Adaptive Text Recognition through Visual Matching

Chuhan Zhang
University of Oxford
czhang@robots.ox.ac.uk

Ankush Gupta
DeepMind, London
ankushgupta@google.com

Andrew Zisserman
University of Oxford
az@robots.ox.ac.uk

Abstract

This work addresses the problems of generalization and flexibility for text recognition in documents. We introduce a new model that exploits the repetitive nature of characters in languages, and decouples the visual decoding and linguistic modelling stages through intermediate representations in the form of similarity maps. By doing this, we turn text recognition into a visual matching problem, thereby achieving generalization in appearance and flexibility in classes.

We evaluate the model on both synthetic and real datasets across different languages and alphabets, and show that it can handle challenges that traditional architectures are unable to solve without expensive re-training, including: (i) it can change the number of classes simply by changing the exemplars; and (ii) it can generalize to novel languages and characters (not in the training data) simply by providing a new glyph exemplar set. In essence, it is able to carry out one-shot sequence recognition. We also demonstrate that the model can generalize to unseen fonts without requiring new exemplars from them.

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

1. Introduction

Our objective in this work is *generalization* and *flexibility* in text recognition. Modern text recognition methods [4, 9, 27, 40] achieve excellent performance in many cases, but generalization to unseen data, *i.e.*, novel fonts and new languages, either requires large amounts of data for primary training or expensive fine-tuning for each new case.

The text recognition problem is to map an image of a line of text $\boldsymbol{x}$ into the corresponding sequence of characters $\boldsymbol{y} = (y_1, y_2, \ldots, y_k)$, where k is the length of the string and $y_i \in \mathcal{A}$ are characters in alphabet $\mathcal{A}$ (*e.g.*, {a,b,...,z,<space>}). Current deep learning based methods [9, 27, 40] cast this in the encoder-decoder framework [10, 45], where first the text-line image is encoded through a visual ConvNet [26], followed by a recurrent neural network decoder, with alignment between the visual features and text achieved either through attention [5] or Connectionist Temporal Classification (CTC) [16].

English: alphabet-size: 26 characters

Greek: novel glyph shapes and alphabet-size: 24 characters

Figure 1: **Visual matching for text recognition.** Current text recognition models learn discriminative features specific to character shapes (*glyphs*) from a pre-defined (fixed) alphabet. We train our model instead to establish *visual similarity* between given character glyphs **(top)** and the text-line image to be recognized **(left)**. This makes the model highly adaptable to unseen glyphs, new alphabets (different languages) and extensible to novel character classes, *e.g.*, English → Greek, *without* further training. Brighter colors correspond to higher visual similarity.

Impediments to generalization. The conventional methods for text recognition train the visual encoder and the sequence decoder modules in an end-to-end manner. While this is desirable for optimal co-adaptation, it induces monolithic representations which confound visual and linguistic functions. Consequently, these methods suffer from the following limitations: (1) Discriminative recognition models specialize to fonts and textures in the training set,

hence generalize poorly to novel visual styles. (2) The decoder discriminates over a fixed alphabet/number of characters. (3) The encoder and decoder are tied to each other, hence are not inter-operable across encoders for new visual styles or decoders for new languages. Therefore, current text recognition methods generalize poorly and require re-initialization or fine-tuning for new alphabets and languages. Further, fine-tuning typically requires new training data for the target domain and does not overcome these inherent limitations.

Recognition by matching. Our method is based on a key insight: text is a sequence of repetitions of a finite number of discrete entities. The repeated entities are *characters* in a text string, and *glyphs*, *i.e*., visual representations of characters/symbols, in a text-line image. We re-formulate the text recognition problem as one of *visual matching*. We assume access to *glyph exemplars* (*i.e*., cropped images of characters), and task the visual encoder to localize these repeated glyphs in the given text-line image. The output of the visual encoder is a *similarity map* which encodes the visual similarity of each spatial location in the text-line to each glyph in the alphabet as shown in Figure 1. The decoder ingests this similarity map to infer the most probable string. Figure 2 summarizes the proposed method.

Overcoming limitations. The proposed model overcomes the above mentioned limitations as follows: (1) Training the encoder for *visual matching* relieves it from learning specific visual styles (fonts, colors *etc*.) from the training data, improving generalization over novel visual styles. (2) The similarity map is agnostic to the number of different glyphs, hence the model generalizes to novel alphabets (different number of characters). (3) The similarity map is also agnostic to visual styles, and acts as an interpretable interface between the visual encoder and the decoder, thereby disentangling the two.

Contributions. Our main contributions are threefold. First, we propose a novel network design for text recognition aimed at generalization. We exploit the repetition of glyphs in language, and build this similarity between units into our architecture. The model is described in Sections 3 and 4. Second, we show that the model outperforms state-of-the-art methods in recognition of novel fonts unseen during training (Section 5). Third, the model can be applied to novel languages without expensive fine-tuning at test time; it is only necessary to supply glyph exemplars for the new font set. These include languages/alphabets with different number of characters, and novel styles *e.g*., characters with accents or historical characters *'ſ'* (also in Section 5).

Although we demonstrate our model for *document OCR* where a consistent visual style of glyphs spans the entire document, the method is applicable to scene-text/text-in-the-wild (*e.g*., SVT [49], ICDAR [21, 22] datasets) where each instance has a unique visual style (results in Appendix E).

2. Related Work

Few-shot recognition. Adapting model behavior based on class exemplars has been explored for few-shot object recognition. Current popular few-shot classification methods, *e.g*., Prototypical Nets [42], Matching Nets [48], Relation Nets [44], and MAML [14], have been applied only to recognition of *single instances*. Our work addresses the unique challenges associated with one-shot classification of *multiple instances in sequences*. To the best of our knowledge this is the first work to address one-shot *sequence recognition*. We discuss these challenges and the proposed architectural innovations in Section 3.4. A relevant work is from Cao *et al*. [7] which tackles few-shot video classification, but similar to few-shot object recognition methods, they classify the whole video as a *single* instance.

Text recognition. Recognizing text in images is a classic problem in pattern recognition. Early successful applications were in reading handwritten documents [6, 26], and document optical character recognition (OCR) [41]. The OCR industry standard—*Tesseract* [41]—employs specialized training data for each supported language/alphabet.[1] Our model enables rapid adaptation to novel visual styles and alphabets and does not require such expensive fine-tuning/specialization. More recently, interest has been focussed towards text in natural images. Current methods either directly classify word-level images [19], or take an encoder-decoder approach [10, 45]. The text-image is encoded through a ConvNet, followed by bidirectional-LSTMs for context aggregation. The image features are then aligned with string labels either using Connectionist Temporal Classification (CTC) [16, 18, 38, 43] or through attention [5, 8, 9, 27, 39]. Recognizing irregularly shaped text has garnered recent interest which has seen a resurgence of dense character-based segmentation and classification methods [13, 31]. Irregular text is rectified before feature extraction either using geometric transformations [29, 39, 40, 51] or by re-generating the text image in canonical fonts and colors [30]. Recently, Baek *et al*. [4] present a thorough evaluation of text recognition methods, unifying them in a four-stage framework—input transformation, feature extraction, sequence modeling, and string prediction.

[1] Tesseract's specialized training data for 103 languages: https://github.com/tesseract-ocr/tesseract/wiki/Data-Files

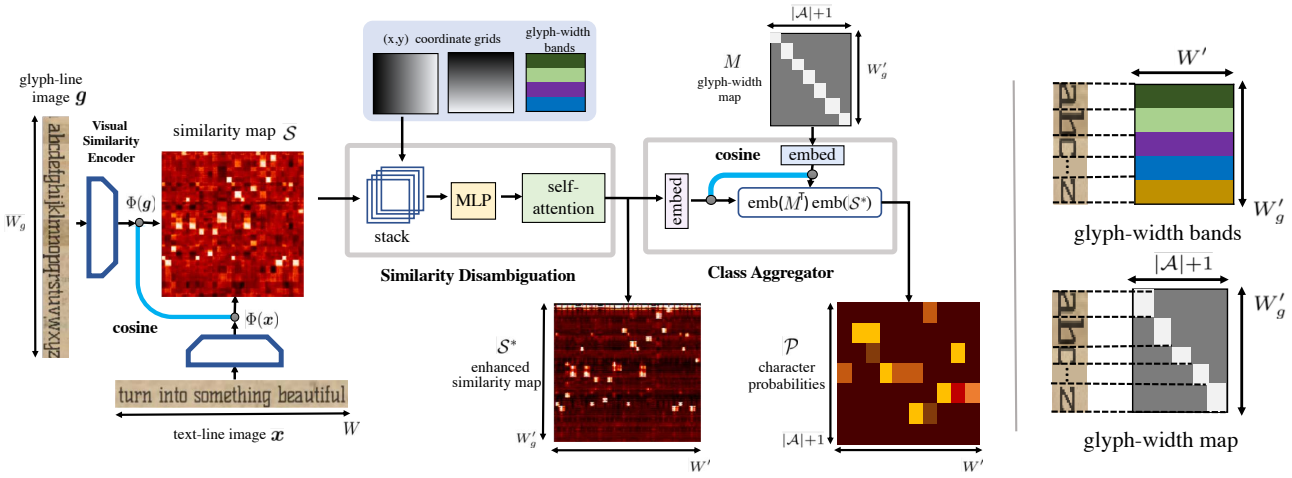


Figure 2: **Architecture for adaptive visual matching.** We cast the problem of text recognition as one of visual matching of glyph exemplars in the given text-line image. **Left**: Overview of the architecture. The visual encoder Φ embeds the glyph-line $\boldsymbol{g}$ and text-line $\boldsymbol{x}$ images and produces a similarity map $\mathcal{S}$, which scores the similarity of each glyph against each position along the text-line. Then, ambiguities in (potentially) imperfect visual matching are resolved to produce the enhanced similarity map $\mathcal{S}^*$. Finally, similarity scores are aggregated to output class probabilities $\mathcal{P}$ using the ground-truth glyph width contained in $\mathcal{M}$. **Right**: Illustration of how glyph-widths are encoded into the model. The glyph-width bands *(top)* have the same height as the width of their corresponding glyph exemplars, and their scalar values are the glyph widths in pixels. The glyph-width map *(bottom)* is a binary matrix with a column for each character in the alphabet $\mathcal{A}$; the columns indicate the extent of glyphs in the glyph-line image by setting the corresponding rows to a non-zero value (=1).

3. Model Architecture

Our model recognizes a given text-line image by localizing glyph exemplars in it through visual matching. It takes both the text-line image and an alphabet image containing a set of exemplars as input, and predicts a sequence of probabilities over N classes as output, where N is equal to the number of exemplars given in the alphabet image. For inference, a glyph-line image is assembled from the individual character glyphs of a reference font simply by concatenating them side-by-side, and text-lines in that font can then be read.

The model has two main components: (1) a visual similarity encoder (Section 3.1) which outputs a similarity map encoding the similarity of each glyph in the text-line image, and (2) an alphabet agnostic decoder (Section 3.2) which ingests this similarity map to infer the most probable string. In Section 3.3 we give details for the training objective. Figure 2 gives a concise schematic of the model.

3.1. Visual Similarity Encoder

The visual similarity encoder is provided with a set of glyphs for the target alphabet, and tasked to localize these glyphs in the input text-line image to be recognized. It first embeds the text-line and glyphs using a shared visual encoder Φ and outputs a *similarity map* $\mathcal{S}$ which computes the visual similarity between all locations in the text-line against all locations in every glyph in the alphabet.

Mathematically, let $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$ be the text-line image, with height H, width W and C channels. Let the glyphs be $\{g_i\}_{i=1}^{i=|\mathcal{A}|}$, $g_i \in \mathbb{R}^{H \times W_i \times C}$, where $\mathcal{A}$ is the exemplar set – the alphabet plus the blank space and image padding, and W_i is the width of the i^{th} glyph. The glyphs are stacked along the width to form a *glyph-line image* $\boldsymbol{g} \in \mathbb{R}^{H \times W_g \times C}$. Embeddings are obtained using the visual encoder Φ for both the text-line $\Phi(\boldsymbol{x}) \in \mathbb{R}^{1 \times W' \times D}$ and the glyph-line $\Phi(\boldsymbol{g}) \in \mathbb{R}^{1 \times W'_g \times D}$, where D is the embedding dimensionality. The output widths are downsampled by the network stride s (*i.e.*, $W' = \frac{W}{s}$). Finally, each spatial location along the width in the glyph-line image is scored against the every location in the text-line image to obtain the similarity map $\mathcal{S} \in [-1, 1]^{W'_g \times W'}$:

$$S_{ij} = \langle \Phi(\boldsymbol{g})_i, \Phi(\boldsymbol{x})_j \rangle = \frac{\Phi(\boldsymbol{g})_i^T \Phi(\boldsymbol{x})_j}{||\Phi(\boldsymbol{g})_i|| \cdot ||\Phi(\boldsymbol{x})_j||} \quad (1)$$

where score is the cosine similarity, and $i \in \{1, \ldots, W'_g\}$, $j \in \{1, \ldots, W'\}$.

3.2. Alphabet Agnostic Decoder

The alphabet agnostic decoder discretizes the similarity maps into probabilities for each glyph in the exemplars for

all spatial locations along the width of the text-line image. Concretely, given the visual similarity map $\mathcal{S} \in \mathbb{R}^{W'_g \times W'}$ it outputs logits over the glyph exemplars for each location in the text-line: $\mathcal{P} \in \mathbb{R}^{|A| \times W'}$, $\mathcal{P}_{ij} = \log p(y_i | \boldsymbol{x}_j)$, where $\boldsymbol{x}_j$ is the j^{th} column in text-line image (modulo encoder stride) and y_i is the i^{th} exemplar in $\mathcal{A}$.

A simple implementation would predict the argmax or sum of the similarity scores aggregated over the extent of each glyph in the similarity map. However, this naive strategy does not overcome ambiguities in similarities or produce smooth/consistent character predictions. Hence, we proceed in two steps: first, **similarity disambiguation** resolves ambiguities over the glyphs in the alphabet producing an enhanced similarity map ($\mathcal{S}^*$) by taking into account the glyph widths and position in the line image, and second, **class aggregator** computes glyph class probabilities by aggregating the scores inside the spatial extent of each glyph in $\mathcal{S}^*$. We detail the two steps next; the significance of each component is established empirically in Section 5.4.

Similarity disambiguation. An ideal similarity map would have square regions of high-similarity. This is because the width of a character in the glyph and text-line images will be the same. Hence, we encode glyph widths along with local x, y coordinates using a small MLP into the similarity map. The input to the MLP at each location is the similarity map value $\mathcal{S}$ stacked with: (1) two channels of x, y coordinates (normalized to $[0, 1]$), and (2) *glyph-width bands* $\mathcal{G}$: $\mathcal{G} = \boldsymbol{w}_g \mathbb{1}^T$, where $\boldsymbol{w}_g \in \mathbb{R}^{W'_g}$ is a vector of glyph widths in pixels; see Figure 2 for an illustration. For disambiguation, we use a self-attention module [46] which attends over columns of $\mathcal{S}$ and outputs the final enhanced similarity map $\mathcal{S}^*$ of the same size as $\mathcal{S}$.

Class aggregator. The class aggregator Λ maps the similarity map to logits over the alphabet along the horizontal dimension in the text-line image: $\Lambda : \mathbb{R}^{W'_g \times W'} \mapsto \mathbb{R}^{|A| \times W'}$, $\mathcal{S}^* \mapsto \mathcal{P}$. This mapping can be achieved by multiplication through a matrix $M \in \mathbb{R}^{|\mathcal{A}| \times W'_g}$ which aggregates (sums) the scores in the span of each glyph: $\mathcal{P} = M\mathcal{S}^*$, such that $M = [m_1, m_2, \ldots, m_{|\mathcal{A}|}]^T$ and $m_i \in \{0, 1\}^{W'_g} = [0, \ldots, 0, 1, \ldots, 1, 0, \ldots, 0]$ where the non-zero values correspond to the span of the i^{th} glyph in the glyph-line image.

In practice, we first embed columns of $\mathcal{S}^*$ and M^T independently using learnt linear embeddings. The embeddings are ℓ_2-normalized before the matrix product (equivalent to cosine similarity). We also expand the classes to add an additional "boundary" class (for CTC) using a learnt $m_{|\mathcal{A}|+1}$. Since, the decoder is agnostic to the number of characters in the alphabet, it generalizes to novel alphabets.

3.3. Training Loss

The dense per-pixel decoder logits over the glyph exemplars $\mathcal{P}$ are supervised using the CTC loss [15] ($\mathcal{L}_{CTC}$) to align the predictions with the output label. We also supervise the similarity map output of the visual encoder $\mathcal{S}$ using an auxiliary cross-entropy loss ($\mathcal{L}_{sim}$) at each location. We use ground-truth character bounding-boxes for determining the spatial span of each character. The overall training objective is the following two-part loss,

$$\mathcal{L}_{pred} = \mathcal{L}_{CTC}\left(\mathrm{SoftMax}(\mathcal{P}), \boldsymbol{y}_{gt}\right) \tag{2}$$

$$\mathcal{L}_{sim} = -\sum_{ij} \log(\mathrm{SoftMax}(S_{y_i j})) \tag{3}$$

$$\mathcal{L}_{total} = \mathcal{L}_{pred} + \lambda \mathcal{L}_{sim} \tag{4}$$

where, $\mathrm{SoftMax}(\cdot)$ normalization is over the glyph exemplars (rows), $\boldsymbol{y}_{gt}$ is the string label, and y_i is the ground-truth character associated with the i^{th} position in the glyph-line image. The model is insensitive to the value of λ within a reasonable range (see Appendix B.2), and we use $\lambda = 1$ for a good balance of losses.

3.4. Discussion: One-shot Sequence Recognition

Our approach can be summarized as a method for one-shot sequence recognition. Note, existing few-shot methods [20,42,44,48] are not directly applicable to this problem of one-shot sequence recognition, as they focus on classification of the whole of the input (*e.g.* an image) as a single instance. Hence, these cannot address the following unique challenges associated with (text) sequences: (1) segmentation of the imaged text sequence into characters of different widths; (2) respecting language-model/sequence-regularity in the output. We develop a novel neural architectural solutions for the above, namely: (1) A neural architecture with *explicit reasoning over similarity maps* for decoding sequences. The similarity maps are key for *generalization* at both ends—novel fonts/visual styles and new alphabets/languages respectively. (2) Glyph width aware *similarity disambiguation*, which identifies contiguous square blocks in noisy similarity maps from novel data. This is critical for robustness against imprecise visual matching. (3) *Class aggregator*, aggregates similarity scores over the reference width-spans of the glyphs to produce character logit scores over the glyph exemplars. It operates over a variable number of characters/classes and glyph-widths. The importance of each of these components is established in the ablation experiments in Section 5.4.

layer	kernel	channels in / out	pooling	output size H×W
conv1	3×3	1 / 64	max = (2, 2)	16 × W/2
resBlock1	3×3	64 / 64	max = (1, 2)	8 × W/2
resBlock2	3×3	64 / 128	max = (2, 2)	4 × W/4
upsample	–	–	(2, 2)	8 × W/2
skip	3×3	128+64 / 128	–	8 × W/2
pool	–	–	avg = (2, 1)	4 × W/2
conv2	1×1	128 / 64	–	4 × W/2
reshape	–	64 / 256	–	1 × W/2

Table 1: **Visual encoder architecture** (Sections 3.1 and 4.1). The input is an image of size $32 \times W \times 1$ (height×width×channels).

4. Implementation details

The architectures of the visual similarity encoder and the alphabet agnostic decoder are described in Section 4.1 and Section 4.2 respectively, followed by training set up in Section 4.3.

4.1. Visual Similarity Encoder

The visual similarity encoder (Φ) encodes both the text-line ($\boldsymbol{x}$) and glyph-line ($\boldsymbol{g}$) images into feature maps. The inputs of height 32 pixels, width W and 1 channel (grayscale images) are encoded into a tensor of size $1 \times \frac{W}{2} \times 256$. The glyph-line image's width is held fixed to a constant $W_g = 720$ px: if $\sum_{i=1}^{i=|\mathcal{A}|} W_i < W_g$ the image is padded at the end using the <space> glyph, otherwise the image is downsampled bilinearly to a width of $W_g = 720$ px. The text-line image's input width is free (after resizing to a height of 32 proportionally). The encoder is implemented as a U-Net [37] with two residual blocks [17]; detailed architecture in Table 1. The visual similarity map ($\mathcal{S}$) is obtained by taking the cosine distance between all locations along the width of the encoded features from text-line $\Phi(\boldsymbol{x})$ and glyph-line $\Phi(\boldsymbol{g})$ images.

4.2. Alphabet Agnostic Decoder

Similarity disambiguation. We use the self-attention based *Transformer* model [46] with three layers with four attention heads of vector dimension 360 each. The input to this module is the similarity map $\mathcal{S}$ stacked with with local positions (x, y) and glyph widths, which are then encoded through a three-layer (4×16, 16×32, 32×1) MLP with ReLU non-linearity [35].

Class aggregator. The columns of $\mathcal{S}^*$ and glyph width templates (refer to Section 3.2) are embedded independently using linear embeddings of size $W'_g \times W'_g$, where $W'_g = \frac{W_g}{s} = \frac{720}{2} = 360$ (s = encoder stride).

Figure 3: **Left: FontSynth splits.** Randomly selected fonts from each of the five font categories – (1) *regular* (R), (2) *bold* (B), (3) *italic* (I), (4) *light* (L) – used for generating the synthetic training set, and (5) *other* (*i.e.* none of the first four) – used for the test set. **Right: Synthetic data.** Samples from *FontSynth* **(top)** generated using fonts from MJSynth [19], and *Omniglot-Seq* **(bottom)** generated using glyphs from Omniglot [25] as fonts (Section 5.2).

Inference. We decode greedily at inference, as is common after training with CTC loss. No additional language model (LM) is used, except in Experiment VS-3 (Section 5.5.3), where a 6-gram LM learnt from over 10M sentences from the WMT News Crawl (2015) English corpus [1] is combined with the model output with beam-search using the algorithm in [32] (parameters: α=1.0, β=2.0, beam-width=15).

4.3. Training and Optimization

The entire model is trained end-to-end by minimizing the training objective Equation (4). We use online data augmentation on both the text-line and glyph images, specifically random translation, crops, contrast, and blur. All parameters, for both ours and SotA models, are initialized with random weights. We use the Adam optimizer [24] with a constant learning rate of 0.001, a batch size of 12 and train until validation accuracy saturates (typically 100k iterations) on a single Nvidia Tesla P40 GPU. The models are implemented in PyTorch [36].

5. Experiments

We compare against state-of-the-art text-recognition models for generalization to novel fonts and languages. We first describe the models used for comparisons (Section 5.1), then datasets and evaluation metrics (Section 5.2), followed by an overview of the experiments (Section 5.3), and a thorough component analysis of the model architecture (Section 5.4). Finally, we present the results (Section 5.5) of all the experiments.

5.1. State-of-the-art Models in Text Recognition

For comparison to state-of-the-art methods, we use three models: (i) Baek *et al.* [4] for scene-text recognition; (ii) Tesseract [41], the industry standard for document OCR;

language →	EN	FR	IT	ES
# books	780	40	40	140
alphabet size	26	35	29	32
% accented letters	0	2.6	0.7	1.5

Table 2: **Google1000 dataset summary.** Total number of books, alphabet size and percentage of letters with accent (counting accented characters a new) for various languages in the Google1000.

and (iii) Chowdhury *et al*. [11] for handwritten text recognition.

For (i), we use the open-source models provided, but without the transformation module (since documents do not have the scene-text problem of non-rectilinear characters). Note, our visual encoder has similar number of parameters as in the encoder ResNet of [4] (theirs: 6.8M, ours: 4.7M parameters). For (ii) and (iii) we implement the models using the published architecture details. Further details of these networks, and the verification of our implementations is provided in the Appendix D.

5.2. Datasets and Metrics

FontSynth. We take fonts from the MJSynth dataset [19] and split them into five categories by their appearance attributes as determined from their names: (1) regular, (2) bold, (3) italic, (4) light, and (5) others (*i.e*., all fonts with none of the first four attributes in their name); visualized in Figure 3 (left). We use the first four splits to create a training set, and (5) for the test set. For training, we select 50 fonts at random from each split and generate 1000 text-line and glyph images for each font. For testing, we use all the 251 fonts in category (5). LRS2 dataset [12] is used as the text source. We call this dataset *FontSynth*; visualization in Figure 3 (right) and further details in the Appendix D.

Omniglot-Seq. Omniglot [25] consists of 50 alphabets with a total of 1623 characters, each drawn by 20 different writers. The original one-shot learning task is defined for *single* characters. To evaluate our sequence prediction network we generate a new *Omniglot-Seq* dataset with *sentence* images as following. We randomly map alphabets in Omniglot to English, and use them as 'fonts' to render text-line images as in FontSynth above. We use the original alphabet splits (30 training, 20 test) and generate data online for training, and 500 lines per alphabet for testing. Figure 3 (right) visualizes a few samples.

Google1000. Google1000 [47] is a standard benchmark for document OCR released in ICDAR 2007. It constitutes scans of 1000 public domain historical books in English (EN), French (FR), Italian (IT) and Spanish (ES) languages; Table 2 provides a summary. Figure 4 visualizes a few samples from this dataset. This dataset poses significant challenges due to severe degradation, blur, show-through (from behind), inking, fading, oblique text-lines etc. Type-faces from 18^{th} century are significantly different from modern fonts, containing old ligatures like *"ſt,ct,Qi"*. We use this dataset only for evaluation; further details in Appendix D.

Evaluation metrics. We measure the character (CER) and word error rates (WER); definitions in Appendix A.

5.3. Overview of Experiments

The goal of our experiments is to evaluate the proposed model against state-of-the-art models for text recognition on their generalization ability to (1) novel visual styles **(VS)** (*e.g*., novel fonts, background, noise *etc*.), and (2) novel alphabets/languages **(A)**. Specifically, we conduct the following experiments:

1. **VS-1: Impact of number of training fonts.** We use FontSynth to study the impact of the number of different training fonts on generalization to novel fonts when the exemplars from the testing fonts are provided.
2. **VS-2: Cross font matching.** In this experiment, we do *not* assume access to the testing font. Instead of using exemplars from the test font, the most similar font from the training set is selected automatically.
3. **VS-3: Transfer from synthetic to real data.** This evaluates transfer of models trained on synthetic data to real data with historical typeface and degradation.
4. **A-1: Transfer to novel Latin alphabets.** This evaluates transfer of models trained on English to new Latin languages in Google1000 with additional characters in the alphabet (*e.g*., French with accented characters).
5. **A-2: Transfer to non-Latin glyphs.** The above experiments both train and test on Latin alphabets. Here we evaluate the generalization of the models trained on English fonts to non-Latin scripts in Omniglot-Seq (*e.g*., from English to Greek).

5.4. Ablation Study

We ablate each major component of the proposed model on the VS-1 experiment to evaluate its significance. Table 4 reports the recognition accuracy on the FontSynth test set when trained on one (R) and all four (R+B+L+I) font attributes. Without the decoder (last row), simply reporting the argmax from the visual similarity map reduces to nearest-neighbors or one-shot Prototypical Nets [42] method. This is ineffective for unsegmented text recognition (49% CER vs. 9.4% CER for the full model). Excluding the position encoding in the similarity disambiguation

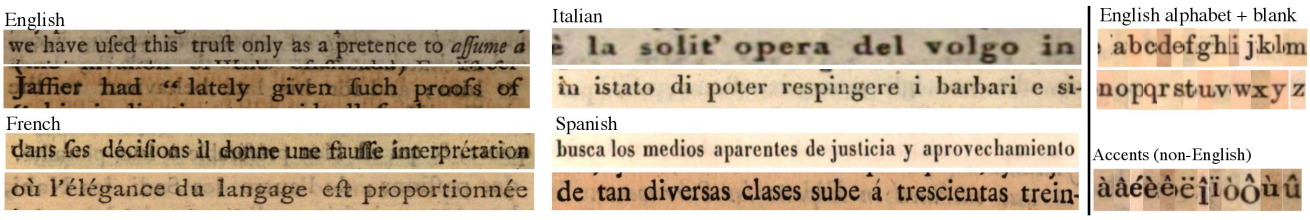


Figure 4: **Google1000 printed books dataset. (left):** Text-line image samples from the Google1000 [47] evaluation set for all the languages, namely, English, French, Italian and Spanish. **(right):** *Common* set of glyph exemplars used in our method for *all* books in the evaluation set for English and accents for the other languages.

training set →			R		R+B		R+B+L		R+B+L+I		R+B+L+I+OS	
model		test glyphs known	CER	WER	CER	WER	CER	WER	CER	WER	CER	WER
CTC Baek *et al.* [4]		✗	17.5	46.1	11.5	30.3	10.4	28.2	10.4	27.7	—	—
Attn. Baek *et al.* [4]		✗	16.5	41.0	12.7	34.5	11.1	27.4	10.3	23.6	—	—
Tesseract [41]		✗	19.2	48.6	12.3	37.0	10.8	31.7	9.1	27.8	—	—
Chowdhury *et al.* [11]		✗	16.2	39.1	12.6	28.6	11.5	29.5	10.5	24.2	—	—
ours-cross	mean (std)	✗	11.0 (2.9)	33.7 (9.8)	9.3 (1.4)	30.8 (5.9)	9.1 (1.1)	28.6 (2.2)	7.6 (0.2)	22.2 (0.9)	7.0 (0.9)	25.8 (3.7)
ours-cross	selected	✗	9.8	30.0	8.4	29.4	8.4	27.8	7.2	21.8	5.3	18.3
ours		✓	**9.4**	**30.2**	**8.3**	**28.8**	**8.1**	**27.3**	**5.6**	**22.4**	**3.5**	**12.8**

Table 3: **VS-1, VS-2: Generalization to novel fonts with/without known test glyphs and increasing number of training fonts.** The error rates (in %; ↓ is better) on **FontSynth** test set. *Ours-cross* stands for cross font matching where test glyphs are unknown and training fonts are used as glyph exemplars, mean and standard-dev reported when the exemplar fonts are randomly chosen from the training set, while *selected* shows results from the best matched exemplars automatically chosen based on confidence measure. *R*, *B*, *L* and *I* correspond to the FontSynth training splits; *OS* stands for the Omniglot-Seq dataset (Section 5.2).

module leads to a moderate drop. The similarity disambiguation *(sim. disamb.)* and linear embedding in class aggregator *(agg. embed.)* are both important, especially when the training data is limited. With more training data, the advantage brought by these modules becomes less significant, while improvement from position encoding does not have such a strong correlation with the amount of training data.

5.5. Results

5.5.1 VS-1: Impact of number of training fonts.

We investigate the impact of the number of training fonts on generalization to unseen fonts. For this systematic evaluation, we train the models on an increasing number of FontSynth splits–regular, regular + bold, regular + bold + light, *etc*. and evaluate on FontSynth test set. These splits correspond to increments of 50 new fonts with a different appearance attribute. Table 3 summarizes the results. The three baseline SotA models have similar CER when trained on the same amount of data. *Tesseract* [41] has a slightly better performance but generalizes poorly when there is only one attribute in training. Models with an attention-based LSTM (Attn. Baek *et al.* [4], Chowdhury *et al.* [11]) achieve lower WER than those without due to better language modelling. Notably, our model achieves the

sim. enc. $\mathcal{S}$	sim. disamb.		agg. embed.	training data			
				R		R+B+L+I	
	pos. enc.	self-attn		CER	WER	CER	WER
✓	✓	✓	✓	**9.4**	**30.1**	**5.6**	**22.3**
✓	✗	✓	✓	11.8	37.9	7.9	22.9
✓	✗	✗	✓	23.9	68.8	13.0	52.0
✓	✓	✓	✗	22.9	65.8	8.5	26.4
✓	✗	✗	✗	25.8	63.1	18.4	45.0
✓	—	—	—	49.0	96.2	38.3	78.9

Table 4: **Model component analysis.** The first row corresponds to the full model; the last row 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 respectively.

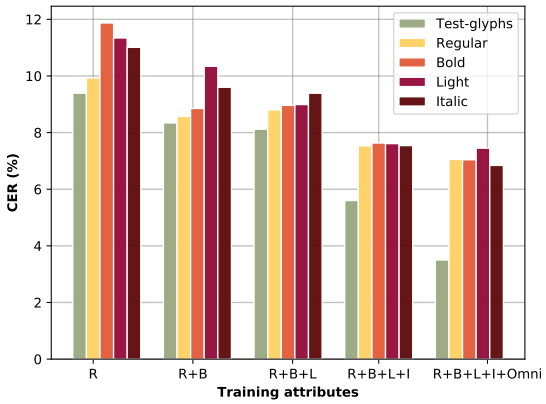


Figure 5: **VS-2: Performance on FontSynth when using different exemplars for cross-font matching.** When using glyphs from other fonts (*e.g.* regular, bold *etc.*) as exemplars, the accuracy is insensitive to the attribute. Further, using test-glyphs as exemplars significantly improve the performance. On the x-axis we show the FontSynth training splits (Figure 3 left).

same accuracy with 1 training attribute (CER=9.4%) as the SotA's with 4 training attributes (CER>10%), *i.e.*, using 150 (=3×50) less training fonts, proving the strong generalization ability of the proposed method to unseen fonts.

Leveraging visual matching. Since, our method does not learn class-specific filters (unlike conventional discriminatively trained models), but instead is trained for visual matching, we can leverage non-English glyphs for training. Hence, we further train on Omniglot-Seq data and drastically reduce the CER from 5.6% (4 attributes) to 3.5%. Being able to leverage language-agnostic data for training is a key strength of our model.

5.5.2 VS-2: Cross font matching.

In VS-1 above, our model assumed privileged access to glyphs from the test image. Here we consider the setting where glyphs exemplars from *training fonts* are used instead. This we term as *cross matching*, denoted 'ours-cross' in Table 3. We randomly select 10 fonts from each font attribute and use those as glyph exemplars. In Table 3 we report the aggregate mean and standard-deviation over all attributes. To automatically find the best font match, we also measure the similarity between the reference and unseen fonts by computing the column-wise entropy in the similarity map $\mathcal{S}$ during inference: Similarity scores within each glyph span are first aggregated to obtain logits $\mathcal{P} \in \mathbb{R}^{|\mathcal{A}| \times W'}$, the averaged entropy of logits over columns $\frac{1}{W'} \sum_i^{W'} -P_i \log(P_i)$ is then used as the criterion to choose the best-matched reference font. Performance from the best-matched exemplar set is reported in 'ours-cross selected' in Table 3. With CER close to the last row where test glyphs are provided, it is shown that the model does not rely on extra information from the new fonts to generalize to different visual styles. Figure 5 details the performance for each attribute separately. The accuracy is largely insensitive to particular font attributes—indicating the strong ability of our model to match glyph shapes. Further, the variation decreases as expected as more training attributes are added.

	CTC Baek [4]		Attn. Baek [4]		Tesseract [41]		Ch. *et al.* [11]		**ours**	
LM	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓
CER	3.5	3.1	5.4	5.4	4.7	3.8	5.5	5.6	**3.1**	**2.4**
WER	**12.9**	11.4	13.1	13.8	15.9	12.2	14.9	15.6	14.9	**8.0**

Table 5: **VS-3: Generalization from synthetic to real data.** Mean error rates (in %; ↓ is better) on Google1000 English document for models trained only on synthetic data (Section 5.5.3). LM stands for 6-gram language model.

5.5.3 VS-3: Transfer from synthetic to real data.

We evaluate models trained with synthetic data on the real-world Google1000 test set for generalization to novel visual fonts and robustness against degradation and other nuisance factors in real data. To prevent giving per test sample specific privileged information to our model, we use a common glyph set extracted from Google1000 (visualized in Figure 4). This glyph set is used for *all* test samples, *i.e.*, is not sample specific. Table 5 compares our model trained on FontSynth+Omniglot-Seq against the SotAs. These models trained on modern fonts are not able to recognize historical ligatures like long s: '*ſ*' and usually classify it as the character '*f*'. Further, they show worse ability for handling degradation problems like fading and show-through, and thus are outperformed by our model, especially when supported by a language model (LM) (CER: ours = 2.4% vs. CTC = 3.14%).

5.5.4 A-1: Transfer to novel Latin alphabets.

We evaluate our model trained on English FontSynth + Omniglot-Seq to other languages in Google1000, namely, French, Italian and Spanish. These languages have more characters than English due to accents (see Table 2). We expand the glyph set from English to include the accented glyphs shown in Figure 4. For comparison, we pick the CTC Baek *et al.* [4] (the SotA with the lowest CER when training data is limited), and adapt it to the new alphabet size

by fine-tuning the last linear classifier layer on an increasing number of training samples. Figure 6 summarizes the results. Images for fine-tuning are carefully selected to cover as many new classes as possible. For all three languages, at least 5 images with new classes are required in fine-tuning to match our performance without fine-tuning; Depending on the number of new classes in this language (for French 16 samples are required). Note that for our model we do not need fine-tuning at all, just supplying exemplars of new glyphs gives a good performance.

5.5.5 A-2: Transfer to non-Latin glyphs.

In the above experiments, the models were both trained and tested on English/Latin script and hence, are not tasked to generalize to completely novel glyph shapes. Here we evaluate the generalization ability of our model to new glyph shapes by testing the model trained on FontSynth + Omniglot-Seq on the Omniglot-Seq test set, which consists of novel alphabets/scripts. We provide our model with glyph exemplars from the randomly generated alphabets (Section 5.2). Our model achieves CER=1.8%/7.9%, WER=7.6%/31.6% (with LM/without LM), which demonstrates strong generalization to novel scripts. Note, the baseline text recognition models trained on FontSynth (English fonts) cannot perform this task, as they cannot process completely new glyph shapes.

6. Conclusion

We have developed a method for text recognition which generalizes to novel visual styles (*e.g.*, fonts, colors, backgrounds *etc.*), and is not tied to a particular alphabet size/language. It achieves this by recasting the classic text recognition as one of visual matching, and we have demonstrated that the matching can leverage random shapes/glyphs (*e.g.*, Omniglot) for training. Our model is perhaps the first to demonstrate one-shot sequence recognition, and achieves superior generalization ability as compared to conventional text recognition methods without requiring expensive adaptation/fine-tuning. Although the method has been demonstrated for text recognition, it is applicable to other sequence recognition problems like speech and action recognition.

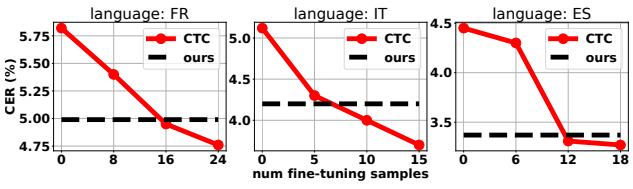


Figure 6: **A-2: Transfer to novel alphabets in Google1000.** We evaluate models trained over the English alphabet on novel languages in the Google1000 dataset, namely, French, Italian and Spanish. CER is reported (in %; ↓ is better).

Acknowledgements. This research is funded by a Google-DeepMind Graduate Scholarship and the EPSRC Programme Grant Seebibyte EP/M013774/1. We would like to thank Triantafyllos Afouras, Weidi Xie, Yang Liu and Erika Lu for discussions and proof-reading.

References

[1] EMNLP 2015 Tenth Workshop On Statistical Machine Translation. http://www.statmt.org/wmt15/. 5

[2] Emmanuel Augustin, Matthieu Carré, Emmanuèle Grosicki, J-M Brodin, Edouard Geoffrois, and Françoise Prêteux. Rimes evaluation campaign for handwritten mail processing. 2006. 23

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 23

[4] Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis. In *Proc. ICCV*, 2019. 1, 2, 5, 6, 7, 8, 22, 23, 24

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 1, 2

[6] Horst Bunke, Samy Bengio, and Alessandro Vinciarelli. Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *PAMI*, 2004. 2

[7] Kaidi Cao, Jingwei Ji, Zhangjie Cao, Chien-Yi Chang, and Juan Carlos Niebles. Few-shot video classification via temporal alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10618–10627, 2020. 2

[8] Zhanzhan Cheng, Fan Bai, Yunlu Xu, Gang Zheng, Shiliang Pu, and Shuigeng Zhou. Focusing attention: Towards accurate text recognition in natural images. In *Proc. ICCV*, 2017. 2

[9] Zhanzhan Cheng, Yangliu Xu, Fan Bai, Yi Niu, Shiliang Pu, and Shuigeng Zhou. Aon: Towards arbitrarily-oriented text recognition. In *Proc. CVPR*, 2018. 1, 2

[10] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, 2014. 1, 2

[11] Arindam Chowdhury and Lovekesh Vig. An efficient end-to-end neural model for handwritten text recognition. *Proc. BMVC*, 2018. 6, 7, 8, 23

[12] Joon Son Chung and Andrew Zisserman. Lip reading in the wild. In *Proc. ACCV*, 2016. 6

[13] Wei Feng, Wenhao He, Fei Yin, Xu-Yao Zhang, and Cheng-Lin Liu. Textdragon: An end-to-end framework for arbitrary shaped text spotting. In *Proc. ICCV*, 2019. 2

[14] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. ICML*, 2017. 2

[15] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proc. ICML*, 2006. 4, 14

[16] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 2005. 1, 2

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 5

[18] Pan He, Weilin Huang, Yu Qiao, Chen Change Loy, and Xiaoou Tang. Reading scene text in deep convolutional sequences. In *Thirtieth AAAI conference on artificial intelligence*, 2016. 2

[19] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. In *Workshop on Deep Learning, NIPS*, 2014. 2, 5, 6, 28

[20] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *Proc. NIPS*, 2016. 4

[21] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, Faisal Shafait, Seiichi Uchida, and Ernest Valveny. ICDAR 2015 robust reading competition. In *Proc. ICDAR*, pages 1156–1160, 2015. 2, 24, 25

[22] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis G. Bigorda, Sergi R. Mestre, Joan Mas, David F. Mota, Jon A. Almazan, and Llus P. de las Heras. ICDAR 2013 robust reading competition. In *Proc. ICDAR*, 2013. 2, 24, 25

[23] Jaeyoung Kim, Mostafa El-Khamy, and Jungwon Lee. Residual lstm: Design of a deep recurrent architecture for distant speech recognition. *arXiv preprint arXiv:1701.03360*, 2017. 23

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[25] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015. 5, 6

[26] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. 1, 2

[27] Chen-Yu Lee and Simon Osindero. Recursive recurrent nets with attention modeling for OCR in the wild. In *Proc. CVPR*, 2016. 1, 2

[28] V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet Physics Doklady*, 1966. 12

[29] Wei Liu, Chaofeng Chen, and Kwan-Yee K Wong. Charnet: A character-aware neural network for distorted scene text recognition. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2

[30] Yang Liu, Zhaowen Wang, Hailin Jin, and Ian Wassell. Synthetically supervised feature learning for scene text recognition. In *Proc. ECCV*, 2018. 2

[31] Pengyuan Lyu, Minghui Liao, Cong Yao, Wenhao Wu, and Xiang Bai. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. In *Proc. ECCV*, 2018. 2

[32] Andrew Maas, Ziang Xie, Dan Jurafsky, and Andrew Ng. Lexicon-free conversational speech recognition with neural networks. In *NAACL-HLT*, 2015. 5

[33] U-V Marti and Horst Bunke. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, 2002. 23

[34] A. Mishra, K. Alahari, and C. V. Jawahar. Scene text recognition using higher order language priors. In *BMVC*, 2012. 24, 25

[35] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. ICML*, 2010. 5

[36] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 5

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, pages 234–241. Springer, 2015. 5

[38] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *PAMI*, 2016. 2

[39] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Robust scene text recognition with automatic rectification. In *Proc. CVPR*, 2016. 2

[40] Baoguang Shi, Mingkun Yang, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Aster: An attentional scene text recognizer with flexible rectification. *PAMI*, 2018. 1, 2

[41] Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007. 2, 5, 7, 8, 22

[42] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Proc. NIPS*, 2017. 2, 4, 6

[43] Bolan Su and Shijian Lu. Accurate scene text recognition based on recurrent neural network. In *Proc. ACCV*, 2014. 2

[44] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018. 2, 4

[45] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. 1, 2

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. NIPS*, 2017. 4, 5

[47] L. Vincent. Google book search: Document understanding on a massive scale. In *PROC. ninth International Conference on Document Analysis and Recognition (ICDAR)*, pages 819–823, Washington, DC, 2007. 6, 7, 27

[48] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Proc. NIPS*, 2016. 2, 4

[49] Kai Wang and Serge Belongie. Word spotting in the wild. In *Proc. ECCV*, 2010. 2, 24, 25

[50] Cong Yao, Xiang Bai, Baoguang Shi, and Wenyu Liu. Strokelets: A learned multi-scale representation for scene text recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4042–4049, 2014. 24

[51] Fangneng Zhan and Shijian Lu. Esir: End-to-end scene text recognition via iterative image rectification. In *Proc. CVPR*, 2019. 2

Appendix

Contents

A. 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* [28]; $\text{Length}\left(\boldsymbol{y}_{\text{gt}}\right)$ 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.

B. Ablation study

B.1. Ablation on modules at a larger scale

In Section 5.4, 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 6, we ablate the same model components, but couple it with increasing number of training fonts from the FontSynth dataset (Section 5.2), 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.

B.2. Ablation on balance of losses

Figure 7 shows the impact of λ when training with one FontSynth attribute, where λ is the weight on L_{sim} as in Equation (4). The use of L_{sim} is essential as the model does not converge when its ratio is smaller than 0.05, labelled by 'x' mark in Figure 7. The CER is also increased by about 1% when its weight is five times larger than that on L_{pred}.

	sim. enc. $\mathcal{S}$	sim. disamb.		agg. embed.	training data									
					R		**R+B**		**R+B+L**		**R+B+L+I**		**R+B+L+I+OS**	
		pos. enc.	self-attn		**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 6: **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.

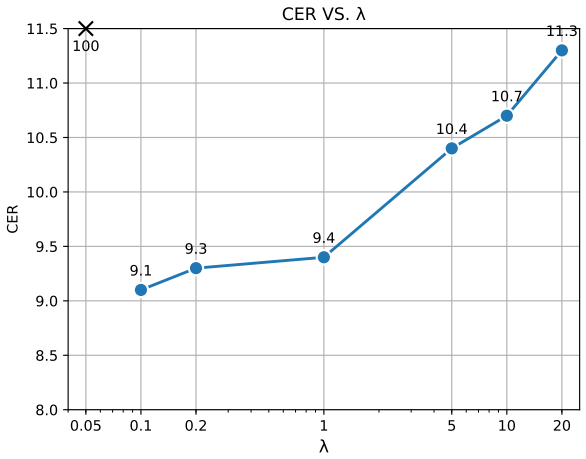


Figure 7: **Impact of λ on CER when training with one FontSynth attribute.**

C. Examples and Visualizations

In Figures 9 to 13 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). 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 8 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 [15] 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.

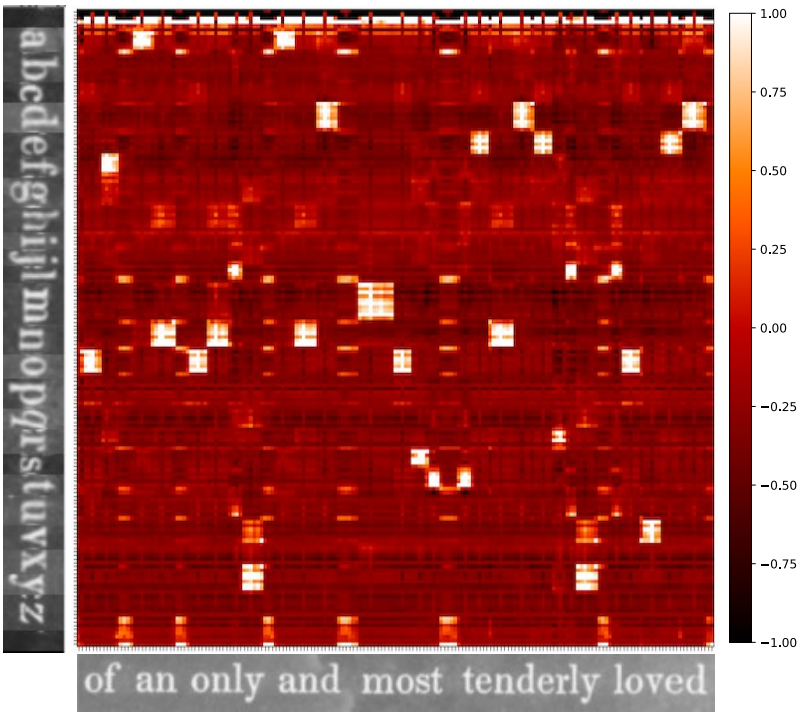


Figure 8: **An example of a similarity map $\mathcal{S}^*$ after the self-attention module.**

Figure 9: **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.

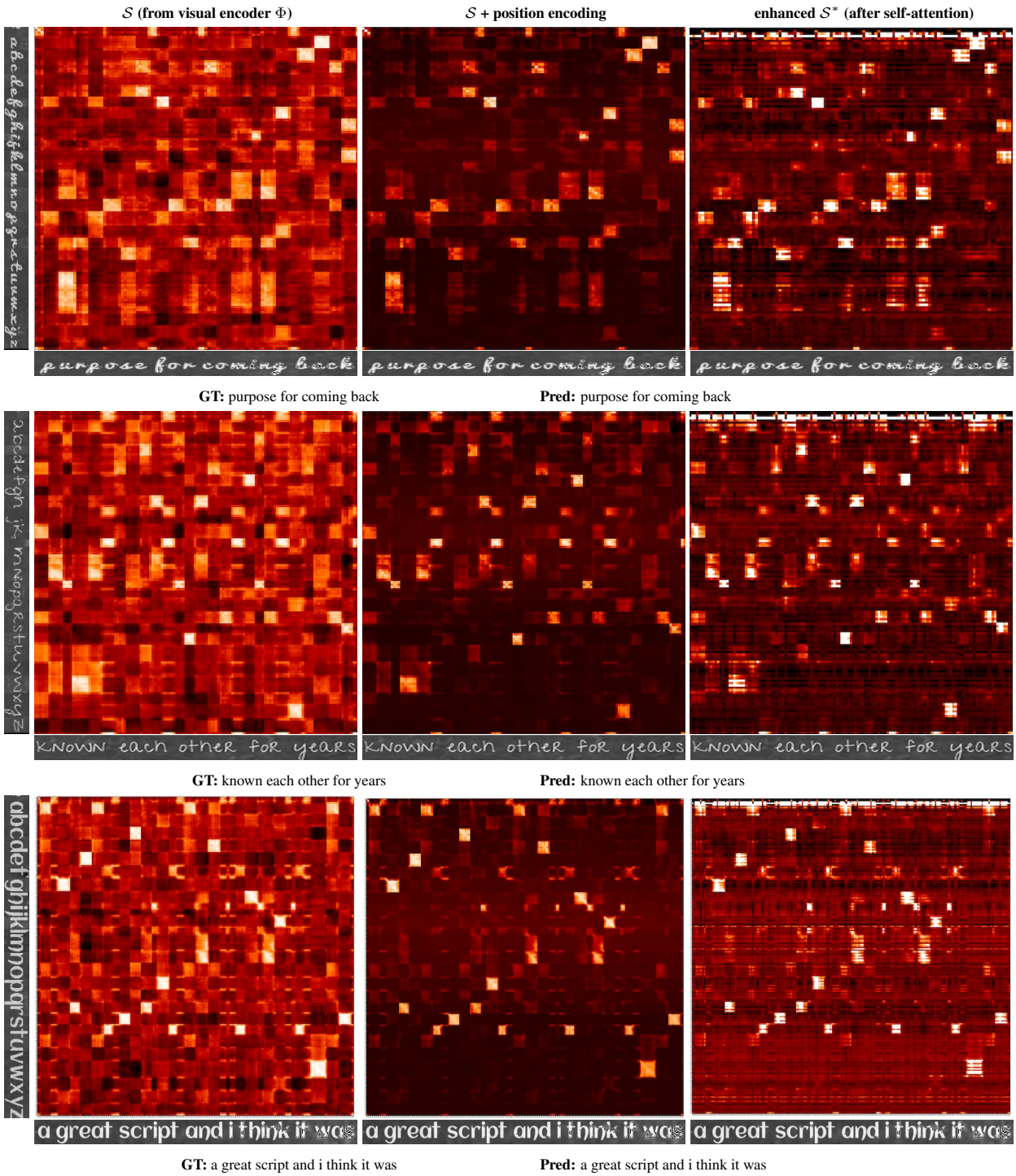


Figure 10: **Experiment VS-2: Cross font matching.** Text recognition on the FontSynth test set with *cross-font matching* – using exemplars from the training set when we do not have access to the test font. The model succeeds at cross-font matching even when the exemplars and line images are in very 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.

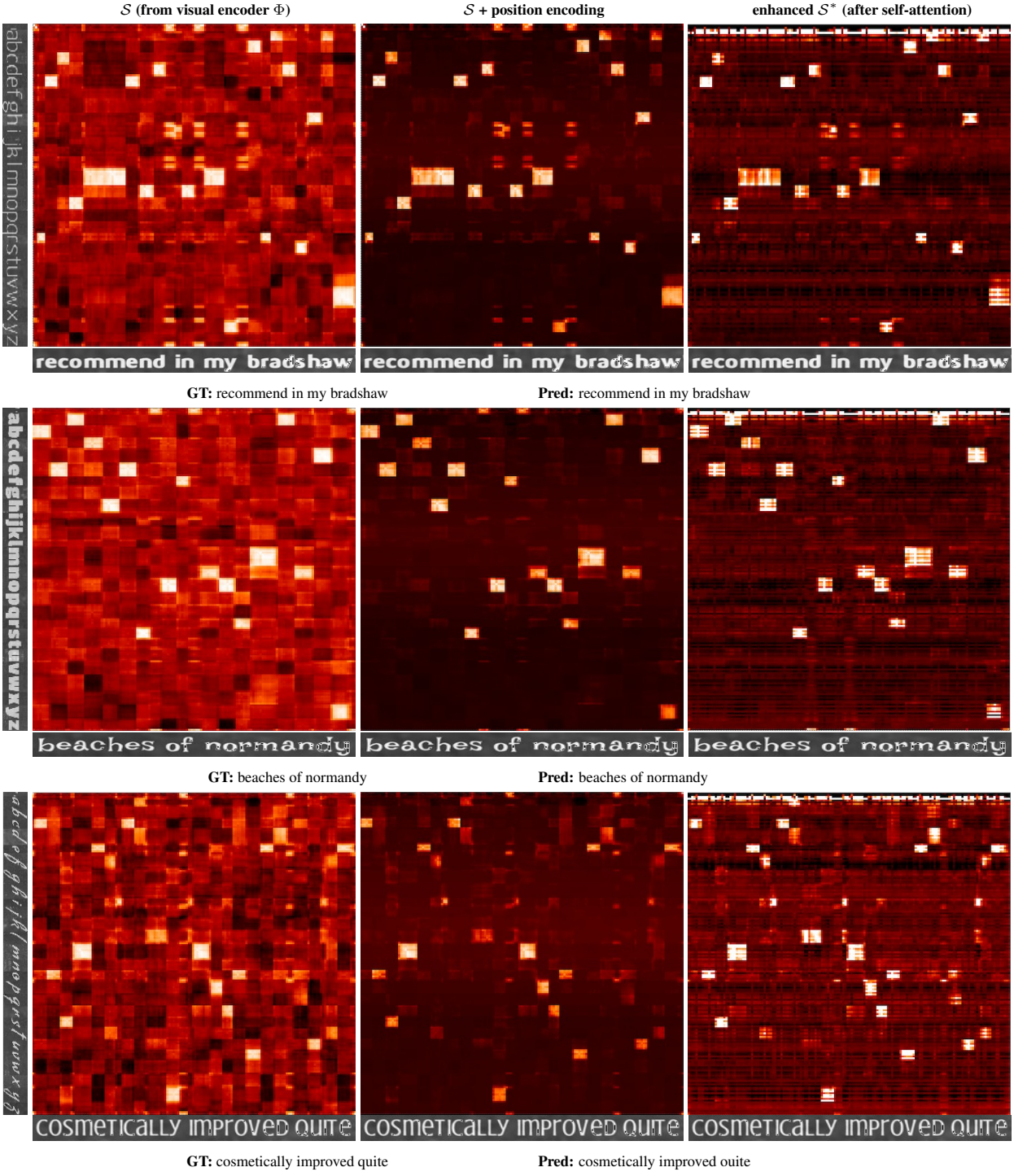


Figure 11: **Experiment VS-3: Transfer from synthetic to real data.** We test our model trained purely on *synthetic data* for generalization to *real world* Google1000 English books. The model is robust to nuisance factors prevalent in scans of real historical books, *e.g.* over-exposure and low contrast, as well as degradation and show-through.

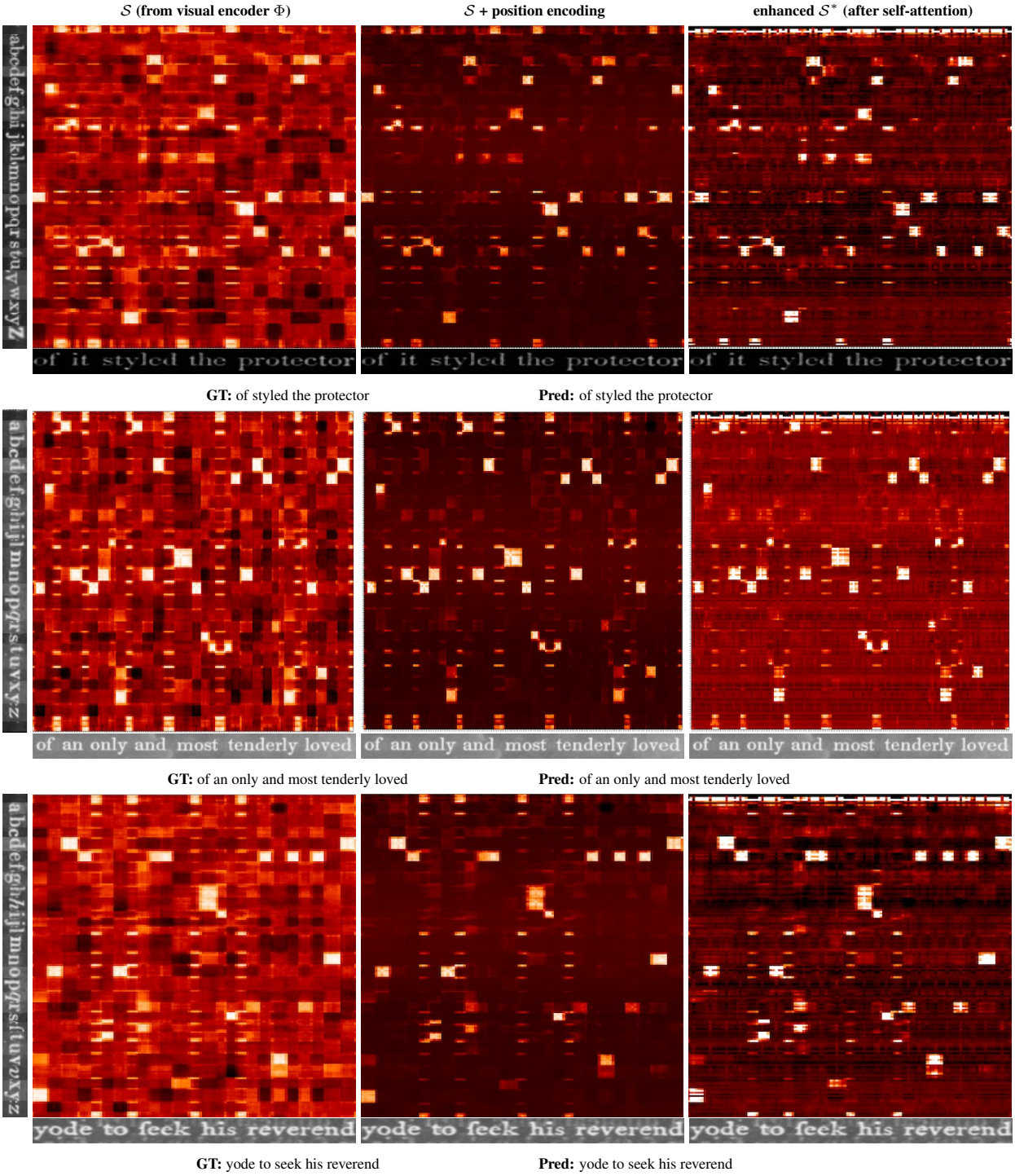
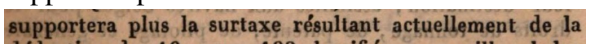


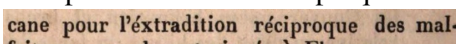
Figure 12: **Experiment A-1: Transfer to novel Latin alphabets.** To test generalization to unseen alphabets/novel languages, we test how well our model trained on English transfers to all the other languages in Google1000, namely, French, Spanish and Italian. Below, we show four text-line samples from 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

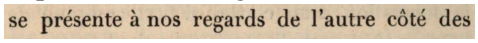
GT: supportera plus la surtaxe résultant actuellement de la
Pred: supportera plus la surtaxe résultant actuellement de la



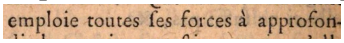
GT: cane pour léxtradition réciproque des mal
Pred: cane pour léxtradition réciproque des mal



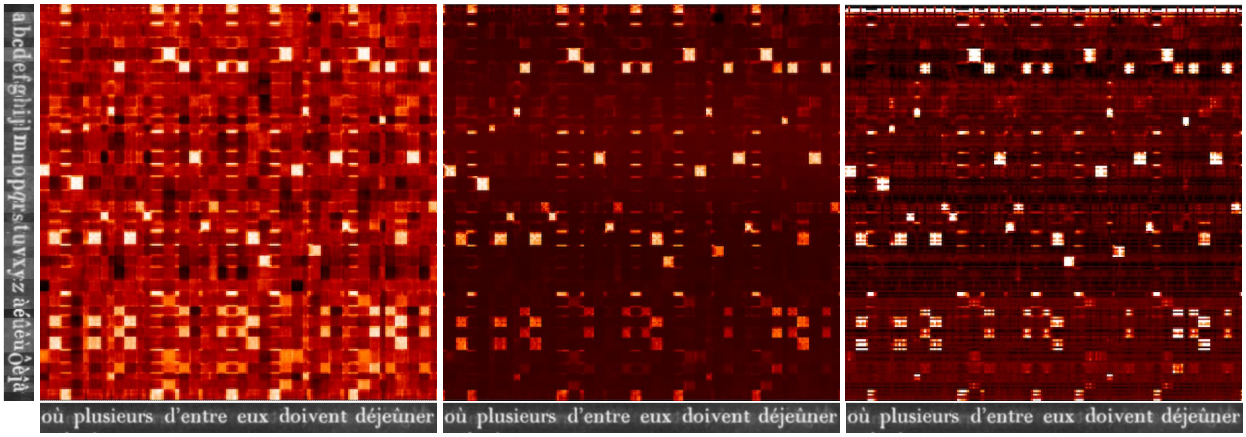
GT: se présente à nos regards de lautre côté des
Pred: se présente à nos regards de lautre côté des



GT: emploie toutes les forces à approfon
Pred: emploie toutes res forces à approfon



$\mathcal{S}$ **(from visual encoder** Φ**)** $\mathcal{S}$ **+ position encoding** **enhanced** $\mathcal{S}^*$ **(after self-attention)**

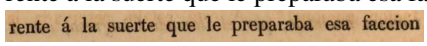


GT: où plusieurs dentre eux doivent dêjeûner **Pred:** où plusieurs dentre eux doivent dêjeûner

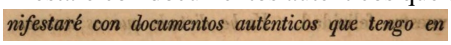
Spanish

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

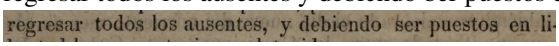
GT: rente á la suerte que le preparaba esa faccion
Pred: rente á la suerte que le preparaba esa faccion



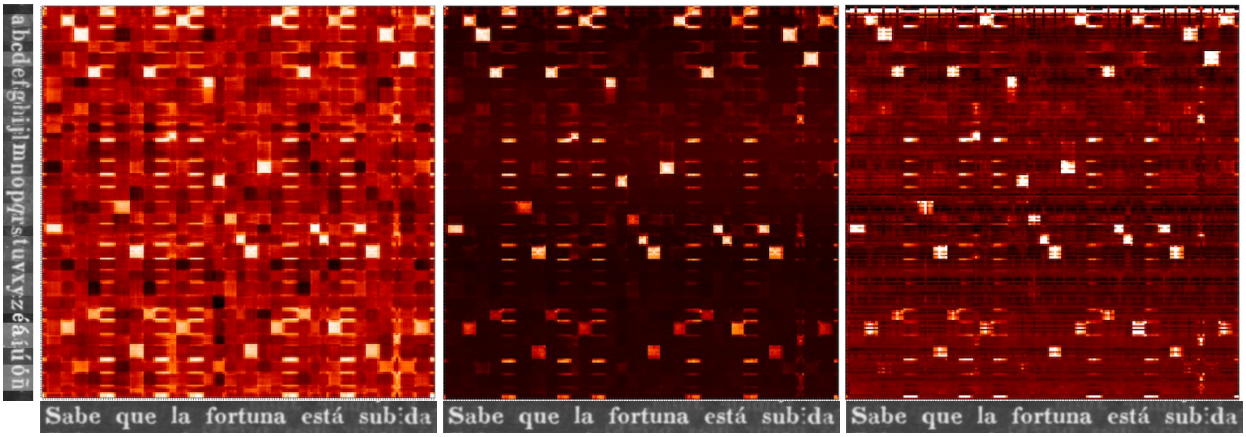
GT: nifestaré con documentos auténticos que tengo en
Pred: nifestaré con documentos auténticos que tengo en



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



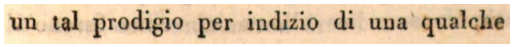
$\mathcal{S}$ **(from visual encoder** Φ**)** $\mathcal{S}$ **+ position encoding** **enhanced** $\mathcal{S}^*$ **(after self-attention)**



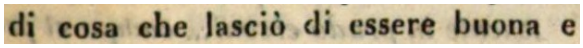
GT: sabe que la fortuna está subda **Pred:** sabe que la fortuna está subda

Italian

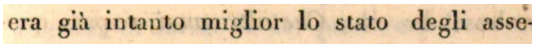
GT: un tal prodigio per indizio di una qualche
Pred: un tal prodigio per indizio di una qualche



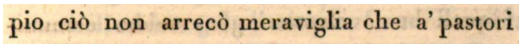
GT: di cosa che lasciò di essere buona e
Pred: di cosa che lasciò di essere buona e



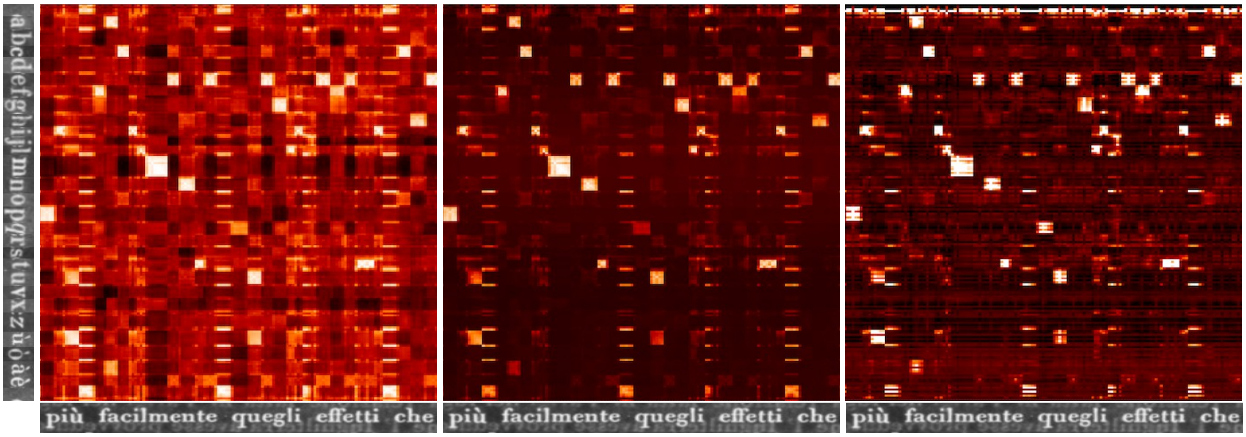
GT: era già intanto miglior lo stato degli asse
Pred: era già intanto miglior lo stato degli asse



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

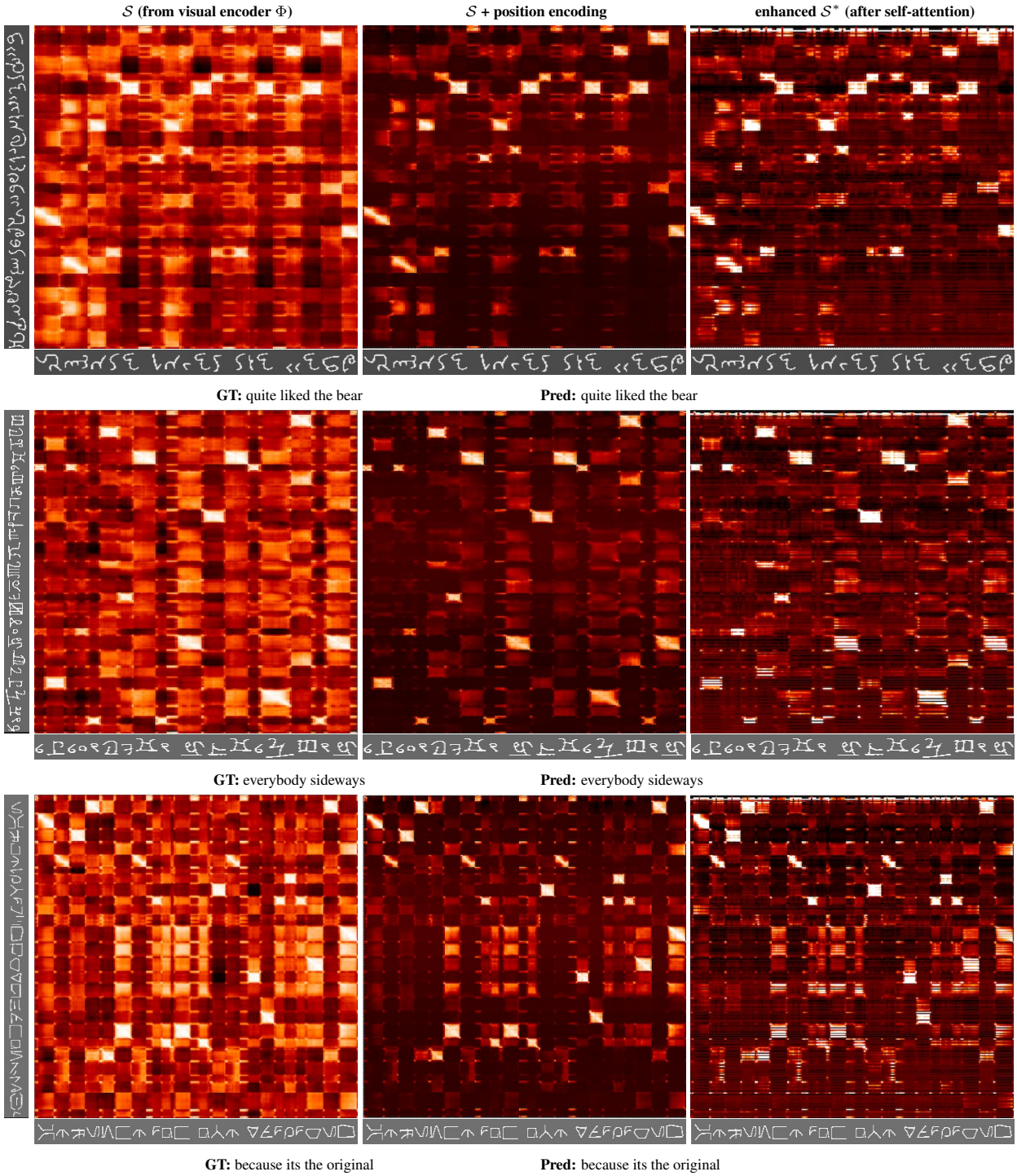


$\mathcal{S}$ **(from visual encoder** Φ**)** $\mathcal{S}$ **+ position encoding** **enhanced** $\mathcal{S}^*$ **(after self-attention)**



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

Figure 13: **Experiment A-2: 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.



D. 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.

D.1. Attn. and CTC model – Baek *et al*. [4]

We follow the recent work of Baek *et al*. [4] 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. We do not use the first transformation stage as

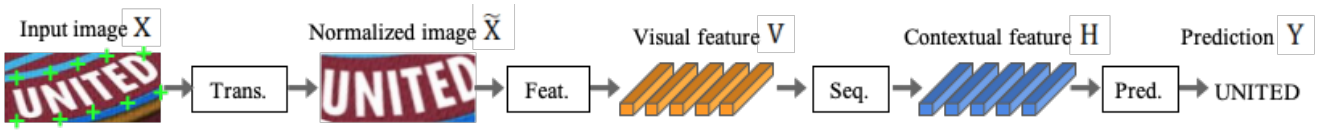


Figure 14: **Four stages of modern text-recognition methods.** State-of-the-art text recognition models constitute of four stages – transformation, feature extraction, sequence modelling and prediction. We compare our proposed method against two of the strongest text-recognition models. Figure reproduced from [4]

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): 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.

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 7: Architecture details of the visual encoder used for feature extraction.

D.2. Industry Standard Model – Tesseract [41]

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 the model is trained on our four training attributes 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 [2], details are shown in Table 8.

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 8: Architecture details of Tesseract.

D.3. Handwritten Text Model – Chowdhury *et al*. [11]

The architecture of the handwritten text recognition model from Chowdhury *et al*. [11] is similar to the *Attn.* model in Baek *et al*. [4] – 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 [23] and Layer Normalization [3] in their LSTMs. Codes of this model are not available online. Therefore, we verify our implementation by training and testing on handwriting benchmarks IAM [33] and RIMES [2]. We compare the numbers they report 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 consistently 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 9: Verification of our implementation of the SotA model [11] in handwritten text recognition. We compare the error rates they report in their paper with those from our implementation on IAM and RIMES.

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

E. Performance on Scene Text

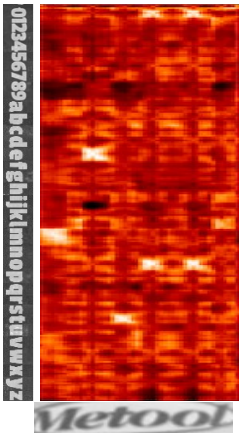
Although our model are trained to recognize rectilinear text images, it can be applied to scene text with a transformation module. We use the pre-trained transformation module from Baek *et al*. [4] to transform the images and test our model on commonly used scene text dataset: SVT [49], ICDAR [21, 22] and IIIT5k [34]. 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*. [50] which also learn to recognizes texts by using visual primitives, and show the results in Table 10.

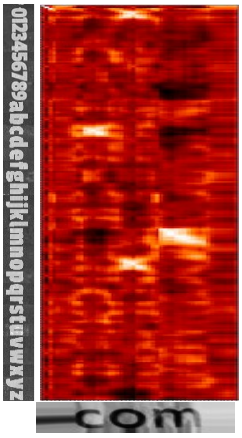
Model	**IIIT5K (small)**	**IIIT5K (medium)**	**IC03 (full)**	**IC03 (50)**	**SVT**
Yao *et al*. [50]	80.2	69.3	80.3	88.5	75.9
Ours	**96.2**	**92.8**	**93.3**	**98.2**	**92.4**

Table 10: Results on scene text dataset IIIT5K, IC03, SVT with lexicon.

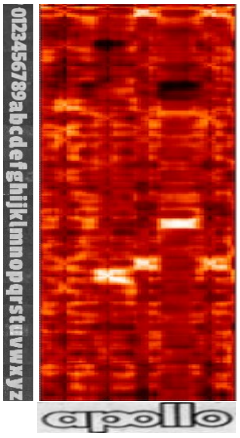
Figure 15: **Visualization of our model's performance on scene text datasets: SVT [49], ICDAR [21,22] and IIIT5k [34].** We show the alphabet image, scene text image and the similarity map from the feature encoder for each example.



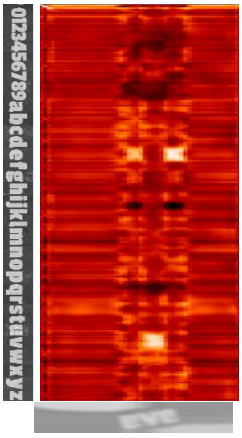
GT: metool **Pred:** metool



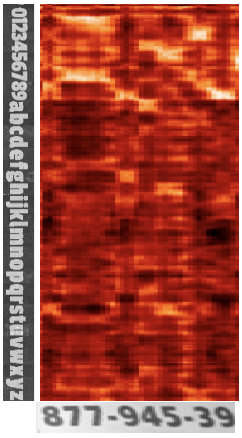
GT: com **Pred:** com



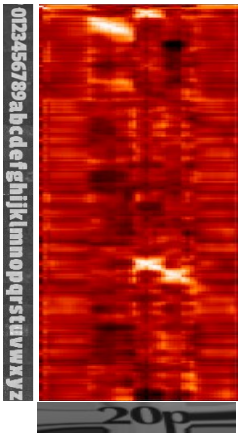
GT: apollo **Pred:** apollo



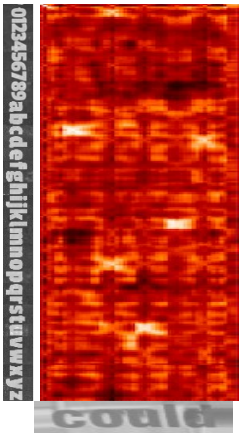
GT: eve **Pred:** eve



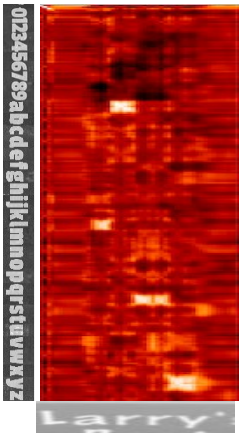
GT: 87794539 **Pred:** 87794539



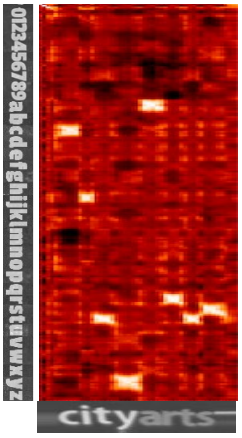
GT: 20p **Pred:** 20p



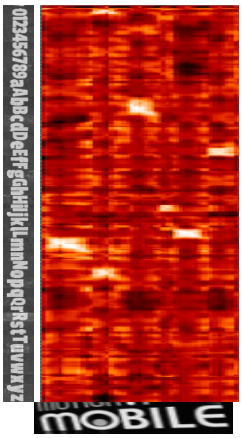
GT: could **Pred:** could



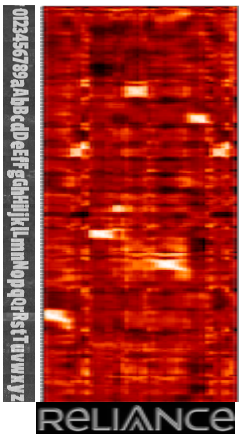
GT: larry **Pred:** larry



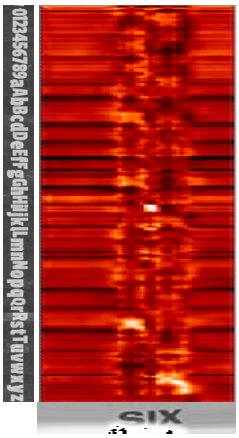
GT: cityarts **Pred:** cityarts



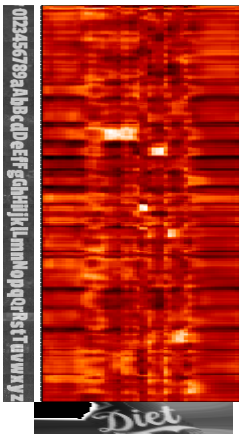
GT: mobile **Pred:** mobile



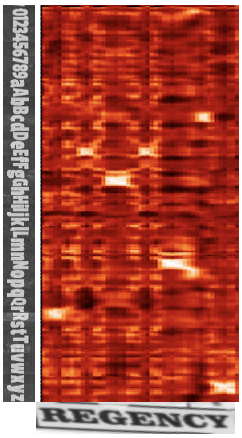
GT: reliance **Pred:** reliance



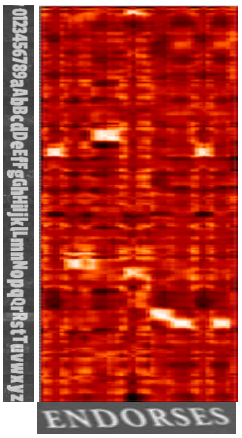
GT: six **Pred:** six



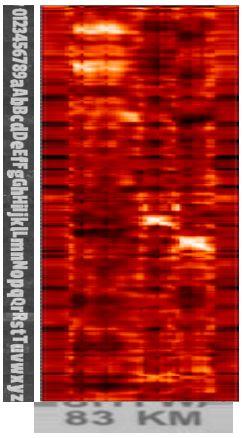
GT: diet **Pred:** diet



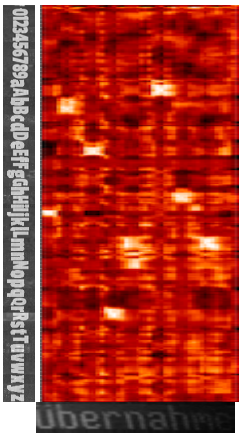
GT: regency **Pred:** regency



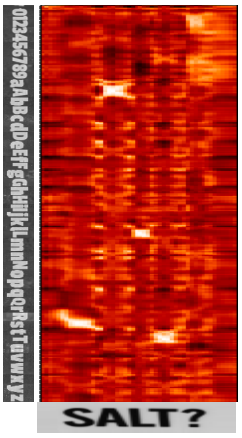
GT: endorses **Pred:** endorses



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



GT: ubernahme **Pred:** jbernahme



GT: salt **Pred:** salt2

F. Dataset Details

F.1. Training and Testing Datasets

We summarize the distribution and size of all our training and testing datasets in Table 11.

datasets	distribution	number of images	language
training			
FontSynth	200 fonts	200000	English
Omniglot-Seq	30 alphabets $\times$ 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 $\times$ 20 writers	40000	Various

Table 11: Details of training and testing datasets used in experiments VS1-3 and A1-2.

F.2. Google1000 Dataset Details

We benchmark our method on a large-scale real world dataset of historical books – Google1000 [47] (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 Latin 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

F.3. FontSynth Dataset Details

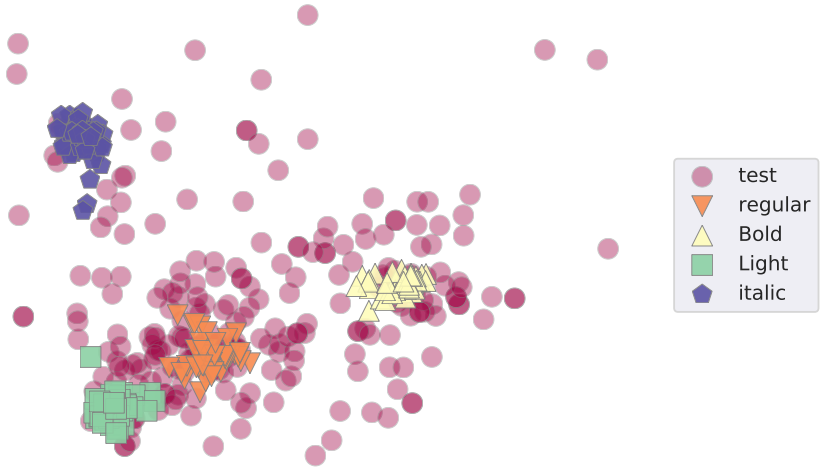


Figure 16: **FontSynth font embeddings.** The training font splits (**regular**, **bold**, **light**, **italic**) are distinct from the **test** fonts.

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 fonts used in MJSynth [19] 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 16 visualizes font-embeddings from a font classifier 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.

Please refer to Figure 3 for a visualization of the fonts in these splits.