Adaptive Text Recognition through Visual Matching

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

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 retraining, 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.

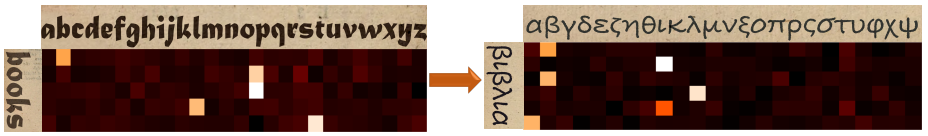
Code, data, and model checkpoints are available at: http://www.robots.ox.ac.uk/~vgg/research/FontAdaptor20/.

Keywords: text recognition, sequence recognition, similarity maps

1 Introduction

Our objective in this work is *generalization* and *flexibility* in text recognition. Modern text recognition methods [2, 7, 23, 32] 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>}).



English: alphabet-size: 26 characters **Greek:** novel glyph shapes and alphabet-size: 24 characters

Fig. 1: **Visual matching for text recognition.** Current text recognition models learn discriminative features specific to character shapes (*glyphs*) from a predefined (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/languages, and extensible to novel character classes, e.g., English → Greek, *without* further training. Brighter colors correspond to higher visual similarity.

Current deep learning based methods [7, 23, 32] cast this in the encoder-decoder framework [8, 37], where first the text-line image is encoded through a visual ConvNet [22], followed by a recurrent neural network decoder, with alignment between the visual features and text achieved either through attention [3] or Connectionist Temporal Classification (CTC) [13].

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 [41], ICDAR [18,19] datasets) where each instance has a unique visual style (results in supplementary material).

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 [34], Matching Nets [40], Relation Nets [36], and MAML [11], 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. [5] 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 [4, 22], and document optical character recognition (OCR) [33]. The OCR industry standard—*Tesseract* [33]—employs specialized training data for each supported language/alphabet.[3] Our model enables rapid adaptation to

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

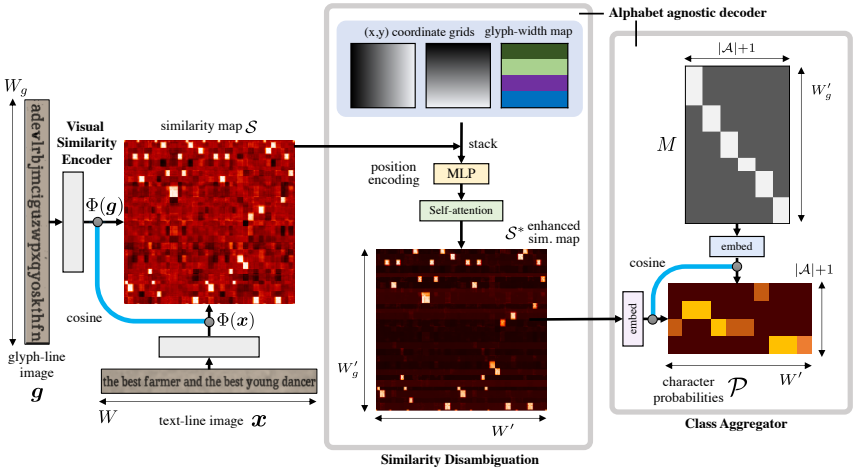


Fig. 2: **Visual matching for text recognition.** We cast the problem of text recognition as one of visual matching of glyph exemplars in the given text-line image. The visual encoder Φ embeds the glyph-line $\boldsymbol{g}$ and text-line $\boldsymbol{x}$ images and produces a similarity map $\mathcal{S}$, 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 $\mathcal{M}$.

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 [16], or take an encoder-decoder approach [8,37]. 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) [13,15,30,35] or through attention [3,6,7,23,31]. Recognizing irregularly shaped text has garnered recent interest which has seen a resurgence of dense character-based segmentation and classification methods [28,42]. Irregular text is rectified before feature extraction either using geometric transformations [24,31,32,44] or by re-generating the text image in canonical fonts and colors [43]. Recently, Baek et al. [2] present a thorough evaluation of text recognition methods, unifying them in a four-stage framework—input transformation, feature extraction, sequence modeling, and string prediction.

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 alphabet, 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 $\mathcal{S} \in [-1, 1]^{W'_g \times W'}$:

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

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 character in the alphabet 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 alphabet 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} character in the alphabet $\mathcal{A}$.

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 character 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) a *glyph width-map* $\mathcal{G}$: $\mathcal{G} = \boldsymbol{w}_g \mathbb{1}^T$, 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 over all the glyphs (columns of $\mathcal{S}$), we use a self-attention module [38] which 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 alphabet 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 alphabet $\mathcal{P}$ are supervised using the CTC loss [12] ($\mathcal{L}_{CTC}$) 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,

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

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

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

where, SoftMax(·) normalization is over the alphabet (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 reasonbale range (see supplementary), 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 [17, 34, 36, 40] 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 alphabet. 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.

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 $1\times\frac{W}{2}\times256$. 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

Table 1: **Visual encoder architecture** (Sections 3.1 and 4.1). The input is an image of size $32\times W\times1$ (height×width×channels).

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

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 [29] with two residual blocks [14]; 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 [38] with three layers with four attention heads 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 [26].

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).

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), 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 [25] (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 [20] 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 [27].

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.

Fig. 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 [16], and *Omniglot-Seq* **(bottom)** generated using glyphs from Omniglot [21] as fonts (Section 5.2).

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. [2] for scene-text recognition; (ii) Tesseract [33], the industry standard for document OCR; and (iii) Chowdhury et al. [9] 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 [2] (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 verifcation of our implementations is provided in the supplementary material.

5.2 Datasets and Metrics

FontSynth. We take 1400 fonts from the MJSynth dataset [16] 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 [10] is used as the text source. We call this dataset *FontSynth*; visualization in Figure 3 (right) and further details in the supplementary.

Omniglot-Seq. Omniglot [21] 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.

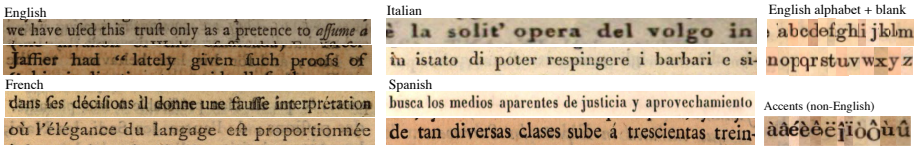


Fig. 4: **Google1000 printed books dataset. (left):** Text-line image samples from the Google1000 [39] evaluation set for all the languages, namely, English, French, Italian and Spanish. **(right):** *Common* set of glyph exemplars used in our method for *all* books in the evaluation set for English and accents for the other languages.

Google1000. Google1000 [39] is a standard benchmark for document OCR released as part of 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 supplementary.

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.

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

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

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 glyph 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 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 3 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 Protypical Nets [34] 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 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.

Table 3: **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.

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

5.5 Results

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 4 summarizes the results. The three baseline SotA models have similar CER when trained on the same amount of data. *Tesseract* [33] 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. [2], Chowdhury et al. [9]) achieve lower WER than those without due to better language modelling. Notably, our model achieves the same accuracy

Table 4: **VS-1, VS-2: Generalization to novel fonts with/without known test glyphs and increasing number of training fonts.** The mean error rates (in %; ↓ is better) on **FontSynth** test set. For cross matching (*ours-cross*), standard-dev is reported in parenthesis. *R*, *B*, *L* and *I* correspond to the FontSynth training splits; *OS* stands for the Omniglot-Seq dataset (Section 5.2).

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 Beak et al. [2]	✗	17.5	46.1	11.5	30.3	10.4	28.2	10.4	27.7	—	—
Attn. Beak et al. [2]	✗	16.5	41.0	12.7	34.5	11.1	27.4	10.3	23.6	—	—
Tesseract [33]	✗	19.2	48.6	12.3	37.0	10.8	31.7	9.1	27.8	—	—
Chowdhury et al. [9]	✗	16.2	39.1	12.6	28.6	11.5	29.5	10.5	24.2	—	—
ours-cross mean	✗	11.0	33.7	9.3	30.8	9.1	28.6	7.6	22.2	7.0	25.8
ours-cross std	✗	(2.9)	(9.8)	(1.4)	(5.9)	(1.1)	(2.2)	(0.2)	(0.9)	(0.9)	(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**

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.

VS-2: Cross glyph 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 4. We randomly select 10 fonts from each font attribute and use those as glyph exemplars. In Table 4 we report the aggregate mean and standard-deviation over all attributes. To automatically find the best font match, we also measure the

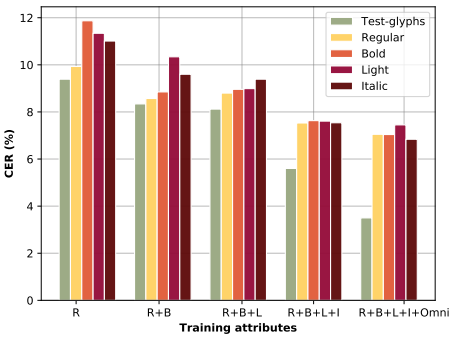


Fig. 5: **VS-2: Cross matching on FontSynth.** Our model maintains its performance when using training fonts as glyph exemplars instead of test-image glyphs (refer to Section 5.5). On the x-axis we show the FontSynth training splits (Figure 3 left).

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 4. 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.

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%).

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). LM stands for 6-gram language model.

	CTC Beak [2]		Attn. Beak [2]		Tesseract [33]		Ch. et al. [9]		**ours**	
LM	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓
CER	3.5	3.14	5.4	5.4	4.65	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**

A-1: Transfer to novel 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 comparsion, we pick the CTC Baek et al. [2] (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

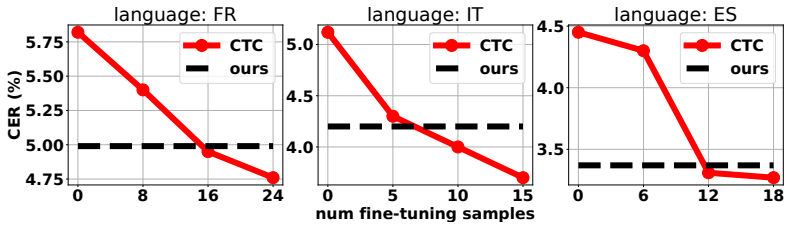


Fig. 6: **A-2: Transfer to novel alphabets in Google1000.** We evaluate models trained over the English alphabet on novel languages in the Google1000 dataset, namely, French, Italian and Spanish. CER is reported (in %; ↓ is better).

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.

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.

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/ 8
2. 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) 1, 4, 8, 9, 11, 12, 13
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014) 2, 4
4. Bunke, H., Bengio, S., Vinciarelli, A.: Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. PAMI **26**(6), 709–720 (2004) 3
5. Cao, K., Ji, J., Cao, Z., Chang, C.Y., Niebles, J.C.: Few-shot video classification via temporal alignment. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10618–10627 (2020) 3
6. Cheng, Z., Bai, F., Xu, Y., Zheng, G., Pu, S., Zhou, S.: Focusing attention: Towards accurate text recognition in natural images. In: Proc. ICCV (2017) 4
7. Cheng, Z., Xu, Y., Bai, F., Niu, Y., Pu, S., Zhou, S.: Aon: Towards arbitrarily-oriented text recognition. In: Proc. CVPR (2018) 1, 2, 4
8. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: EMNLP (2014) 2, 4
9. Chowdhury, A., Vig, L.: An efficient end-to-end neural model for handwritten text recognition. Proc. BMVC (2018) 8, 11, 12, 13
10. Chung, J.S., Zisserman, A.: Lip reading in the wild. In: Proc. ACCV (2016) 9
11. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: Proc. ICML (2017) 3
12. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proc. ICML. ACM (2006) 6
13. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional lstm and other neural network architectures. Neural Networks **18**(5-6), 602–610 (2005) 2, 4
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. CVPR (2016) 7
15. He, P., Huang, W., Qiao, Y., Loy, C.C., Tang, X.: Reading scene text in deep convolutional sequences. In: Thirtieth AAAI conference on artificial intelligence (2016) 4
16. 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) 4, 9
17. Jia, X., De Brabandere, B., Tuytelaars, T., Gool, L.V.: Dynamic filter networks. In: Proc. NIPS (2016) 7
18. 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. pp. 1156–1160 (2015) 3
19. 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. In: Proc. ICDAR (2013) 3

20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 8
21. Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B.: Human-level concept learning through probabilistic program induction. Science (2015) 9
22. LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Computation **1**(4), 541–551 (1989) 2, 3
23. Lee, C.Y., Osindero, S.: Recursive recurrent nets with attention modeling for OCR in the wild. In: Proc. CVPR (2016) 1, 2, 4
24. Liu, W., Chen, C., Wong, K.Y.K.: Char-net: A character-aware neural network for distorted scene text recognition. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018) 4
25. Maas, A., Xie, Z., Jurafsky, D., Ng, A.: Lexicon-free conversational speech recognition with neural networks. In: NAACL-HLT (2015) 8
26. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proc. ICML (2010) 8
27. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017) 8
28. Pengyuan, L., Minghui, L., Cong, Y., Wenhao, W., Xiang, B.: Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. In: Proc. ECCV (2018) 4
29. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Proc. MICCAI. pp. 234–241. Springer (2015) 7
30. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. PAMI (2016) 4
31. Shi, B., Wang, X., Lyu, P., Yao, C., Bai, X.: Robust scene text recognition with automatic rectification. In: Proc. CVPR (2016) 4
32. Shi, B., Yang, M., Wang, X., Lyu, P., Yao, C., Bai, X.: Aster: An attentional scene text recognizer with flexible rectification. PAMI (2018) 1, 2, 4
33. 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) 3, 8, 11, 12, 13
34. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Proc. NIPS (2017) 3, 7, 11
35. Su, B., Lu, S.: Accurate scene text recognition based on recurrent neural network. In: Proc. ACCV (2014) 4
36. Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P.H., Hospedales, T.M.: Learning to compare: Relation network for few-shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1199–1208 (2018) 3, 7
37. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in neural information processing systems. pp. 3104–3112 (2014) 2, 4
38. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Proc. NIPS (2017) 6, 8
39. 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) 9, 10
40. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. In: Proc. NIPS (2016) 3, 7

41. Wang, K., Belongie, S.: Word spotting in the wild. In: Proc. ECCV (2010) 3
42. Wei, F., Wenhao, H., Fei, Y., Xu-Yao, Z., Cheng-Lin, L.: Textdragon: An end-to-end framework for arbitrary shaped text spotting. In: Proc. ICCV (2019) 4
43. Yang, L., Zhaowen, W., Hailin, J., Ian, W.: Synthetically supervised feature learning for scene text recognition. In: Proc. ECCV (2018) 4
44. Zhan, F., Lu, S.: Esir: End-to-end scene text recognition via iterative image rectification. In: Proc. CVPR (2019) 4