On the Generalization of Handwritten Text Recognition Models

Supplementary Material

7. Complementary results

In this section, we provide additional metrics to the Character Error Rate (CER) presented in the main paper. We provide the following information regarding the results: the main table of the paper (Table 3) is presented in terms of Word Error Rate (WER) in Table 4 and the dataset source for the best test result obtained for each architecture is reported in Table 5. Additionally, Tables 6 and 7 present all cross-domain results (both real and synthetic, respectively) for each architecture individually. In Table 6, the in-domain (ID) results are displayed on the main diagonals and highlighted in gray.

We observe in Table 4 that the ID results remain generally consistent, comparable to those reported in the literature. However, the performance in the OOD scenario remains significantly poor. Additionally, the numerical values highlight that the WER, being a stricter metric than the CER, amplifies the performance gap in the OOD scenario. While the average ID to OOD gap was 37.6% in terms of CER, it increases to 60.3% when measured using WER. Table 5 presents the best-performing source domain for each target domain in the OOD scenario. The dominance of the IAM dataset is noticeable, accounting for nearly 60% of cases, followed by Bentham with slightly over 20%. However, as reported in Section 4.3, the choice of source domain has minimal impact on OOD performance, as all results remain poor even when the target domain is known (oracle scenario). Additionally,

Lastly, we compare three state-of-the-art VLMs [22, 48, 59] including TrOCR [46] against the best-reported OOD results in the paper (HTR_{OOD} column) in Table 8. As observed, the zero-shot performance of these models is very low, as they are not originally designed for HTR tasks, with TrOCR being on par with the HTR models on the English datasets. Moreover, there is no established pipeline for effectively applying VLMs to such specific HTR datasets, highlighting the need for further investigation.

8. Hyperparameters

8.1. Architectures implementation

As stated in the main paper, we aimed to follow the implementation closest to the original papers using the available information for those that did not provide code. In all cases, the most significant architectural change occurred in the final prediction layer, where the output vocabulary size was adjusted to match the vocabulary size (94) reported in Section 3 of the main text.

8.2. Data augmentation

We detail the parameters used for data augmentation during training. No transformations are applied during validation or testing, except for padding, which is applied equally across the validation, training, and test splits. All transformations are applied independently with a 50% probability. For the transformations, we utilized those available in version 2 of transformations in torchvision (`torchvision.transforms.v2`). To simplify visualization and shorten the names, we directly referenced the v2 submodule. For operations involving OpenCV, we employed the opencv-python library (`cv2` module) to execute OpenCV transformations directly.

- **Dilation** (`Custom transformation`):
  - Parameters: kernel size = 3; iterations = 1.
- **Erosion** (`Custom transformation`):
  - Parameters: kernel size = 2; iterations = 1.
- **Elastic Transform** (`v2.ElasticTransform`):
  - Parameters: sigma = 5.0; alpha = 5.0; fill = 255 (white).
- **Random Affine (Rotation, Translation, Shear)** (`v2.RandomAffine`):
  - Parameters: rotation degrees = ± 1; translation = 1% horizontally and up to 5% vertically; shear = ± 1 pixels (sheared by a factor of 5); fill = 255 (white).
- **Perspective** (`v2.RandomPerspective`):
  - Parameters: Distortion scale = 0.1; fixed probability of applying the distortion = 100%; fill = 255 (white).
- **Gaussian Blur (Noise)** (`v2.GaussianBlur`):
  - Parameters: kernel size = 3; sigma = 2.0.
- **Padding** (`v2.Pad`):
  - Parameters: padding = 15 pixels on the left and right; fill = 255 (white).
- **Grayscale** (`v2.Grayscale`):
  - Parameters: num_output_channels = 1
- **Convert to Tensor** (`v2.ToTensor`):
  - Converts the input data to a PyTorch tensor format.

Dilation details. The image is first inverted using `cv2.bitwise_not`. Then, `cv2.dilate` is applied with the selected kernel, expanding the white areas in the image. The process is repeated for the specified number of iterations. Finally, the image is inverted again to restore its original colors.

Erosion details. The image is inverted using `cv2.bitwise_not`, followed by `cv2.erode` with the selected kernel, shrinking the white areas. This operation is also repeated for the given number of iterations.

Table 4. In-distribution (ID) and out-of-distribution (OOD) results (WER %) for HTR models across datasets. The OOD result is reported from the best-performing source. Results marked with $*$ indicate outliers, meaning that the model did not converge in the ID setting. Average results (bottom row) are computed filtering out outliers. † denotes architectures implemented from the papers (no code provided).

Dataset	**CRNN** [64]		**VAN** [20]		**C-SAN**† [26]		**HTR-VT** [47]		**Kang**† [39]		**Michael**† [54]		**LT**† [10]		**VLT**† [11]	
	ID	OOD	ID	OOD	ID	OOD	ID	OOD	ID	OOD	ID	OOD	ID	OOD	ID	OOD
IAM	22.4	68.2	24.2	76.8	50.7	83.6	18.2	83.7	23.2	87.1	20.2	82.9	23.4	72.0	27.0	70.3
Rimes	11.5	74.2	18.2	66.9	45.1	80.7	24.5	78.7	14.8	78.3	20.0	84.5	13.2	77.4	13.3	76.1
G.W.	26.0	68.8	31.2	74.7	33.8	93.9	71.7*	83.3	104.3*	77.6	80.1*	76.1	195.4*	65.7	51.4*	65.8
Bentham	13.2	54.9	20.5	62.1	30.2	82.2	24.6	70.8	19.8	68.2	19.8	69.6	14.3	60.6	15.5	60.5
S.G.	31.1	96.8	33.1	95.3	41.0	96.4	59.5	97.4	203.3*	98.7	134.7*	98.8	45.0	97.9	37.6	97.3
Rodrigo	8.3	92.1	12.4	92.9	22.3	97.2	16.3	92.2	11.1	96.8	15.4	96.2	9.6	94.0	10.7	94.3
ICFHR_{2016}	22.1	104.5	33.9	100.4	64.2	100.7	43.3	103.1	31.7	106.8	33.9	104.8	24.2	110.3	24.6	109.7
Average	**19.2**	**79.9**	24.8	81.3	37.2	90.7	25.4	87.0	20.1	87.6	22.3	87.6	21.6	82.6	25.7	82.0

Table 5. Best source domain for each target (rows) across all architectures studied in the paper.

Dataset	**CRNN** [64]	**VAN** [20]	**C-SAN**† [26]	**HTR-VT** [47]	**Kang**† [39]	**Michael**† [54]	**LT**† [10]	**VLT**† [11]
IAM	Bentham	Rimes	Rimes	Rimes	Rimes	Bentham	Bentham	Bentham
G.W.	IAM	Bentham	IAM	Bentham	IAM	IAM	IAM	IAM
Bentham	IAM	IAM	IAM	IAM	IAM	IAM	IAM	IAM
Rimes	IAM	IAM	IAM	IAM	IAM	IAM	IAM	IAM
S.G.	Rodrigo	IAM	Rodrigo	Rodrigo	IAM	IAM	Rodrigo	Rodrigo
Rodrigo	IAM	IAM	Bentham	IAM	IAM	IAM	IAM	IAM
ICFHR 2016	Bentham	Bentham	Bentham	Bentham	Rimes	IAM	Rimes	Bentham

Afterward, the image is inverted back to its original color scheme.

9. Visual and textual divergences

In this section, we present the specific numerical metrics for visual and textual divergence across the various domains used in the factor analysis. Prior to presenting these results, we first describe the training procedure for the Convolutional Autoencoder (AE) (ϕ_{θ_S}) employed to measure reconstruction error (visual divergence).

9.1. Convolutional Autoencoder

9.1.1. Architecture

Regarding the Autoencoder (AE) used, we employed a rather simple convolutional architecture. The encoder progressively downsamples and compresses the input image into a 512-dimensional latent vector using four 3×3 convolutional layers, each followed by leaky ReLU activation and 2×2 max-pooling. The feature channels increase sequentially from 1 to $16, 32, 64$, and 128, with a fully connected layer producing the final latent representation. The decoder reconstructs the image from the latent vector by reversing the encoder's process. It uses a fully connected layer to reshape the latent vector into a tensor, followed by four transposed convolutional layers that upsample the feature map to the original image size. Feature channels decrease from 128 to $64, 32, 16$, and finally 1, with leaky ReLU activations applied after each layer, except the final layer, which uses a sigmoid activation to normalize output pixel values.

Despite the simplicity of the architecture, the input images are rescaled to dimensions of 64 pixels in height and 1024 pixels in width, result in a model with approximately 33 million parameters. Note that due to the large image size, the pre-flattened vector resulting from the encoder's downsampling has 32,768 dimensions (flattening the final feature map of the encoder with 128 channels, a height of 4, and a width of 64). Using an MLP to reduce this vec-

Table 6. Complete CER results in all datasets using real data.

Method	S/T	IAM	Rimes	G.W.	Bentham	S.G.	Rodrigo	ICFHR$_{2016}$
CRNN [64]	IAM	6.4	25.0	31.1	25.3	45.5	40.9	86.2
	Rimes	35.4	3.7	49.0	50.2	52.3	47.1	87.9
	G.W.	55.6	61.5	8.2	59.2	69.3	66.2	100.0
	Bentham	34.9	45.3	32.2	4.7	57.8	43.8	78.7
	S.G.	77.7	74.5	89.3	78.0	7.2	52.8	100.0
	Rodrigo	65.7	61.4	71.8	66.3	33.6	1.7	85.3
	ICFHR$_{2016}$	74.9	78.4	81.6	75.4	77.9	75.6	5.2
VAN [20]	IAM	6.6	21.3	34.5	26.6	39.8	38.5	82.9
	Rimes	28.6	5.6	46.1	45.0	47.2	43.7	88.4
	G.W.	73.7	67.4	9.3	59.3	67.1	69.6	100.0
	Bentham	37.2	41.7	32.0	7.4	49.4	38.6	75.3
	S.G.	96.1	85.0	93.1	83.1	7.8	57.7	100.0
	Rodrigo	76.5	70.7	78.2	68.1	41.1	2.3	87.3
	ICFHR$_{2016}$	70.8	74.8	76.4	67.8	72.1	71.4	7.5
C-SAN [26]	IAM	28.6	29.8	49.8	38.9	50.2	46.6	90.9
	Rimes	31.5	21.3	50.7	45.9	51.1	45.7	87.0
	G.W.	60.7	60.2	32.0	63.2	68.4	66.8	96.9
	Bentham	54.7	51.3	58.3	26.6	64.8	45.2	83.4
	S.G.	75.3	72.5	81.8	75.5	39.8	50.8	90.2
	Rodrigo	72.7	69.1	78.1	68.2	35.0	38.5	86.9
	ICFHR$_{2016}$	78.4	81.8	86.6	78.7	86.1	81.2	75.3
HTR-VT [47]	IAM	5.8	28.3	40.0	33.3	44.2	38.5	86.1
	Rimes	33.7	7.9	46.2	48.2	51.0	46.1	81.9
	G.W.	70.1	73.3	34.9	76.3	79.2	76.9	85.9
	Bentham	44.4	49.8	38.6	8.4	58.1	45.4	79.6
	S.G.	78.7	78.2	89.8	78.1	17.1	60.1	100.0
	Rodrigo	66.4	64.1	68.7	67.8	36.5	3.9	85.4
	ICFHR$_{2016}$	74.9	77.1	78.8	73.3	76.2	71.9	11.6
Kang [39]	IAM	8.0	32.0	44.0	39.4	51.8	60.6	96.2
	Rimes	42.1	5.7	65.5	63.5	62.1	65.2	92.6
	G.W.	82.8	81.8	78.4	77.7	78.3	77.4	100.0
	Bentham	53.4	59.7	46.6	8.5	80.8	71.1	95.5
	S.G.	100.0	100.0	100.0	100.0	78.7	86.5	100.0
	Rodrigo	81.7	78.6	85.9	78.6	61.1	2.6	95.9
	ICFHR$_{2016}$	74.7	76.5	75.3	74.1	75.9	74.9	7.8
Michael [54]	IAM	7.5	35.5	43.6	43.5	55.3	65.3	85.2
	Rimes	54.5	6.9	63.9	70.2	64.3	66.8	86.3
	G.W.	78.9	80.5	53.8	73.4	79.5	76.7	100.0
	Bentham	49.1	60.0	50.7	8.5	74.5	70.7	91.2
	S.G.	100.0	100.0	100.0	100.0	76.9	79.5	100.0
	Rodrigo	100.0	100.0	87.2	100.0	92.9	3.8	92.4
	ICFHR$_{2016}$	89.3	89.7	81.6	80.4	77.9	77.9	9.5
LT [10]	IAM	7.9	30.8	32.3	33.8	51.4	48.4	98.1
	Rimes	42.9	5.0	52.5	61.9	60.8	56.4	90.5
	G.W.	83.5	85.8	79.6	87.9	75.0	73.3	100.0
	Bentham	42.0	55.4	39.6	6.0	65.4	51.7	92.4
	S.G.	86.6	82.1	95.6	84.9	12.5	65.9	96.7
	Rodrigo	73.1	67.6	80.9	70.8	37.8	2.0	90.7
	ICFHR$_{2016}$	78.0	81.3	81.3	77.9	77.4	76.5	5.9
VLT [11]	IAM	8.9	29.4	32.1	33.3	48.2	47.4	89.7
	Rimes	44.9	5.1	56.4	63.2	62.4	59.1	96.4
	G.W.	69.5	72.2	25.2	69.6	76.3	75.9	100.0
	Bentham	41.3	53.7	43.7	6.1	65.8	53.5	85.1
	S.G.	91.8	83.3	98.4	86.9	9.2	58.9	100.0
	Rodrigo	71.7	67.2	80.2	72.0	38.7	2.2	90.7
	ICFHR$_{2016}$	78.3	80.3	80.8	81.7	79.5	81.8	6.0

Table 7. Complete CER results in all datasets using synthetic data.

Method	S/T	IAM	Rimes	G.W.	Bentham	S.G.	Rodrigo	ICFHR$_{2016}$
CRNN [64]	WIT-en	11.9	22.3	16.9	26.9	25.8	28.1	78.5
	WIT-fr	19.5	17.0	26.3	31.7	26.3	27.2	78.1
	WIT-es	20.0	22.7	27.4	32.9	27.2	22.2	79.4
	WIT-la	20.7	24.6	27.4	34.5	21.4	27.4	79.0
	WIT-de	20.4	25.3	27.3	32.1	28.8	28.3	77.6
VAN [20]	WIT-en	16.9	24.3	25.8	26.1	26.3	28.4	75.0
	WIT-fr	22.4	19.4	31.7	33.1	25.3	25.8	76.9
	WIT-es	23.1	23.5	32.6	34.8	26.0	23.2	77.5
	WIT-la	22.7	24.5	34.1	34.8	23.5	28.2	77.7
	WIT-de	21.9	26.0	30.8	33.8	28.2	29.6	74.3
C-SAN [26]	WIT-en	32.0	38.0	47.8	42.7	35.7	38.8	83.5
	WIT-fr	35.7	35.9	47.4	46.5	36.4	40.4	81.5
	WIT-es	36.2	38.0	46.4	47.4	35.0	37.3	82.7
	WIT-la	36.6	38.8	49.6	48.0	35.8	40.7	83.4
	WIT-de	34.8	38.6	48.9	46.0	36.6	40.5	81.1
HTR-VT [47]	WIT-en	20.7	31.5	26.3	28.3	29.4	32.2	77.5
	WIT-fr	27.5	26.6	34.2	38.1	29.3	32.1	78.1
	WIT-es	28.2	30.3	35.2	38.6	30.1	26.6	77.3
	WIT-la	28.9	33.1	35.4	39.6	27.8	30.6	78.2
	WIT-de	29.2	34.1	36.6	39.2	33.0	34.2	76.7
Kang [39]	WIT-en	28.7	44.5	45.1	51.3	40.0	46.1	85.3
	WIT-fr	38.6	33.1	47.7	57.4	36.2	41.9	87.8
	WIT-es	37.7	49.1	70.3	66.4	42.8	48.7	100.0
	WIT-la	26.2	29.6	39.6	41.2	22.7	35.3	95.3
	WIT-de	32.3	40.1	42.0	43.9	32.4	43.0	84.2
Michael [54]	WIT-en	20.6	34.0	30.5	36.6	35.2	43.4	83.6
	WIT-fr	32.9	25.2	42.8	52.4	36.8	41.1	83.5
	WIT-es	33.8	33.9	45.0	54.0	36.6	33.4	85.2
	WIT-la	37.7	39.5	47.3	58.5	30.7	45.6	82.9
	WIT-de	39.1	44.1	48.4	61.7	40.3	49.7	79.8
LT [10]	WIT-en	13.8	25.1	16.3	21.4	23.4	28.0	80.6
	WIT-fr	22.5	18.9	26.2	35.5	24.1	27.7	79.8
	WIT-es	23.3	25.0	27.1	38.0	22.5	24.8	81.4
	WIT-la	24.0	26.4	27.9	39.3	22.5	28.5	83.0
	WIT-de	23.1	27.6	24.9	36.3	26.0	28.2	78.8
VLT [11]	WIT-en	15.3	26.7	19.0	25.0	23.5	29.1	80.6
	WIT-fr	23.0	19.5	25.9	37.3	23.3	27.9	79.5
	WIT-es	24.9	27.0	28.4	40.6	25.1	24.2	79.8
	WIT-la	26.0	28.3	29.9	42.0	26.6	31.0	82.1
	WIT-de	23.6	25.9	27.0	39.2	24.4	28.2	79.2

Table 8. Zero-shot performance (CER) of VLMs on HTR datasets vs. the best-reported OOD results in the paper (HTR$_{OOD}$ column).

Dataset	LLaVA1.6	Kosmos-2	TrOCR$_M$	InstructBlip	HTR$_{OOD}$
IAM	74.9	80.3	6.8	78.9	28.6
Rimes	93.5	81.5	27.2	80.4	21.3
G.W.	78.6	79.7	17.3	83.3	31.1
S.G.	80.4	82.5	44.1	87.5	25.3
Bentham	85.4	78.4	17.9	76.7	33.6
Rodrigo	76.4	81.2	38.1	86.2	38.5
ICFHR$_{2016}$	95.3	87.2	92.6	88.7	75.3

tor to 512 dimensions significantly increases the parameter count. These two layers (one in the encoder and one in the decoder) account for 99% of the model's parameters.

9.1.2. Training details

We train the AE to minimize the Mean Squared Error (MSE) between the input and reconstructed images. We employ the Adam optimizer with a learning rate of 0.001 for a maximum of 100 epochs. To avoid overfitting, we save the best-performing model according to the validation loss of the same source domain at the end of each epoch.

9.2. Visual divergence

This section presents the results of visual domain divergence, measured by the reconstruction error obtained from the autoencoder described in previous sections. Fig. 8 illustrates the divergence (calculated as the average MSE per image) between each domain pair, with the source represented on the Y-axis and target on the X-axis. Divergences are computed between training and test splits for each pair. To facilitate interpretability, the values are normalized, such that a value of 100 reflects high divergence (darker colors), while a value of 0 denotes indicating low divergence (lighter colors). To validate the visual divergence results in the OOD scenario, Fig. 11 presents images from three pairs of domains with low visual divergence (left) and three domains with high one (right) with their respective scores. Note that the left column features writing styles with very similar stroke densities, while the right column displays styles that differ significantly in both stroke appearance and density. The domain pairs were selected based on the scores presented in Fig. 8, ensuring minimal repetition of domains to better highlight the differences.

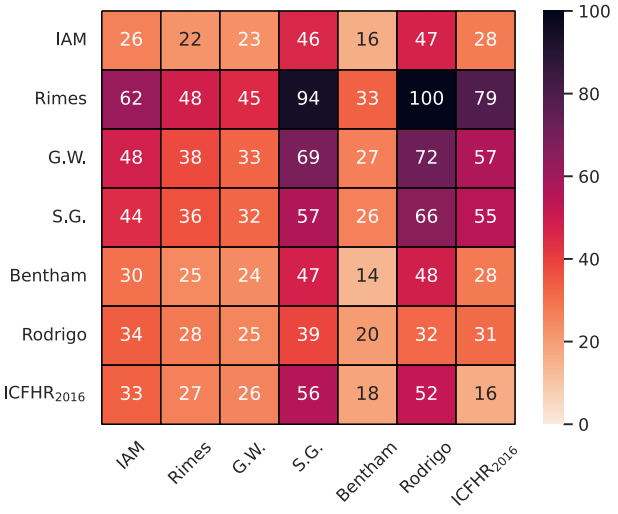

Figure 8. Heatmap of visual divergence between source (rows) and target (columns) from real HTR domains. Divergence values are normalized, with higher scores indicating greater divergence and lower scores reflecting lower divergence.

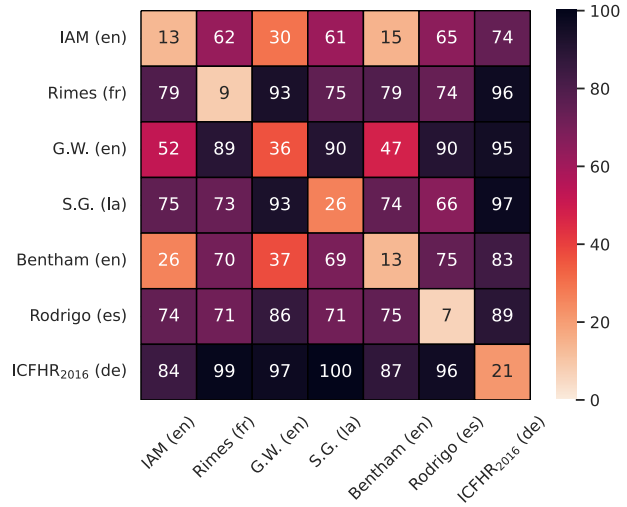

Figure 9. Heatmap of textual divergence from real HTR domains. Rows correspond to source domains, while columns represent target domains. The values are normalized, with 100 indicating maximum divergence and 0 representing minimum divergence.

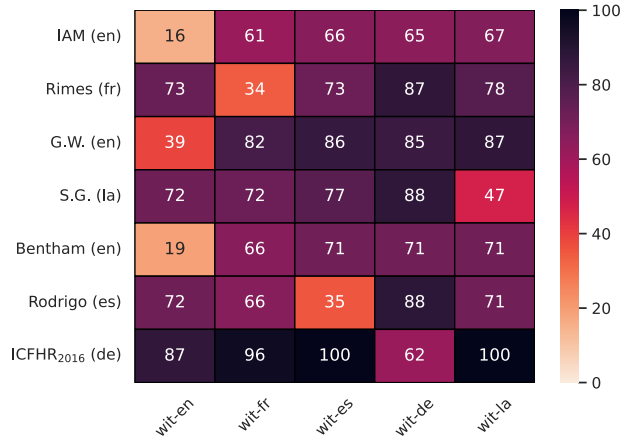

Figure 10. Heatmap of textual divergence between real and synthetic domains. The values are normalized, where 100 represents maximum divergence and 0 represents minimum divergence. Note that each source domain corresponds to a target domain that matches its language, except for English, where three target domains are used: IAM, George Washington, and Bentham.

9.3. Textual divergence

We present the results of textual domain divergences, quantified as the averaged KL-divergence across n-grams as described in the main text. Fig. 9 shows the divergence between textual distributions across domains, with the source represented on the Y-axis and the target the X-axis. Divergences are computed between training and test splits for each pair. Fig. 10 presents the textual divergences be-

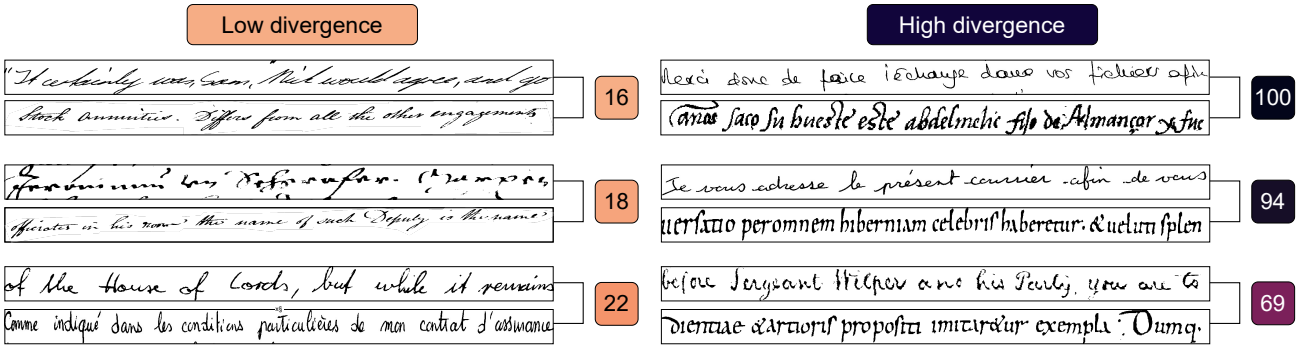

Figure 11. Representation of visual divergence between domains. Examples are presented in two columns: on the left, three domain pairs with low visual divergence, and on the right, three pairs with high visual divergence. For each domain pair, the source domain is shown at the top and the target domain at the bottom. Left (top to bottom): First pair (IAM-Bentham), second pair (ICFHR$_{2016}$-Bentham), third pair (IAM-Rimes). Right (top to bottom): First pair (Rimes-Rodrigo), second pair (Rimes-S.G.), third pair (G.W.-S.G.). The divergence percentage (see Fig. 8) is displayed for each pair.

tween real source domains (Y-axis) and synthetic domains (X-axis) for each language. In this case, the divergence is calculated between the training split of the source domain and the training split of the synthetic data, as these are used to compute the n-grams and train the models in the synthetic experiments. Both figures display normalized values, where a value of 100 indicates maximum divergence (darker colors) and 0 minimum divergence between texts.

10. Factor analysis

The selection of the number of factors is a crucial criterion for analyzing the outcomes of the factor analysis. Note that the first k factors span the subspace defined by the first k eigenvectors of the data matrix. To determine the number of factors (n), the simplest rule of thumb involves retaining all eigenvectors with eigenvalues ≥ 1. This can be simply visualized by plotting the eigenvalues in descending order using a scree plot, as shown in Fig. 12. Based on this analysis, we decided to retain four factors as stated in the main text of the paper.

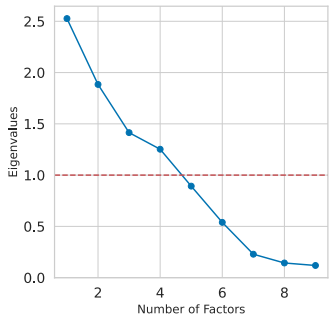

Figure 12. Scree plot: Eigenvalues of the standardized values used for factor analysis, ordered in descending magnitude. We chose to retain 4 factors, as these are the ones with eigenvalues ≥ 1.