Deep Structured Output Learning For Unconstrained Text Recognition

Li Honglin
Maxime Buron

17 février 2017

Table des matières

1 Résumé du papier

Nous avons choisi de travailler sur le papier intitulé Deep Structured Output Learning Dor Unconstrained Text Recognition de Max Jaderberg, Karen Simonnyan, Andrea Vedaldi et Andrew Zisserman. Voici un résumé de ce que nous en avons compris. L'article présente trois méthodes pour résoudre le problème suivant, dont la dernière combine les deux premières.

1.1 introduction du problème

Ce papier s'attaque au problème très général suivant : détecter et reconnaître du texte sur une image. Plus précisément, ce papier suppose la phase de détection est déjà réalisée, en d'autres termes l'on ne considère que des images contenant uniquement un seul mot sans autre contenu. Le sujet principal est donc la reconnaissance de mot, et particulièrement de mot sans contrainte, c'est à dire des mots qui ne sont pas obligatoirement issue d'une liste de vocabulaires.

1.2 encodage de l'entrée

Comme dit précedemment, l'entrée du problème est un image, seulement pour des questions de standardisation, notre algorithme nécessite des images de taille fixe 32×100 en noir et blanc. Malheuresement les images des ensembles de données ne sont pas toutes la même taille, c'est pourquoi elle sont étirées pour atteindre ces dimensions et cela sans préserver les proportions. Les images, une fois chargée sous forme de matrices de dimension 32×100 sont normalisées en leur soustrayant leur moyenne et en les divisant par leur écart type.

1.3 l'approche par caractère

L'approche par caractère considère que chaque caractère identifié selon sa position indépendemment des autres caractères.

La modélisation de la sortie de l'algorithme est une liste de N_{max} distribution de probabilités,où N_{max} est la longueur maximale qu'un mot sur une image en entrée. Les distribution de probalités modèlisent les lois de variables aléatoires à valeurs dans l'alphabet C considèré pour identifier les mots augmenté par un caractère vide ϕ. Un mot ω composé des caractères $c_1c_2\dots c_n$ avec $n \leq N_{max}$ est encodé par la suite de N_{max} caractères suivantes $c_1, c_2, \dots, c_n, \phi, \dots, \phi$. La prédiction de l'algorithme pour une image donnée x est alors l'encodage ω^* défini par :

$$\omega^* = \arg\max_{\omega} P(w|x) = \arg \max_{c_1,c_2,\dots,c_{N_{max}}} \prod_{i=1}^{N_{max}} P(c_i|\Phi(x))$$

où $\Phi(x)$ est un ensemble de paramètres.

L'algorithme du calcul des distributions est constitué d'un réseau de neurore conditionnel, qui sera réutiliser dans la méthode suivante. Ce réseau de neurone est constitué de 5 couches convolutionnelles et 2 couches complétementement connectées de la dimension de l'image en entrée.

Au bout du réseau de neurones spécialisé pour cette méthode, il y a N_{max} couches complétements connectées et indépendantes avec $|C|+1$ neurones où C est l'alphabet de sortie. On peut alors interpréter les valeurs de chacune de ces couches comme les distributions de probabilités mentionnées plus haut.

En ce qui concerne la fonction de score S_1 de cette approche, on peut la définir de la manière suivante :

$$S_1(\omega, x) = \log P(\omega|x) = \sum_{i=1}^{N_{max}} S^i(c, x)$$

où S^i est le logarithme de la probabilité que i^e caractères de la sur l'entrée x soit c d'après l'algorithme.

Le réseau de neurones est entraîné par une descente de gradient stochastique.

1.4 l'approche par sac de mots

L'approche par sac de mots utilise la même entrée, que l'approche précédentes.

La sortie de cette approche est la probalité pour une sous-chaîne de caractères d'être présente dans l'image en entrée. On considère en particulier uniquement les sous-chaînes de taille inférieur à N un entier, en pratique on a choisi 4. On définit le sac de mots d'un mot comme étant l'ensemble des sous-chaîne de caractères d'un mot. Pour des raisons pratiques, on ne considère que les sous-chaînes les plus présentes dans les datasets considérés et l'on nomme leur ensemble G_N.

L'algorithme pour déterminer cette probabilité utilise le même premier réseau de neurones de l'approche précédent, mais se terminer avec une unique couche complétement connectée possédant exactement un neurone pour chaque sous-chaîne de caractères de G_N.

Ici la fonction de score S_2 peut être définie de la manière suivante :

$$S_2(\omega, x) = \sum_{i=1}^{|\omega|} \sum_{k=1}^{\min(N,|\omega|-i+1)} S_e(c_ic_{i+1}\dots c_{i+k-1}, x)$$

où ω composé des caractères $c_1c_2\dots c_n$ et $S_e(c_ic_{i+1}\dots c_{i+k-1}, x)$ est le logarithme de la probabilité d'avoir la sous-chaîne $c_ic_{i+1}\dots c_{i+k-1}$ dans l'entrée x. Si $s \notin G_N$, alors $S_e(s, x) = 0$

1.5 l'approche jointe

La méthode jointe effectue en parallèle est deux méthodes précédentes et définit un nouveau score S pour une entrée x définie par :

$$S(\omega, x) = S_1(\omega, x) + S_2(\omega, x)$$

Encore une fois, une sortie pour cette approche est un mot qui maximise la fonction de score sachant l'entré x.

Ici, nous allons parler de la fonction du perte appliquée dans l'approche jointe, elle s'adapte facilement aux autres approches en remplaçant simplement S par S_1 ou S_2. La function de perte utilisée ici est une fonction de perte de type Hinge. C'est à dire que la fonction de perte L sur une entrée x pour un mot ω est définie par :

$$L(x, \omega) = \max_{\omega \neq \omega'} \max(0, \mu + S(\omega', x) - S(\omega, x))$$

où μ est une marge fixée.

1.6 les différents ensembles de données

Les ensembles de données utilisés pour effectuer l'évaluation des différents approches sont les suivants
— ICDAR 2003
— ICDAR 2013
— Street View Text : textes extraits de Google Street View
— Synth90k : textes générés par ordinateur
Le derniers ensemble de données est particulièrement important. Il sert d'ensemble d'entraînement, puisqu'il est assez important.

2 La méthode choisie

2.1 Notre projet

Nous avons choisi de réaliser la première approche, qui évalue les caractères les uns après les autres. Et nous avons décidé d'utiliser le dataset ICDAR2013, qui semble d'une taille et d'une difficulté convenable.

FIGURE 1 – Exemple d'images en entrée

On remarque tout de suite la différence de taille des images, c'est pourquoi comme décrit précédemment les images sont déformées vers une taille précise et sauver dans une matrice à valeurs réelles.

3 L'implémentation

Nous avons décidé d'utiliser Tensor Flow pour réaliser le réseau de neurones.

4 Les résultats

5 Conclusion