Adaptive Text Recognition through Visual Matching
Supplementary Material

Chuhan Zhang[1], Ankush Gupta[2], and Andrew Zisserman[1]

[1] Visual Geometry Group,Department of Engineering Science
University of Oxford
{czhang,az}@robots.ox.ac.uk
[2] DeepMind, London
ankushgupta@google.com

Contents

Code, data, and model checkpoints are available at: http://www.robots.ox.ac.uk/~vgg/research/FontAdaptor20/.

1 Evaluation Metrics

We measure the character (CER) and word error rates (WER):

$$\text{CER} = \frac{1}{N}\sum_{i=1}^{N}\frac{\text{EditDist}\left(\boldsymbol{y}_{\text{gt}}^{(i)}, \boldsymbol{y}_{\text{pred}}^{(i)}\right)}{\text{Length}\left(\boldsymbol{y}_{\text{gt}}^{(i)}\right)}$$

where, $\boldsymbol{y}_{\text{gt}}^{(i)}$ and $\boldsymbol{y}_{\text{pred}}^{(i)}$ are the i^{th} ground-truth and predicted strings respectively in a dataset containing N strings; EditDist is the *Levenshtein distance* [10]; $\text{Length}\left(\boldsymbol{y}_{\text{gt}}\right)$ is the number of characters in $\boldsymbol{y}_{\text{gt}}$. WER is computed as CER above with words (i.e. contiguous characters separated by whitespace) in place of characters as tokens.

2 Ablation study

2.1 Ablation on modules at a larger scale

In section 5.4 of the main paper, we ablated each major component of the proposed model architecture to evaluate its relative contribution to the model's performance. However, there the training set was limited to only one FontSynth attribute (regular fonts). Here in table 1, we ablate the same model components, but couple it with increasing number of training fonts from the FontSynth dataset (section 5.2 of the main paper), still evaluating on the FontSynth test set.

Predictions straight from the output of the visual encoder (not using the decoder) are quite noisy (row 6). Using a class aggregator ('agg. embed') lends robustness to noise in the similarity maps which leads to consistent improvements across all training data settings (compare rows 3 and 5). Using position encoding ('pos. enc.') which encodes glyph extents also leads to consistent improvements (compare rows 1 and 2). Self-attention in the similarity disambiguator is a critical component, without which error increases at least twofold (compare rows 2 and 3). With increasing training data, the performance generally improves for all the ablated versions of the model explored here. However, it is evident that all the model components retain their functional significance even with large amounts of training data.

	sim. enc. $\mathcal{S}$	sim. disamb.		agg. embed.	training data									
		pos. enc.	self-attn		**R**		**R+B**		**R+B+L**		**R+B+L+I**		**R+B+L+I+OS**	
					CER	**WER**	**CER**	**WER**	**CER**	**WER**	**CER**	**WER**	**CER**	**WER**
1.	✓	✓	✓	✓	**9.4**	**30.2**	**8.3**	**28.8**	**8.1**	**27.3**	**5.6**	**22.3**	**3.5**	**12.8**
2.	✓	✗	✓	✓	11.8	37.9	12.4	41.1	9.4	27.6	7.9	22.9	4.2	16.8
3.	✓	✗	✗	✓	23.9	68.8	15.4	39.4	13.2	43.5	13.0	52.0	11.4	49.6
4.	✓	✓	✓	✗	22.9	65.8	8.6	34.3	8.3	28.5	8.5	26.4	4.5	20.2
5.	✓	✗	✗	✗	25.8	63.1	19.3	54.9	18.5	48.7	18.4	45.0	20.1	62.1
6.	✓	–	–	–	49.0	96.2	33.7	67.7	31.8	72.0	38.3	78.9	41.8	98.0

Table 1. Model component analysis. We ablate various components of the model and report performance on the FontSynth test set when trained on an increasing number of FontSynth attributes. The first row corresponds to the full model; the last row (#6) corresponds to reading out characters using the CTC decoder from the output of the visual encoder. 'R', 'B', 'L' and 'I' correspond to the FontSynth training splits: 'Regular', 'Bold', 'Light' and 'Italic'; while 'OS' stands for the Omniglot-Sequence dataset.

2.2 Ablation on balance of loses

Figure 1 shows the impact of the weights on two losses when training with one FontSynth attribute. The use of L_{sim} is essential as the model does not converge

when its ratio is too small. The CER is aslo increased by about 1% when its weight is five times larger than that on L_{pred}. Therefore, ratio close to 1 gives a good balance of the two losses.

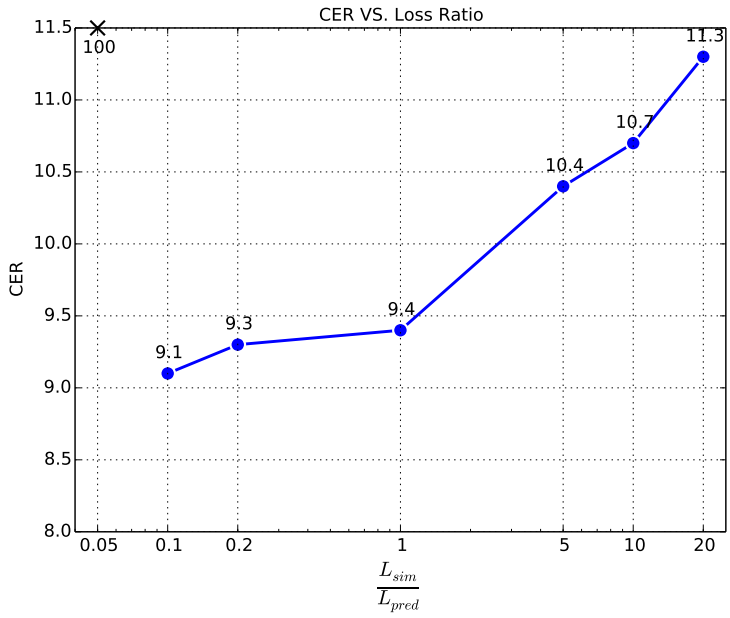


Fig. 1. Impact of balance of weights on CER when training with one FontSynth attribute.

3 Examples and Visualizations

In figs. 3 to 7 we show the similarity maps at various stages in the model for all the main experiments presented in the paper **VS1-3, A1-2** (section 5.3 of the main paper). The three similarity maps correspond to (from left to right): (1) $\mathcal{S}$ similarity map from cosine similarity of features from the visual encoder (Φ), (2) similarity map after local position encoding, and (3) $\mathcal{S}^*$ similarity map after the self-attention module. All maps are in the range [-1,1].

By comparing maps at different stages, it is clear that the *position encoding* removes some ambiguities in shape matching, e.g. 'w' and 'v', 'm' and 'n', by using the positional and width information of exemplars. Further, *self-attention* is crucial as it compares the confidence across all glyphs and suppresses the lower confidence glyphs, while boosting the higher confidence glyphs, as is evident by increased contrast of the maps.

Figure 2 shows a zoomed in version of $\mathcal{S}^*$. The self-attention module (coupled with the class aggregator; see fig. 2 in the paper) introduces the boundary token for the CTC loss [5] to separate characters, especially repeated characters. Training with CTC is known to result in peaky distribution along the sequence – it predicts character classes when the confidence is very high, usually at the center of the character, while predicting all the other positions between characters as boundary token. This effect can be seen from the top rows of the map, where white pixels correspond to boundary token and the gaps are where the model looks at the central column in each 'glyph-square' with high confidence.

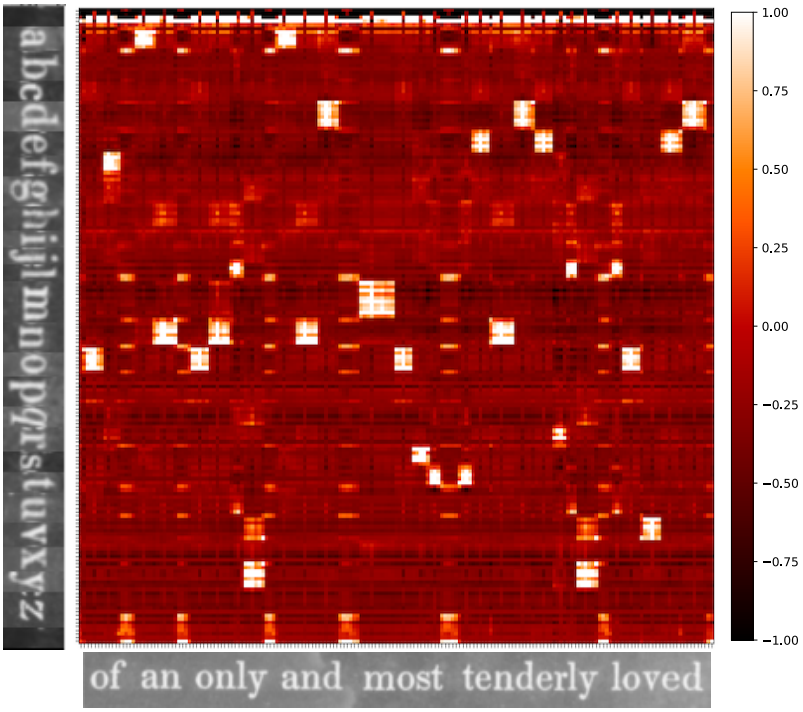


Fig. 2. An example of a similarity map $\mathcal{S}^*$ after the self-attention module.

Fig. 3. Experiment VS-1: Generalization to novel fonts with known glyph exemplars. We use FontSynth to study generalization to novel fonts when the exemplars from test fonts are provided in matching. Challenging samples from the FontSynth test set (novel fonts) are visualized below. Note the marked improvement in the confidence and coherence of the similarity maps through the through various processing stages.

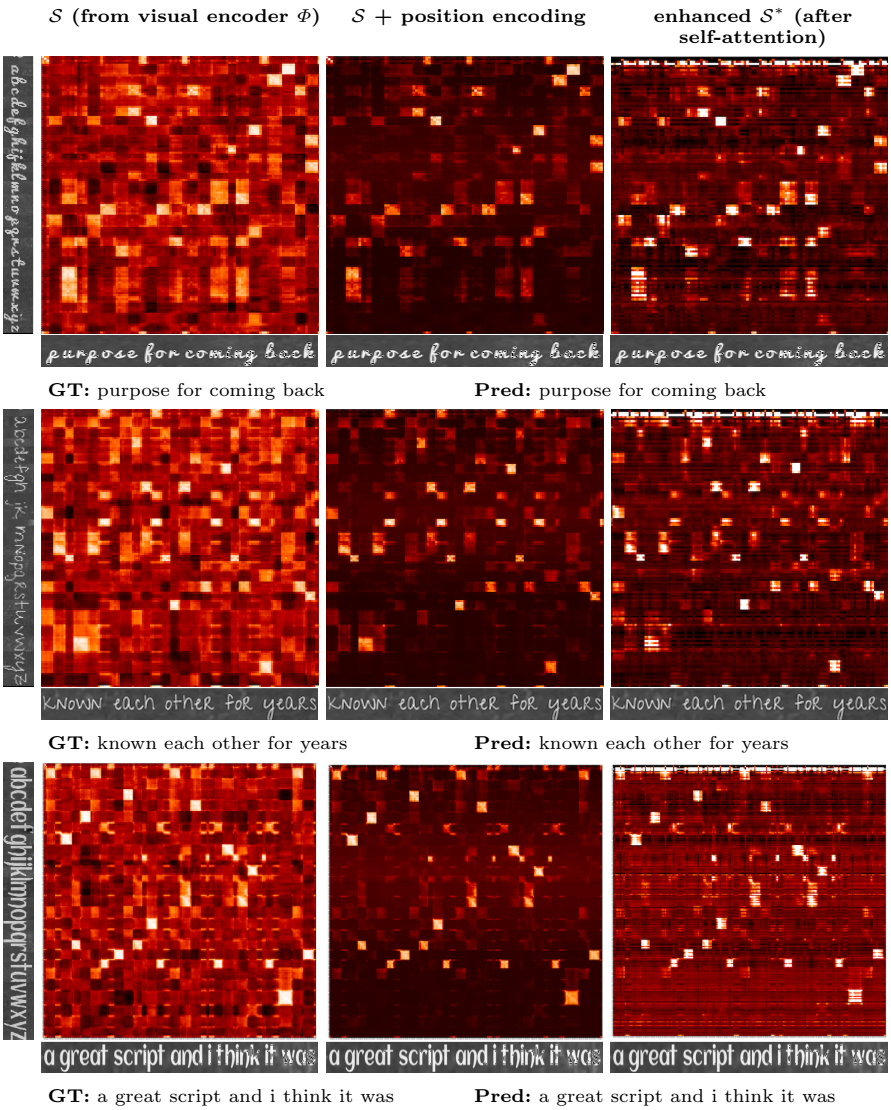


Fig. 4. Experiment VS-2: Cross glyph matching. Text recognition on the FontSynth test set with *cross-font matching* – using exemplars from the training set when we do not have access to the test font. The model succeeds at cross-font matching even when the exemplars and line images are in very different visual styles. A difficulty occurs when the same glyph in the two fonts are not quite visually similar, e.g. due to differences in upper vs. lower case. One example is shown in the last row where the capital 'Q' is matched to lower case 'o' instead of 'q'. However, using consensus from multiple training exemplars can help overcome this limitation.

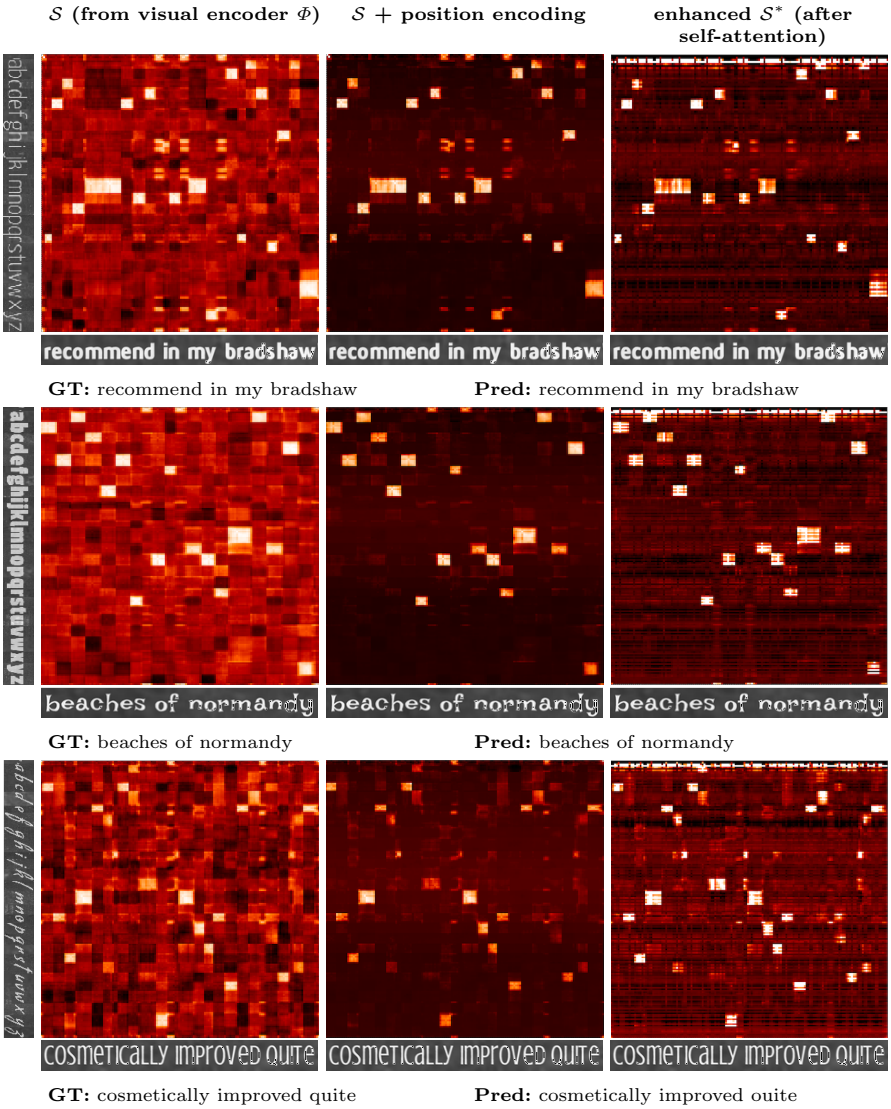


Fig. 5. Experiment VS-3: Transfer from synthetic to real data. We test our model trained purely on *synthetic data* for generalization to *real world* Google1000 English books. The model is robust to nuisance factors prevalent in scans of real historical books, e.g. over-exposure and low contrast, as well as degradation and show-through.

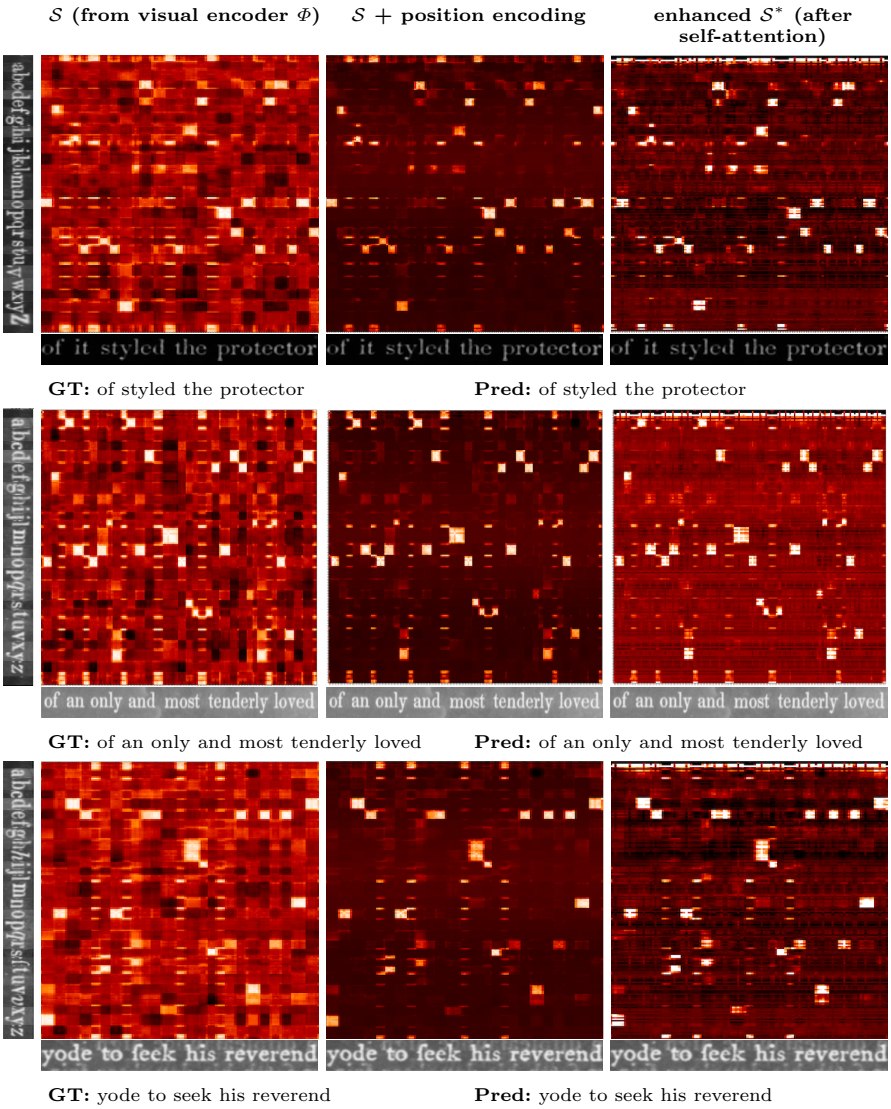


Fig. 6. Experiment A-1: Transfer to non-Latin glyphs. We evaluate our model on the novel glyph styles in the Omniglot-Sequence test set. The Omniglot-Sequence dataset is constructed by mapping Omniglot glyphs randomly to the English alphabet and then rendering text-line images using those as characters (refer to section 5.2 in the paper). The ground-truth English sentence used for assembling the text-line image is shown underneath the similarity maps. This demonstrates strong generalization to novel visual styles. The position encoder and self-attention module resolve many ambiguities and predict accurate matches.

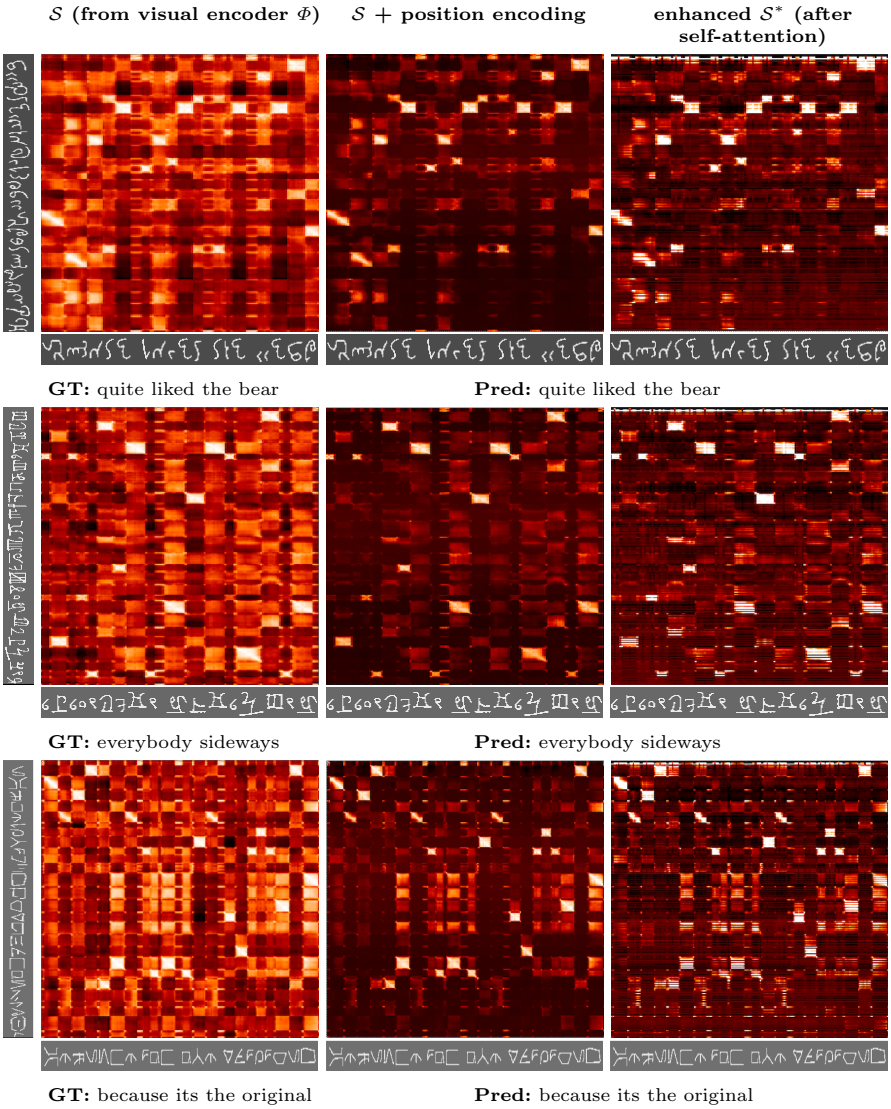
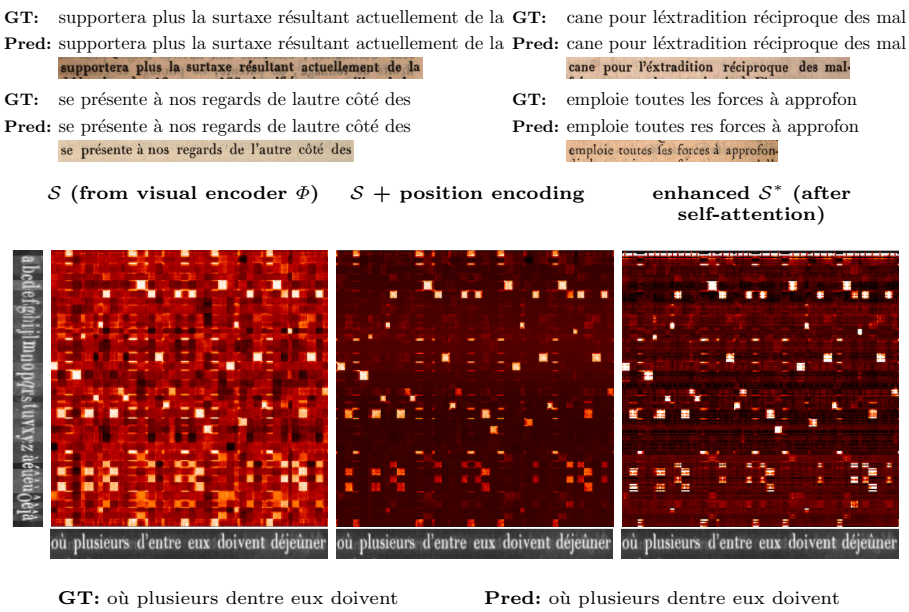


Fig. 7. Experiment A-2: Transfer to novel alphabets. To test generalization to unseen alphabets/novel languages, we test how well our model trained on English transfers to all the other languages in Google1000, namely, French, Spanish and Italian. Below, we show four text-line samples from each language and also similarity map progression as above for one sample. These new languages have an expanded alphabet due to accented letters, e.g. é, ê, á, û, ù, etc. These are especially challenging as the visual differences between these letters is quite subtle. Our model is able to successfully recognize such fine-grained differences and decode these completely unseen languages. For example, in the French similarity map example below, the model successfully distinguishes between 'u', 'ù' and 'û' when their exemplars are provided.

French



GT: où plusieurs dentre eux doivent dêjeûner

Pred: où plusieurs dentre eux doivent dêjeûner

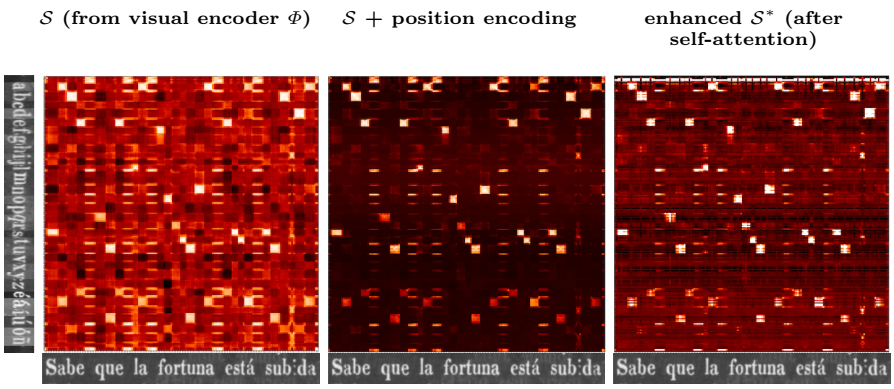
Spanish

GT: declaró haber recibido en dote de su segunda mujer
Pred: declaró haber recibido en dote de su segunda mujer

GT: nifestaré con documentos auténticos que tengo en
Pred: nifestaré con documentos auténticos que tengo en

GT: rente á la suerte que le preparaba esa faccion
Pred: rente á la suerte que le preparaba esa faccion

GT: regresar todos los ausentes y debiendo ser puestos en li
Pred: regresar todos los ausentes y debicndo ber puestos en li



GT: sabe que la fortuna está subda **Pred:** sabe que la fortuna está subda

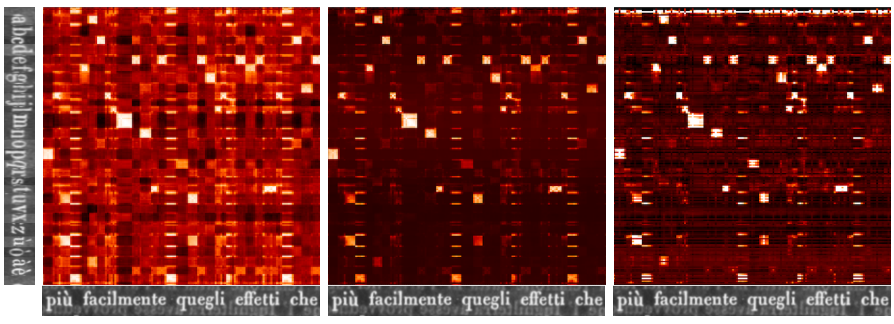
Italian

GT: un tal prodigio per indizio di una qualche
Pred: un tal prodigio per indizio di una qualche

GT: era già intanto miglior lo stato degli asse
Pred: era già intanto miglior lo stato degli asse

GT: di cosa che lasciò di essere buona e
Pred: di cosa che lasciò di essere buona e

GT: pio ciò non arrecò meraviglia che a pastori
Pred: piociò nop arrecò meraviglia che a pastori



GT: più facilmente quegli effetti che **Pred:** più facilmente quegli effetti che

4 Implementation of SOTA models

In the experiments in the main paper, we compare our model with four state-of-the-art models from three different domains: scene text, handwritten text and document text recognition. In the following subsections, we describe the details of how we implement and use them.

4.1 Attn. and CTC model – Baek et al. [3]

We follow the recent work of Baek et al. [3] to identify the two of the strongest state-of-the-art text recognition models, named *Attn.* and *CTC* model in the main paper, to benchmark against our method on generalization to novel fonts and alphabets. They unify modern text recognition methods in a four-stage framework, visualized below – (1) input transformation, (2) feature extraction, (3) sequence modeling and (4) string prediction.

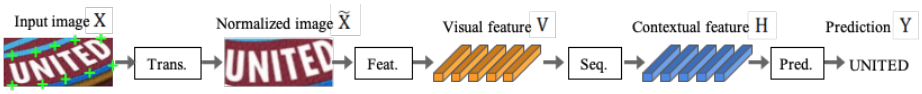


Fig. 8. Four stages of modern text-recognition methods [3]. State-of-the-art text recognition models constitute of four stages – transformation, feature extraction, sequence modelling and prediction. We compare our proposed method against two of the strongest text-recognition models.

We do not use the first transformation stage as document line images are mostly rectilinear and are not severely curved/distorted. For the remaining three stages, we use: (2) ResNet visual encoder ('Feat.'), (3) 2-layer bidirectional-LSTM (256 state size) ('Seq.'), and for the last 'Pred.' stage (4), we consider both: (1) CTC, and (2) attention based (with a 128-state LSTM) sequence prediction. The ResNet visual encoder in the baseline models is designed to have the same (or more) number of parameters as our visual encoder (please refer to table 1 in the paper): baseline ResNet: 6.8M; our encoder: 4.7M parameters. Detailed architecture of the visual encoder used in the baseline models is given in the table below.

4.2 Industry Standard Model – Tesseract [13]

Tesseract is a widely-used engine for text recognition in documents like pdf files. They provide free user interface where people can render texts to train the engine on a wide range fonts and languages. However, 1) the online engine does not support parallel training in gpu, this restricts the batch size to be 1 and training usually runs into divergence , 2) it does not use much data augmentation during training, which leads to serve overfitting problem, 3) checkpoints are saved by selecting the ones with low training error rates, not from validation results on other fonts. It results in large error rate in generalization to unseen fonts. When

layer	**kernel**	**channels in / out**	**pool**	**# conv layers**	**output size** $H \times W$
conv1	3×3	1 / 64	max = (2, 2)	4	$16 \times W/2$
resBlock1	3×3	64 / 64	max = (2, 1) x 2	6	$4 \times W/2$
resBlock2	3×3	64 / 128	max = (2, 1) x 2	6	$2 \times W/4$
resBlock3	3×3	128 / 256	max = (1, 2)	4	$2 \times W/8$
resBlock4	3×3	256 / 512	max = (1, 1)	2	$2 \times W/8$
AvgPool	-	-	avg = (2, 1)	-	$1 \times W/8$

Table 2. Architecture details of the visual encoder used for feature extraction.

the model is trained on our four training attributess in Experiment VS-1, the CER on unseen fonts is 33.5%. Therefore, we implement Tesseract in Pytorch and train it in the same way as we do for other models. We follow the standard network architecture they use for English, details are given on Tesseract's github page [3], details are shown in table 3.

layers	kernel	channels in / out	hidden size
conv	3x3	1 / 16	–
tanh			
maxpool	3x3	16 / 16	–
LSTM	–	16 / 48	48
Bi-LSTM	–	48 / 192	96
LSTM	–	192 / 256	256
linear	–	256 / #classes	–

Table 3. Architecture details of Tesseract.

4.3 Handwritten Text Model – Chowdhury et al. [4]

The archiecture of the handwritten text recognition model from Chowdhury et al. [4] is similar to the *Attn.* model in Baek et al. [3] – it uses CNNs as the feature extractor, Bi-LSTM as the sequence modelling module and attention-based LSTM as the predictor. However, it adds residual connection [9] and Layer Normalization [2] in their LSTMs. Codes of this model are not available online. Therefore, we verify our implementation by training and testing on handwriting benchmarks IAM [11] and RIMES [1]. We compare the numbers they report

[3] Tesseract's specification of architecure details.
https://github.com/tensorflow/models/blob/master/research/street/g3doc/vgslspecs.md

in their paper to ours. Please note that their best performance is achieved with a beam search algorithm in the final LSTM, we do not implement this stage because in our paper we use beam search with an external n-gram language model on every model for a fair comparison.

Our implementation is able to achieve similar or lower CER, but has a consistenly higher WER, especially in the IAM dataset. It might be caused by different methods of measuring WER, e.g. the inclusion of punctuations, numbers and abbreviations. Since no details are given in their paper, we treat everything between two white spaces as words when computing the WER, which takes punctuations, numbers as well as words into account.

		IAM		RIMES	
		CER	WER	CER	WER
Baseline	reported	17.4	25.5	12.0	19.1
	implemented	13.5	30.9	6.2	16.0
+ LN	reported	13.1	22.9	9.7	15.8
	implemented	11.4	26.6	5.1	12.6
+ LN + Focal Loss	reported	11.4	21.1	7.3	13.5
	implemented	10.9	25.4	4.6	12.8

Table 4. Verification of our implementation of the SOTA model [4] in handwritten text recognition. We compare the error rates they report in their paper with those from our implementation on IAM and RIMES.

5 Performance on Scene Text

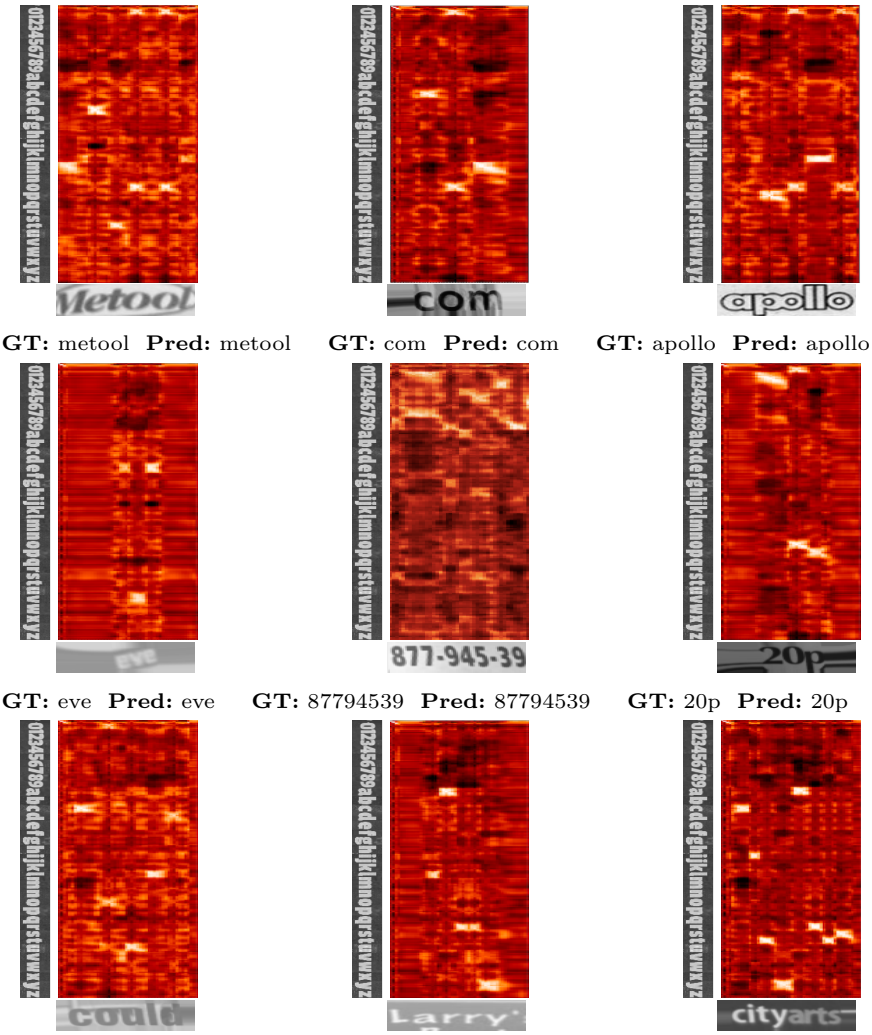
Although our model are trained to recognize rectilinear text images, it can be applied to scene text with a transformation module. We use the pretrained transformation module from Baek et al. [3] to transform the images and test our model on commonly used scene text dataset: SVT [15], ICDAR [7, 8] and IIIT5k [12]. Given that the visual style in scene texts images is inconsistent and we don't have access to the fonts, we cross-match the images with exemplars from a randomly selected font. We show the similarity maps and predictions from our model below, with examples of both arabic numbers and words in various visual styles.

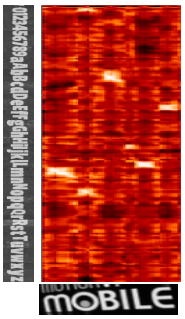
We compare our results with Yao et al. [16] which also learn to recognizes texts by using visual primitives, and show the results in table 5.

Model	**IIIT5K (small)**	**IIIT5K (medium)**	**IC03 (full)**	**IC03 (50)**	**SVT**
Yao et al. [16]	80.2	69.3	80.3	88.5	75.9
Ours	**96.2**	**92.8**	**93.3**	**98.2**	**92.4**

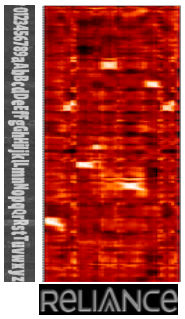
Table 5. Results on scene text dataset IIIT5K, IC03, SVT with lexicon.

Fig. 9. experiment VS-1: Visualization of our model's performance on scene text datasets: SVT [15], ICDAR [7, 8] and IIIT5k [12]. We show the alphabet image, scene text image and the similarity map from the feature encoder for each example.

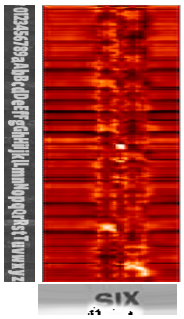




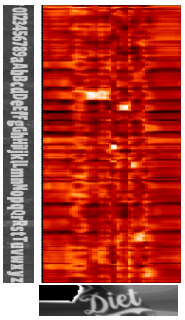
GT: mobile **Pred:** mobile



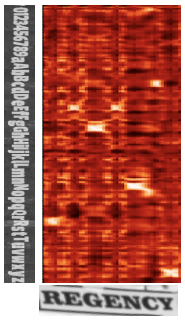
GT: reliance **Pred:** reliance



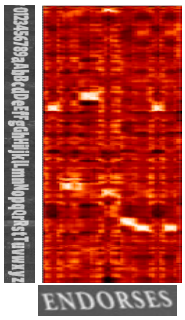
GT: six **Pred:** six



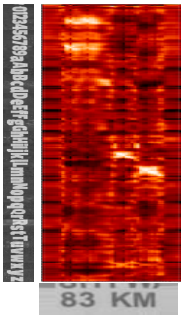
GT: diet **Pred:** diet



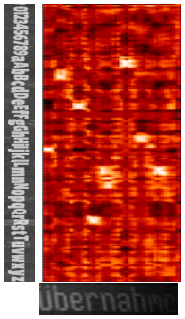
GT: regency **Pred:** regency



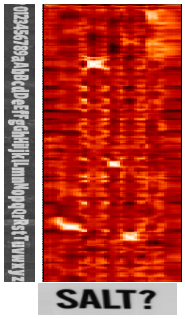
GT: endorses **Pred:** endorses



GT: 83km **Pred:** 83km



GT: ubernahme **Pred:** jbernahme



GT: salt **Pred:** salt2

6 Dataset Details

6.1 Training and Testing Datasets

We summarize the distribution and size of all our training and testing datasets in table 6.

datasets	distribution	number of images	language
training			
FontSynth	200 fonts	200000	English
Omniglot-Seq	30 alphabets × 20 writers	60000	Various
testing			
FontSynth	251 fonts	12550	English
Google1000_EN	40 books	2000	English
Google1000_FR	40 books	2000	French
Google1000_ES	40 books	2000	Spanish
Google1000_IT	40 books	2000	Italian
Omniglot-Seq	20 alphabets × 20 writers	40000	Various

Table 6. Details of training and testing datasets used in expeiments VS1-3 and A1-2.

6.2 Google1000 Dataset Details

We benchmark our method on a large-scale real world dataset of historical books – Google1000 [14] (section 5.2). Specifically, we employ Google1000 for two experiments: (1) **experiment VS-3:** to evaluate our models trained on synthetic data for generalization to nuisance factors encountered in the real data like degradation, blur, show-through (from behind), inking, fading, oblique text-lines etc and (2) **experiment A-1:** to evaluate generalization from English (training language) to novel alphabets/new languages in Google1000, namely French, Spanish and Italian.

For this, we randomly select 40 volumes for each language; Below, we list the volume-ids for completeness.

English

0194	0701	0343	0025	0006	0100	0663	0255	0147	0054
0612	0315	0034	0084	0128	0513	0441	0709	0224	0587
0050	0047	0093	0676	0448	0347	0511	0335	0218	0438
0287	0131	0174	0380	0059	0151	0340	0068	0176	0569

French

0930	0951	0931	0927	0954	0937	0946	0935	0952	0949
0957	0940	0925	0942	0932	0922	0950	0944	0936	0933
0947	0921	0926	0959	0953	0943	0934	0924	0938	0941
0923	0948	0929	0928	0956	0958	0945	0955	0920	0939

Spanish

0843	0795	0820	0827	0797	0810	0864	0872	0898	0857
0836	0878	0818	0875	0875	0916	0845	0888	0818	0808
0865	0860	0793	0788	0826	0897	0871	0824	0845	0883
0918	0864	0846	0899	0874	0850	0910	0870	0790	0901

Italian

0967	0983	0986	0978	0972	0964	0962	0970	0971	0995
0968	0969	0997	0991	0981	0979	0975	0993	0990	0977
0963	0961	0976	0988	0965	0998	0960	0999	0982	0973
0987	0994	0996	0984	0985	0980	0992	0989	0966	0974

6.3 FontSynth Dataset Details

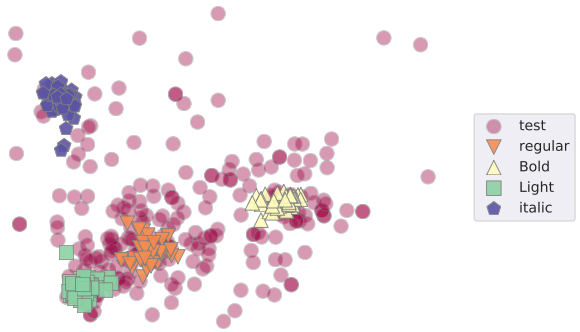


Fig. 10. FontSynth font embeddings. The training font splits (**regular**, **bold**, **light**, **italic**) are distinct from the **test** fonts.

The FontSynth dataset (section 5.2) used for evaluating the effect of increasing diversity of training fonts on generalization to novel fonts, is created by taking 1400 fonts used in MJSynth [6] and splitting them per *attributes* as determined from their font names, namely – *regular*, *bold*, *light*, *italic*, and *other* (i.e. rest of the fonts with none of the four attributes in their name). We select 50 fonts at random from each of the four attributes to create the training set, while all of the *other* 251 fonts constitute the test set. Data are available to download at: http://www.robots.ox.ac.uk/~vgg/research/FontAdaptor20/.

Figure 10 visualizes font-embeddings from a font classifier (1400-way categorization from font glyph images) trained on the MJSynth fonts. The training font embeddings form tight clusters, while the test embeddings are spread out, indicating that they are visually distinct from the training set. Hence, FontSynth forms a good benchmark for evaluating generalization to novel fonts.

Below, we list the names of the fonts in each split for completeness.
Please refer to fig. 3 in the main paper for a visualization of the fonts in these splits.

Regular (R)

Arimo-Regular.ttf
Calligraffitti-Regular.ttf
Cousine-Regular.ttf
MountainsofChristmas-Regular.ttf
NotoSans-Regular.ttf
RobotoCondensed-Regular.ttf
RobotoSlab-Regular.ttf
Rochester-Regular.ttf
Smokum-Regular.ttf
AllertaStencil-Regular.ttf
AveriaSansLibre-Regular.ttf
Balthazar-Regular.ttf
Basic-Regular.ttf
Baumans-Regular.ttf
Belgrano-Regular.ttf
BenchNine-Regular.ttf
BrunoAceSC-Regular.ttf
BubblegumSans-Regular.ttf
ButterflyKids-Regular.ttf
CabinSketch-Regular.ttf
Cambo-Regular.ttf
CantataOne-Regular.ttf
Clara-Regular.ttf
ClickerScript-Regular.ttf
Codystar-Regular.ttf
Comfortaa-Regular.ttf
ConcertOne-Regular.ttf
Condiment-Regular.ttf
Convergence-Regular.ttf
Cookie-Regular.ttf
Copse-Regular.ttf
DuruSans-Regular.ttf
Dynalight-Regular.ttf
Eater-Regular.ttf
EaterCaps-Regular.ttf
EBGaramond-Regular.ttf
Economica-Regular.ttf
EmblemaOne-Regular.ttf
Englebert-Regular.ttf
Jura-Regular.ttf
Kameron-Regular.ttf

KellySlab-Regular.ttf
KottaOne-Regular.ttf
KronaOne-Regular.ttf
Lancelot-Regular.ttf
Lato-Regular.ttf
Lekton-Regular.ttf
Lemon-Regular.ttf
UbuntuMono-Regular.ttf

Bold (B)

Alef-Bold.ttf
Allan-Bold.ttf
AmaticSC-Bold.ttf
Amiri-Bold.ttf
Andada-Bold.ttf
Antonio-Bold.ttf
Comfortaa-Bold.ttf
Corben-Bold.ttf
DancingScript-Bold.ttf
Economica-Bold.ttf
FiraMono-Bold.ttf
Gorditas-Bold.ttf
Gudea-Bold.ttf
Inika-Bold.ttf
IstokWeb-Bold.ttf
JosefinSans-Bold.ttf
Judson-Bold.ttf
Kameron-Bold.ttf
Karla-Bold.ttf
Lekton-Bold.ttf
LibreCaslonText-Bold.ttf
Lora-Bold.ttf
Magra-Bold.ttf
Merienda-Bold.ttf
MerriweatherSans-Bold.ttf
Monda-Bold.ttf
MontserratSubrayada-Bold.ttf
NanumGothicCoding-Bold.ttf
NanumMyeongjo-Bold.ttf
PassionOne-Bold.ttf
Podkova-Bold.ttf
Puritan-Bold.ttf
Quattrocento-Bold.ttf
Raleway-Bold.ttf
Rokkitt-Bold.ttf
Rosario-Bold.ttf
Rufina-Bold.ttf
Scada-Bold.ttf
SeoulHangang-Bold.ttf
SeoulHangangCondensed-Bold.ttf
Share-Bold.ttf
SignikaNegative-Bold.ttf
Sintony-Bold.ttf
Skranji-Bold.ttf
SourceCodePro-Bold.ttf
Thabit-Bold.ttf
UnifrakturCook-Bold.ttf
YanoneKaffeesatz-Bold.ttf
Ubuntu-Bold.ttf
UbuntuMono-Bold.ttf

Light (L)

OpenSans-Light.ttf
Roboto-Light.ttf
RobotoCondensed-Light.ttf
RobotoSlab-Light.ttf
AdventPro-Light.ttf
AlegreyaSans-Light.ttf
AlegreyaSansSC-Light.ttf
Antonio-Light.ttf
AveriaLibre-Light.ttf

AveriaSerifLibre-Light.ttf
BenchNine-Light.ttf
Buda-Light.ttf
Codystar-Light.ttf
Comfortaa-Light.ttf
Dosis-Light.ttf
Exo-Light.ttf
Exo2-Light.ttf
FiraSans-Light.ttf
Flamenco-Light.ttf
Jura-Light.ttf
Kreon-Light.ttf
Lato-Light.ttf
Merriweather-Light.ttf
MerriweatherSans-Light.ttf
Muli-Light.ttf
Neuton-Light.ttf
Oswald-Light.ttf
Oxygen-Light.ttf
Raleway-Light.ttf
Rokkitt-Light.ttf
Sansation-Light.ttf
SeoulHangang-Light.ttf
SeoulHangangCondensed-Light.ttf
SeoulNamsan-Light.ttf
SeoulNamsanCondensed-Light.ttf
Signika-Light.ttf
SignikaNegative-Light.ttf
SourceCodePro-Light.ttf
SourceSansPro-Light.ttf
Stoke-Light.ttf
TitilliumWeb-Light.ttf
Tulpen-Light.ttf
YanoneKaffeesatz-Light.ttf
Ubuntu-Light.ttf
YanoneKaffeesatz-ExtraLight.ttf
Exo2-ExtraLight.ttf
Neuton-ExtraLight.ttf
Raleway-ExtraLight.ttf
TitilliumWeb-ExtraLight.ttf
YanoneKaffeesatz-ExtraLight.ttf

Italic (I)

NotoSans-Italic.ttf
NotoSansUI-Italic.ttf
OpenSans-Italic.ttf
OpenSansHebrew-Italic.ttf
Amaranth-Italic.ttf
Andada-Italic.ttf
AnonymousPro-Italic.ttf
Caudex-Italic.ttf
ChangaOne-Italic.ttf
ChauPhilomeneOne-Italic.ttf
Exo-Italic.ttf
FiraSans-Italic.ttf
Fondamento-Italic.ttf
IstokWeb-Italic.ttf
JosefinSans-Italic.ttf
JosefinSlab-Italic.ttf
Karla-Italic.ttf
Lekton-Italic.ttf
Lora-Italic.ttf
Muli-Italic.ttf
Neuton-Italic.ttf
NoticiaText-Italic.ttf
Oregano-Italic.ttf
Overlock-Italic.ttf
PlayfairDisplay-Italic.ttf
QuattrocentoSans-Italic.ttf
Quicksand-Italic.ttf
Radley-Italic.ttf
Sanchez-Italic.ttf
Sarabun-Italic.ttf
Sedan-Italic.ttf
SourceSansPro-Italic.ttf

Trochut-Italic.ttf
Vollkorn-Italic.ttf
Chivo-BlackItalic.ttf
Exo-BlackItalic.ttf
Exo2-BlackItalic.ttf
AlegreyaSans-MediumItalic.ttf
AlegreyaSansSC-MediumItalic.ttf
Share-BoldItalic.ttf
TitilliumWeb-SemiBoldItalic.ttf
Volkhov-BoldItalic.ttf
AlegreyaSansSC-LightItalic.ttf
AveriaLibre-LightItalic.ttf
AveriaSansLibre-LightItalic.ttf
AveriaSerifLibre-LightItalic.ttf
CrimsonText-BoldItalic.ttf
CrimsonText-SemiboldItalic.ttf
Cuprum-BoldItalic.ttf
Exo-ExtraBoldItalic.ttf

Test fonts

Aclonica.ttf
CherryCreamSoda.ttf
CherryCreamSoda-VTT.ttf
Chewy.ttf
Chewy-VTT.ttf
ComingSoon.ttf
ComingSoon-TTX.ttf
CraftyGirls.ttf
CraftyGirls-TTX.ttf
Crushed.ttf
DroidSans.ttf
DroidSansMono.ttf
DroidSerif.ttf
FontdinerSwanky.ttf
HomemadeApple.ttf
IrishGrover.ttf
IrishGrowler.ttf
JustAnotherHand.ttf
Kranky.ttf
LuckiestGuy.ttf
MaidenOrange.ttf
PermanentMarker.ttf
Redressed.ttf
Roboto-Black.ttf
Roboto-Medium.ttf
Roboto-Thin.ttf
RobotoSlab-Thin.ttf
RockSalt.ttf
Schoolbell.ttf
Slackey.ttf
SpecialElite.ttf
Sunshiney.ttf
Ultra.ttf
WalterTurncoat-VTT.ttf
WalterTurncoat.ttf
AdventPro-Medium.ttf
AdventPro-Thin.ttf
Alegreya-Black.ttf
AlegreyaSans-Black.ttf
AlegreyaSans-Medium.ttf
AlegreyaSans-Thin.ttf
AlegreyaSansSC-Black.ttf
AlegreyaSansSC-Medium.ttf
AlegreyaSansSC-Thin.ttf
AlegreyaSC-Black.ttf
Amiri-Slanted.ttf
Andika-R.ttf
AnnieUseYourTelescope.ttf
Anton.ttf
Anton-VTT.ttf
ArchitectsDaughter.ttf
Asset.ttf
Bangers.ttf
Bevan.ttf
Bevan.ttf

Bevan_ik.ttf
Bevan_VTT - Copy.ttf
Bevan_VTT.ttf
Bevan_VTT2.ttf
BevanFLAB.ttf
BigshotOne.ttf
Cabin-Medium.ttf
CabinCondensed-Medium.ttf
Candal.ttf
Cantarell-Oblique.ttf
CarterOne.ttf
Cedarville-Cursive.ttf
Chivo-Black.ttf
Chivo-Black-VTT.ttf
Cinzel-Black.ttf
CinzelDecorative-Black.ttf
Coda-Heavy.ttf
CodaCaption-Heavy.ttf
Coustard-Black.ttf
CoveredByYourGrace.ttf
CrimsonText-Roman.ttf
cwTeXFangSong.ttf
cwTeXHei-ofl.ttf
cwTeXHei.ttf
cwTeXKai-ofl.ttf
cwTeXKai.ttf
cwTeXMing-ofl.ttf
cwTeXMing.ttf
cwTeXYen-ofl.ttf
cwTeXYen.ttf
DawningofaNewDay.ttf
DidactGothic.ttf
Dosis-Medium.ttf
Elsie-Black.ttf
ElsieSwashCaps-Black.ttf
Exo-Black.ttf
Exo-Medium.ttf
Exo-Thin.ttf
Exo2-Black.ttf
Exo2-Medium.ttf
Exo2-Thin.ttf
ExpletusSans-Medium.ttf
FiraSans-Medium.ttf
FrancoisOne.ttf
GenBasB.ttf
GenBasBI.ttf
GenBasI.ttf
GenBasR.ttf
GenBkBasB.ttf
GenBkBasBI.ttf
GenBkBasI.ttf
GenBkBasR.ttf
Geo-Oblique.ttf
GFSDidot.ttf
GFSNeohellenic.ttf
GFSNeohellenic.ttf
GiveYouGlory.ttf
GloriaHallelujah.ttf
GoblinOne.ttf
GoudyBookletter1911.ttf
GravitasOne.ttf
BM-HANNA.ttf
Hanuman.ttf
Hanumanb.ttf
HoltwoodOneSC.ttf
IMFeDPit28P.ttf
IMFeDPrm28P.ttf
IMFeDPsc28P.ttf
IMFePIit28P.ttf
IMFePIrm28P.ttf
IMFePIsc28P.ttf
IMFeENit28P.ttf
IMFeENrm28P.ttf
IMFeENsc28P.ttf
IMFeFCit28P.ttf
IMFeFCrm28P.ttf
IMFeFCsc28P.ttf
IMFeGPit28P.ttf
IMFeGPrm28P.ttf
IMFeGPsc28P.ttf
IndieFlower.ttf
Jomolhari-alpha3c-0605331.ttf
JosefinSans-Thin.ttf
JosefinSlab-Thin.ttf
Jura-Medium.ttf
JustMeAgainDownHere.ttf
Kristi.ttf

LaBelleAurore.ttf
LateefRegOT.ttf
Lato-Black.ttf
Lato-Hairline.ttf
LeagueScript.ttf
Lobster.ttf
Lobster-Cyrillic-TTF.ttf
LovedbytheKing.ttf
LoveYaLikeASister.ttf
MavenPro-Black.ttf
MavenPro-Medium.ttf
MavenPro-Black-VTT.ttf
MavenPro-Medium-VTT.ttf
Meddon.ttf
MedievalSharp.ttf
Megrim.ttf
Merriweather-Black.ttf
Metrophobic.ttf
Miama.ttf
Michroma.ttf
Monofett.ttf
Neucha.ttf
Neucha-hints.ttf
Nobile-Medium.ttf
NothingYouCouldDo.ttf
NovaCut.ttf
NovaFlat.ttf
NovaMono.ttf
NovaOval.ttf
NovaRound.ttf
NovaScript.ttf
NovaSlim.ttf
NovaSquare.ttf
OFLGoudyStMTT.ttf
Orbitron-Black.ttf
Orbitron-Medium.ttf
Overlock-Black.ttf
OvertheRainbow.ttf
Pacifico.ttf
padauk_src.ttf
padaukbook_src.ttf
PassionOne-Black.ttf
PaytoneOne.ttf
Pecita.ttf
PlayfairDisplay-Black.ttf
PlayfairDisplaySC-Black.ttf
PollerOne.ttf
PTM55FT.ttf
PTF55F.ttf
PTF56F.ttf
PTF75F.ttf
PTF76F.ttf
PTZ55F.ttf
PTZ56F.ttf
Raleway-Heavy.ttf
Raleway-Medium.ttf
Raleway-Thin.ttf
ReenieBeanie.ttf
ReenieBeanie-VTT.ttf
Ruda-Black.ttf
RuslanDisplay.ttf
SansitaOne.ttf
ScheherazadeRegOT.ttf
SeoulHangang-Medium.ttf
SeoulHangang.ttf
SeoulHangangB.ttf
SeoulHangangCondensed-Medium.ttf
SeoulHangang.ttf
SeoulHangangB.ttf
SeoulNamsan-Medium.ttf
SeoulNamsan.ttf
SeoulNamsanB.ttf
SeoulNamsanCondensed-Black.ttf
SeoulNamsanCondensed-Medium.ttf
SeoulNamsan.ttf
SeoulNamsanB.ttf
SigmarOne.ttf
Simonetta-Black.ttf
SixCaps.ttf
Snippet.ttf
SourceCodePro-Black.ttf
SourceCodePro-Medium.ttf

SourceSansPro-Black.ttf
SueEllenFrancisco.ttf
SwankyandMooMoo.ttf
TerminalDosis-Medium.ttf
Thabit-Oblique.ttf
Thabit.ttf
TheGirlNextDoor.ttf
Tienne-Heavy.ttf
TitilliumWeb-Black.ttf
UnifrakturMaguntia-Book.ttf
WaitingfortheSunrise.ttf
WireOne.ttf
Zeyada.ttf
Ubuntu-B-hinting.ttf
Ubuntu-BI-hinting.ttf
Ubuntu-C-hinting.ttf
Ubuntu-L-hinting.ttf
Ubuntu-LI-hinting.ttf
Ubuntu-M-hinting.ttf
Ubuntu-MI-hinting.ttf
Ubuntu-R-hinting.ttf
Ubuntu-RI-hinting.ttf
Ubuntu-Medium.ttf
UbuntuMono-B-hinting.ttf
UbuntuMono-BI-hinting.ttf
UbuntuMono-R-hinting.ttf
UbuntuMono-RI-hinting.ttf

References

1. Augustin, E., Carré, M., Grosicki, E., Brodin, J.M., Geoffrois, E., Prêteux, F.: Rimes evaluation campaign for handwritten mail processing (2006) 13
2. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016) 13
3. Baek, J., Kim, G., Lee, J., Park, S., Han, D., Yun, S., Oh, S.J., Lee, H.: What is wrong with scene text recognition model comparisons? dataset and model analysis. In: Proc. ICCV (2019) 12, 13, 15
4. Chowdhury, A., Vig, L.: An efficient end-to-end neural model for handwritten text recognition. Proc. BMVC (2018) 13, 14
5. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proc. ICML (2006) 4
6. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Synthetic data and artificial neural networks for natural scene text recognition. In: Workshop on Deep Learning, NIPS (2014) 20
7. Karatzas, D., Gomez-Bigorda, L., Nicolaou, A., Ghosh, S., Bagdanov, A., Iwamura, M., Matas, J., Neumann, L., Chandrasekhar, V.R., Lu, S., Shafait, F., Uchida, S., Valveny, E.: ICDAR 2015 robust reading competition. In: Proc. ICDAR (2015) 15, 16
8. Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., Bigorda, L.G., Mestre, S.R., Mas, J., Mota, D.F., Almazan, J.A., de las Heras, L.P.: ICDAR 2013 robust reading competition (2013) 15, 16
9. Kim, J., El-Khamy, M., Lee, J.: Residual lstm: Design of a deep recurrent architecture for distant speech recognition. arXiv preprint arXiv:1701.03360 (2017) 13
10. Levenshtein, V.: Binary codes capable of correcting deletions, insertions and reversals. In: Soviet Physics Doklady 1
11. Marti, U.V., Bunke, H.: The iam-database: an english sentence database for offline handwriting recognition. International Journal on Document Analysis and Recognition **5**(1), 39–46 (2002) 13
12. Mishra, A., Alahari, K., Jawahar, C.V.: Scene text recognition using higher order language priors. In: BMVC (2012) 15, 16
13. Smith, R.: An overview of the tesseract ocr engine. In: Ninth international conference on document analysis and recognition (ICDAR 2007). vol. 2, pp. 629–633. IEEE (2007) 12
14. Vincent, L.: Google book search: Document understanding on a massive scale. In: PROC. ninth International Conference on Document Analysis and Recognition (ICDAR). pp. 819–823. Washington, DC (2007) 18
15. Wang, K., Belongie, S.: Word spotting in the wild. In: Proc. ECCV (2010) 15, 16
16. Yao, C., Bai, X., Shi, B., Liu, W.: Strokelets: A learned multi-scale representation for scene text recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4042–4049 (2014) 15