Reconstruction dense via des convolutions trouées

Adrien Chan-Hon-Tong

Reconstruction dense via des convolutions trouées

Adrien CHAN-HON-TONG
ONERA

27 septembre 2018

Résumé

La reconstruction d'images est un sujet bien étudié notamment via des méthodes de deep learning. Cependant, la plupart des approches apprennent leur réseau de reconstruction en présentant une image troué en entrée et l'image d'origine en sortie. Ainsi, le passage de l'image dans le réseau ne sera utilisé que pour actualiser le réseau à la lumière de sa performance sur une toute petite zone. À l'inverse, on propose ici une astuce dite de convolution trouée permettant d'apprendre le réseau comme pour de l'auto encodage (l'entrée est l'image la sortie aussi) tout en actualisation le réseau à la lumière de sa performance sur l'ensemble de la zone.

1 Introduction

L'auto encodage consiste à prédire une image I à partir d'une image I via un réseau de neurones D (plus généralement un algorithme) [4]. Le réseau D est typiquement un réseau encodeur décodeur comme par exemple UNET [3]. Durant l'apprentissage, le réseau est mis à jour par descente de gradient stochastique en fonction de l'écart (disons quadratique) entre $D(I, w)$ et I (w étant les poids courants du réseau). Typiquement, la structure de D force à perdre de l'information vis à vis de I (l'auto encodage peut alors être utilisé pour compresser le signal en récupérant la interne couche la plus compressée de D).

Étant donnée, une base d'apprentissage d'images $I_1, ..., I_N$, et un réseau D initialisé avec des poids w, l'apprentissage de D pour de l'auto encodage correspond grossièrement à :

`app_auto_encodage`$(D, w, I_1, ..., I_N)$:
  `pour i de 1 à` $nbEpoch \times N$
    `tirer n dans 1,...,N`
    `calculer` $metric = (D(I_n, w) - I_n) \times (D(I_n, w) - I_n)$
    `calculer` $\nabla metric_w$
    `calculer` $w = w - \frac{lr}{1+i/nbEpoch} \times \nabla metric_w$

Cette tache est très étudié dans la littérature [4, 1].

Indépendamment, la reconstruction d'images est aussi un problème très étudié [5, 2]. L'objectif est de reconstruire l'image I à partir d'une image dégradée I' où une partie a été masqué i.e. formellement $\forall i, j \in 1, ..., L \times 1, ..., l - E, I'(i, j) = I(i, j)$ et $\forall i, j \in E, I'(i, j) = 0$ (L, l étant la longueur et largeur de l'image).

À noter qu'ici, D ne va généralement pas forcer à perdre de l'information vis à vis de I comme c'est le cas en auto encodage.

Dans ce contexte, l'algorithme d'apprentissage correspond grossièrement à :

`app_reconstruction`$(D, w, I_1, ..., I_N)$:

  `pour i de 1 à` $nbEpoch \times N$

    `tirer n dans 1,...,N`

    `construire` I' en retirant une partie à I_n

    `calculer` $metric = (D(I', w) - I_n) \times (D(I', w) - I_n)$

    `calculer` $\nabla metric_w$

    `calculer` $w = w - \frac{lr}{1+i/nbEpoch} \times \nabla metric_w$

L'inconvénient de cet algorithme d'apprentissage est que chaque passe de I' dans le réseau ne sert à actualiser les poids w qu'au regarde de l'erreur sur la zone supprimée (on peut faire l'hypothèse que le réseau reconstruit parfaitement les zones non masquées).

On propose ici une astuce pour accélérer l'apprentissage. Il n'est pas trivial que cette astuce soit pertinente du point de vue de l'erreur de généralisation et/ou qu'elle permette d'autres applications. Ces autres questions sont hors de la portée de ce papier.

2 Contribution

2.1 Reconstruction dense

L'idée de base de ce papier est de chercher à apprendre de la reconstruction comme on apprendrait de l'auto encodage.

Ainsi, on veut que l'algorithme d'apprentissage soit :

`app_reconstruction_dense`$(D, w, I_1, ..., I_N)$:

  `pour i de 1 à` $nbEpoch \times N$

    `tirer n dans 1,...,N`

    `calculer` $metric = (D(I_n, w) - I_n) \times (D(I_n, w) - I_n)$

    `calculer` $\nabla metric_w$

    `calculer` $w = w - \frac{lr}{1+i/nbEpoch} \times \nabla metric_w$

Cependant, cela n'est pas possible trivialement puisque l'information à produire (I_n) est incluse dans (I_n)... Un tel réseau pourrait donc simplement converger vers l'identité - qui ne permet aucune reconstruction quand on lui présente une image tronqué I'.

Il faudrait donc que ce soit le réseau qui se force à reconstruire le pixel i, j en utilisant l'information autour de i, j mais pas i, j.

2.2 Convolution trouée

Il est possible de forcer le réseau à ne pas utiliser une information avec des couches classiques en forçant certain coefficient des convolutions à être nul.

Typiquement, ici l'idée serait de forcer le centre de la dernière convolution à être nul.

L'apprentissage est simplement modifié en :

`app_reconstruction_dense`$(D, w, I_1, ..., I_N)$:

  `pour i de 1 à` $nbEpoch \times N$

    `tirer n dans 1,...,N`

    `calculer` $metric = (D(I_n, w) - I_n) \times (D(I_n, w) - I_n)$

    `calculer` $\nabla metric_w$

    `calculer` $w = w - \frac{lr}{1+i/nbEpoch} \times \nabla metric_w$

    `pour tous les coefficients` k `qui doivent rester` $w_k = 0$

(ce qui ne change pas le gain de temps)

Plus précisément, soit I est une image $16L \times 16l \times 3$. On la fait passer dans un réseau convolutionnel D classique. Ici, on considèrera le debut de l'encodeur d'un UNET i.e. le début d'un VGG, jusqu'à la convolution 5_1 (en supprimant les convolutions 4_1 et 4_2). On obtient une feature map $F(I)$ de taille $L \times l \times 512$ dans laquelle chaque pixel i, j ne dépend que du bloc de pixels $\{16i-16, ..., 16i+32\} \times \{16j-16, ..., 16j+32\}$ de l'image.

Si on convolue cette carte avec une convolution de taille $(3 \times 16 \times 16) \times 512 \times 5 \times 5$ dont la sortie $3 \times 16 \times 16$ est reconvertie en une imagette $16 \times 16 \times 3$, on obtient une image qui a le même format que I.

En d'autre mot, chaque bloc 5×5 de la feature map est utilisé pour produire une image $16 \times 16 \times 3$, et donc reconstruire une image qui a le même format que I.

Maintenant si la sous convolution 3×3 de cette convolution 5×5 est forcé à être nulle, cela signifie que le bloc $16 \times 16 \times 3$ est reconstruit sans jamais utilisé l'information correspondante dans l'image !

En effet, l'imagette $\{16i, ..., 16i+16\} \times \{16j, ..., 16j+16\}$ va être reconstruite en utilisant l'information de la feature map au indice $i+\delta_i, j+\delta_j$ avec $|\delta_i|+|\delta_j| > 3$.

Le réseau appris est donc bien un réseau de reconstruction. Supprimer un bloc 16×16 de l'image, ne modifiera pas le bloc 16×16 correspond dans l'image produite (cela peut modifier les blocs voisins). Et pourtant l'apprentissage est réalisé exactement comme pour un auto encodeur.

3 Perspective

On introduit dans ce papier une astuce de convolution trouée qui permet de faire de la reconstruction dense. Cette astuce doit maintenant être évaluée expérimentalement.

Références

[1] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori. Speech enhancement based on deep denoising autoencoder. In *Interspeech*, pages 436–440, 2013.

[2] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders : Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[4] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[5] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*, pages 341–349, 2012.