**Arbitrary Shape Scene Text Detection with Adaptive Text Region Representation**

روش پیشنهادی در این مقاله دارای دو مرحله هست که این دو مرحله شامل:

1. پیشنهاد مکان متن[[1]](#footnote-1)
2. بهبود پیشنهاد‌ها[[2]](#footnote-2)

در مرحله‌ی اول، از شبکه‌ی Text\_RPN استفاده می‌شود که پیشنهاد‌های مکان متن از تصویر ورودی را تولید می‌کند. نقشه‌ ویژگی[[3]](#footnote-3) تصویر ورودی در این بخش به دست می‌آید که در مراحل بعدی از آن استفاده می‌شود. در مرحله‌‌ی دوم، پیشنهادهای مکان متن از طریق یک شبکه تأیید و تصحیح می‌شود. این مرحله شامل طبقه بندی متن / غیرمتن ، رگرسیون کادر محدوده[[4]](#footnote-4) و مکان متن که حاصل از یک شبکه ی RNN است، می‌باشد. سرانجام، خروجی این روش، تشخیص مکان متن در تصویر ورودی است که با چند ضلعی‌هایی در تصویر مشخص می‌شود.

ایده‌ای که در این مقاله مطرح می‌شود این است که برای بازنمایی مکان متن در تصویر، منطقی است که از چند ضلعی‌هایی استفاده شود که نقاط آن متغیر و متناسب با شکل و محدوده‌ی متن باشد. کار ساده‌ای که می‌‌توان انجام داد این است که از نقاط گوشه‌ای موجود بر روی مرز متن برای نمایش مکان متن استفاده کنیم. اما مشکلی که وجود دارد این است که یادگیری و پیدا کردن این نقاط مشکل است. برای حل این مشکل، مسئله را این گونه در نظر می‌گیرند که تقریبا مرز بالا و پایین یک متن با یکدیگر تقارن دارند و کافی است ما زوج نقاطی از مرز بالا و پایین متن را به دست آوریم تا بتوان مکان متن را با یک چند ضلعی نشان داد. اکنون به شرح کامل دو مرحله روش پیشنهادی این مقاله می‌پردازیم.

**پیشنهاد مکان متن**

همان‌طور که قبلا هم اشاره کردیم، در این مرحله برای به دست آوردن پیشنهاد مکان متن از شبکه‌ی Text\_RPN استفاده می‌شود که مشابه‌ی شبکهRPN [[5]](#footnote-5) در Faster R\_CNN است با این تفاوت که شبکه‌ی اصلی آن و سایز anchor‌ها متفاوت است. شبکه‌ی اصلی Text\_RPN شبکه‌ی SE\_VGG16 است که در واقع همان VGG16 است که بلوک‌های squeeze-and-excitation به آن اضافه شده است. علت استفاده از بلوک‌های SE این است که بلوک‌های SE با در نظر گرفتن وابستگی بین کانال‌ها و مدل‌سازی آن، ویژگی های کانال را کالیبره می‌کنند، که به طور قابل توجهی باعث بهبود عملکرد می‌شود. سایز anchor‌های این شبکه 32، 64، 128، 256 و 512 با نسبت های 0.5، 1 و 2 می‌باشد.

**بهبود پیشنهادها**

در این مرحله، از پیشنهادهایی که در مرحله‌ی قبل از تصویر ورودی تولید شده‌اند، به عنوان ورودی به یک شبکه‌ی تصحیح[[6]](#footnote-6) داده می‌شود. این شبکه دارای سه شاخه است. شاخه اول طبقه بندی می‌کند که کاندید به دست آمده دارای متن هست یا نه. شاخه دوم محدوده‌ی متن را خروجی می‌دهد و در نهایت شاخه‌ی سوم چند ضلعی که مکان متن را بازنمایی می‌کند را خروجی می‌دهد. دو شاخه‌ی اول همانند روش‌های قبلی که در این زمینه وجود دارند؛ پیاده‌سازی می‌شود. ورودی شاخه‌ی سوم ویژگی‌هایی هستند که بعد از اعمال ROI pooling بر روی نقشه‌ی ویژگی‌های شبکه‌ی SE\_VGG16 به دست می‌آید. خروجی این شاخه هم نقاط مرزی هستند که مکان متن را نشان می‌دهند. چون تعداد این نقاط برای هر متن متفاوت است، منطقی است که از یک شبکه‌ی RNN استفاده شود که در اینجا از شبکه‌ی LSTM استفاده می‌شود که یک شبکه‌ی معروف از RNN هاست. همان طور که قبلا توضیح دادیم، برای راحتی، نقاط را به صورت زوج‌هایی که بر روی مرز بالا و پایین متن قرار دارند در نظر می‌گیریم که به صورت (xi, yi, xi+1, yi+1) نمایش می‌دهیم. ورودی همه‌ی گام‌های زمانی در LSTM، ویژگی‌های بدست آمده بعد از اعمال ROI Pooling هستند و خروجی هر یک از این گام‌های زمانی یک زوج نقطه از نقاط مرزی هستند. چون تعداد نقاط مرزی برای هر متن متفاوت است از یک برچسب توقف[[7]](#footnote-7) استفاده می کنیم تا زمان توقف را به شبکه اعلام کنیم. به دلیل این که پیش‌بینی برچسب توقف یک مسئله‌ی طبقه بندی هست و پیش‌بینی مختصات نقاط مرزی یک مسئله‌ی رگرسیون است؛ پس بهتر است که هر گام زمانی در شبکه‌ی LSTM دو شاخه خروجی بدین منظور داشته باشد. در نهایت همانند سایر روش‌های تشخیص اشیا، در این روش هم لازم است که از روش [[8]](#footnote-8)NMS استفاده کنیم. که در اینجا از NMS معمولی نمی‌توانیم استفاده کنیم و باید از Polygon NMS استفاده کنیم که محاسبات آن براساس مساحت چند ضلعی نشان دهنده ی مکان متن می‌باشد.

**تابع خطا**

گفتیم که شبکه‌ی Text\_RPN مشابه‌ی شبکه RPN در faster R\_CNN است و خطای آموزش آن هم مشابه همان محاسبه می‌شود. اما باید برای شبکه‌ی تصحیح پیشنهادی، خطای آموزش را محاسبه کنیم که برابر است با مجموع خطای طبقه بندی متن/ غیر‌متن، خطای رگرسیون کادر محدوده، خطای رگرسیون نقاط مرزی و خطای طبقه بندی برچسب توقف/ ادامه. بنابراین می‌توان نوشت:

که در عبارت بالا ، و پارامتری‌هایی برای ایجاد توازن بین ترم‌های خطا هستند که در اینجا مقدار همه برابر 1 است.

در ترم مربوط به خطای طبقه بندی متن / غیرمتن، پارامتر نشان دهنده ی برچسب کلاس است که اگر کلاس متن باشد؛ مقدار آن یک و اگر پس زمینه باشد؛ مقدار آن صفر می‌شود. پارامتر هم نشان دهنده‌ی احتمال کلاس متن و پس زمینه، پس از اعمال تابع softmax است. بنابراین مقدار عبارت برابر است با یعنی منفی لگاریتم خطای کلاس درست است.

در ترم مربوط به خطای رگرسیون کادر محدوده، پارامتر نشان‌دهنده‌ی کادر محدوده‌ی درست است و نشان‌دهنده‌ی مختصات کادر محدوده‌ی پیش‌بینی شده است.

**Efficient and Accurate Arbitrary-Shaped Text Detection with Pixel Aggregation Network**

**د**ر این مقاله یک شبکه به نام PAN ارائه شده است که می‌تواند متن‌ها با شکل‌های دلخواه را در تصویر تشخیص دهد. این شبکه شامل دو مرحله می‌شود:

1. پیش‌بینی مکان متن، kernel و بردارهای شباهت توسط یک شبکه segmentation.
2. بازسازی نمونه‌های کامل متن از kernel های پیش‌بینی شده.

در این مقاله بیان می‌کند که برای این که بهره‌وری را بالا ببریم؛ لازم است که زمان این دو مرحله را کاهش دهیم. بدین منظور یک شبکه‌ی سبک به عنوان شبکه اصلی [[9]](#footnote-9)انتخاب می‌کنند. در اینجا از شبکه‌یResNet18 استفاده می‌شود. اما استفاده از شبکه‌ی سبک دو ایراد دارد. اول این که در استخراج ویژگی ضعیف عمل می‌کند و دوم این که ویژگی‌های بدست آمده دارایreceptive

field کمی است و قابلیت بازنمایی ضعیفی دارد. برای حل این مشکل بعد از شبکه‌ی اصلی، دو ماژول قرار داده‌اند:

1. FPEM [[10]](#footnote-10)
2. FFM[[11]](#footnote-11)

FPEM یک ماژول U شکل است که با استفاده از کانولوشن‌های جداکننده[[12]](#footnote-12) ساخته‌شده‌است. بنابراین FPEM با ترکیب ویژگی‌های سطح پایین و بالا می‌تواند ویژگی‌های مقیاس‌های مختلف را بهبود دهد. علاوه بر این چون FPEM به صورت آبشاری است می‌تواند سبک بودن شبکه اصلی را جبران کند. پس ازFPEM شبکه‌ی FFM قرار می‌گیرد که ویژگی‌های ایجاد‌شده توسط FPEM را با یکدیگر ادغام می‌کند. در آخر، برای بازسازی نمونه‌های متن به طور دقیق، از روش[[13]](#footnote-13)PA استفاده می‌شود که می‌تواند پیکسل‌های متن را برای تصحیح kernel از طریق بردارهای شباهت پیش‌بینی شده، راهنمایی کند.

به صورت خلاصه، نوآوری‌های این مقاله سه مورد زیر می‌باشد:

1. استفاده از FPEM و FFM که هر دو باعث می‌شوند تا ویژگی‌های بهتری بدست آورند.
2. استفاده ازPA .
3. روش پیشنهادی این مقاله توانسته است عملکرد بهتری نسبت به روش‌های قبلی بر روی دو مجموعه‌داده \* و \*که شامل تصاویری با متن‌های خمیده هستند، شود‌. ضمناً inference time این روش نسبت به روش‌های قبلی تغییر نکرده است و روش پیشنهاد شده در این مقاله مکان متن ها را در تصویر به صورت real time انجام می‌دهد.

**FPEM**

FPEM یک ماژول U شکل می‌باشد که شامل دو مرحله است:

1. Up-scale Enhancement
2. Down-scale Enhancement

در بخش اول بهبود به صورت تکراری بر روی ویژگی‌های ورودی با گام های 32، 16، 8 و 4 اعمال می‌شود و خروجی این بخش که هرمی از ویژگی‌ها[[14]](#footnote-14) است را به عنوان ورودی به مرحله‌ی بعدی می‌دهد و این بار بهبود را برعکس بخش قبل از گام 4 تا 32 انجام می‌دهد. به منظور ساخت قسمت اتصال در ماژول FPEM از کانولوشن‌های جداکننده به جای کانولوشن‌های معمولی استفاده شده‌است که می‌تواند با استفاده از محاسبات کمی receptive field را افزایش دهد (به علت وجود depth-wise کانولوشن‌های 3\*3) و همچنین می تواند شبکه را عمیق کند(به علت وجود کانولوشن‌های 1\*1).

اگر بخواهیم ماژول FPEM را با ماژولFPN مقایسه کنیم؛ می‌توانیم یک شباهت و دو تفاوت را برای این دو ماژول نام ببریم. شباهت این دو در این است که هر دوی آن ها می توانند با ترکیب ویژگی‌های سطح پایین و سطح بالا، ویژگی‌های بهتری را بدست آورند. اما تفاوت آن ها در این است که FPEM یک ماژول آبشاری است که با افزایش تعداد این ماژول‌ها در آبشار () می‌تواند ویژگی‌ها را در مقیاس‌های مختلف به صورت مناسب‌تری ترکیب کند و همچنین receptive field را افزایش دهد. تفاوت دوم در این است که از نظر هزینه محاسباتی ماژول FPEM ارزان‌تر است و علت آن هم استفاده از کانولوشن‌های جدا کنندهاست.

**FFM**

این ماژول قرار است خروجی مرحله‌ی قبل که تا هرم ویژگی شامل است را با یکدیگر ترکیب کند. عمق هر هرم ویژگی متفاوت از بقیه هرم‌ها می‌باشد. ساده‌ترین روشی که برای ترکیب این هرم ویژگی‌ها به ذهن می‌رسد این است که همه‌ی ویژگی‌ها را upsample کنیم و سپس آن‌ها را با هم concatenate کنیم. اما این روش یک ایراد دارد و ایراد آن هم این است که تعداد کانال‌های خروجی زیاد می‌شود. راه‌حل این مشکل این است که ویژگی‌هایی که از نظر مقیاس با یکدیگر متناظر هستند به صورت دو به دو به هم ترکیب شوند و سپس عمیات upsampling و concatenation را انجام دهیم. در این صورت تعداد کانال‌ها از به تا کاهش می‌یابد.

**Pixel Aggregation**

همانطورکه می‌دانید معمولا برای یک متن در تصویر چندین کاندید نزدیک به هم به دست می‌آید که ما باید از این کاندید‌ها یکی را انتخاب کنیم. در اینجا برای این کار از روش PA استفاده می‌کنیم. ایده‌ی این روش استفاده از kernel است که به خوبی نشا‌ن‌دهنده‌ی یک نمونه خاص از متن در تصویر است. اما یک kernel یک کاندید کامل از متن نیست. بنابراین لازم است که کاندید کامل متن را از روی kernel بازسازی کنیم و برای این کار باید پیکسل‌های موجود در کاندید‌های متن را با kernel ادغام کنیم. این ایده از روش clustering الهام گرفته شده‌است که هر نمونه از متن یک cluster است و kernel آن نمونه از متن، مرکز این cluster می باشد. پیکسل‌های کاندید‌های متن هم نمونه‌هایی هستند که باید cluster شوند. بنابراین فاصله‌ی پیکسل‌های متناظر با یک kernel باید کوچک باشد. در زمان آموزش، خطای تجمیع به صورت زیر تعریف می‌شود:

که N تعداد نمونه‌های متن است و برابر است با iامین نمونه از متن. هم نشان‌دهنده‌ی فاصله‌ی پیکسل pتا کرنل که مربوط به نمونه متن است. یک عدد ثابت است که به صورت تجربی در اینجا برابر 0.5 قرار داده شده است. بردار مشابهت برای پیکسل p و بردار مشابهت برای کرنل است که به صورت زیر محاسبه می‌شود:

علاوه بر خطای تجمیع لازم است که مراکز cluster ها متمایز باشند به همین دلیل خطای دیگری به نام خطای تمایز نیز تعریف می‌کنیم که به صورت زیر بیان می‌شود:

سعی می‌کند که فاصله‌ی مراکز clusterها کمتر از نباشد که در اینجا به صورت تجربی مقدار آن را برای همه‌ی آزمایش‌ها برابر 3 قرار داده شده‌است.

در مرحله آزمایش، از بردار شباهت برای هدایت پیکسل ها در کاندید متن به کرنل مربوطه استفاده می‌کنیم. بنابراین به صورت دقیق مراحل پس پردازش[[15]](#footnote-15) به شرح زیر است:

* 1. یافتن مؤلفه‌های متصل در kernel’s segmentation result که هر کدام از این مولفه‌های متصل یک kernel جداگانه محسوب می‌شوند.
  2. برای هر کرنل ، پیکسل همسایه‌ی p را با کاندید متن ادغام می‌کند در حالی که فاصله اقلیدسی از بردارهای شباهت آن‌ها کمتر از d است.
  3. تکرار مرحله 2 تا زمانی که هیچ پیکسل همسایه‌ای واجد شرایط نباشد.

**تابع هزینه[[16]](#footnote-16)**

**باید تکمیل شود.**

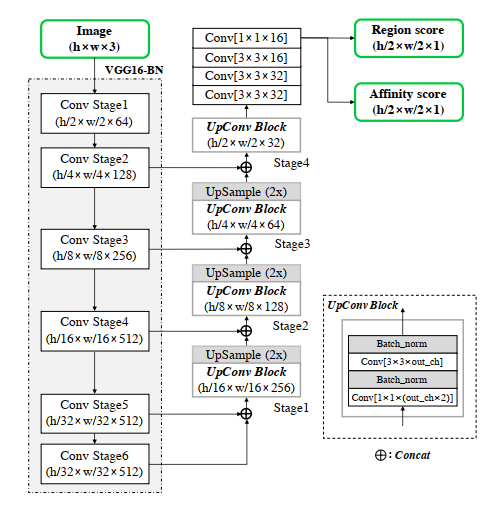
**Character Region Awareness for Text Detection**

روشی که در این مقاله پیشنهاد شده‌است به این صورت است که مکان هر کاراکتر را به صورت جداگانه تشخیص دهد و سپس با ترکیب کاراکتر‌های تشخیص داده‌شده، مکان متن را مشخص کنند. اسم این روش را [[17]](#footnote-17)CRAFT نامیدند. خوبی این روش این است که می‌تواند همه اشکال متن، متن‌های منحنی شکل و بد شکل را تشخیص دهد.

این روش از یک شبکه کانوولوشنی استفاده می‌کند که دو امتیاز را خروجی می‌دهد: امتیاز ناحیه هر کاراکتر[[18]](#footnote-18) و امتیاز نزدیکی کاراکترها[[19]](#footnote-19). امتیاز ناحیه‌ هر کاراکتر، مکان هر کاراکتر را مشخص می‌کند و امتیاز نزدیکی دو کاراکتر، برای گروه‌بندی هر کاراکتر در یک نمونه استفاده می‌شود. برای جبران مشکل مشخص نبودن مکان کاراکترها در مجموعه‌داده اصلی، این مقاله یک یادگیری با نظارت ضعیف[[20]](#footnote-20) را برای تخمین مکان صحیح کاراکترها با استفاده از مکان واقعی کلمات مشخص شده در مجموعه‌داده پیشنهاد می‌دهد.

**معماری شبکه**

شبکه پایه این روش، یک شبکه کاملا کانوولوشنی بر اساس VGG-16 است. این شبکه دارای skip connection در قسمت رمزگشا[[21]](#footnote-21) می‌باشد که شبیه شبکه U-Net، ویژگی‌های سطح پایین را نیز در نظر می‌گیرد. خروجی این شبکه دارای دو کانال است که شامل امتیاز ناحیه هر کاراکتر و امتیاز نزدیکی دو کاراکتر می‌باشد. ساختار این شبکه را می توان در زیر مشاهده کرد:



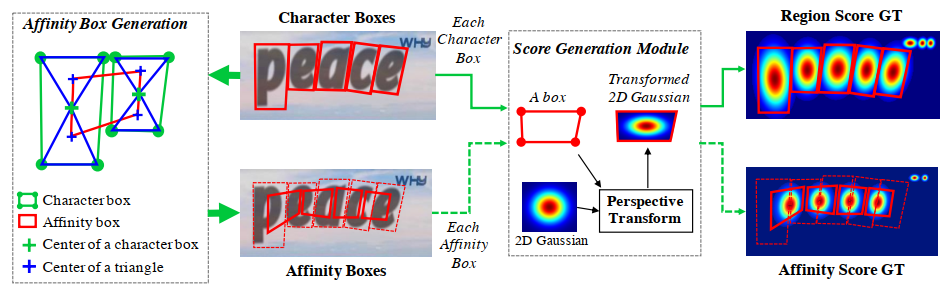
**آموزش**

یادگیری این روش دارای دو بخش زیر است :

1. Ground Truth Label Generation

در این بخش به ازای هر عکس، باتوجه به امتیاز ناحیه هر کاراکتر و امتیاز نزدیکی دو کاراکتر، برای کادرمحدوده هر کاراکتر یک برچسب تولید می‌شود. امتیاز ناحیه هر کاراکتر احتمال مرکز کاراکتر بودن هر پیکسل را نشان می‌دهد و امتیاز نزدیکی دو کاراکتر برای هر پیکسل، احتمال اینکه مرکز بین دو کاراکتر مجاور باشد را بازنمایی می‌نماید.

مراحل تولید برچسب را در شکل زیر می‌توان دید. احتمال‌ها در این شکل به صورت heatmap گوسی نمایش داده شده‌اند.



محاسبه تابع توزیع گوسی به ازای هر پیکسل در کادر محدوده، بسیار زمانبر است.

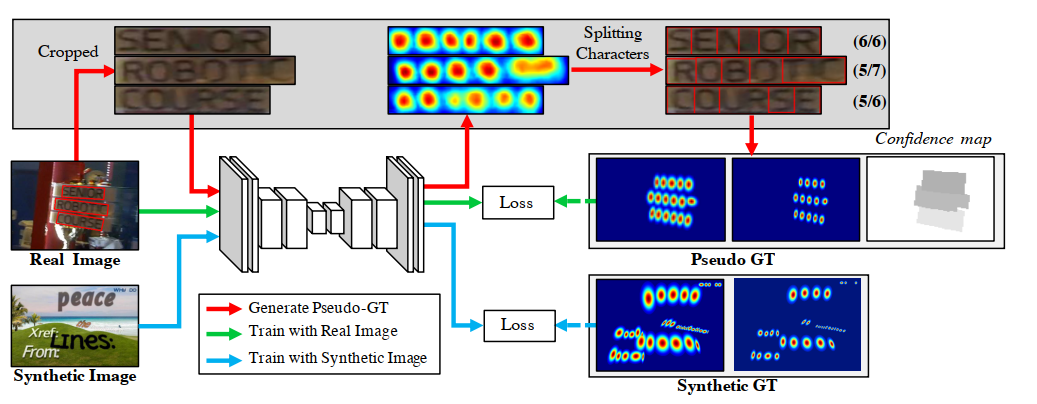
برای تخمین زدن و تولید ground truth برای امتیاز ناحیه هر کاراکتر و امتیاز نزدیکی دو کاراکتر، از سه مرحله استفاده می‌شود :

1. آماده‌سازی یک نقشه گوسی دو بعدی
2. محاسبه یک تبدیل perspective بین نقشه گوسی ناحیه‌ای و محدوده هر کاراکتر
3. نگاشت هر توزیع گوسی به محدوده هر ناحیه

تکمیل شود.

1. یادگیری با نظارت ضعیف

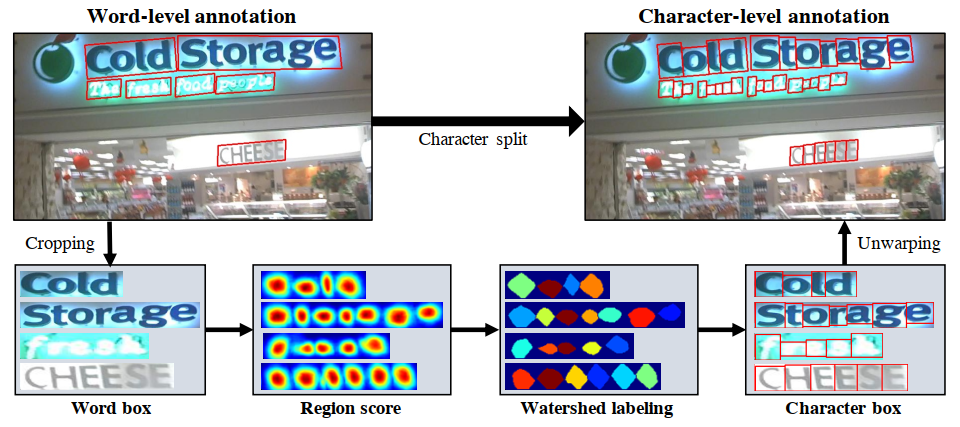
در این بخش، از تصاویر واقعی که در آن ها فقط محدوده متن‌ها مشخص شده‌است، استفاده کرده و محدوده کاراکتر‌های آن متن را به دست می‌آورد.



روند جداسازی کاراکترها دارای 4 مرحله است:

1. بخش‌های دارای متن از تصویر ورودی بریده می‌شود.
2. مدل برای پیش‌بینی امتیاز ناحیه هر کاراکتر آموزش می‌بیند.
3. الگوریتم watershid برای جداسازی ناحیه‌های کاراکتر استفاده می‌شود.
4. مختصات محدوده کاراکترها با استفاده از تبدیل معکوس مرحله جداسازی، بر روی تصویر اصلی مشخص می‌شود.

این مراحل را در تصویر زیر می‌توان مشاهده کرد :



به ازای هر نمونه w از مجموعه‌داده آموزشی، R(w) و l(w) به ترتیب نشان دهنده ناحیه هر کلمه و طول آن کلمه می‌باشد. در طول روند جداسازی کاراکترها، می توان محدوده و طول متناظر هر کاراکتر (lc (w)) را به دست آورد. سپس Sconf(w) را که confidence score برای هر نمونه است به شکل زیر تعریف می‌شود :

سپس به ازای هر پیکسل تصویر ورودی، confidence map را که Sc می‌نامند، به شکل زیر محاسبه می‌شود:

که در آن p، هر پیکسل از ناحیه R(w) می باشد و در نهایت Loss به شکل زیر تعریف می‌شود :

که در آن Sr\*(p) و Sa\*(p) به ترتیب نشان‌دهنده psudo-ground truth امتیاز ناحیه هر کاراکتر و امتیاز نزدیکی دو کاراکتر می باشد و Sr(p) و Sa(p) به ترتیب نشان‌دهنده امتیاز ناحیه هر کاراکتر و امتیاز نزدیکی دو کاراکتر پیش‌بینی شده هستند.

یک خوبی روش CRAFT این است که هیچ پس پردازشی مثل NMS ندارد. همچنین در این روش یک چند ضلعی محصور برای هر متن موجود در تصویر می توان تولید کرد.

1. Text proposal [↑](#footnote-ref-1)
2. Proposal refinement [↑](#footnote-ref-2)
3. Feature map [↑](#footnote-ref-3)
4. Bounding box [↑](#footnote-ref-4)
5. Region Proposal Network [↑](#footnote-ref-5)
6. Refinement network [↑](#footnote-ref-6)
7. Stop label [↑](#footnote-ref-7)
8. Non Maximum Suppression [↑](#footnote-ref-8)
9. Backbone network [↑](#footnote-ref-9)
10. Feature Pyramid Enhancement Module [↑](#footnote-ref-10)
11. Feature Fusion Module [↑](#footnote-ref-11)
12. Separable convolution [↑](#footnote-ref-12)
13. Pixel Aggregation [↑](#footnote-ref-13)
14. Feature pyramid [↑](#footnote-ref-14)
15. Post processing [↑](#footnote-ref-15)
16. Loss function [↑](#footnote-ref-16)
17. Character Region Awareness For Text detection [↑](#footnote-ref-17)
18. Region score [↑](#footnote-ref-18)
19. Affinity score [↑](#footnote-ref-19)
20. Weakly supervised [↑](#footnote-ref-20)
21. Decoding [↑](#footnote-ref-21)