Text segmentation in manga

Babacar Toure
CentraleSupélec
babacar.toure@student-cs.fr

Abstract

La détection et la reconnaissance de texte dans les BDs est une tâche assez difficile. Les BDs les plus populaires, et malheureusement les complexes à traiter automatiquement aujourd'hui sont les mangas japonais. En effet dans les mangas, le texte n'est pas contraint dans des bulles de dialogue et le style est assez variable selon l'auteur. Nous allons tenter d'améliorer le travail déjà fait en segmentation binaire de texte à caractères japonais grâce à des architechtures basées sur Unet.

1. Introduction

Les mangas sont aujourd'hui très populaires. Pour un manga donné, la publication de nouveau chapitre en japonais se fait à une fréquence fixe, souvent hebdomadaire. Il faut ensuite un temps de traduction qui est assez variable selon le succès du manga en question. Le japonais est une langue assez complexe à l'écrit. Il comporte enormément de lettres dont des logogrammes et sinogrammes. De plus, les auteurs de manga n'ont pas réellement de contraintes sur l'organisation du texte. Le texte peut être dans une bulle de dialogue ou pas, il peut traverser des scè es différentes et avoir des tailles, formes et orientations diverses. Ainsi, une problématique majeure et difficile de cette industrie est la traduction automatique de manga du japonais vers d'autres langes. Pour cela, il faut être capable de détecter automatiquement le texte sur la page en japonais, de l'effacer, de le traduire puis de réécrire ce texte traduit sur la page. Etant donné qu'il faut effacer le texte, la meilleure manière manière de réaliser la détection est la segmentation binaire au niveau des pixels.

Dans ce projet, nous faisons donc de la segmentation binaire afin classer un pixel en "texte" ou "background". Nous implémentons et entrainons d'abord la meilleure solution que nous avons trouvé pour segmenter du texte japonais, puis dans un second temps, nous essayons d'enrichir cette méthode par des approches semi-supervisées afin de contribuer à l'état de l'art.

Figure 1. Variété de styles de texte

2. État de l'art

L'approche classique de détection de texte dans les bandes dessinées est la détection de bulle de parole. Cette technique est utilisée dans l'état de l'art sur des comics à alphabet latin et marche bien [1]. La méthode consiste à un entrainer un Unet avec VGG16 comme encoder. Mais les auteurs ont dû écarter les mangas de leur dataset pour booster les performances. Nous avons approfondi cela en entrainant les réseaux sur un dataset uniquement composé de manga et les résultats n'étaient pas acceptables.

Une autre approche développée dans [2], consiste à en-

trainer un Unet à segmenter les pixels directement avec un resnet38 comme encoder. Les auteurs ont créé leur propre dataset annoté pour cela.

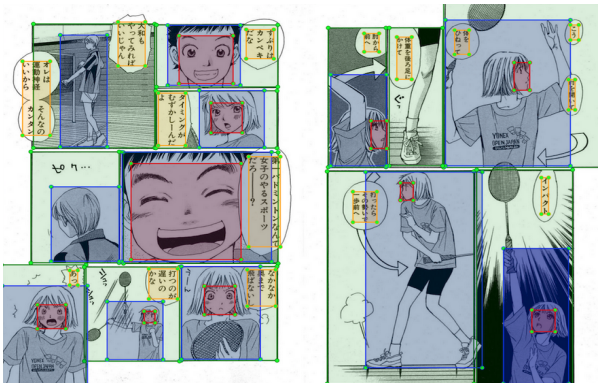
Par ailleurs, la segmentation binaire connait beaucoup d'application sur des cartes, des documents ou encore de la imagerie médicale. Les approches sont en général basées sur Unet combiné à un backbone pré-entrainé sur ImageNet.

3. Dataset

Le dataset que nous utilisons provient de plusieurs sources et a subi quelques pré-traitements. Il est très difficile de trouver des images labellisées de mangas. C'est d'autant plus le cas pour une tâche de segmentation car l'annotation consomme énormément de temps et demande beaucoup de précision. Nous avons utilisé un dataset très populaire appelé Manga109. Ce dataset est disponible en demandant aux auteurs de le fournir, mais il est interdit de le partager. Ce dataset est annoté mais pas pour la segmentation. Nous disposons aussi d'une version annotée de dataset (environ 5% du dataset) par les auteurs de [2] et aussi une version traitée par nous-mêmes de Manga109.

3.1. Manga109

Manga109 ([3]) regroupe 109 volumes de manga, soit plus de 10000 images. Ces volumes sont parus entre 1970 et 2010 et correspondent à des oeuvres de tout genre. Chaque image est annotée à l'aide de "bounding boxes" autour du texte, des visages, des corps et des cadres.



Manga109 annotation

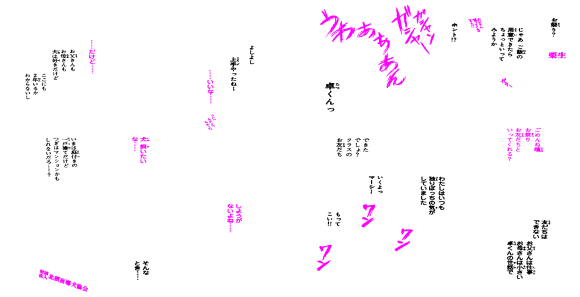
3.2. Ensemble complet annoté

Nous disposons également de 400 images qui correspondent à des masques de texte. Chaque pixel est classé en 1 s'il s'agit de texte et 0 si il s'agit de fond. C'est sur cet ensemble que vont s'appuyer l'essentiel de nos méthodes.

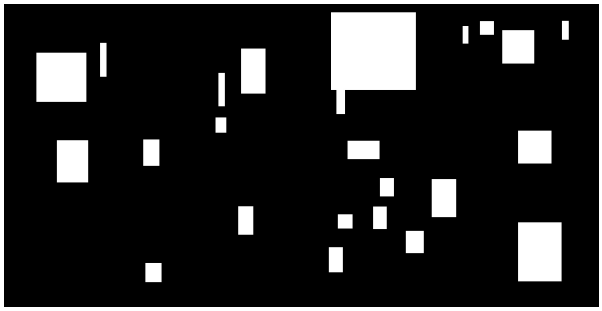
3.3. Dataset traité final

Nous rajoutons à tout ceci un traitement qui extrait de chaque image un masque correspondant au "bounding box" autour du texte donnée dans la version originale de Manga 109. En résumé, nous disposons donc d'une part d'un ensemble complet avec 400 images, leur masque de texte et le masque donnant la position du texte (boite englobante autour du texte); d'autre part, nous avons 9550 images accompmpagnées uniquement de la position du texte (boite anglobante) mais sans la labellisation pour la segmentation. Nous gardons également 20% de l'ensemble complet comme ensemble de test servant à évaluer notre modèle.

Image originale



Masque de texte



Masque de position du texte

Ceci est un élément de l'ensemble complet. S'il était dans l'ensemble faible ou incomplet, il n'aurait juste pas de masque de texte.

4. Méthodes

Notre première approche est d'utiliser la méthode des auteurs de l'article qui nous a servi de base [2]. Nous avons donc implémenter une architecture de type Unet avec un encoder pré-entrainé. Nous avons ensuite essayer de tirer

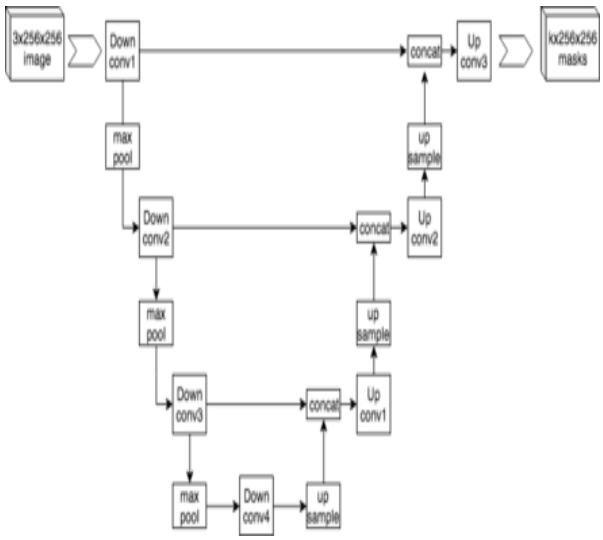
partie de la connaissance de la position du texte grâce aux ”bounding boxes” pour améliorer ce modèle grâce à une approche semi-supervisée.

Quelques précisions sur les termes. Nous appelons le dataset faible le dataset contenant uniquement les images et les boites englobantes donnant la position du texte. Le dataset complet est le dataset dans lequel on dispose des images, des boites englobantes mais aussi la segmentation de texte.

4.1. Unet34

4.1.1 Modèle

Dans cette approche, le réseau convolutionnel est entrainé pour prédire si un pixel en niveau de gris est du texte ou non. La prédiction est donc une image binaire. Unet34 est une variation du réseau Unet avec comme encoder le backbone Resnet34. L'encoder génére 5 représentations progressivement contractées mais avec un nombre de canaux de plus en plus grand. Le décoder fait une expansion des représentations combinées à une concaténation aux répresentations de l'encoder (voir figure).

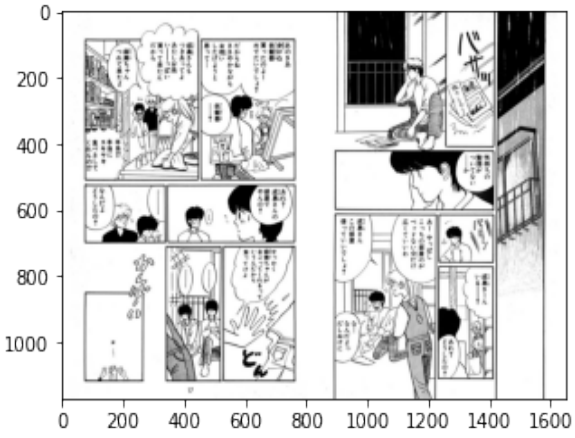


Architechture Unet

Le modèle Resnet34 quant à lui est un réseau convolutionnel utilisé comme classique pour la reconnaissance d'image. Il a été entrainé sur ImageNet et peut donc distinguer un objet parmi 200 classes. Sa particularité est l'usage de la sortie de chaque couche dans les couches suivantes (connexion ”sautée”).

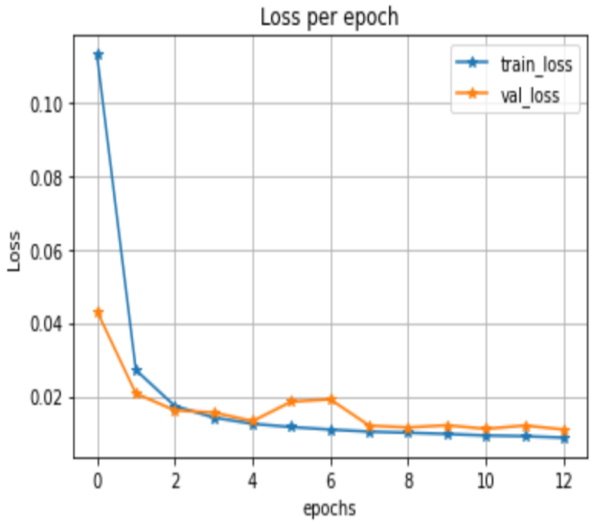
4.1.2 Optimisation du modèle

Tout d'abord nous avons choisi la fonction de cout la plus classique pour les tâches de segmentation qui est la diceloss. Cette fonction de coût est basée sur le coefficient de Dice: $Diceloss = 1 - Dice$. Le coefficient de Dice est une métrique évaluant le nombre de pixels ayant la même valeur: $Dice = 2 * \frac{y_{pred} \cap y}{y_{pred} + y}$. Il s'agit donc du double du rapport entre le nombre de pixels ayant la même valeur dans les deux images et le nombre total de pixels. C'est une metrique allant de O à 1 avec la la meilleure valeur possible.

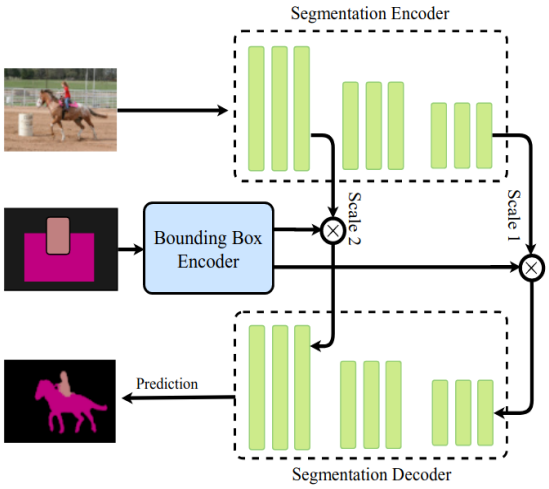


Exemple d'image du dataset

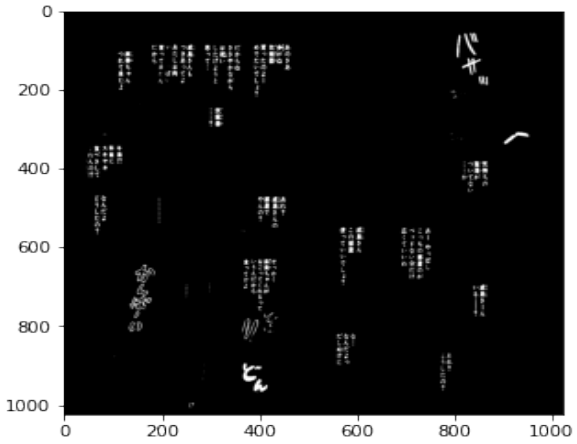
Pour l'entrainement de ce modèle nous avons plus ou moins repris les recommandations de l'article de Rosana et al [2]. En particulier, nous avons utilisé l'optimizer Adam. Nous avons fixé le taux d'apprentissage à 1.10^{-3} en faisant des entrainements avec les taux 1.10^{-3}, 1.10^{-2} et 1.10^{-4}. Avec un petit taux d'apprentissage, la convergence était lente. Nous avons également choisi la taille du batch d'entrainement à 4 après avoir comparé les résultats avec un entrainement à taille de batch 10. D'autre part nous avons testé notre réseau en l'entrainant sur des images de taille 1024 et 512. Nous avons conclut qu'avec des images de taille 1024, nous avions une meilleure erreur de validation. Nous précisions que la taille de l'ensemble de validation représente 15% de la taille de l'ensemble d'entraineement. Après avoir fixé ces hyperparamètres nous avons réalisé un entrainement long, d'environ 30 epochs pour trouver le meilleur nombre d'épochs sans faire de sur-apprentissage. Nous avons constaté qu'entre 10 et 15 epochs suffisent pour avoir un bon modèle et avons donc fixé le nombre d'epochs à 13 en gardant le modèle qui a la meilleure erreur de validation jusque-là.



Évolution du cout



Architure du réseau auxiliaire

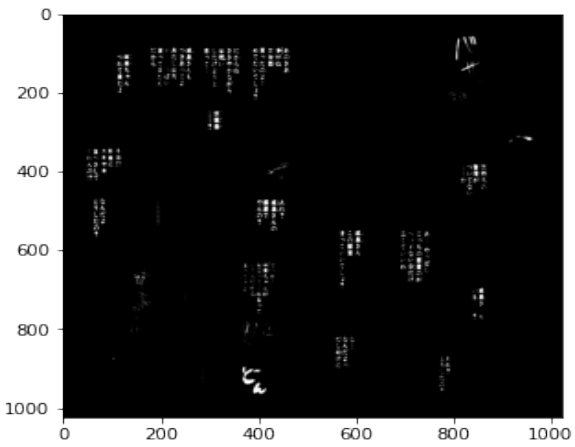


Prédiction sur l'exemple de base

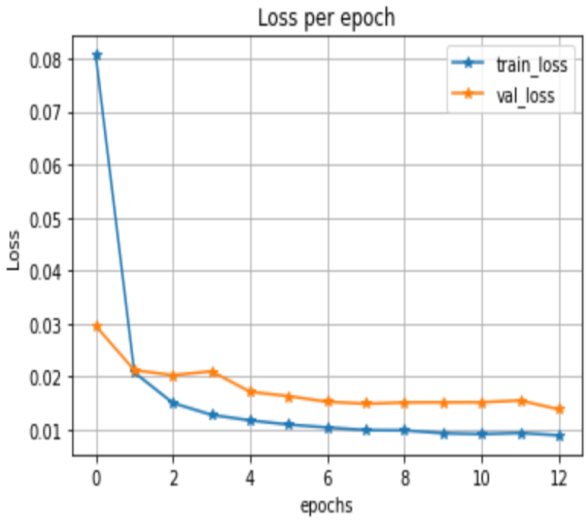
Dans notre application, notre réseau auxiliaire prend en entrée une image, un masque de position de texte et produit une segmentation de texte. Pour l'architcture, on garde la même architecture que précédemment mais on multiplie les tenseurs de "skip-connexions" classiques de Unet par le masque de position du texte, qui contient, rappelons le des 0 et des 1. Ceci peut être vu en fait comme une attention map. Le schéma ci-dessous l'illustre bien (cf [4].

4.2. Approche semi-supervisée

L'objectif de l'approche semi-supervisée est de corriger le manque de données labellisées. En effet, nous ne disponsons que de 400 images labellisées sur 10000. Si nous réussissons à faire un modèle, meilleur que notre modèle de base, on pourra l'utiliser pour prédire les labels du dataset faible. La première étape est donc faire un modèle plus performant en ajoutant de l'information. Nous proposons cette approche, afin d'améliorer la méthode utilisée par les auteurs de [2]. Lidée est tout simplement d'apporter de l'information sur la position du texte pendant l'entrainement. Cette méthode nous est inspirée par l'article de Ibrahim et al [4], qui utilise un Unet qui prend en entrée les images et les boites englobantes ("bounding boxes") des classes qu'ils veulent segmenter, ce qui augmente la précision de leur segmentation. Ce réseau est appelé le réseau auxilaire.



Prédiction sur l'exemple de base avec modèle auxiliaire



Évolution du cout du modèle auxiliaire

4.2.1 Entrainement du réseau auxiliaire

L'entraiment du modèle auxiliaire est fait dans les mêmes conditions que celui du modèle de base. Il est réalisé sur l'ensemble complet constitué de 400 images, masques de position de texte et masques de texte. Nous entrainons sur 13 épochs, des batches de taille 4 et un taux d'apprentissage de à $1e^3$.

4.2.2 Prédiction et ré-entrainement du modèle de base

Grâce au réseau auxiliaire, nous prédisons des masques de texte sur le dataset faible. Nous disposons ainsi d'un nouveau dataset. Dans ce dataset, on dispose des 400 images originales mais aussi des images du dataset faible labellisée pour la segmentation par le réseau auxiliaire. Malheureusement, ce nouveau dataset est trop grand et nous n'avons eu les capacités de calcul permettant de faire l'entrainement de notre premier modèle siple Unet34. Néanmoins, nous sommes convaincus ce modèle aurait une meilleure capacité de généralisation et des meilleurs résultats.

5. Expériences et résultats

Dans cette section, nous exposons les résultats obtenus par les différentes méthodes. Nous notons "base" le modèle de base et "ancillary" le modèle auxiliaire pour reprendre la notation des auteurs de la méthode. Nous rappelons que le modèle "ancillary" n'est pas exploitable dans un cas général car il a aussi besoin de boites anglobantes donnant la position du texte en entrée. Nous le camparons quand même au modèle de base car la détection de boites anglobantes de texte n'est pas une tâche dure.

5.1. Critères d'évaluation

Pour comparer les différentes approches, nous utilisons différentes métriques suivantes. Sachant que l'on peut considérer notre problème comme un problème de classification binaire de pixels, nous utilisons la précision, le recall et la F1-score comme premières métriques. Nous n'allons pas détailler ces métriques car elles ne sont pas spécifiques à la segmentation et sont assez classiques. Une autre métrique que nous utilisons est le score IoU pour "intersection over union" ou indice de Jaccard. C'est un indice allant de 0 à 1 avec 1 la valeur idéale. On calcule sur une fenetre le cardinal des pixels correspondants entre la segmentation prédite et la vérité-terrain divisé par le cardinal de l'union (ici la somme). Cette métrique est plus performante et directe que les métriques classiques de la classification pour une tâche de segmentation.

Les implémentations des métriques utilisées proviennent de la librairie segmentation-models-pytorch

5.1.1 Performances

Nous rappelons que nous évaluons nos modèles sur un ensemble de test fixé dès le départ et tiré au hasard dans le dataset complet. Il contient 91 instances.

En évaluant nos modèles, nous nous rendons compte qu'ils sont déjà tous très performants. Même si leurs objectifs sont différents. Le modèle de base est très performant et fournit une segmentation en réalité très propre. Par contre le modèle auxiliaire arrivent à distinguer le texte correspondant à du dialogue au texte présent dans le dessin (sur un mur par exemple). Ce qui est normal car on fournit la position du texte de dialogue dans le masque de position du texte. Voici les perfomances que nous avons obtenus sur notre ensemble d'entrainement:

Modèles	IoU	Precision	Recall	F1
Base	97.40 %	98.04 %	99.32 %	98.67
Ancillary	98.13 %	99.13 %	98.32 %	99.04

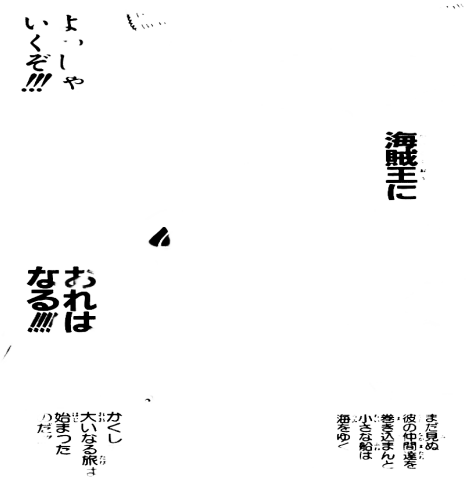
Nous constatons que nous avons de meilleurs scores sur toutes les métriques avec le modèle auxiliaire, sauf le recall. Ce qui est très encourageant sachant que le modèle auxiliaire prédit du background pour le texte qui n'est pas du dialogue (par exemple une écriture sur un mur dans le dessin), alors que dans la vérité terrain cette écriture est annotées comme "texte" par les auteurs du dataset faible [2]. C'est d'ailleurs à cause de ce phénomène que le modèle auxiliaire prédit beaucoup de faux négatif. Ce qui explique pourquoi son recall est plus faible que celui du modèle de base. Globalement, les résultats sur l'ensemble de test sont excellents pour les deux modèles.

6. Conclusion

Ce projet a été l'occasion de disséquer en une expérience deux réseaux backbones avec des tâches différentes. Nous entendons là Unet et Resnet34. Nous avons bien compris leurs implémentations à travers notre propre implémentation d'un réseau de segmentation. En implémentant notre approche semi-supervisée pour réaliser la tâche de segmentation nous avons aussi pu charger un réseau déjà entrainer et y ajouter entrées et des "skipped connexions", ce qui a eu un grand apport pour nous. Par ailleurs nous avons constaté que rien qu'en réglant les hyperparamètres de nos modèles nous pouvons améliorer de manière significative l'apprentissage. Néanmoins le constat le plus important a été la puissance des modèles préentrainés. En effet, dès le départ, les scores obtenus uniquement avec du transfer learning étaient excellent (aux alentours de 98%), ce qui surclasse largement les performances humaines sur une tâche de segmentation. Nous en avons déduit qu'il ne vaut donc pas toujours le coup de collecter massivement de la donnée et d'entrainer son propre modèle. Nous prenons ce fait d'ailleurs avec des pincettes car nous avons conscience que nos modèles ont été testé sur un ensemble de test défini. En outre, nous avons constaté que les approches semi-supervisées ou le rajout d'information présente dans le dataset peut améliorér de manière significative les résultats. Il peut donc être très intéressant de passer du temps sur le prétraitement afin de rajouter le maximum d'information possible.

Par ailleurs, notre modèle avec ces performances peut directement servir afin d'aider la tâche de traduction automatique de manga. D'une part il détecte très bien le texte japonais, ce qui est la première étape de traduction. La suite de ce projet peut être de l'associer à un lecteur de caractère et un traducteur. En outre, l'intêret de la segmentation se révèle principalement au moment d'écrire le texte traduit dans l'image. Pour conclure ce projet nous affichons ci-dessous le résultat du détecteur de texte sur mon manga préféré:

Page du manga One Piece



Page du manga One Piece avec texte segmenté

References

[1] Jochen Laubrock David Dubray. Deep CNN-based Speech Balloon Detection and Segmentation for Comic Books. 1

[2] Rosana Matuk Herrera Julián Del Gobbo. Unconstrained Text Detection in Manga: a New Dataset and Baseline. 1, 2, 3, 4, 5

[3] Atsushi Otsubo Toru Ogawa Yusuke Matsui Koki Tsubota Hikaru Ikuta Kiyoharu Aizawa, Azuma Fujimoto. Building a Manga Dataset "Manga109" with Annotations for Multimedia Applications. 2

[4] Mani Ranjbar William G. Macready Mostafa S. Ibrahim, Arash Vahdat. Semi-Supervised Semantic Image Segmentation with Self-correcting Networks. 4