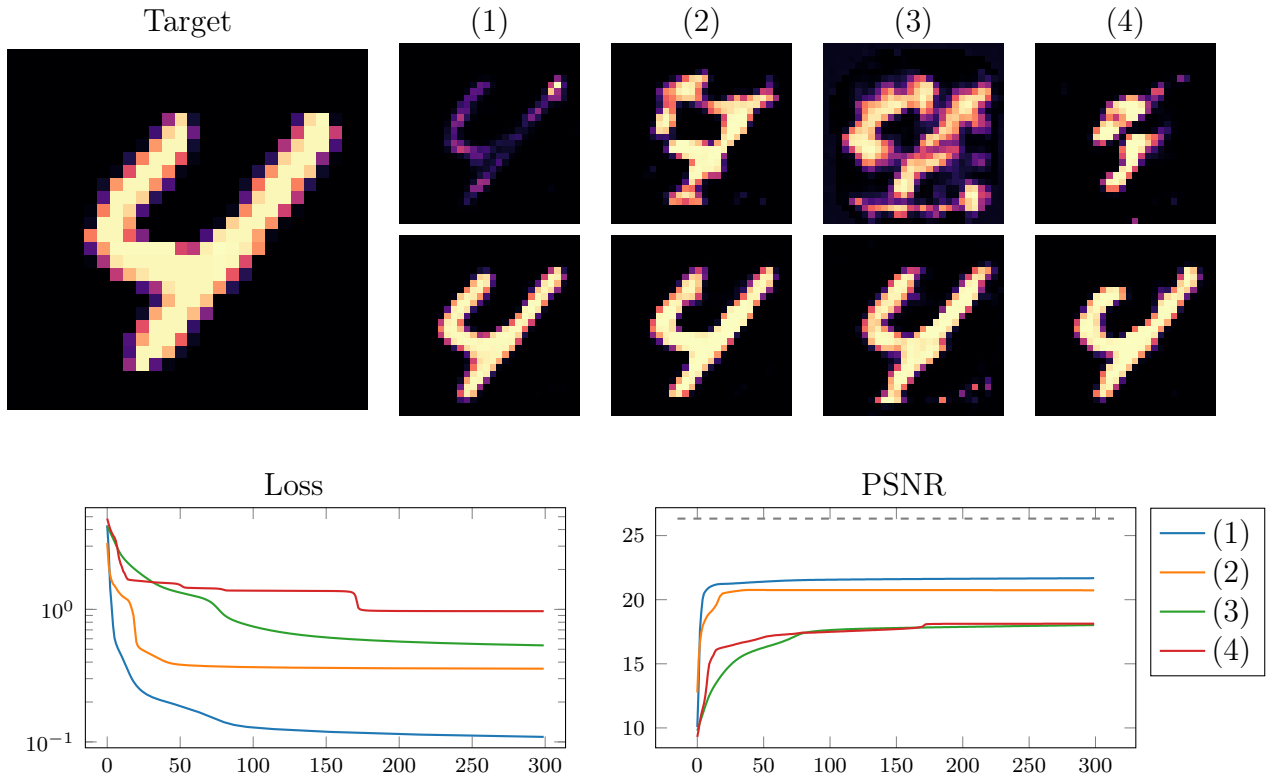


fig. 8 — En première lignes différentes initialisation avec en dessous le résultats de la LGD après 300 itérations – sans filtre passe-bas

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

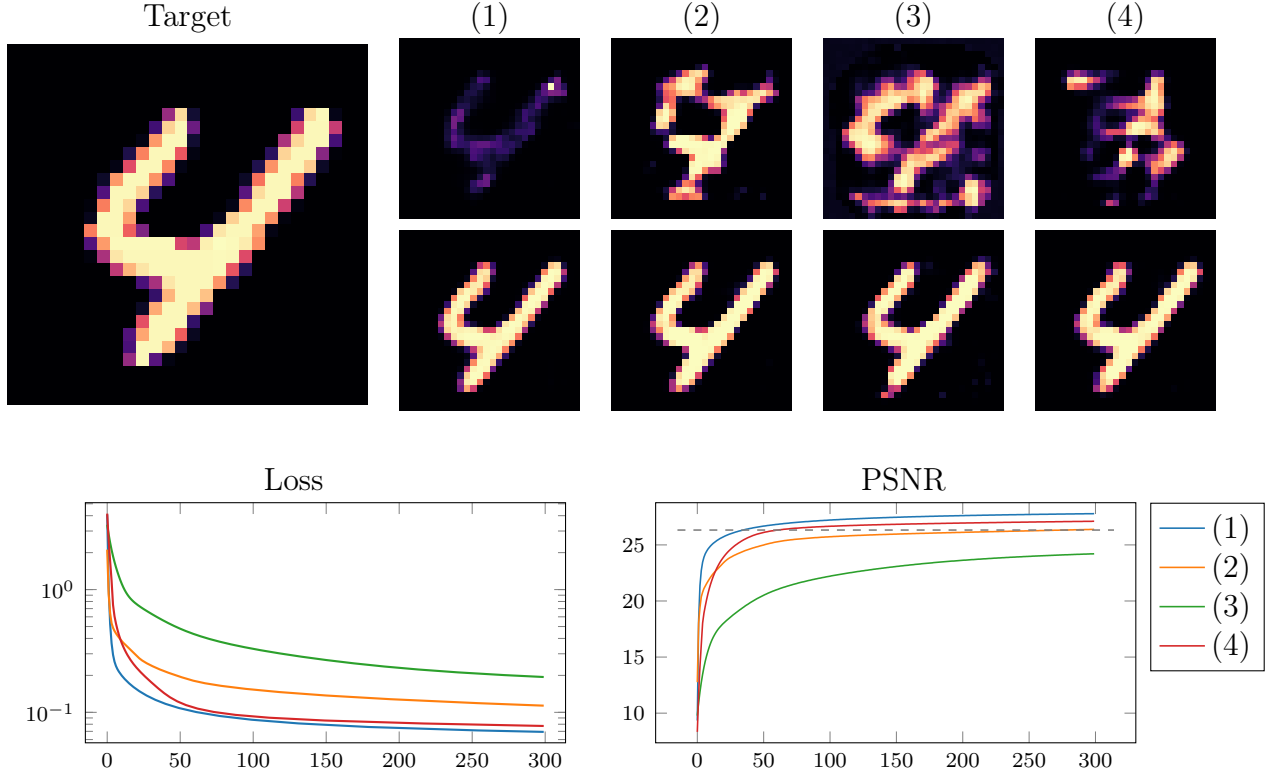


fig. 9 — En première lignes différentes initialisation avec en dessous le résultats de la LGD après 300 itérations – avec passe-bas gaussien ($\sigma = 0.5$)

Il va de soit que le calcul de PSNR est à but purement indicatif et n'intervient pas dans la LGD elle-même. Les ligne en pointillé représentent $\text{PSNR}(\mathbf{x}_0, f(\mathbf{x}_0))$, le PSNR entre la cible \mathbf{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_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) $\mathbf{u}_0 = f_E({}^t A \mathbf{y}_0)$, la backprojection encodée de \mathbf{y}_0
- (2) $\mathbf{u}_0 = f_E(\mathbf{x}_0) + e$, l'image cible encodée puis bruitée (bruit uniforme sur $] - 0.5, 0.5[$)
- (3) idem, avec un bruit gaussien d'écart-type 0.5
- (4) $\mathbf{u}_0 = \mathbf{u}_{\text{rand}}$, vecteur aléatoire de l'espace latent

La première initialisation est la plus naturelle puisqu'elle utilise ${}^t A$, ce qui se rapproche le plus d'un inverse pour A (voir figure 30 et 30 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 de la cible pour voir à quel point il possible de s'éloigner de \mathbf{x}_0 tout en conservant la convergence de l'algorithme.

Cela étant dit, le premier constat est que la méthode fonctionne et ceux même en partant d'une vecteur aléatoire. Aussi, même s'il sont meilleurs avec un passe-bas, les résultats restent satisfaisant 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 suggèrent que le résultats pourrait être amélioré avec plus d'itération. Celles de la figure 8 suggère une convergence mais comme les courbes fond des saut, il possible que g puisse encore diminuer en échappant à ces plateaux. On pourrait par exemple ajouter de l'inertie à la descente.

Autre chose remarquable, le résultats (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 \mathbf{u}_{rand} . Pour rester cohérent avec la structure des images, et le réseau f (qui est se termine par des sigmoïdes), les coefficients de \mathbf{u}_{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 $[0, 1]$.

Pour le voir, dans la figure 11 ci-dessous, les deux premières initialisations, (1) et (2), sont faites avec un vecteur aléatoires tirés suivant une loi uniforme. Les deux suivantes, (3) et (4), le sont suivant des lois normales d'écart-type respectif 1 et 5. Il apparaît clairement que l'auto-encodeur se comporte mal avec des initialisations trop loin de $[0, 1]$. *blablabla il te faut les résultats pour avancer là*

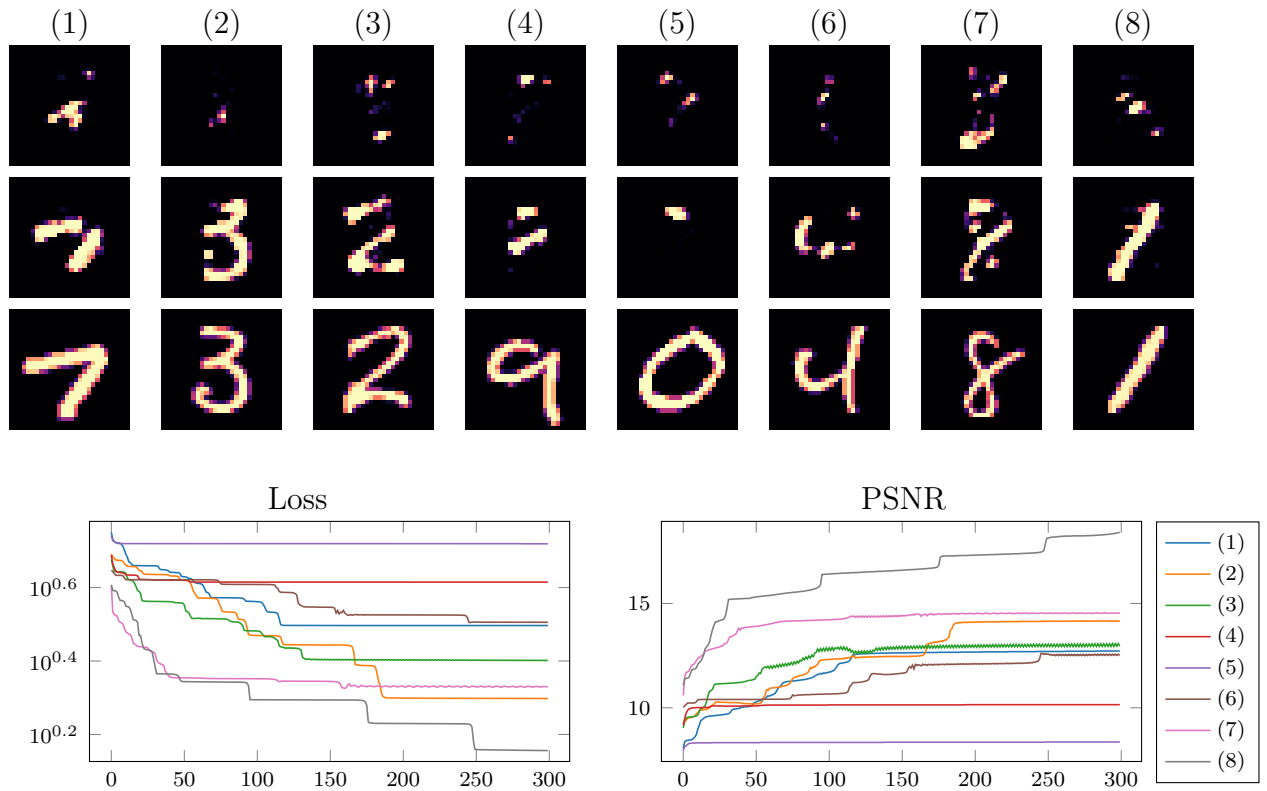


fig. 10 — Plein de descentes — sans passe-bas

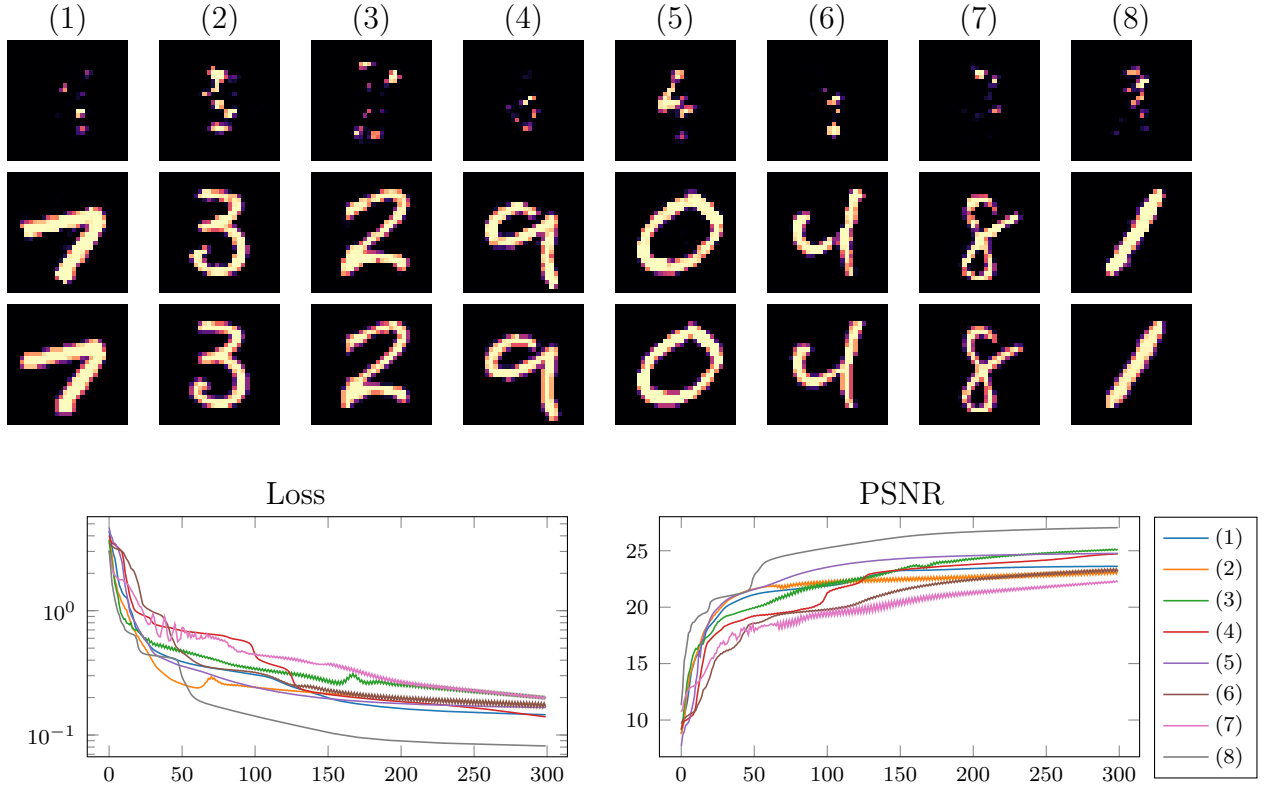


fig. 11 — Plein de descentes — avec passe-bas gaussien ($\sigma = 0.6$)

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 ajusté 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. ??

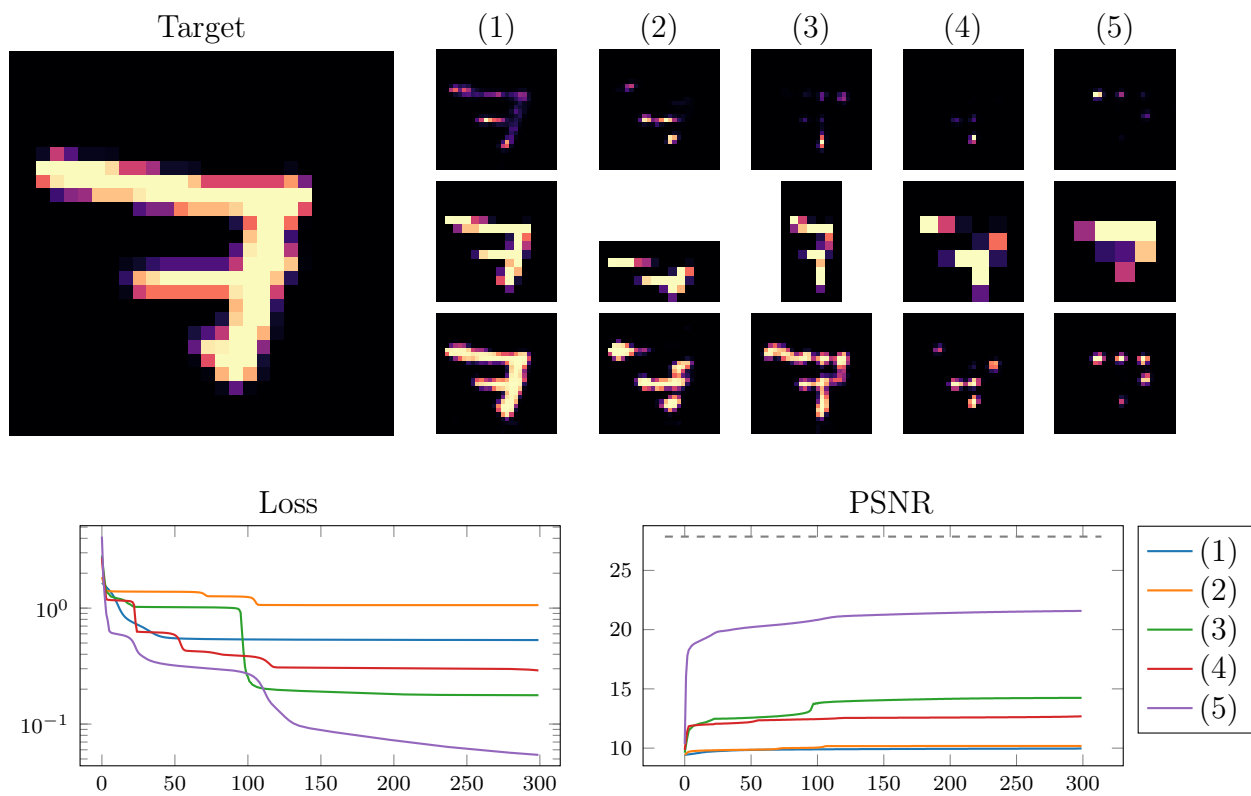


fig. 12 — Resultats de la LGD avec différents niveau de compression — sans passe-bas

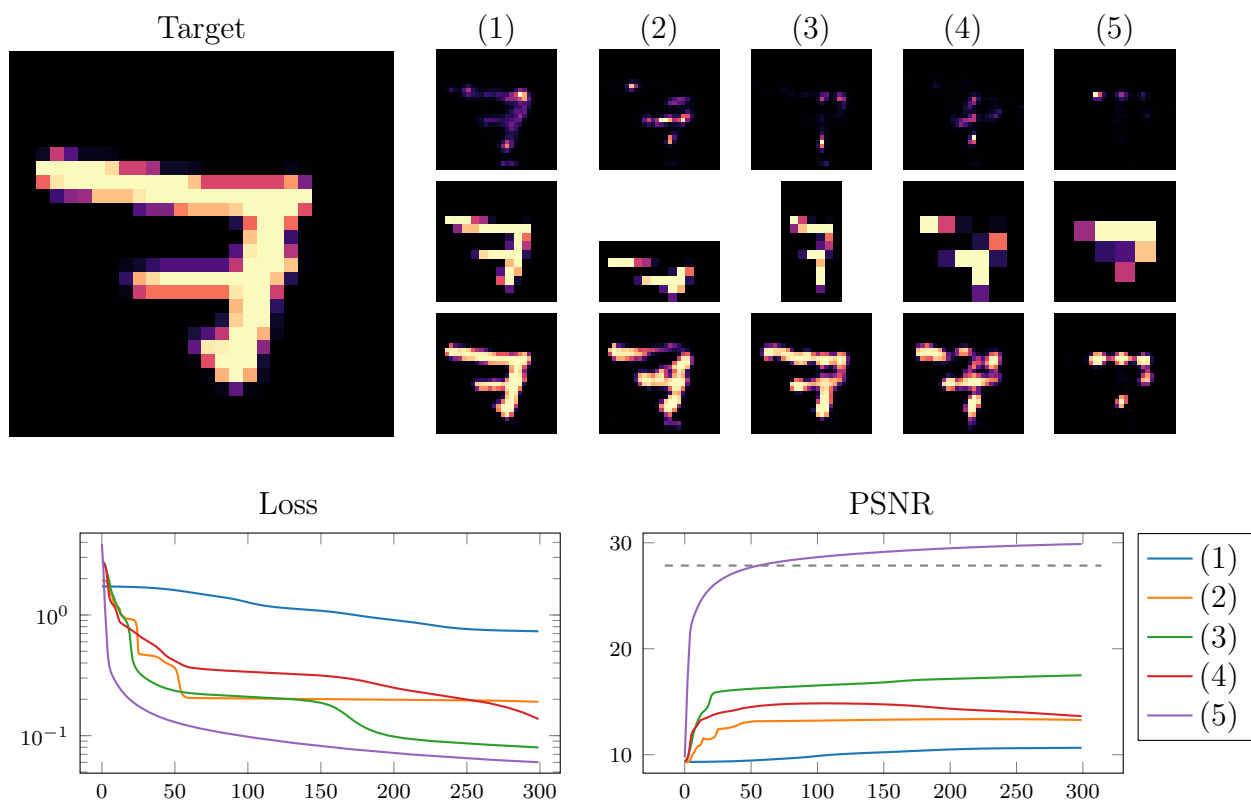


fig. 13 — Idem, cette fois avec un passe-bas gaussien ($\sigma = 0.5$)

2.3. Variation de la taille de l'espace latent

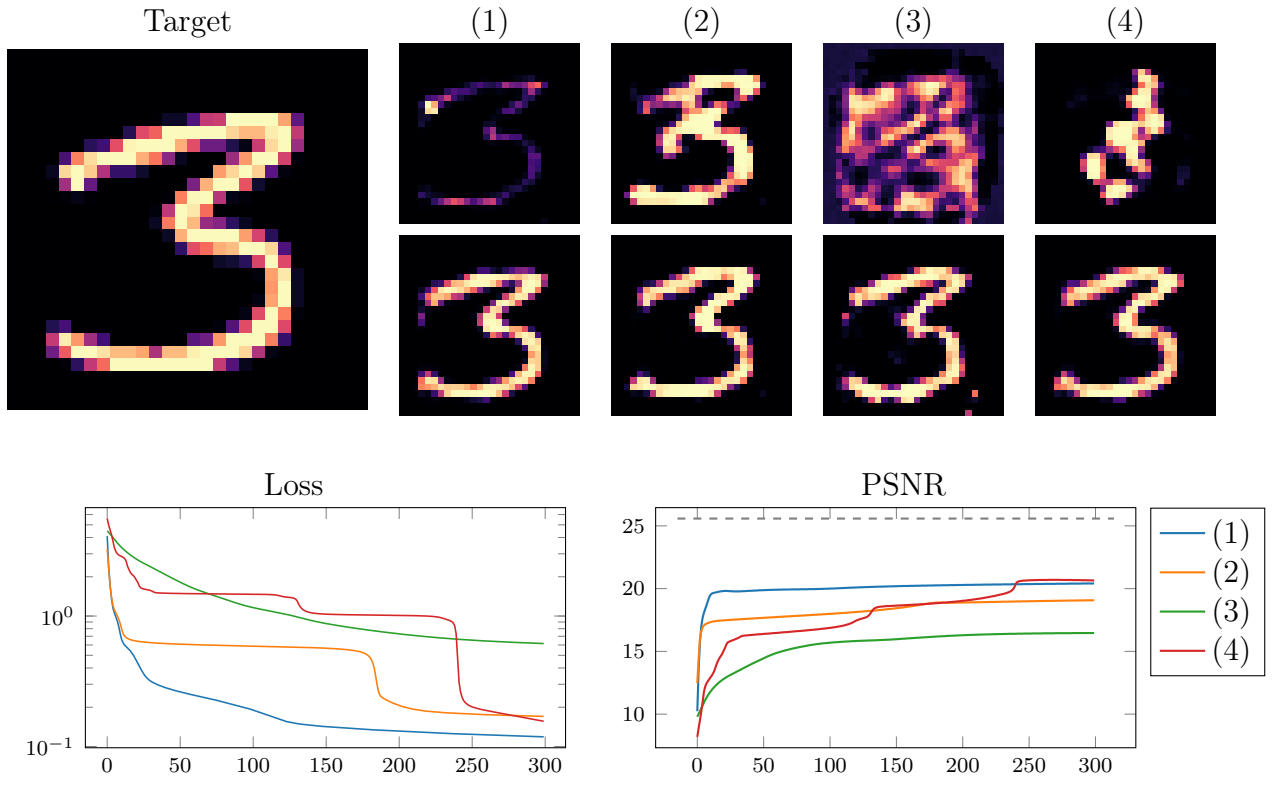


fig. 14 — $d = 100$, sans passe-base

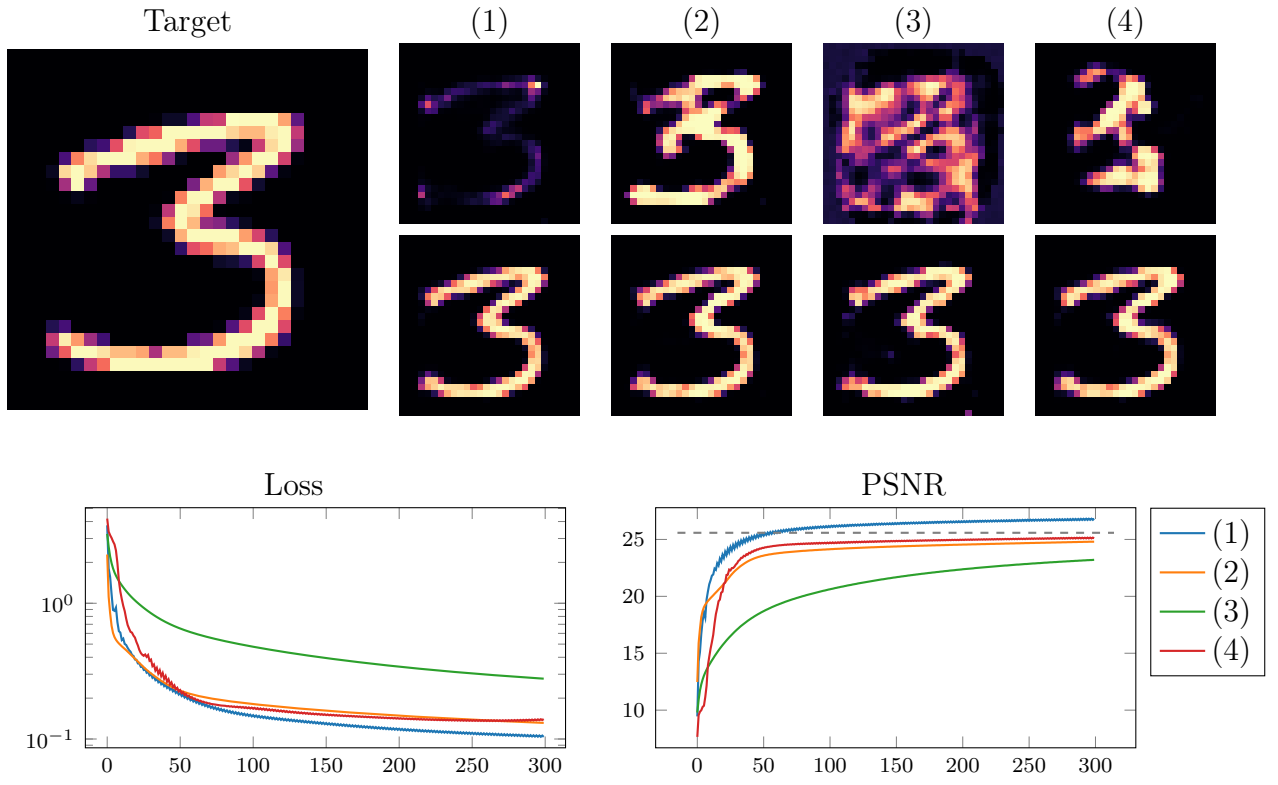


fig. 15 — $d = 100$, avec passe-base

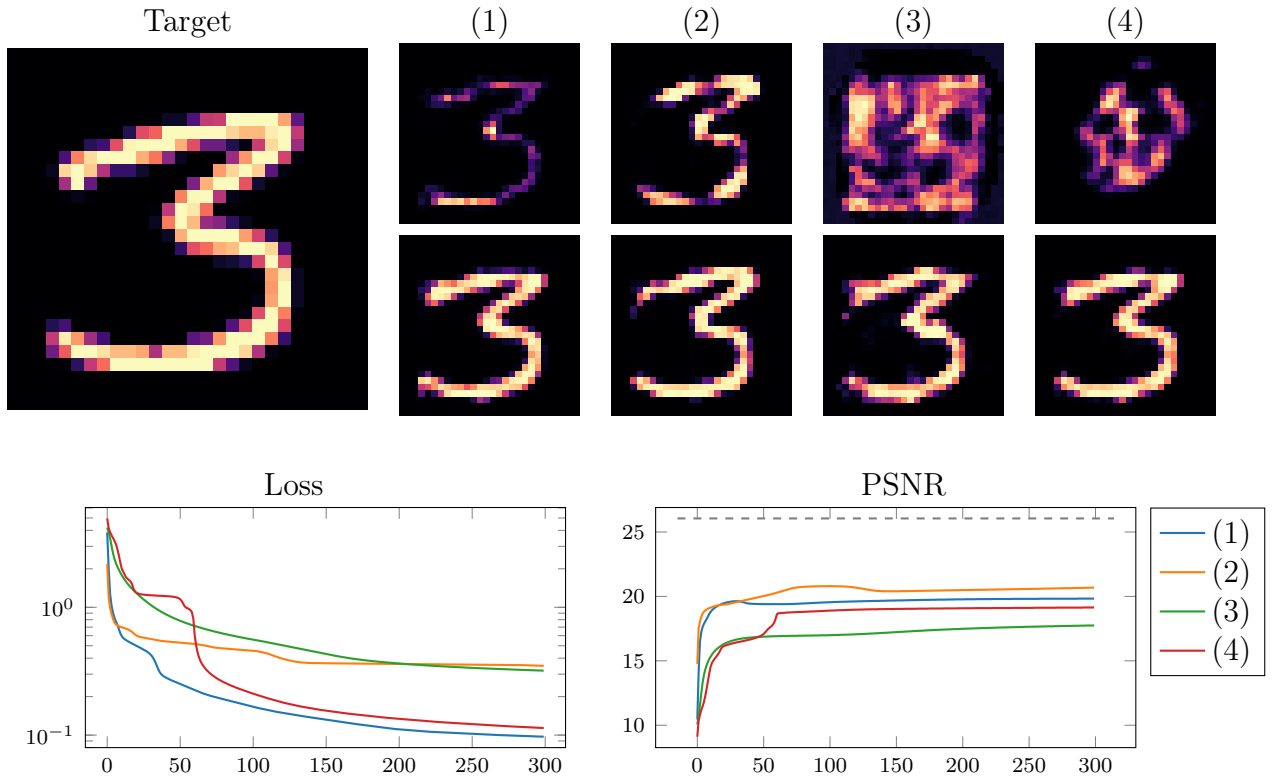


fig. 16 — $d = 200$, sans passe-base

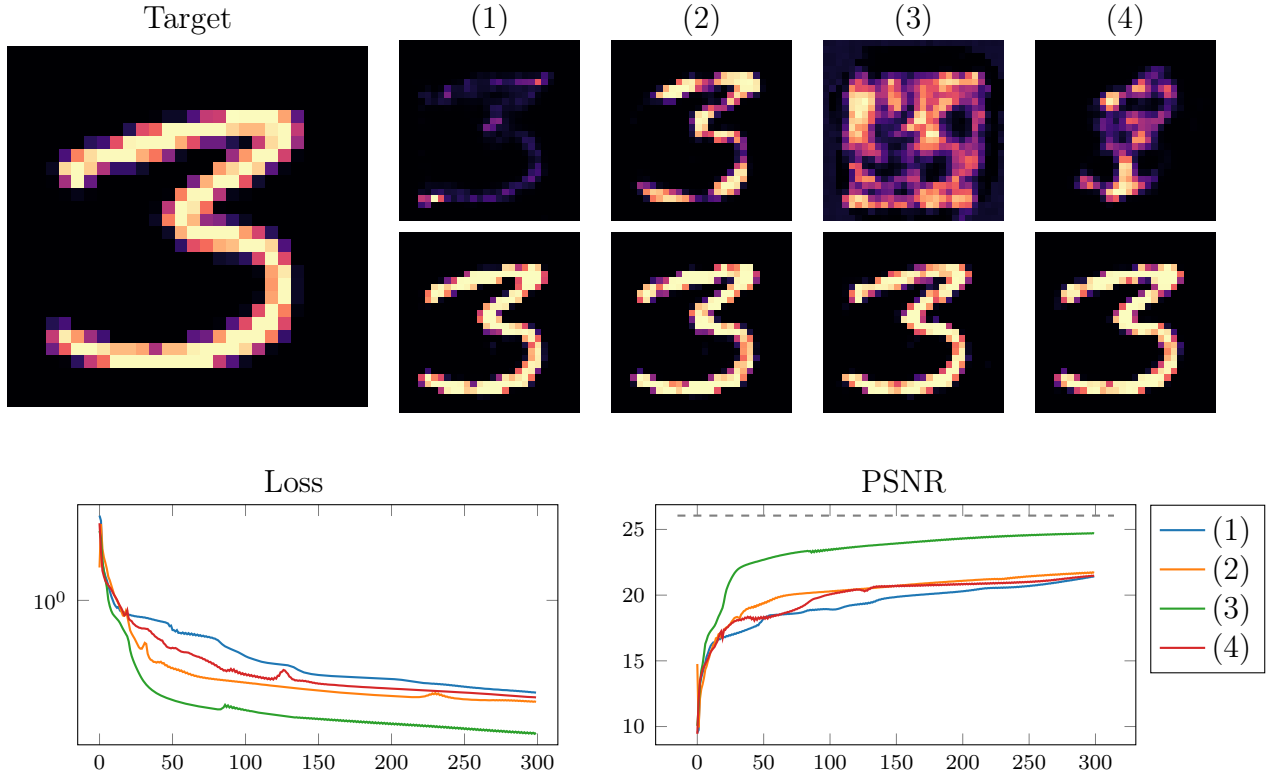


fig. 17 — $d = 200$, avec passe-base

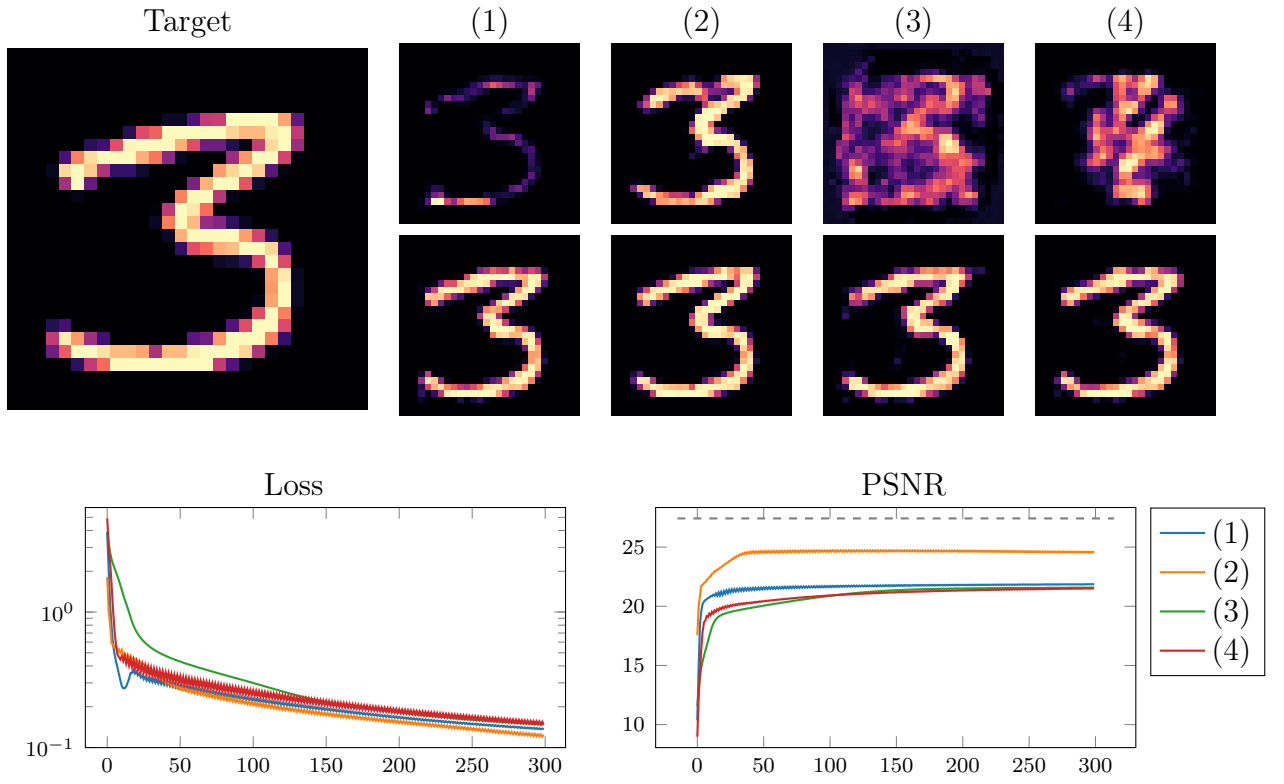


fig. 18 — $d = 400$, sans passe-base

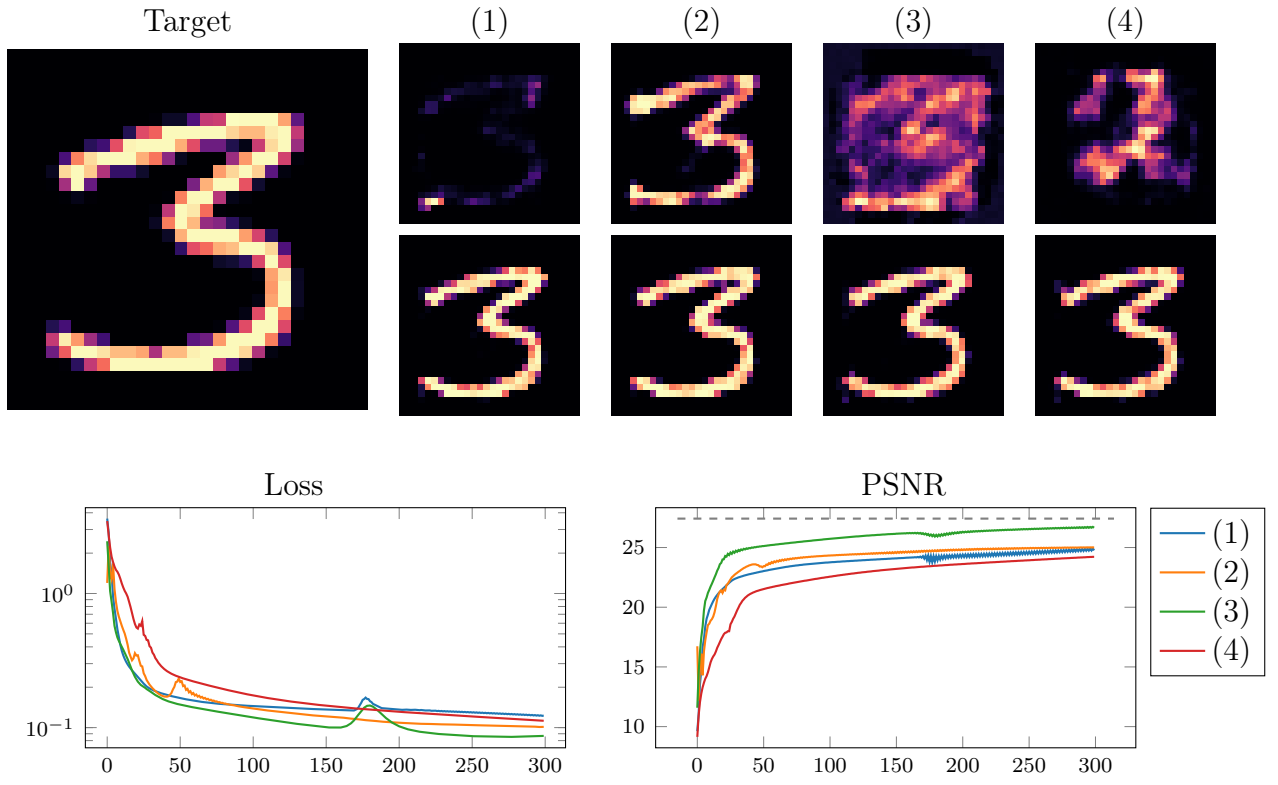


fig. 19 — $d = 400$, avec passe-base

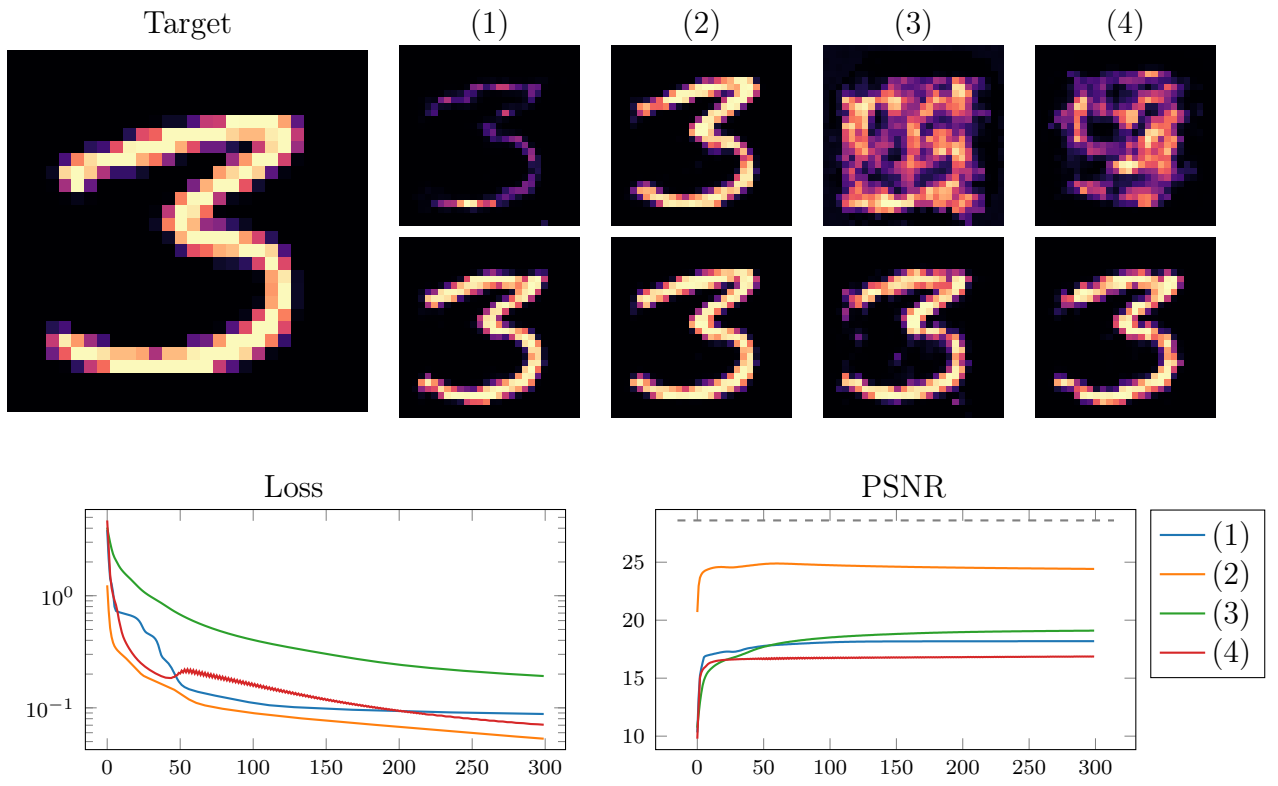


fig. 20 — $d = 800$, avec passe-base

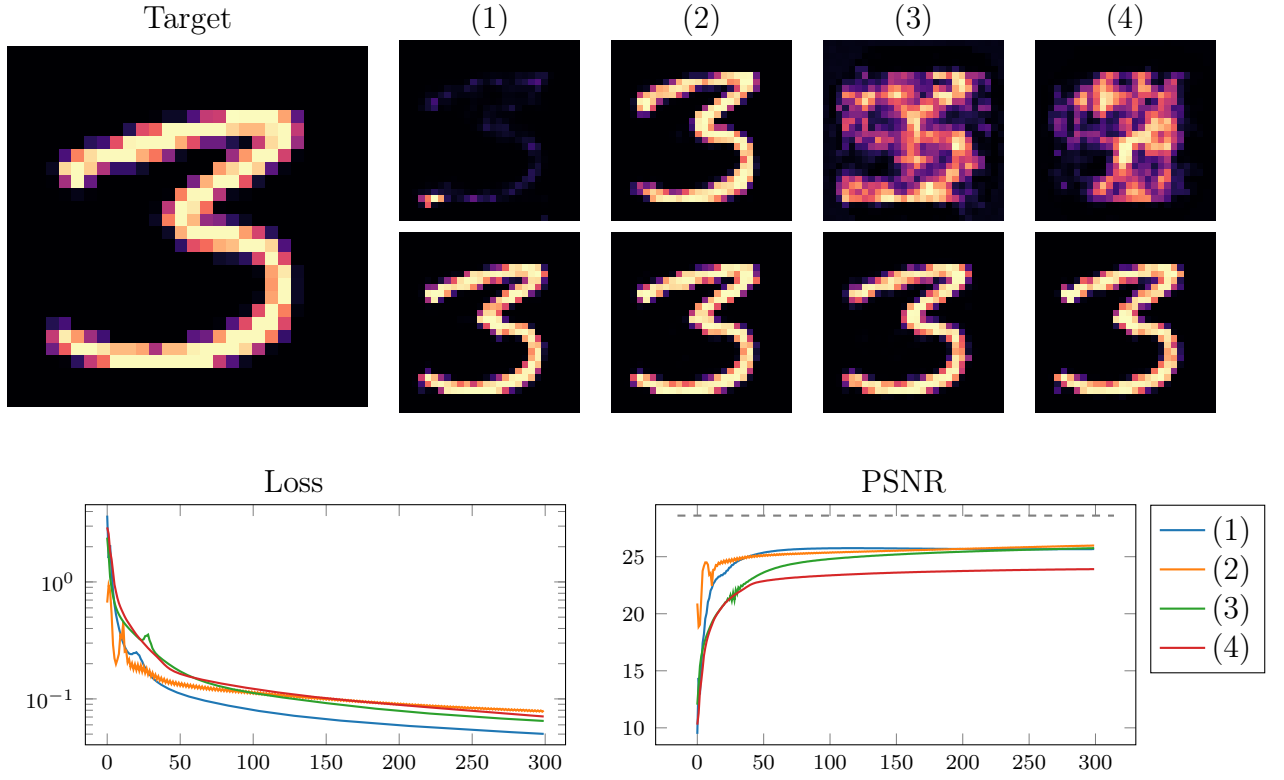


fig. 21 — $d = 800$, sans passe-base

À noter que sur la loss de la figure *fig. 21* les algorithmes stagnent ce qui sous entend que l'on se trouve dans des vallées (ou des crêtes). Problème qui peut être évité en ajoutant de l'inertie à la descente avec des méthodes type heavy-ball ou Nesterov.

III. Comparaison avec la descente de gradient projetée

3.1. Le principe et les résultats

La descente de gradient projeté par auto-encodeur était sensé faire office d’algorithme de contrôle. Comme son nom l’indique, la descente de gradient projeté (PGD) consiste en une descente de gradient classique à cela près qu’à itération, le vecteur obtenu est projeté par une fonction f (voir *fig. 22* ci-contre).

Dans notre cas, f est un auto-encodeur dont on notera f_E et f_D les parties encodage et décodage respectivement, de sorte que :

$$f = f_D \circ f_E$$

Cette méthode est tirée de l’article [?] de P. Peng, S. Jalali and X. Yuan, et leurs résultats étaient satisfaisant. Pourtant comme on va le voir dans la section 3.2. plus bas, nous n’avons pas réussi à les reproduire. Cela étant dit, nous allons malgré tout détailler l’algorithme et essayer de comprendre les résultats obtenus.

Ce rapport étant surtout porté sur la descente de gradient depuis l’espace latent de f , 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 le bon argmin. C’est là que composer par f peut aider à diriger la descente vers le “bon” minimum.

- Converge beaucoup plus vite
- Sensible à l’excès d’itérations (sûrement parce que f n’est pas une vraie identité, c’est un problème classique)
- Très sensible au pas image-wise
- Robuste à l’initialisation

3.2. Comparaison avec la LGD

fig. 23 — Résultat de la PGD avec un pas de descente $\rho = 10^{-10}$, sans passe-bas

```
 $x_0$  : initialisation
 $x$  : image à reconstruire
 $\rho$  : pas de descente
 $N$  : nombre d’itération
 $y \leftarrow A(x)$ 

for  $n \in \llbracket 1, N - 1 \rrbracket$  do
     $\nabla F(x_n) \leftarrow {}^t A(Ax_n - y_0)$ 
     $x_{n+1} \leftarrow f(x_n - \rho \nabla F(x_n))$ 
end for
return  $(x_n)_{0 \leq n \leq N}$ 
```

fig. 22 — Algorithme de PGD

fig. 24 — Résultat de la PGD avec un pas de descente $\rho = 10^{-10}$, passe-bas gaussien ($\sigma = 0.6$)

fig. 25 — Comparaison de la PGD avec le résultat attendu (Target) après 100 itérations et avec différent pas — initialisation ${}^tA(Ax)$, sans passe-bas

fig. 26 — Comparaison de la PGD avec le résultat attendu (Target) après 100 itérations et avec différent pas — initialisation ${}^tA(Ax)$, passe-bas gaussien ($\sigma = 0.6$)

IV. Conclusion

Par manque de temps, il n'a pas été trop possible de trop rentrer dans la théorie mais au vue des résultats il est clair que qu'il y a quelque chose à voir.

Il a été fait le choix de de supposer f_D inversible pour exprimer le fait que f_D vienne qu'un auto-encodeur. Mais peut-être qu'une formalisation moins dur, pas exemple en terme de voisinage, de cette idée pourrait être moins restrictive. En particulier, avoir $d(\theta, \theta^*) = \|\theta - \theta^*\|_2$ plutôt qu'une égalité permettrait d'élargir le bassin Λ_β , ce qui serait plus cohérent avec les résultats

A voir sur des images plus complexes mais *a priori*, les modèles très résistant à l'aliasing

Il faudrait faire marcher le pas adaptatif voir ajouter de l'inertie pour éviter les cas les plateaux qui arrive de temps à autre.

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.*, tout les fonctions d'activations sont des sigmoïdes, ce qui, retrospectivement, 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 :

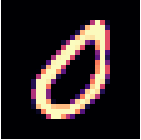
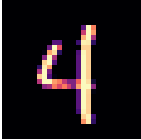

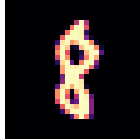
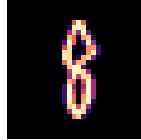
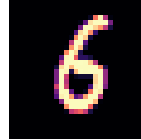
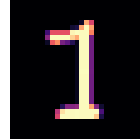
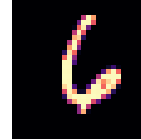
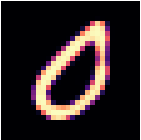
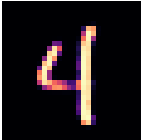

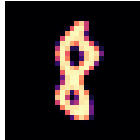
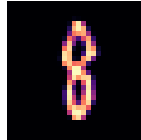
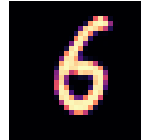
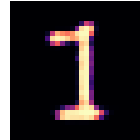
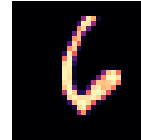
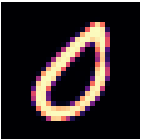
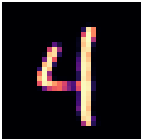

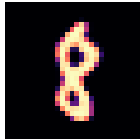
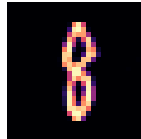
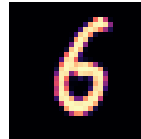
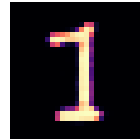
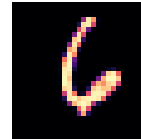
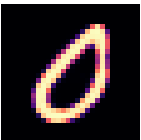
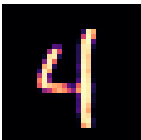

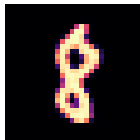
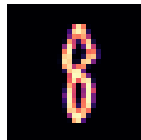
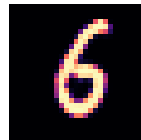

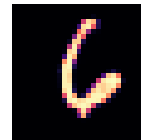
Input								
	PSNR=28.74	PSNR=29.68	PSNR=26.2	PSNR=29.65	PSNR=26.76	PSNR=28.69	PSNR=28.41	PSNR=27.76
100								
	PSNR=29.64	PSNR=29.24	PSNR=27.29	PSNR=30.21	PSNR=28.16	PSNR=29.71	PSNR=30.85	PSNR=29.12
200								
	PSNR=30.28	PSNR=30.76	PSNR=27.43	PSNR=30.53	PSNR=29.54	PSNR=29.83	PSNR=31.53	PSNR=31.3
400								

fig. 27 — Reconstruction par auto-encodeur des images Input en fonction de la taille de l'espace latent du dit auto-encodeur et avec les PSNR associés

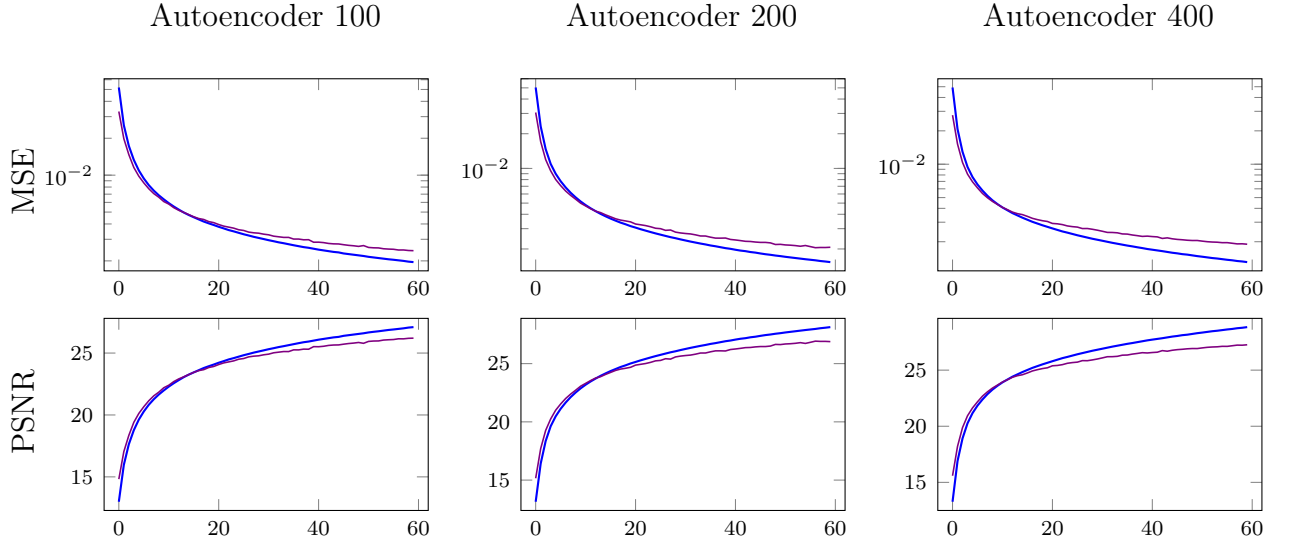


fig. 28 — Évolution de la MSE et du PSNR par batch des auto-encodeurs au cours des epochs. En bleu sur le set d'entraînement et en violet sur celui de test

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 $\mathbf{x} \in \mathbb{R}^{n \times m}$, sa représentation matricielle elle prendre un vecteur $x \in \mathbb{R}^{nm}$. Par souci 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 calcul explicitement le gradient de F à moindre coût :

$$\forall x \in \mathbb{R}^{nm}, \quad \nabla F(x) = {}^t A(Ax - y) \quad (6)$$

Si la formule (6) est mathématiquement très simple, dès que n et m seront un peu trop grand, 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 \mathcal{F} (resp. \mathcal{F}^{-1}) la matrice associée à la transformée (resp. inverse) de Fourier discrète. En notant de plus $\hat{\mathbf{x}} = \mathcal{F}\mathbf{x}$ et \odot le produit terme à terme de vecteur/matrice, A s'écrit alors :

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

Notons D_h l'application/matrice associée au produit $\hat{h} \odot \cdot$. Comme la transformée de

Fourier discrète et D_h sont symétriques³, le gradient de F se réécrit comme :

$$\begin{aligned} \forall x \in \mathbb{R}^{nm}, \quad \nabla F(x) &= {}^t A(Ax - y) = {}^t (S\mathcal{F}^{-1}D_h\mathcal{F})(S\mathcal{F}^{-1}D_h\mathcal{F}x - y) \\ &= {}^t \mathcal{F} {}^t D_h {}^t \mathcal{F}^{-1} {}^t S(S\mathcal{F}^{-1}D_h\mathcal{F}x - y) \\ &= \mathcal{F} D_h \mathcal{F}^{-1} {}^t S(S\mathcal{F}^{-1}D_h\mathcal{F}x - y) \end{aligned}$$

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 :

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

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

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

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

$$\forall (i, j) \in \llbracket 1, n \rrbracket \times \llbracket 1, m \rrbracket, \quad \hat{\mathbf{x}}^{kl} = \mathcal{F}_{ij}^{kl} \mathbf{x}^{ij} = \mathcal{F}_{ij}^{lk} \mathbf{x}^{ij} = \hat{\mathbf{x}}^{lk}$$

Enfin, comme son nom l'indique, il est plus naturel de construire la filtre *passé-bas* directement dans l'espace de Fourier, et ce qui est dans le code. Le figure 30 et 31 montre l'application de A et ${}^t A$ à une image. La figure 31 en particulier mais bien en évidence l'effet de Gibbs qui apparaît parce que les fréquences sont trop brutalement coupées.

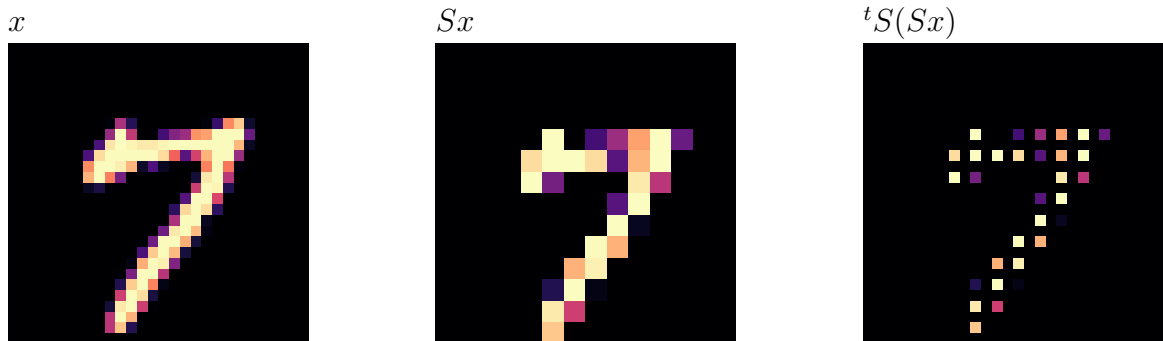


fig. 29 — Application successive de S et ${}^t S$ à une image

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

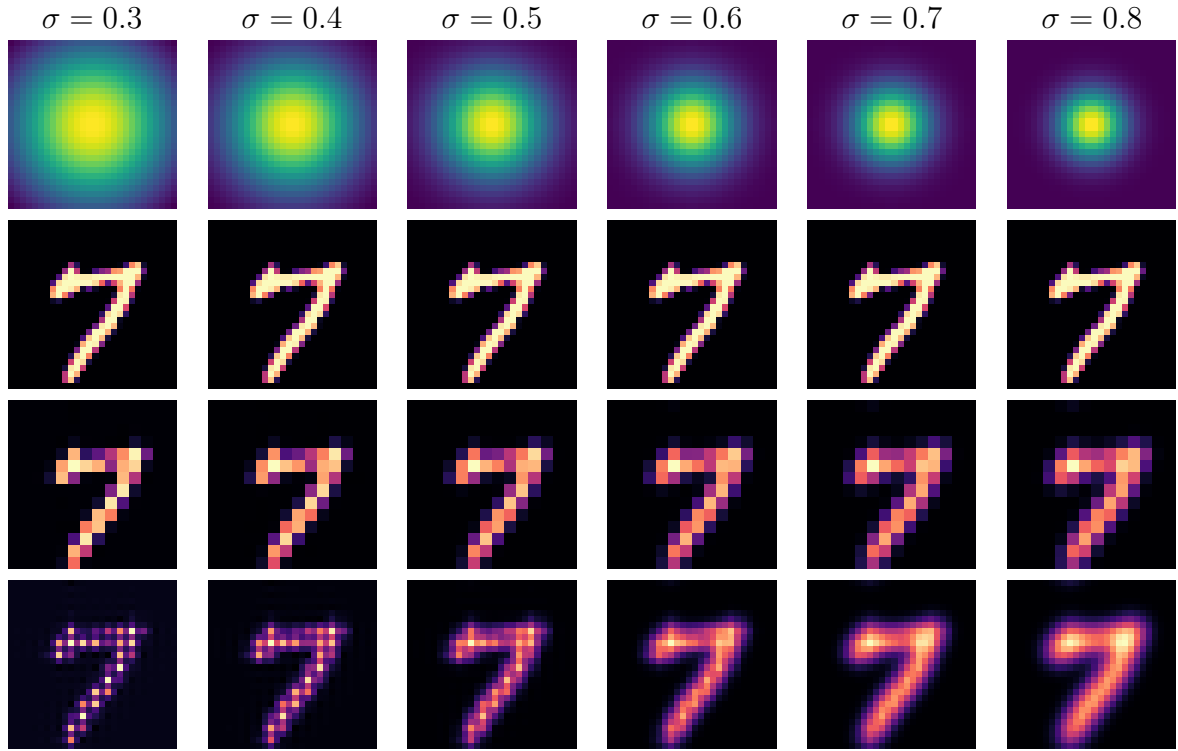


fig. 30 — En première ligne la transformée du filtre \hat{h} gaussien d'écart type σ , en deuxième l'image \mathbf{x} avec en dessous le produit $A(\mathbf{x})$ puis $tA(A\mathbf{x})$.

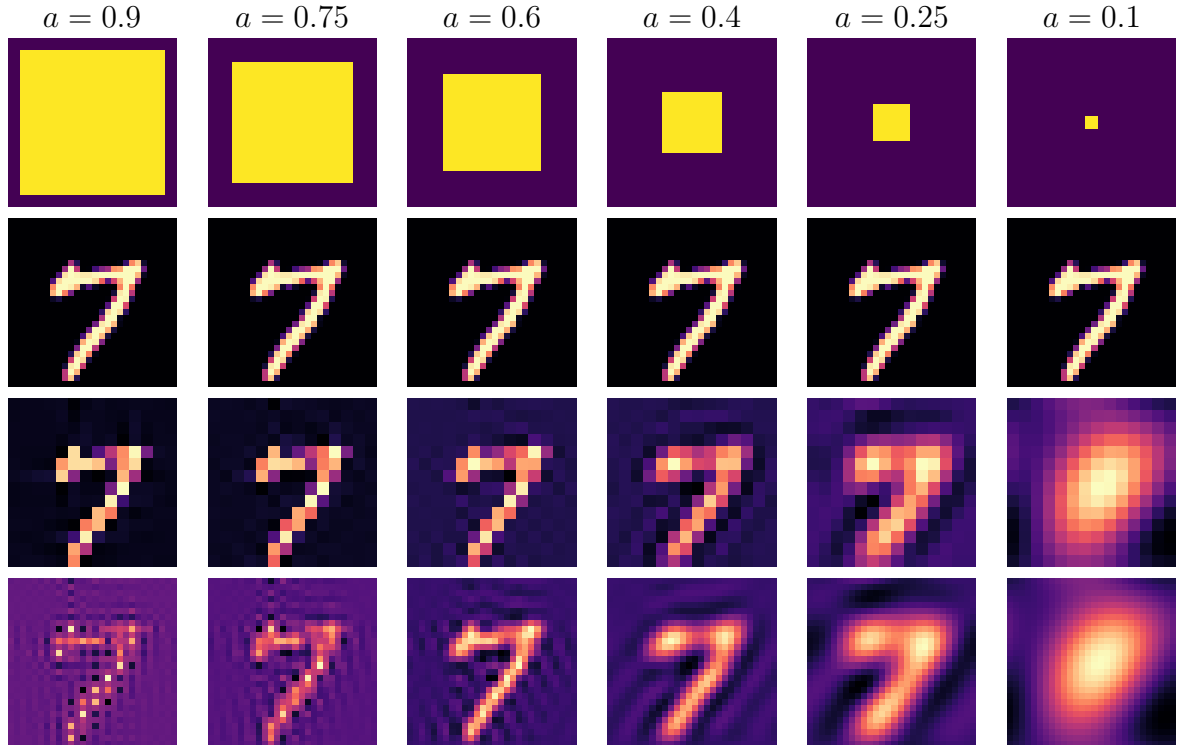


fig. 31 — Idem qu'à la figure 30 avec cette fois \hat{h} qui est l'indicatrice de $[-a, a] \times [-a, a]$

RÉFÉRENCES