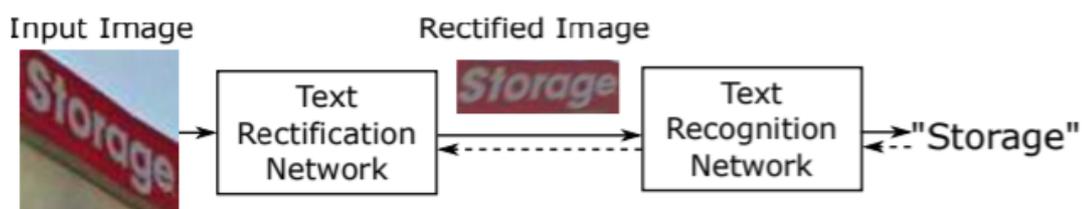


ASTER: An Attentional Scene Text Recognizer with Flexible Rectification

정리

1. Introduction

- 딥러닝 기반 text recognition model 등장(ex. CRNN)
- 하지만 위 모델들은 irregular text(수평적이지 않고, 정면이 아니고, 휘어져있고 등등)를 인식하는데에는 한계가 존재
- 여기서는 이 irregular text 문제를 해결하기 위한 방안으로 ASTER를 제안



▼ 크게 두 파트로 나뉜다.

- **the rectification network** : irregular text를 담은 image → text 정제화(좀 퍼지게)
 - 매개화된(parameterized) Thin-Plate Spline(TPS)을 변환(함수)로 사용
 - 유연해서(flexible) 다양한 text irregularities를 다루는데 이점 존재
 - input image로부터 TPS 파라미터 예측, 해당 image에 적용
 - Spatial Transformer Networks(STN) 을 이용해 network 전체를 train
- **the recognition network** : rectified image → character sequence(문자열)
 - attention mechanism 기반
 - 고전적인 단방향 decoder → 양방향(bidirectional) decoder로 개선
 - 서로 정반대의 decoding 방향을 가지는 두개의 decoder로 구성
- Aster가 regular, irregular text 모두에 대해 압도적인 성능을 보인다고 주장
 - detected box를 필터링하고 정제함으로써 text detection능력 향상

▼ 저자가 주장하는 Aster의 장점(Contribution)

1. irregular text recognition 문제를 해결할 수 있음
2. text recognition에서 attentional seq2seq 모델 사용 + 여기에 bidirectional decoder 이용했다

3. Aster를 이용해서 text detect하는 능력을 향상하는 방법을 제안

▼ 저자가 주장하는 Aster가 RARE보다 나은 이유(Aster는 RARE를 개선한 논문임)

1. rectification network를 수정 → rectification 성능 향상(↔ 우리가 좀 더 irregular한 text를 더 잘 인식할 수 있다.)
2. 기존의 recognition decoder를 bidirectional decoder로 개선
 - bidirectional로 개선한 이유 : 양방향으로 종속성을 활용하기 위해서(??→ maybe... text 추론할 때 두 방향에서 얻은 추론 정보를 활용하기 위해서 라는 의미 같다)
3. end-to-end text recognition에서 ASTER를 적용해서 입증했다. (↔ 실험했더니 우리가 짱짱맨이야)

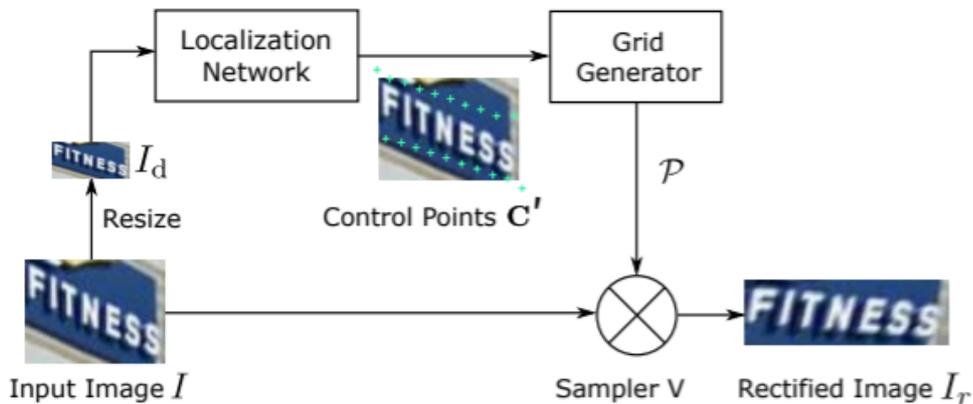
2. related work(생략)

3. Model

- 모델 구성
 - text rectification network + text recognition network (Section 3.1, 3.2)
- training strategy는 (Section 3.3)

3.1 Rectification Network

▼ 내용

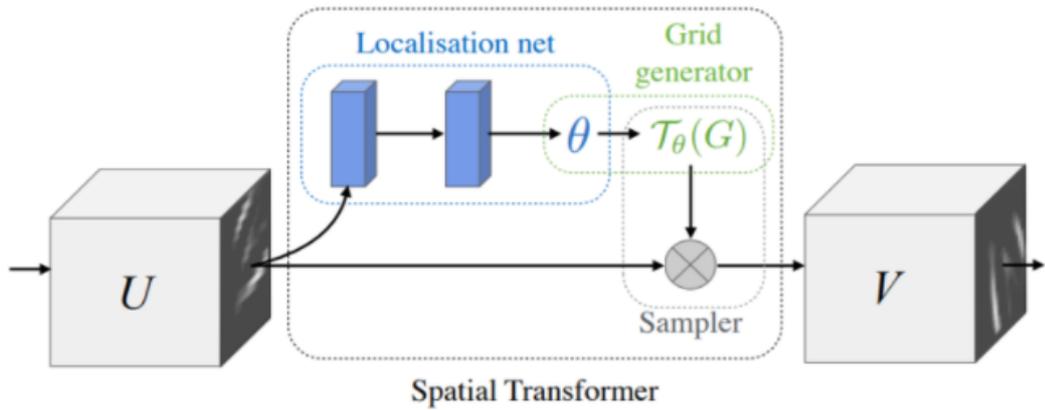


- 이미지 조정(rectify)하는 network
- Thin-Plate-Spline[8](TPS)을 transformer로 이용함
 - 다른 기술보다 유연하다고 함(flexible). 특히 affine변환(점, 직선, 평면, 평행선을 보존하는 변환), projection 처리에 유용
 - → 결론 : TPS가 irregular text 처리에 짱짱맨

▼ Spatial Transformer Network(SPN) 기반

(참조 블로그 : [https://m.blog.naver.com/PostView.naver?](https://m.blog.naver.com/PostView.naver?blogId=worb1605&logNo=221580830661&proxyReferer=http:%2F%2F221.147.67.207%2F)

[blogId=worb1605&logNo=221580830661&proxyReferer=http:%2F%2F221.147.67.207%2F](https://m.blog.naver.com/PostView.naver?blogId=worb1605&logNo=221580830661&proxyReferer=http:%2F%2F221.147.67.207%2F))



(어찌보면 비슷한 느낌이 든다)

- spatial transformation을 learnable network layer로 모델링했음(내 생각엔 위 그림의 localisation net을 의미하는 듯하다. 그리고 localisation net은 이 논문의 localization network에 대응되는 것이고)
- localization network로 control points 예측 → grid generator & sampler 를 거치면서 최종적으로 Rectified Image 뽑아냄

▼ 개인감상

- 이미지 (fig 3)에서 변형한 뒤에 전체적으로 약간 해상도가 낮아지는 현상이 벌어지는 것 같다. → 학습에 해가 될까?
- 확인하는 방법에 대해서 어제 이야기가 나왔는데 고려해봐야 할 수도(중간에 이미지를 시각화 따오기 등)

▼ 3.1.1 Localization Network

▼ TPS transformation

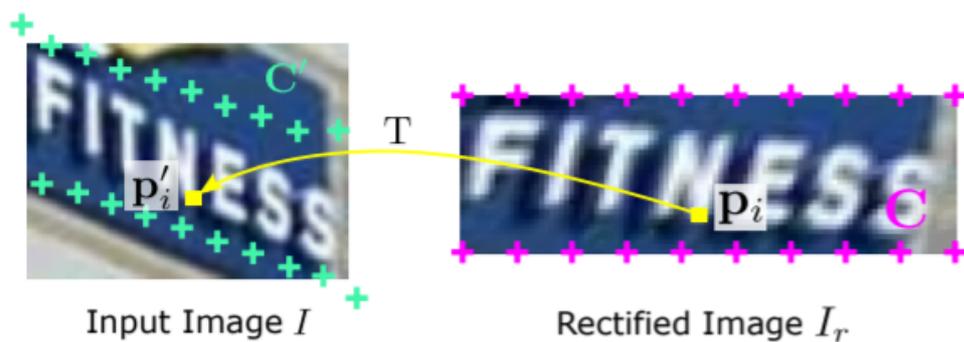
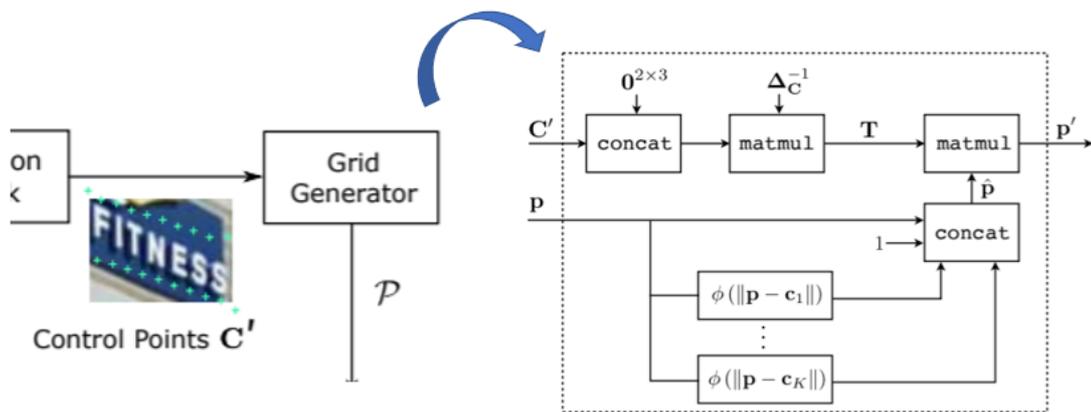


Fig. 5. Text rectification with TPS transformation. Crosses are control points. The yellow arrow represents the transformation T , connecting a point p_i and its corresponding point p'_i .

- K개의 control points로 구성된 두 개의 set으로 정의됨

- output 이미지(Rectified Image)의 control points : 고정된 위치, 일정 간격을 유지
 - 고로, input image의 control points는 text 위 아래 가장자리를 따라 예측될 때 TPS 변환은 regular text로 수정된 이미지를 출력
- 여기서 문제는 **input image의 control points를 예측하는 것** → convolutional neural network로 예측 수행
 - Localization Network 내에서 I_d (input image I 로부터 downsampling 해서 얻은 결과물)로부터 control points C' 예측(regress라고 되어 있는 것으로보아 회귀 분석? 하는 듯하다)
 - Localization Network = few convolution layers(사이사이에 max pooling layers가 끼었다)
- 모든 rectification network는 미분가능 → 고로, Localization Network는 back propagation진행하면서 학습된다.

▼ 3.1.2 Grid Generator



- transformation계산 + I_r 에 존재하는 모든 픽셀 위치값에 transformation적용
 - 이렇게 해서 I 에 대한 sample grid $\mathcal{P} = \{p_i\}$ 생성

▼ 수식 → 이해 부족....

- A 2D TPS transformation : $2 \times (K + 3)$ matrix

$$\mathbf{T} = \begin{bmatrix} a_0 & a_1 & a_2 & \mathbf{u} \\ b_0 & b_1 & b_2 & \mathbf{v} \end{bmatrix}$$

$$\mathbf{u}, \mathbf{v} \in \mathbb{R}^{1 \times K}$$

- 2D point $p = [x_p, y_p]^T$ (그림으로 치자면 output 이미지)

▼ 3.1.3 Sampler

- 앞서 계산한 sample grid P 와 input image I 를 바탕으로 rectified image(I_r) 생성

$$I_r = V(P, I)$$

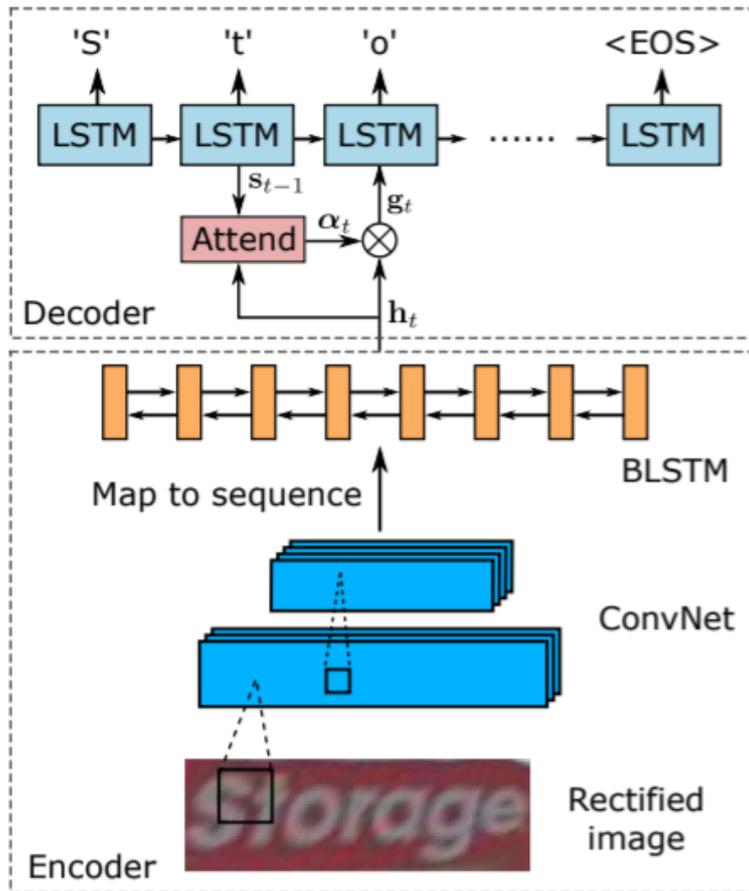
- sampler는 미분가능 → 고로, I_r 에서 P 로 back propagate 가능

▼ 3.1.4 Comparison with STN [28] and RARE [55]

- STN하고 RARE와 비교했을 때 우린 2가지 향상을 이루었음
 1. STN과 달리 localization network와 sampler에 다른 사이즈의 image 사용함
 - localization network → 원본 이미지의 downsampling 한 이미지(원본보다 작음) → 예측에 필요한 파라미터 수를 줄일 수 있음
 - sampler의 경우 원본 이미지에서 작업 → output으로 나오는 rectified 이미지의 quality 보존 가능
 2. RARE와 달리 C' (input image의 control points)를 제한하지 않음
 - 마지막 fully-connected layer에 tanh 활성화 함수(activation function)이용함
 - tanh없으면 control points는 image 경계를 벗어날 수 있음 → 유효한 샘플링을 위해 clipping이 sampler에서 수행한다.(????)
 - 결론 : 가중치 초기화에 안정성을 주고 성능향상에 좋다.....

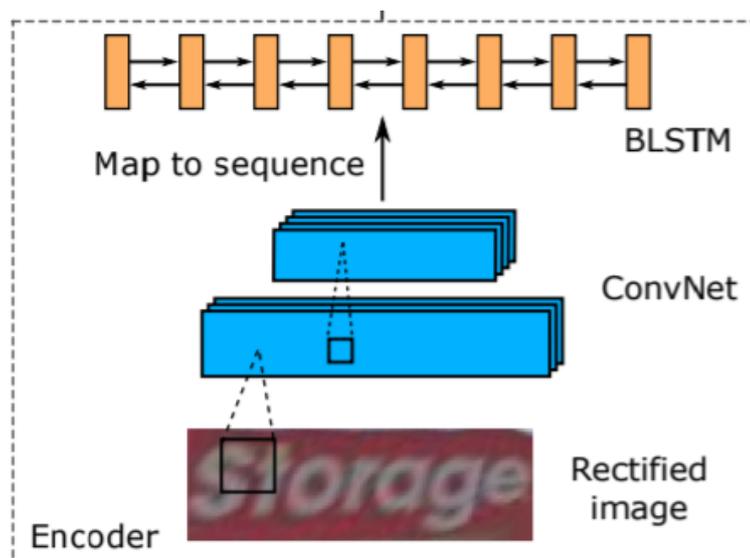
3.2 Recognition Network

- rectified image(수정된 이미지) → character sequence 예측
- end-to-end trainable
- 이전 연구에서 제안한 아이디어의 핵심 : CTC(Connectionist Temporal Classification)
 - 장점 : 수평 문자 배치와 간격에 민감하게 반응하지 않는 미분가능한 loss function을 제공 → end to end trainable sequence recognition 가능하게 함
 - 단점 : output 문자들 간에 연관성(dependencies)을 제공하지 않음 → (의역)다른 언어 모델 (lexicon)을 언어 순서를 파악하는 데 이용해서 recognition 진행
- 그래서 우리는....
 - **bidirectional decoder로 확장한 seq2seq 모델** 이용함
 - 순환 신경망(recurrent neural network)로 seq2seq 결과 도출 → 글자 간 연관성 (dependencies)를 파악함 → 고로, recognition 에 language modeling 결합 가능함(→ 해석 : 우린 이전 연구와 다르게 다른 언어 모델에 의존하지 않아도 된다)
 - bidirectional decoder가 양방향으로 글자 간 연관성(dependencies)을 파악(뒤에 읽어보니까 왼→오, 왼 ←오 두가지 inference 진행)
- network = encoder + decoder



▼ 3.2.1 Encoder: Convolutional-Recurrent Neural Network

- rectified image 에서 feature map을 뽑아낸다.(convolutional layer(ConvNet)을 이용)



▼ ConvNet

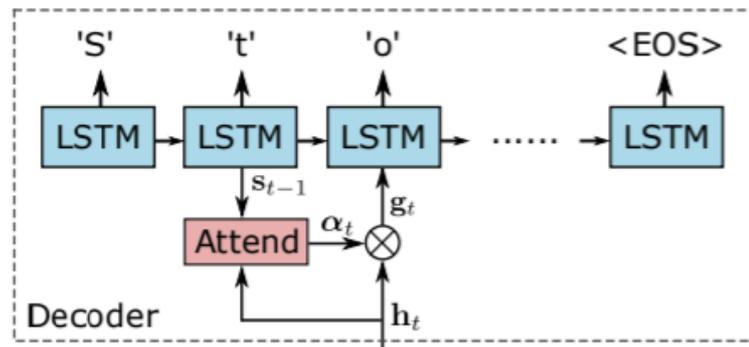
- convolutional layer(ConvNet)은 feature map이 H(height)= 1가지도록 설계

- feature map -> row축을 따라 자르면서 feature sequence(또 다른 feature map)로 변형
 - shape = $h_{conv} * w_{conv} * d_{conv}$ (h=height, w=width, d=depth)
- 잘라서 얻은 feature map → 일련의 w_{conv} 벡터(이때 dimension = $h_{conv} * d_{conv}$)
- 위의 ConvNet을 거쳐 얻은 feature들은 receptive field에 제한적
 - 그래서 feature context를 확장하기 위해 multi-layer Bidirectional LSTM (BLSTM) network 도입
 - BLSTM으로 feature sequence의 넓은 범위의 연관성을 양방향 분석을 통해 파악
- encoder output H = new feature sequence (길이 = w_{conv})

$$\mathbf{H} = [\mathbf{h}_1, \dots, \mathbf{h}_n], \text{ where } n = w_{conv}.$$

▼ 3.2.2 Decoder: Attentional Sequence-to-Sequence Model

- encoder의 output(feature sequence) → character sequence
 - decoder의 input, output모두 가변길이



▼ attentional seq2seq model 기반

- 매 decoder step마다 encoder의 output에 접근함 & 디버깅 및 분석에 용이함
- 단방향 순환 네트워크
- T 번 step을 반복한다고 할 때, 길이 T 인 symbol sequence 생성 ($\rightarrow (y_1, \dots, y_T)$)
- ▼ t 번째 단계 (그림에서는 'o'를 예측하는 부분)
 - encoder output $H \rightarrow$ 1글자 or EOS 예측
 - 매 step마다 decoder는 a_t (attentional weights의 벡터)를 계산
 - a_t 로 encoder output(H)의 각 요소의 importance(의역 : 어떤 요소에 집중해야 하는가)를 명시
 - 수식

$$e_{t,i} = \mathbf{w}^\top \tanh(\mathbf{W}\mathbf{s}_{t-1} + \mathbf{V}\mathbf{h}_i + b)$$

$$\alpha_{t,i} = \exp(e_{t,i}) / \sum_{i'=1}^n \exp(e_{t,i'})$$

s_{t-1} : internal state(위 그림의 't'를 예측하는 부분)

y_{t-1} : 이전 step(t-1)에서 예측한 symbol(위 그림의 't')

a_t : attentional weights의 벡터($a_{t,i}$ 는 벡터의 i번째 원소로 추정됨)

$\mathbf{w}, \mathbf{W}, \mathbf{V}$ = trainable weights

- a_t 와 H 를 선형적으로 결합(원소곱인것으로 추측) → glimpse(g_t)
 - 전체 context의 일부를 표현함?나타냄?(describe)

$$\mathbf{g}_t = \sum_{i=1}^n \alpha_{t,i} \mathbf{h}_i$$

- 계산한 g_t 는 LSTM의 input으로 들어감 →

$$(\mathbf{x}_t, \mathbf{s}_t) = \text{rnn}(\mathbf{s}_{t-1}, (\mathbf{g}_t, f(y_{t-1})))$$

$(g_t, f(y_{t-1}))$: g_t 와 y_{t-1} 의 one hot embedding($f(y_{t-1})$)을 이어붙인 것
 rnn : LSTM, GRU같은 순환 신경망 모듈의 step function(이 논문에서는 LSTM)

- 위에서 계산된 x_t 는 t번째 step의 symbol(y_t)을 예측하는데 이용됨

$$p(y_t) = \text{softmax}(\mathbf{W}_o \mathbf{x}_t + b_o)$$

$$y_t \sim p(y_t)$$

- y_{t-1} 가 계산에 이용되기 때문에 decoder는 output (일련의 문자들) 간의 연관성을 파악할 수 있게됨 → 이런 점이 기존 연구처럼 다른 언어 모델을 빌리지 않고도 language prior를 recognition에 반영할 수 있게 함
- beam search(k=5) 사용 → (요약, 약간 의역 : 매 단계마다 나온 결과 중 가장 잘 나온 결과물 (높은 확률) 5개를 보존하고 나머지 버림)

▼ 3.2.3 Bidirectional Decoder

- (기존) seq2seq decoder : 글자 간 연관성을 파악하지만 한 방향으로만 파악함
 - 이렇게 하면 오답을 예측할 가능성이 존재.(ex. 대문자 I(i), 소문자 l(L)은 구분하기 힘든 경우가 많아 한방향만 보고 판단하기에는 무리가 있음 → 왼 → 오 / 오 → 왼 모두 볼 필요가 있다.
- 두 방향에서 본 글자 간 연관성을 이용하기 위해 bidirectional decoder 도입했다
 - 그냥 두개 decoder가 있는데 하나는 왼 → 오/ 하나는 오 → 왼으로 이동

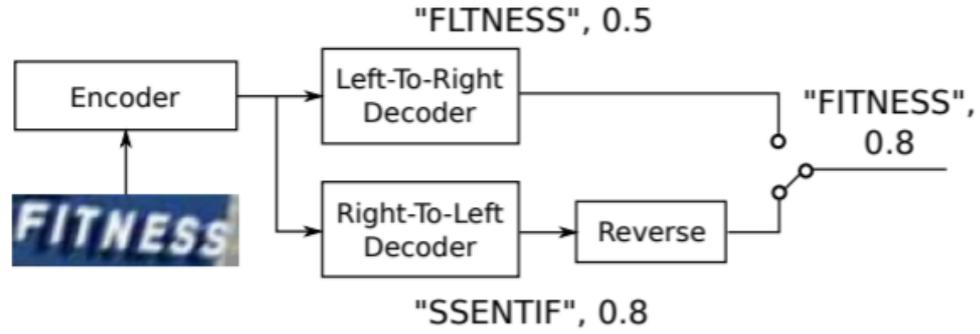


Fig. 8. Bidirectional decoder. "0.5" and "0.8" are recognition scores.

- 두 decoder로 부터 나온 두 개의 결과 중 가장 높은 recognition score 쪽을 채택
- recognition score를 계산한 방법 : 모든 예측 symbol의 log softmax score의 합

3.3 Training

- loss

$$L = -\frac{1}{2} \sum_{t=1}^T (\log p_{ltr}(y_t|I) + \log p_{rtl}(y_t|I))$$

$(y_1, \dots, y_t, \dots, y_T = \text{groundtruth})$

- $L = \text{왼} \rightarrow \text{오}$ 방향 decoder와 $\text{오} \rightarrow \text{왼}$ 방향 decoder 각각의 loss의 평균
- 모든 layer의 weights는 랜덤으로 주었다.(localization network 빼고)
 - localization network 상에서 TPS transformation이 control points로 계산되기 때문에 랜덤으로 초기화시켜버리면 control points도 랜덤으로 놓여질 수 있기 때문에 output image(I_r)가 잘못 나올 수 있다.
 - 그래서 마지막 fully-connected layer(fc2)를 초기화해서 training 처음부터 왜곡된 결과가 나오지 않게 했다.
 - fc2의 weights는 0으로 설정, biases는 특정 값(input image의 control points와 output image control points가 서로 대응될 수 있게 하는)으로 설정

4. Experiments

▼ 4.1 Settings

▼ 4.1.1 datasets

Synth90k, SynthText, IIIT5k-Words, Street View Text, ICDAR 2003, ICDAR 2013, ICDAR 2015 Incidental Text, SVT-Perspective, CUTE80

▼ 4.1.2 Text Rectification Network

Network 들어가기 전 Resize(64×256)

▼ 4.1.3 Text Recognition Network

TABLE 1

Text recognition network configurations. Each block is a residual network block. 's' stands for stride of the first convolutional layer in a block. "*" means dynamic output length. "Out Size" is feature map size for convolutional layers (height×width) and sequence length for recurrent layers. "Att. LSTM" stands for attentional LSTM decoder. Two decoders are instantiated and work in parallel.

	Layers	Out Size	Configurations
Encoder	Block 0	32 × 100	3 × 3 conv, s 1 × 1
	Block 1	16 × 50	$\begin{bmatrix} 1 \times 1 \text{ conv, } 32 \\ 3 \times 3 \text{ conv, } 32 \end{bmatrix} \times 3, s 2 \times 2$
	Block 2	8 × 25	$\begin{bmatrix} 1 \times 1 \text{ conv, } 64 \\ 3 \times 3 \text{ conv, } 64 \end{bmatrix} \times 4, s 2 \times 2$
	Block 3	4 × 25	$\begin{bmatrix} 1 \times 1 \text{ conv, } 128 \\ 3 \times 3 \text{ conv, } 128 \end{bmatrix} \times 6, s 2 \times 1$
	Block 4	2 × 25	$\begin{bmatrix} 1 \times 1 \text{ conv, } 256 \\ 3 \times 3 \text{ conv, } 256 \end{bmatrix} \times 6, s 2 \times 1$
	Block 5	1 × 25	$\begin{bmatrix} 1 \times 1 \text{ conv, } 512 \\ 3 \times 3 \text{ conv, } 512 \end{bmatrix} \times 3, s 2 \times 1$
	BiLSTM 1	25	256 hidden units
	BiLSTM 2	25	256 hidden units
Decoder	Att. LSTM	*	256 attention units 256 hidden units
	Att. LSTM	*	256 attention units 256 hidden units

4.2 Experiments on Text Rectification

4.2.1 Effect of Rectification

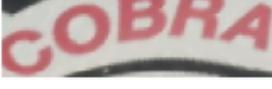
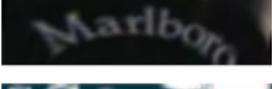
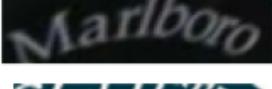
- Rectification 과정이 있으면 결과가 좋더라

TABLE 2
Recognition accuracies with and without rectification.

Variants	IIIT5k	SVT	IC03	IC13	SVTP	CUTE
Without Rect.	91.93	88.76	93.49	89.75	74.11	73.26
With Rect.	92.67	91.16	93.72	90.74	78.76	76.39

+ 우리가 RARE보다 더 잘 recognition 함

TABLE 3
Rectified images and recognition results by [55] and by ASTER.
Recognition errors are marked by red characters.

By [55]	By ASTER	By [55] By ASTER
		w indow wyndham
		optimute optimum
		coera cobra
		warehouse warehouse
		tallob s tailors
		maribon marlboro
		scotting coll s

마지막 fully-connected layer를 zero weights로 주고 특정 biases로 설정(called identity)한 경우가 random하게 설정한 경우 좀더 학습이 잘 되고, 정확도가 올라갔다.

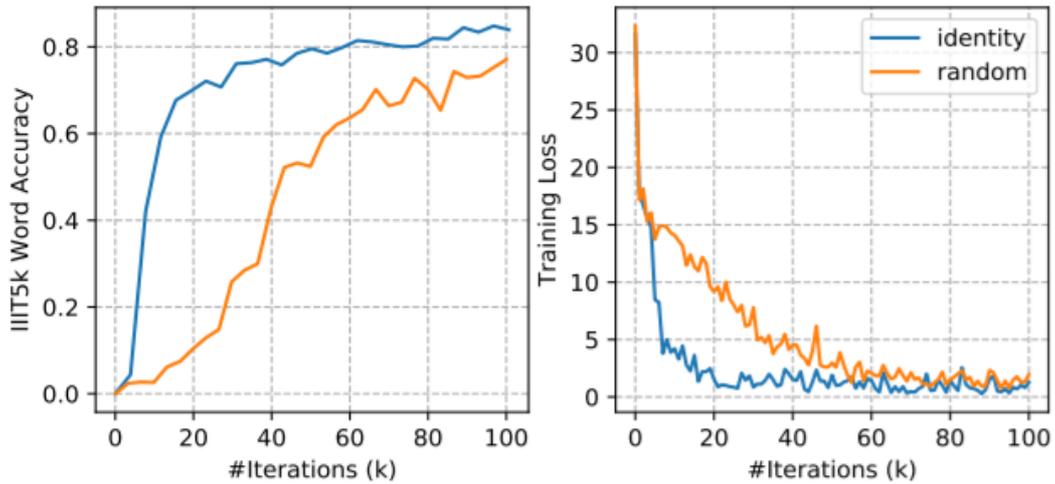


Fig. 10. Word accuracies (left) and training losses (right) with different model initialization schemes.

4.2.1 Effect of Recognition

L2R : left-to-right order

R2L : right-to-left order

Bidirectional : the bidirectional decoder.

정확도가 bidirectional 경우가 더 높게 나온다.

TABLE 5
Recognition accuracies with different decoders. **L2R** and **R2L** stand for left-to-right and right-to-left, respectively.

Variants	IIT5k	SVT	IC03	IC13	SVTP	CUTE
L2R	91.93	88.76	93.49	89.75	74.11	73.26
R2L	91.43	89.96	92.79	89.95	73.95	74.31
Bidirectional	92.27	91.57	93.60	90.54	74.26	74.31

Aster-A, B : ASTER와 차이점 → ConvNet, training data 차이

A : RARE 가 수행했던 실험 + IIT5k(regular text만 존재)

B : ???

Aster : rectification + bidirectional decoder

TABLE 6

Recognition results comparison. "50", "1k", "Full" are lexicons. "0" means no lexicon. *The conference version of this paper. "90k" and "ST" are the Synth90k and the SynthText datasets, respectively. "ST+" means including character-level annotations. "Private" means private training data.

Methods	ConvNet, Data	IIT5k			SVT		IC03			IC13	IC15	SVTP	CUTE
		50	1k	0	50	0	50	Full	0	0	0	0	0
Wang <i>et al.</i> [60]	-	-	-	-	57.0	-	76.0	62.0	-	-	-	-	-
Mishra <i>et al.</i> [44]	-	64.1	57.5	-	73.2	-	81.8	67.8	-	-	-	-	-
Wang <i>et al.</i> [62]	-	-	-	-	70.0	-	90.0	84.0	-	-	-	-	-
Bissacco <i>et al.</i> [7]	-	-	-	-	-	-	90.4	78.0	-	87.6	-	-	-
Almazan <i>et al.</i> [2]	-	91.2	82.1	-	89.2	-	-	-	-	-	-	-	-
Yao <i>et al.</i> [67]	-	80.2	69.3	-	75.9	-	88.5	80.3	-	-	-	-	-
Rodríguez-Serrano <i>et al.</i> [52]	-	76.1	57.4	-	70.0	-	-	-	-	-	-	-	-
Jaderberg <i>et al.</i> [29]	-	-	-	-	86.1	-	96.2	91.5	-	-	-	-	-
Su and Lu [56]	-	-	-	-	83.0	-	92.0	82.0	-	-	-	-	-
Gordo [16]	-	93.3	86.6	-	91.8	-	-	-	-	-	-	-	-
Jaderberg <i>et al.</i> [26]	VGG, 90k	97.1	92.7	-	95.4	80.7	98.7	98.6	93.1	90.8	-	-	-
Jaderberg <i>et al.</i> [25]	VGG, 90k	95.5	89.6	-	93.2	71.7	97.8	97.0	89.6	81.8	-	-	-
Shi <i>et al.</i> [54]	VGG, 90k	97.8	95.0	81.2	97.5	82.7	98.7	98.0	91.9	89.6	-	-	-
*Shi <i>et al.</i> [55]	VGG, 90k	96.2	93.8	81.9	95.5	81.9	98.3	96.2	90.1	88.6	-	71.8	59.2
Lee <i>et al.</i> [36]	VGG, 90k	96.8	94.4	78.4	96.3	80.7	97.9	97.0	88.7	90.0	-	-	-
Yang <i>et al.</i> [64]	VGG, Private	97.8	96.1	-	95.2	-	97.7	-	-	-	-	75.8	69.3
Cheng <i>et al.</i> [11]	ResNet, 90k+ST ⁺	99.3	97.5	87.4	97.1	85.9	99.2	97.3	94.2	93.3	70.6	-	-
ASTER-A	VGG, 90k	98.1	95.7	81.7	97.6	85.5	98.7	97.3	92.2	88.6	67.6	73.2	63.9
ASTER-B	ResNet, 90k	98.7	96.3	83.2	99.2	87.6	99.1	97.6	92.4	89.7	68.9	75.4	67.4
ASTER	ResNet, 90k+ST	99.6	98.8	93.4	99.2	93.6	98.8	98.0	94.5	91.8	76.1	78.5	79.5