

# What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision ?

Clément Bonet

ENS Paris-Saclay

Mars 2020

## 1 Introduction

Un problème assez important en apprentissage est de pouvoir quantifier l'incertitude. En effet, généralement, les modèles renvoient une prédiction sans indicateur sur l'incertitude de celles-ci. Cela peut créer des problèmes comme souligné dans [1] avec l'exemple du modèle de classification d'images qui a confondu des afro-américains avec des gorilles ce qui a soulevé des questions de discrimination. Cela aurait pu être évité si le modèle avait pu quantifier son incertitude, et ainsi éventuellement ne rien prédire si celle-ci est trop grande.

Pour la classification par exemple, on aurait pu penser à utiliser les résultats renvoyées par la softmax pour quantifier l'incertitude liée à la prédiction, mais il s'avère que celle-ci n'est pas du tout adapté à cela comme expliqué dans [2]. Ce qui est alors utilisé est le cadre bayésien, notamment via les réseaux de neurones bayésiens qui permettent de calculer la distribution prédictive afin de quantifier l'incertitude.

Il existe 2 modèles d'incertitudes pour les réseaux de neurones : l'incertitude épistémique qui est liée au modèle et est modélisée avec une loi a priori sur les paramètres du réseaux, et l'incertitude aléatoire qui est inhérente aux données.

Dans [1], ils expliquent dans un premier temps comment calculer ces deux types d'incertitudes indépendamment, en utilisant des réseaux de neurones bayésiens pour l'incertitude épistémique, et en réalisant du Maximum à Posteriori (MAP) pour capturer l'incertitude aléatoire. Puis, dans un second temps, ils présentent une manière de combiner les deux méthodes afin de capturer les deux types d'incertitude.

## 2 Incertitude épistémique

### 2.1 Théorie

L'incertitude épistémique est l'incertitude liée au modèle, et qui peut donc être a priori contrôlée en ayant plus de données. Traditionnellement, pour capturer cette incertitude en apprentissage profond, on utilise des réseaux de neurones bayésien. Ces modèles consistent à placer une loi a priori sur les poids du réseau  $W$ . Souvent, on prendra  $W \sim \mathcal{N}(0, I)$ . On peut alors calculer la loi a posteriori de la façon suivante en utilisant la règle de Bayes, en notant  $(X, Y)$  les données :

$$p(W|X, Y) = \frac{p(Y|X, W)p(W)}{p(Y|X)}$$

Le problème est que  $p(Y|X)$  n'est pas connue analytiquement. Pour l'approcher, on peut utiliser des méthodes d'inférence variationnelles bayésiennes, qui consistent à minimiser un certain critère sur un ensemble choisi de lois. Par exemple, on peut choisir  $q$  minimisant la divergence de Kullback-Leibler sur l'ensemble des loi normales telle que  $q(W) = \prod_i q(w_i)$ . Ces méthodes peuvent être assez coûteuses en termes de calcul, voire inutilisable si il y a trop de paramètres dans le modèle.

Ils proposent d'utiliser dans le papier l'inférence variationnelle dropout qui consiste à entraîner le modèle avec du dropout avant chaque couche contenant des poids, et à continuer d'utiliser le dropout lors du test. Ainsi les résultats renvoyés par le réseau sont aléatoires. Les sorties forment donc une certaine loi sachant les entrées. Et on peut calculer l'espérance et la variance de cette loi.

Cette méthode est en fait un problème d'inférence variationnelle de Bernoulli comme montré dans [3]. En effet, faire du dropout revient à avoir des matrices de poids de la forme :

$$W_i = M_i \cdot \text{diag}((z_{ij})_{i=1}^{K_i}) \quad \text{où } z_{i,j} \sim \text{Ber}(p_i)$$

On peut alors montrer d'après [1] que réaliser de l'inférence variationnelle dropout revient à choisir comme approximation de  $p(W|X, Y)$  une loi q telle que :

$$q \in Q = \{\alpha_1 q_1 + \alpha_2 q_2 | \alpha_1 + \alpha_2 = 1, q_1 = \mathcal{N}(\mu, \sigma_1), q_2 = \mathcal{N}(0, \sigma_2)\}$$

Minimiser la divergence de Kullback-Leibler entre q et  $p(W|X, Y)$  revient alors à maximiser d'après [4] la "Log Evidence Lower Bound" (ELBO) :

$$\mathcal{L}_{VI} = \int q(\omega) \log(p(Y|X, \omega)) d\omega - KL(q||p)$$

où p est la loi a priori sur les poids.

En effet, on a :

$$\begin{aligned} KL(q(W)||p(W|X, Y)) &= \mathbb{E}_{W \sim q} \left[ \log \left( \frac{q(W)}{p(W|X, Y)} \right) \right] \\ &= -\mathbb{E}_{W \sim q} \left[ \log \left( \frac{p(Y|X, W)p(W)}{q(W)p(Y|X)} \right) \right] \\ &= -\mathbb{E}_{W \sim q} [\log(p(Y|X, W))] + \mathbb{E}_{W \sim q} [\log(\frac{q(W)}{p(W)})] + \mathbb{E}_{W \sim q} [\log(p(Y|X))] \\ &= -\int \log(p(Y|X, w)q(w)dw + KL(q||p) + \log(p(Y|X)) \\ &= -\mathcal{L}_{VI} + \log(p(Y|X)) \end{aligned}$$

Cette expression ne peut pas être directement maximiser, car l'on ne peut pas calculer analytiquement tous les termes. En utilisant des estimations de Monte Carlo, on se ramène à la fonction de coût suivante :

$$\mathcal{L}_{dropout} = -\frac{1}{N} \sum_{i=1}^N \log(p(y_i|x_i, W_i)) + \lambda \sum_{l=1}^L (\|W_l\|_2^2 + \|b_l\|_2^2) \quad \text{où } W_i \sim q$$

Le second terme représente une régularisation  $\ell^2$  sur les poids du réseau.

Dans le cadre de mes expérimentations, je me suis concentré sur la régression et ai donc choisi comme loi  $p(y|f^W(x)) = \mathcal{N}(y; f^W(x), \sigma^2)$ . Pour la classification, on aurait utilisé une loi multinomiale, par exemple  $p(y|f^W(x)) = \text{Softmax}(f^W(x))$  ce qui aurait donné comme fonction de coût l'entropie croisée.

Ainsi, on a :

$$\begin{aligned} -\log(p(y_i|x_i, w_i)) &= -\log(p(y_i|f^{W_i}(x))) \\ &= \frac{1}{2} \log(\sigma^2) + \frac{\|y_i - f^{W_i}(x)\|^2}{2\sigma^2} + cst \end{aligned}$$

Et la fonction de coût que j'ai utilisé dans mon réseau est :

$$\mathcal{L}_{dropout} = \frac{1}{N} \sum_{i=1}^N \frac{\|y_i + f^{W_i}(x)\|^2}{2\sigma^2} + \lambda \sum_{l=1}^L (\|W_l\|_2^2 + \|b_l\|_2^2) \quad \text{où } W_i \sim q$$

Afin d'exprimer l'incertitude épistémique, on peut calculer la variance prédictive qui est approximativement donnée dans [2] par :

$$\begin{aligned}
Var(y|x, X, Y) &= \mathbb{E}[y^2|x, X, Y] - \mathbb{E}[y|x, X, Y]^2 \\
&= \int y^2 p(y|x, X, Y) dy - \left( \int y p(y|x, X, Y) dy \right)^2 \\
&= \int y^2 \int p(y, w|x, X, Y) dw dy - \left( \int y \int p(y, w|x, X, Y) dw dy \right)^2 \\
&= \int \int y^2 p(y|f^w(x)) p(w|X, Y) dy dw - \left( \int \int y p(y|f^w(x)) p(w|X, Y) dy dw \right)^2 \quad \text{par Fubini} \\
&\approx \int \int y^2 p(y|f^w(x)) q(w) dy dw - \left( \int y p(y|f^w(x)) q(w) dy dw \right)^2 \\
&\approx \int \mathbb{E}_y[y^2|f^w(x)] q(w) dw - \left( \int \mathbb{E}_y[y|f^w(x)] q(w) dw \right)^2 \\
&\approx \int (\sigma^2 + f^w(x)^2) q(w) dw - \left( \int f^w(x) q(w) dw \right)^2 \quad \text{car } p(y|f^w(x)) = \mathcal{N}(y; f^w(x), \sigma^2) \\
&\approx \sigma^2 + \frac{1}{T} \sum_{t=1}^T f^{W_t}(x)^2 - \left( \frac{1}{T} \sum_{t=1}^T f^{W_t}(x) \right)^2 \quad \text{où } \forall t, W_t \sim q \text{ en utilisant l'estimateur de Monte Carlo}
\end{aligned}$$

$\sigma^2$  correspond à l'incertitude aléatoire que l'on considère dans ce cas comme constante et que l'on peut donc omettre.

Pour la classification, on aurait plutôt utilisé la distribution prédictive :

$$\begin{aligned}
p(y=c|x, X, Y) &= \int p(y=c, W|x, X, Y) dW = \int p(y=c|W, x) p(W|X, Y) dW \\
&\approx \int p(y=c|W, x) q(W) dW \\
&\approx \frac{1}{T} \sum_{t=1}^T p(y=c|W_t, x) \quad \text{où } W_t \sim q \\
&\approx \frac{1}{T} \sum_{t=1}^T \text{Softmax}(f_t^W(x))
\end{aligned}$$

Puis l'on aurait ensuite calculé l'entropie de cette loi  $\mathcal{H}(p) = -\sum_{c=1}^C p_c \log(p_c)$  afin de mesurer l'incertitude du vecteur de probabilité.

## 2.2 Expérimentations

Les différentes expériences peuvent être retrouvées sur le github suivant.

### 2.2.1 MNIST avec défloutage et pixels manquants

J'ai tout d'abord essayé de calculer cette incertitude sur le problème de reconstruction d'images avec des pixels manquants. Pour cela, j'ai utilisé le jeu de données MNIST, et ai utilisé comme modèle un auto-encodeur, auquel j'ai ajouté du dropout entre chaque couche avec probabilités 0,2. En pratique, je n'ai pas utilisé de régularisation  $\ell^2$  sur les couches car cela ne donnait pas de très bons résultats en terme de prédictions.

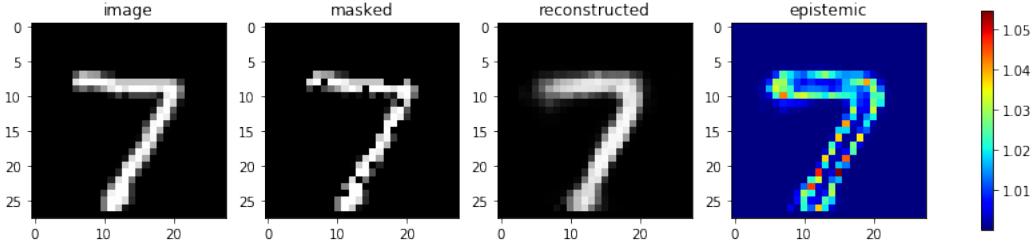


FIGURE 1: Problème de reconstruction à partir de pixels manquants avec l'incertitude épistémique associée

Je l'ai ensuite appliqué au problème de défloutage avec des noyaux de convolutions gaussiens aléatoires.

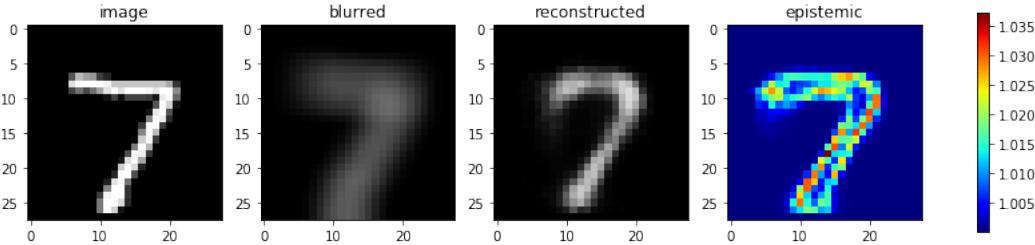


FIGURE 2: Problème de défloutage et l'incertitude épistémique associée

On observe dans les deux cas que la variance est plus forte au niveau des bords des chiffres. Cela se comprend bien dans la mesure où un pixel manquant à l'intérieur du chiffre est assez facile à reconstruire et sera à chaque fois bien reconstruit. Il sera aussi facile de reconstruire un pixel manquant en dehors du chiffre. En revanche, sur les bords, cela varie plus car les contours sont plus dure à reconstruire de manière exacte. Ainsi, il peut y avoir quelques bandes de pixels d'écart d'une réalisation à l'autre.

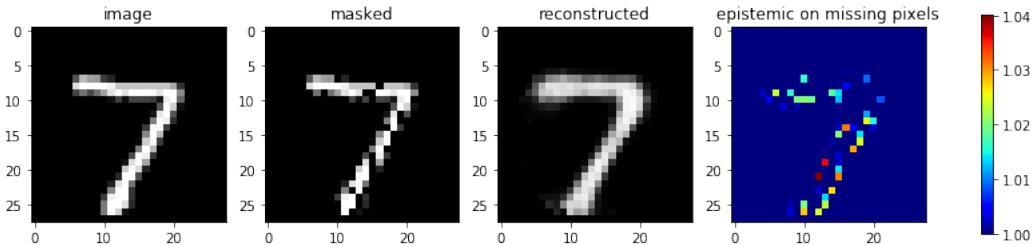


FIGURE 3: Incertitude épistémique sur les pixels manquants

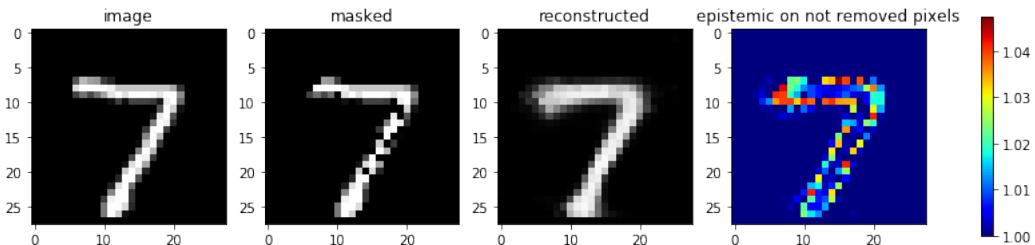


FIGURE 4: Incertitude épistémique sur les pixels non changés

On observe qu'il y a autant, voire plus d'incertitude épistémique sur certains pixels qui n'ont pas été modifiés autour de la figure. Intuitivement, on aurait pu se dire que l'incertitude devrait être faible sur ces pixels. Cela est sans doute vrai pour d'autres types de réseaux, mais j'ai utilisé ici un autoencodeur,

qui donc compresse les images en un vecteur dans un espace latent, avant de reconstruire l'image avec le décodeur. Le réseau reconstruit finalement toute l'image, ce qui peut expliquer que l'incertitude ne prenne pas vraiment en compte où les pixels sont manquants.

### 2.2.2 Super résolution

Dans un second temps, j'ai calculé cette incertitude sur un problème de super résolution. Pour cela, j'ai utilisé un des réseaux les plus simples qui est le SRCNN. Je me suis inspiré de [5] pour le code des réseaux de neurones.

Comme le réseau n'est pas énorme, il ne fonctionne pas très bien si l'on sous échantillonne trop les images. Je me suis donc contenté de sous échantillonner deux fois celles-ci.



FIGURE 5: Image dégradée (un pixel sur deux puis interpolation cubique)

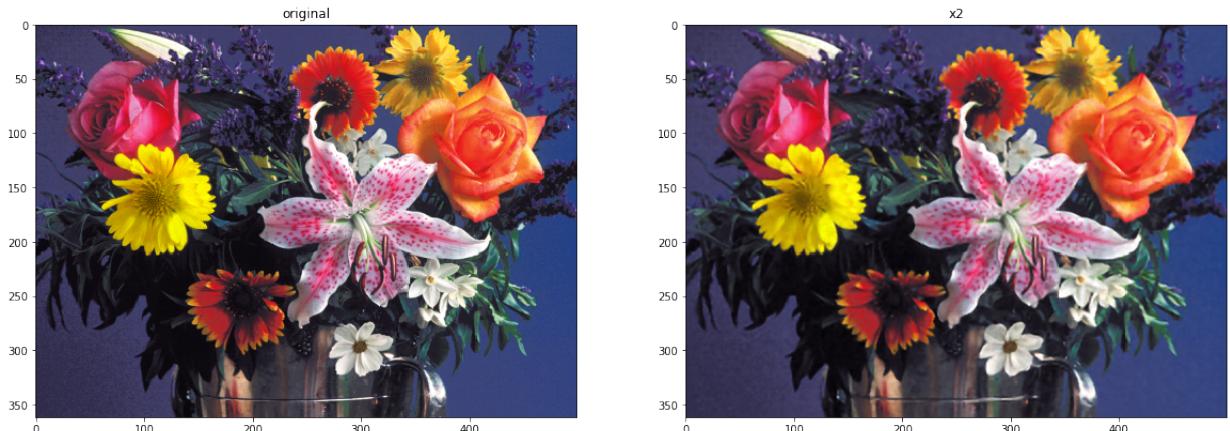


FIGURE 6: Super résolution x2 avec un SRCNN, à gauche l'original et à droit l'image reconstruite

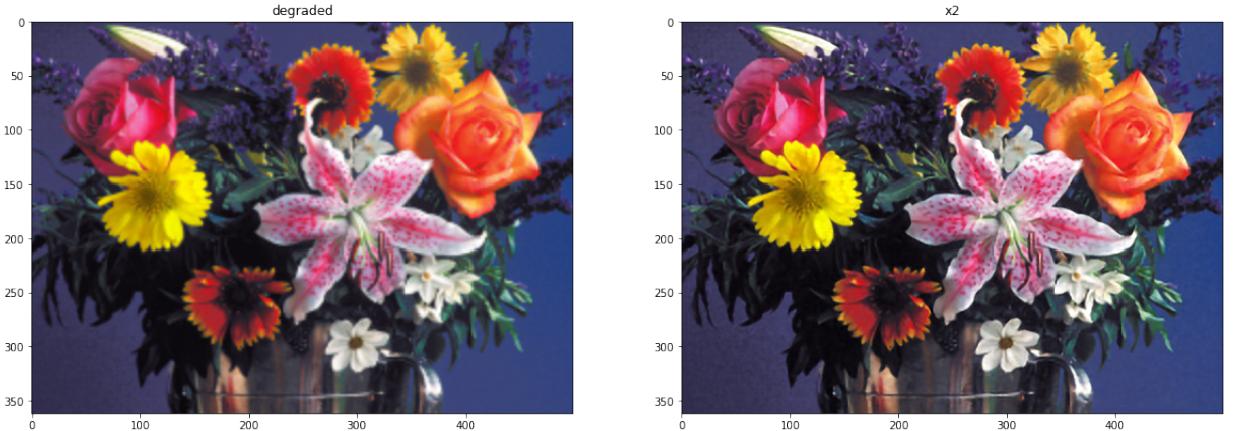


FIGURE 7: Super résolution x2 avec un SRCNN, à gauche l'image dégradée et à droite l'image reconstruite

J'ai rajouté des couches de dropout entre les couches de convolution avec des probabilités de 0.2. J'ai ensuite pu calculé l'incertitude épistémique par pixel. Pour cela, je l'ai calculé par canal puis je les ai sommé afin d'avoir un aperçu global.

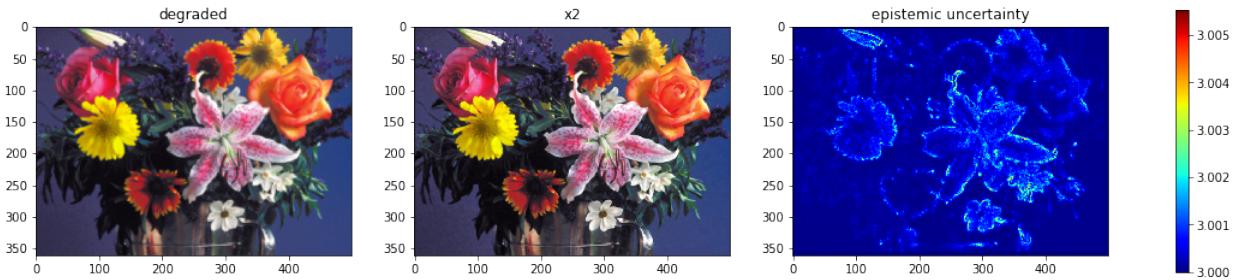


FIGURE 8: Super résolution x2 avec un SRCNN, et l'incertitude épistémique associée sommée sur les 3 canaux

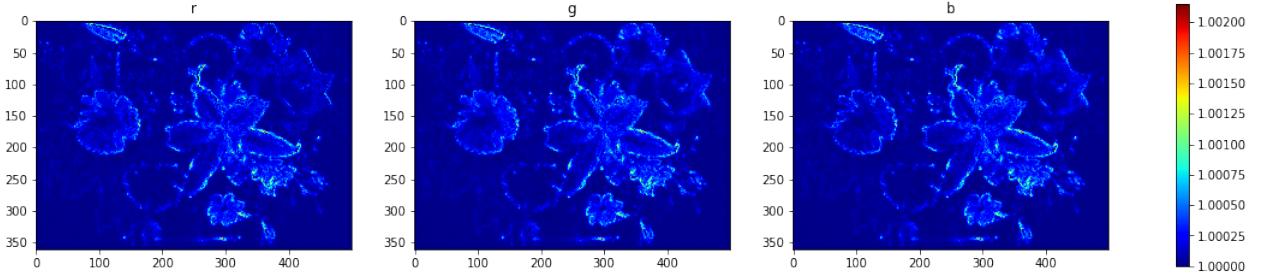


FIGURE 9: Incertitude épistémique associée à la super résolution sur chaque canal R, G, et B

On observe que l'incertitude est plus forte sur les bords des différents objets, notamment où les détails sont assez fins. C'est bien le genre de comportement que l'on pouvait attendre de l'incertitude. Les objets les plus fins sont les plus durs à reconstruire. Où les couleurs sont assez uniformes, sans trop de figures géométriques, l'incertitude est beaucoup plus faible.

On observe toutefois que l'incertitude a l'air assez égal sur chaque canal, alors qu'on aurait pu s'attendre à qu'elle soit différente en fonction des couleurs des objets ou des fleurs. Cela s'explique peut-être par le fait que la reconstruction avec un SRCNN est effectuée dans la base YCbCr.

De plus, elle n'a pas l'air de différencier les pixels qui n'ont pas été détériorées des autres. Cela est sans doute dû à la manière de fonctionner du réseau, car on change la représentation des images à mettre en entrée de celui-ci.



FIGURE 10: Super résolution x2 avec un SRCNN, et l’incertitude épistémique associée sommée sur les 3 canaux

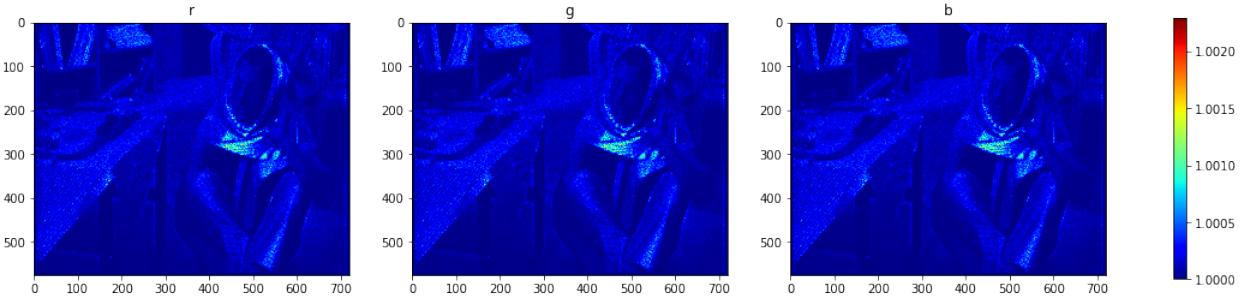


FIGURE 11: Incertitude épistémique associée à la super résolution sur chaque canal R,G, et B

Sur l’image de Barbara, on observe que l’incertitude est plus forte sur les textures qui sont le plus susceptibles d’avoir subi de l’aliasing lors du sous-échantillonage, c’est-à-dire le foulard ou encore le pantalon par exemple.

### 3 Incertitude aléatoire

#### 3.1 Théorie

L’incertitude aléatoire est inhérente aux données, et ne peut donc a priori pas être contrôlée. En revanche, cela peut être utile de la quantifier. Pour des photos, cette incertitude peut être grande quand celle-ci est floue, ou qu’une partie de la photo est trop claire. Cela peut être important par exemple dans le cas de la voiture autonome. En effet, en 2016, le système de perception d’une voiture autonome a confondu le blanc d’une remorque et le ciel très lumineux. On comprend que cela peut être intéressant de quantifier cette incertitude afin de savoir à quel point on peut se fier à une image par exemple.

On peut catégoriser l’incertitude aléatoire en deux incertitudes différentes, l’incertitude homoscédastique lorsque celle-ci est constante pour toutes les données, et hétéroscédistique lorsqu’elle dépend des données. La seconde est la plus intéressante, et celle à laquelle s’intéresse l’article.

On peut donc modéliser cette incertitude de plusieurs façons différentes. Si l’on modélise la loi de la sortie comme  $p(y|f^W(x)) = \mathcal{N}(y; f^W(x), \sigma^2)$ , alors on peut considérer que l’incertitude aléatoire correspond à  $\sigma^2$ . On peut alors essayer de l’apprendre et utilisant la fonction de coût suivante :

$$\mathcal{L}_{NN} = \frac{1}{N} \sum_{i=1}^N \left( \frac{1}{2\sigma(x_i)^2} \|y_i - f(x_i)\|^2 + \frac{1}{2} \log(\sigma(x_i)^2) \right) \quad (1)$$

On peut alors réaliser du maximum a posteriori (MAP) afin d’avoir un modèle fixe, et récupérer  $\sigma(x)^2$  en faisant prédire au réseau cette valeur. On fera plutôt prédire  $s_i = \log(\sigma(x_i)^2)$  au réseau, car cette valeur n’est pas forcément positive.

## 3.2 Expérimentations

### 3.2.1 MNIST avec défloutage et pixels manquants

J'ai rajouté une branche au réseau utilisé dans 2.2.1 permettant de prédire le bruit. Pour ce type d'incertitude, on n'a pas besoin de couches de dropout en test, car l'on veut réaliser du MAP et donc avoir un résultat déterministe. Les différences avec le réseau utilisé dans 2.2.1 sont donc qu'il n'y a pas de dropout en test, et que l'on fait prédire le bruit au réseau. Pour faire cela, on le prend en compte dans la fonction de coût (1). On observe alors que le bruit intervient deux fois dans cette fonction de coût. Dans le terme de droite, on va plutôt chercher à le minimiser, alors que dans le terme de gauche, on va plutôt chercher à l'augmenter. Ainsi, il va y avoir un compromis à trouver.

Avec le réseau précédent, j'ai donc obtenu les résultats suivants :

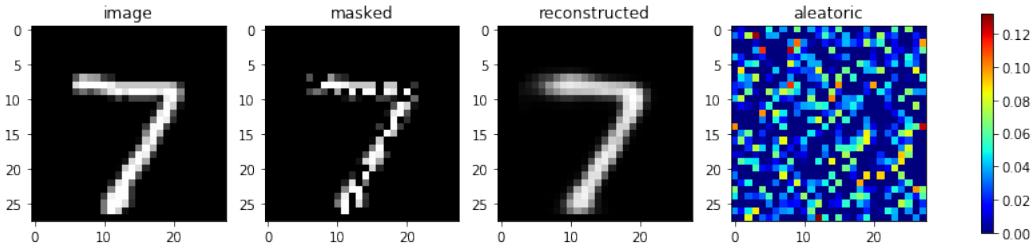


FIGURE 12: Problème de reconstruction à partir de pixels manquants, avec l'incertitude aléatoire associée

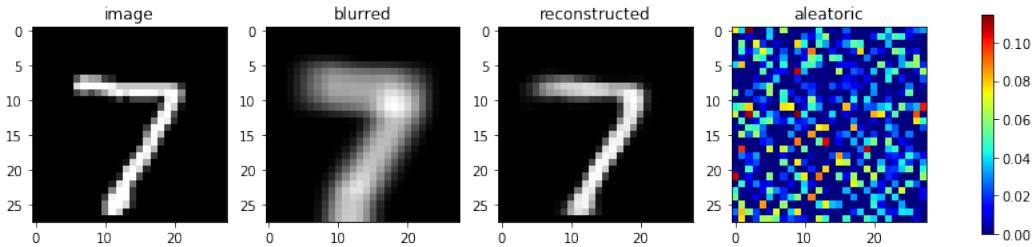


FIGURE 13: Problème de défloutage et l'incertitude aléatoire associée

On observe que cette incertitude n'est pas vraiment interprétable, et l'on observe aucune correspondance avec la forme géométrique. Ce n'est que du bruit. Cela peut être dû au fait que les images sont trop petites et n'ont pas assez de détails.

J'ai ensuite tracé une sorte de courbe de "precision-recall" adaptée à la régression. Pour cela, j'ai calculé le RMSE pour plusieurs images de test. J'ai ensuite trié cette liste d'erreurs, que j'ai tracé.

### 3.2.2 Super résolution

J'ai ensuite effectué les mêmes modifications sur le réseau SRCNN et l'ai appliqué à la super résolution. J'ai obtenu les résultats suivants :

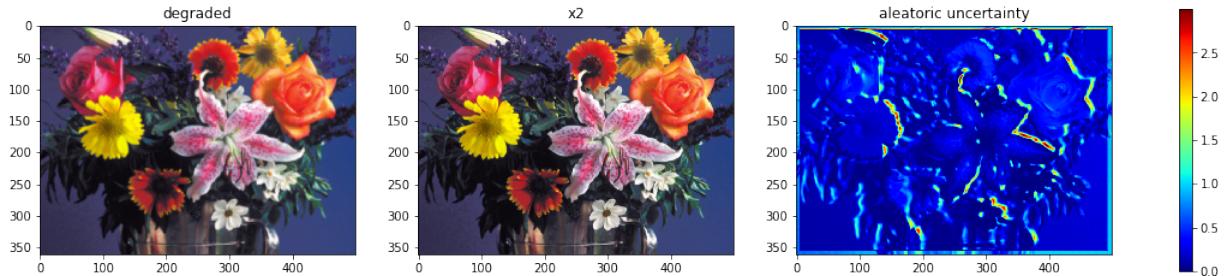


FIGURE 14: Super résolution x2 avec un SRCNN, et l'incertitude aléatoire associée sommée sur les 3 canaux

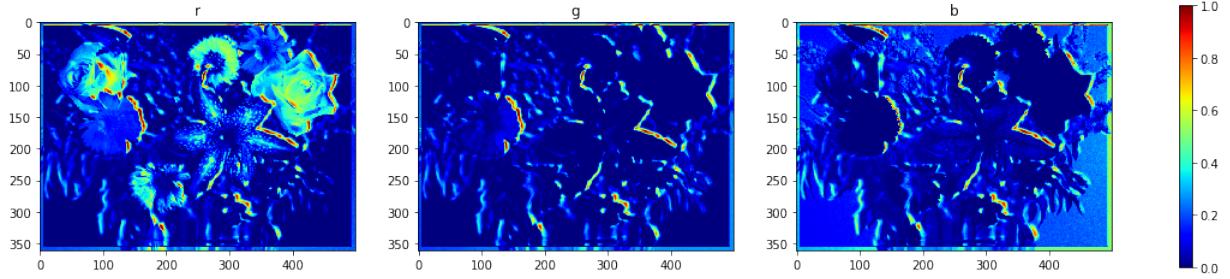


FIGURE 15: Incertitude aléatoire associée à la super résolution sur chaque canal R,G, et B

On observe que l'on peut reconnaître un peu mieux les formes géométriques par rapport à MNIST. De plus, on voit que contrairement à l'incertitude épistémique, l'on obtient pas la même incertitude sur les différents canaux.

## 4 Combinaison des deux

Dans l'article, ils remarquent que c'est l'incertitude aléatoire qui est la plus importante car elle capture la partie de l'incertitude que l'on ne peut pas contrôler, et c'est donc la plus importante en pratique car elle permet de récupérer l'incertitude liée au matériel par exemple, ou à la qualité d'une photo. L'incertitude épistémique diminue avec le nombre de données d'entraînement et est donc moins importante. En revanche, on ne peut pas identifier des exemples qui n'appartiennent pas à la distribution d'entraînement avec l'incertitude aléatoire, alors qu'on peut le faire avec l'épistémique.

Ce n'est pas forcément très flagrant avec les expériences que j'ai réalisées.

Ils présentent ainsi une façon de combiner les deux incertitudes afin de récupérer l'incertitude totale et pouvoir repérer les données n'appartenant pas à la distribution des données d'entraînement.

### 4.1 Théorie

Pour combiner les deux, la méthode proposée dans l'article est d'utiliser un réseau de neurone bayésien utilisant le dropout, permettant ainsi d'estimer l'incertitude épistémique avec la même fonction de coût que celle utilisée pour récupérer l'incertitude aléatoire. Ce réseau aura 2 sorties : la prédiction  $\hat{y}$  et une estimation de l'incertitude aléatoire  $\hat{\sigma}^2$ . On aura aussi besoin d'une loi a priori sur les poids, que l'on choisira de la même façon que dans la section 2, et qui induira donc un terme de régularisation  $\ell^2$  sur les poids.

Ainsi, on obtient, comme pour l'incertitude aléatoire, la fonction de coût suivante :

$$\mathcal{L}_{BNN} = \frac{1}{N} \sum_{i=1}^N \left( \frac{1}{2\sigma(x_i)^2} \|y_i - f(x_i)\|^2 + \frac{1}{2} \log(\sigma(x_i)^2) \right)$$

Notons la sortie du réseau de neurone  $f^W(x) = (\hat{y}, \hat{\sigma}^2)$ . On peut ainsi prédire l'incertitude aléatoire de la même façon que précédemment. Pour prédire l'incertitude épistémique, l'on utilise encore la variance prédictive, mais cette fois-ci en prenant en compte l'incertitude aléatoire hétéroscélastique. La formule est la suivante :

$$Var(y|x, X, Y) \approx \frac{1}{T} \sum_{t=1}^T \hat{y}_t^2 - \left( \frac{1}{T} \sum_{t=1}^T \hat{y}_t \right)^2 + \frac{1}{T} \sum_{t=1}^T \hat{\sigma}_t^2 \quad \text{où } \forall t, W_t \sim q$$

### 4.2 Expérimentations

J'ai combiné les réseaux précédents afin de pouvoir obtenir l'incertitude combinée. J'ai obtenu les résultats suivants :

#### 4.2.1 MNIST avec défloutage et pixels manquants

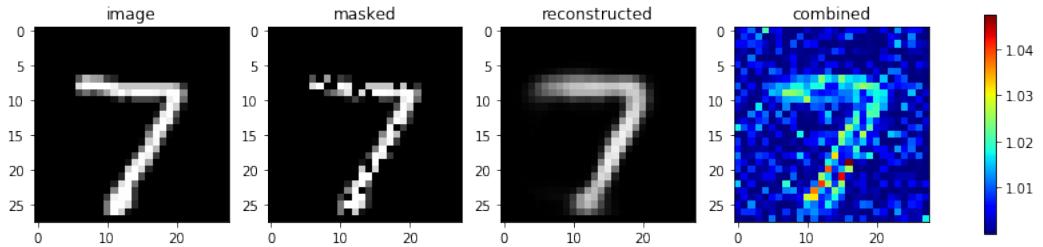


FIGURE 16: Problème de reconstruction à partir de pixels manquants, avec l'incertitude combinée associée

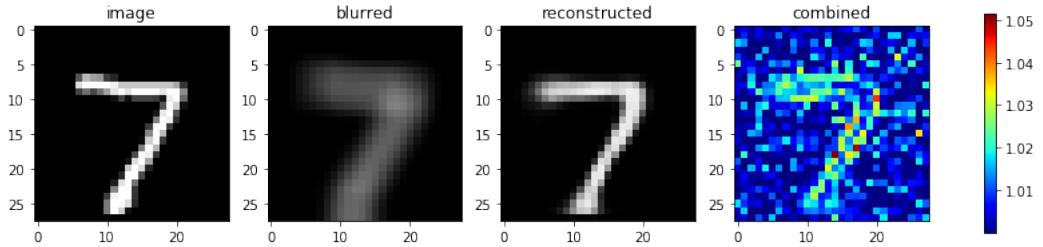


FIGURE 17: Problème de défloutage et l'incertitude combinée associée

On observe que l'on peut bien voir les deux types d'incertitude sur la figure, que l'on avait observé précédemment. En effet, l'on distingue bien les formes de l'image d'origine qui ont une incertitude plus élevée. Cela est du au fait qu'il y avait sur les formes une incertitude épistémique plus élevée, mais à peu près la même incertitude aléatoire partout.

#### 4.2.2 Super résolution

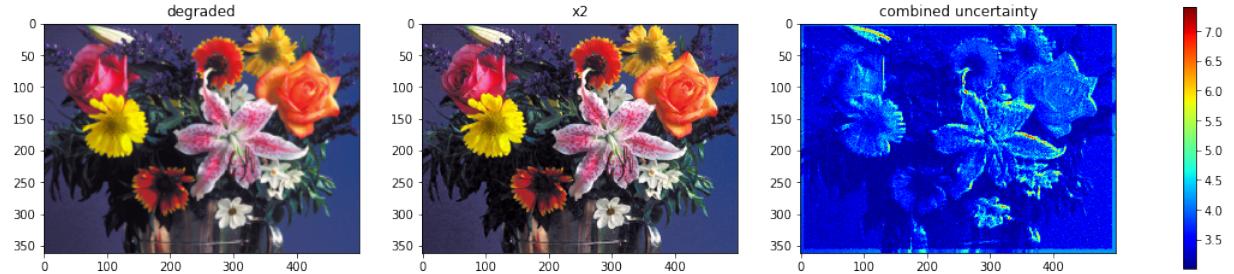


FIGURE 18: Super résolution x2 avec un SRCNN, et l'incertitude combinée associée sommée sur les 3 canaux

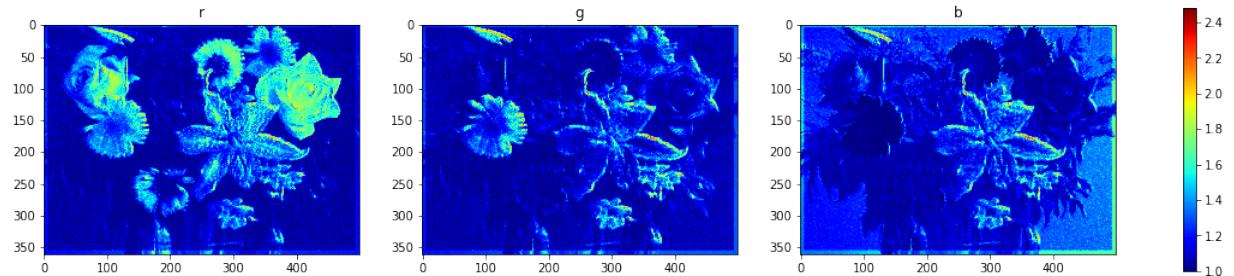


FIGURE 19: Incertitude combinée associée à la super résolution sur chaque canal R, G, et B

On observe que les incertitudes sur les différents canaux sont différentes, ce qui est sans doute dû à l'incertitude aléatoire comme on a pu l'observer précédemment. De plus, on observe bien que l'incertitude est beaucoup plus élevée aux endroits où il y a des formes géométriques, comme on pouvait s'y attendre via l'incertitude épistémique.

## 5 Conclusion

L'intérêt de modéliser et combiner les deux types d'incertitudes n'est pas très clair avec les expériences que j'ai effectué. Dans l'article, ils expliquent que finalement, dans des régimes de big data, ce n'est pas très intéressant de capturer l'incertitude épistémique car celle-ci est censée diminuer avec le nombre de données, et donc presque disparaître à partir d'un moment. Alors que l'incertitude aléatoire va capturer des incertitudes que l'on ne peut pas maîtriser car dû aux données. Le problème de cette dernière est qu'elle ne capture pas les images n'appartenant pas à la distribution des données d'entraînement. Et c'est pour cela qu'il est utile de combiner les deux types d'incertitudes car l'incertitude épistémique le peut. Il existe d'ailleurs d'autres méthodes comme par exemple les Prior Network présentés dans [6] qui permettent de capturer l'incertitude variationnelle pour la classification, c'est-à-dire qu'elle permet d'identifier les exemples n'appartenant pas à la distribution d'entraînement.

---

## Références

- [1] Alex Kendall and Yarin Gal. *What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?*, 2017.
- [2] Yarin Gal and Zoubin Ghahramani. *Dropout as a Bayesian Approximation : Representing Model Uncertainty in Deep Learning*, 2016.
- [3] Yarin Gal and Zoubin Ghahramani. *Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference*, 2016.
- [4] Christopher Bishop. *Pattern Recognition and Machine Learning*, 2006.
- [5] Mark Precursor. <https://github.com/MarkPrecursor/SRCNN-keras>
- [6] Andrey Malinin and Mark Gales *Predictive Uncertainty Estimation via Prior Networks*, 2018.