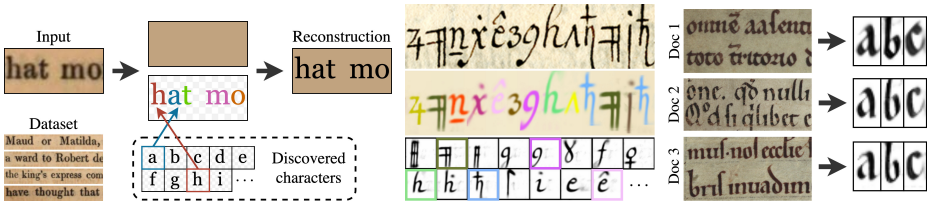
The Learnable Typewriter: A Generative Approach to Text Analysis

Ioannis Siglidis[0009−0002−2278−5825], Nicolas Gonthier[0000−0002−9236−5394], Julien Gaubil[0009−0006−2577−0847], Tom Monnier[0009−0008−1937−6506], and Mathieu Aubry[0000−0002−3804−0193]

LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France
ioannis.siglidis@enpc.fr



(a) The Learnable Typewriter idea (b) Cipher analysis [27] (c) Paleography [6]

Fig. 1: **The Learnable Typewriter. (a)** Given a text line dataset, we learn to reconstruct images by discovering the underlying characters. This generative approach can be used both **(b)** to analyze complex ciphers and **(c)** as an automatic tool for the study of handwriting in historical documents.

Abstract. We present a generative document-specific approach to character analysis and recognition in text lines. Our main idea is to build on unsupervised multi-object segmentation methods and in particular those that reconstruct images based on a limited amount of visual elements, called sprites. Taking as input a set of text lines with similar font or handwriting, our approach can learn a large number of different characters and leverage line-level annotations when available. Our contribution is twofold. First, we provide the first adaptation and evaluation of a deep unsupervised multi-object segmentation approach for text line analysis. Since these methods have mainly been evaluated on synthetic data in a completely unsupervised setting, demonstrating that they can be adapted and quantitatively evaluated on real images of text and that they can be trained using weak supervision are significant progresses. Second, we show the potential of our method for new applications, more specifically in the field of palaeography, which studies the history and variations of handwriting, and for cipher analysis. We demonstrate our approach on four very different datasets: a printed volume of the Google1000 dataset [48,19], the Copiale cipher [2,27], a large scale multi-font benchmark [41], and historical handwritten charters from the 12th and early 13th century [6].

Keywords: Document Analysis · Palaeography · Analysis by Synthesis

1 Introduction

A popular approach to document analysis in the 1990s was to learn document-specific character prototypes, which enabled Optical Character Recognition (OCR) [28,29,49,1] but also other applications, such as font classification [21] or document image compression and rendering [38]. This idea culminated in 2013, with the Ocular system [3] which proposed a generative model for printed text lines inspired by the printing process and held the promise of achieving a complete explanation of their appearance. These document-specific generative approaches were however overshadowed by discriminative approaches, whose sole purpose is to perform predictions, and which lead to higher performance at the cost of interpretability, e.g. [16,33]. In this paper, we explore how modern deep approaches enable revisiting and extending model-based approaches to text line analysis. In particular, we demonstrate an approach that can deal with challenging examples of handwritten documents, opening a new perspective for the study of historical handwriting, that of palaeography.

While discriminative approaches are largely dominant in today's deep learning-based computer vision, a recent set of works revisited generative approaches for unsupervised multi-object object segmentation [5,9,18,17,50,7,8,10,23,42,37]. Most of them provide results on synthetic data or simple real images [37], and sometimes show qualitative results on simple printed text images [42,40]. Surprisingly, images of handwritten characters, which were notoriously used in the development of convolutional neural networks [31,32] and generative adversarial networks [13] were largely overlooked by these approaches.

We build on recent sprite-based unsupervised image decomposition approaches [42,37] that provide an interpretable decomposition of images into a dictionary of visual elements, called sprites. These methods jointly optimize both the sprites and the neural networks that predict their position and color. Intuitively, we would like to adapt these methods so that, from text lines that are extracted from any given document, they could learn sprites that correspond to each character. By adapting MarioNette [42] to perform text line analysis, we provide quantitative evaluation on real data and an analysis of the limitations of a state-of-the-art unsupervised multi-object segmentation approach. We argue that text line recognition should be used as a benchmark for this task in future work.

Because unsupervised performances are not completely satisfactory, we combine this approach with a weak supervision from line-level transcriptions. Transcriptions are widely available and easy to produce with dedicated software, e.g. [24], and we show that this dramatically improves the results, while preserving their interpretability. We believe that similar weak (i.e., image-level) annotations could also be considered for other image decomposition problems.

Contributions. To summarize, we present:

- a deep generative approach to text line analysis, inspired by deep unsupervised multi-object segmentation approaches and adapted to work in both an unsupervised and a weakly supervised setting,

- a demonstration of the potential of our approach in challenging applications, particularly in ciphered documents and for palaeographic analysis,
- experiments on three very different types of datasets: the printed volume of Google1000 [48,19] and the MFGR [41] dataset, the Copiale cipher [2,27], and historical handwritten charters from the 12th and early 13th century [6].

Our implementation can be found at `github.com/ysig/learnable-typewriter`.

2 Related Work

Text analysis. Image Text Recognition, including Optical Character Recognition (OCR) and Handwritten Text Recognition (HTR), is a classic pattern recognition problem and one of the earliest successful applications of deep learning [31,32]. The mainstream approaches for text line recognition rely on discriminative supervised learning. Typically, a Convolutional Neural Network (CNN) encoder will map the input image to a sequence of features, and a decoder will associate them to the ground truth, e.g., through a recurrent architecture trained with a Connectionist Temporal Classification (CTC) loss [15,16,39,4,44], or a transformer trained with cross entropy [25,33].

More related to our work, ScrabbleGAN [11] proposed a generative adversarial approach for semi-supervised text recognition, but their method is neither able to reconstruct an input text line nor to decompose it into individual characters. Also related are approaches that use already annotated sprites (referred to as exemplars or supports) to perform OCR/HTR [52,43] by matching them to text lines. Recent unsupervised approaches, either cluster input images in a feature space [2] or rely on an existing text corpus of the recognized language [19].

Closest to our work are classical prototype-based methods [28,29,49,1] and in particular the Ocular system [3] which follows a generative probabilistic approach to jointly model text and character fonts in binarized documents and is optimized through Expectation Maximization (EM). Unlike us, it also relies on a pre-trained n-gram language model, originally from the English language and later extended to multiple languages [12]. Other approaches rely on language models to identify characters [30,3,19]. However, language models do not exist for unknown ciphers or historical manuscripts. Instead, we propose to disambiguate sprites by relying on line-level transcriptions.

Most related to our application in palaeography, [14] proposes a probabilistic model for printed font analysis and [45] for linear scribal hands of linear b. However, both works rely on single isolated and binarized characters as input, whereas the goal of our approach is to be directly applicable to colored text lines.

Unsupervised multi-object segmentation. Unsupervised multi-object segmentation refers to a family of approaches that decompose and segment scenes into multiple objects in an unsupervised manner [26]. Some techniques perform decomposition by computing pixel-level segmentation masks over the whole input image [5,9,18,17,50], while others focus on smaller regions of the input and learn to compose objects in an iterative fashion, mostly relying on a recurrent

architecture [7,8,10,23]. All of these techniques can isolate objects by producing segmentation masks, but our goal is also to capture recurring visual elements.

We thus build on techniques that explicitly model the objects located inside the input image, by associating them to a set of image prototypes referred to as sprites [37,42]. Sprites are color images with an additional transparency channel and are associated to transformation prediction networks that are used to compose them onto a target canvas. However, DTI-Sprites [37] can only predict a small number of sprites for a collection of fixed-size images and fails to scale when the number of objects within each image scales to those of real documents. At the same time, MarioNette [42] suffers from a high reconstruction error and fuzzy sprites that sub-optimally reconstruct a toy text dataset.

3 The Learnable Typewriter

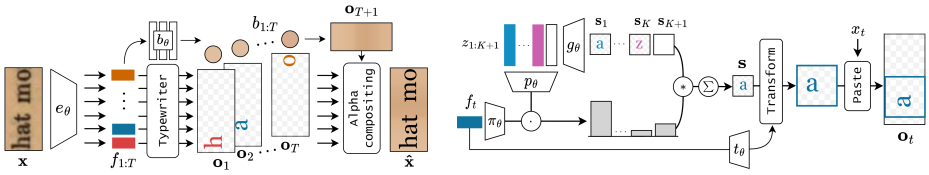
Given a collection of text lines written using consistent font or handwriting, our goal is to learn the shape of all the characters it contains and a deep network that predicts the exact way these characters were used to generate any input text line. Since complete supervision (i.e., the position and shape of every character used in a document) for such a task would be extremely costly to obtain, we propose to proceed in an analysis-by-synthesis fashion and to build on sprite-based unsupervised image decomposition approaches [42,37] which jointly learn a set of character images - called *sprites* - and a network that transforms and positions them on a canvas in order to reconstruct input lines. Due to the potential ambiguity in the definition of sprites, we introduce a complementary weak-supervision from line-level transcriptions.

In this section, we first present an overview of our image model and approach (Section 3.1). Then, we describe the deep architecture we use (Section 3.2). Finally, we discuss our loss and training procedure (Section 3.3).

Notations. We write $a_{1:n}$ the sequence $\{a_1, \ldots, a_n\}$, and use bold letters $\mathbf{a}$ for images. An RGBA image $\mathbf{a}$ corresponds to an RGB image denoted by $\mathbf{a}^c$, alongside an alpha-transparency channel denoted by $\mathbf{a}^\alpha$. We use θ as a generic notation for network parameters and thus any character indexed by θ, e.g., a_θ, is a network.

3.1 Overview and image model

Figure 2a presents an overview of our pipeline. An input image $\mathbf{x}$ of size $H \times W$ is fed to an encoder network e_θ generating a sequence of T features $f_{1:T}$ associated to uniformly-spaced locations $x_{1:T}$ in the image. Each feature f_t is processed independently by our *Typewriter* module (Section 3.2) which outputs an RGBA image $\mathbf{o}_t$ corresponding to a character. The images $\mathbf{o}_{1:T}$ are then composited with a canvas image we call $\mathbf{o}_{T+1}$. This canvas image $\mathbf{o}_{T+1}$ is a completely opaque image (zero transparency). Its colors are predicted by a Multi-Layer Perceptron (MLP) b_θ which takes as input the features $f_{1:T}$ and outputs RGB values $b_{1:T}$. All



(a) Overview of our full pipeline (b) Details of our Typewriter module

Fig. 2: **Overview. (a)** An image is encoded into a sequence of features, each decoded by the Typewriter module into image layers. They are then fused by alpha compositing with a predicted uniform background. **(b)** The Typewriter module takes a feature as input, computes sprites and associated probabilities from learned latent codes, and composes them into a composite sprite that is transformed and positioned onto an image-sized canvas.

resulting images $\mathbf{o}_{1:T+1}$ can be seen as ordered image layers and are merged using alpha compositing, as proposed by both [37,42]. More formally, the reconstructed image $\hat{\mathbf{x}}$ can be written:

$$\hat{\mathbf{x}} = \sum_{t=1}^{T+1} \Big[\prod_{j<t} (1 - \mathbf{o}_j^\alpha) \Big] \mathbf{o}_t^\alpha \mathbf{o}_t^c. \tag{1}$$

During training, the order of $\mathbf{o}_{1:T}$ in the compositing operation is randomized to reduce overfitting, as advocated by the MarioNette approach [42]. The full system is differentiable and can be trained end-to-end.

3.2 Typewriter Module

We now describe in detail the Typewriter module, which takes as input a feature f from the encoder and its position x, and outputs an image layer $\mathbf{o}$, to be composited. An overview of the module is presented in Figure 2b. On a high level, it is similar to the MarioNette architecture [42], but handles blanks (i.e., the generation of a completely transparent image) differently and has a more flexible deformation model, similar to the one used in DTI-Sprites [37]. More specifically, the module learns jointly RGBA images called *sprites* corresponding to character images, and networks that use the features f to predict a probability for each sprite and a transformation of the sprite. In the following, we detail how we obtain the following three elements: the set of K parameterized sprites, the sprites compositing, and the transformation model.

Sprite parametrization. We model characters as a set of K sprites which are defined using a generator network. More specifically, we learn K latent codes $z_{1:K}$ which are used as an input to a generator network g_θ in order to generate sprites $\mathbf{s}_{1:K} = g_\theta(z_{1:K})$. These sprites are images with a single channel that corresponds to their opacity. Similar to DTI-Sprites [37], we model a variable number of

sprites with an empty (i.e., completely transparent) sprite which we write $\mathbf{s}_{K+1}$. Instead of directly learning sprites in the pixel space as in DTI-Sprites [37], we found that using a generator network yields faster and better convergence.

Sprite probabilities and compositing. To predict a probability p_k for each sprite $\mathbf{s}_k$, each latent code z_k is associated through a network p_θ to a probability feature $z^p_k = p_\theta(z_k)$ of the same dimension D as the encoder features ($D = 64$ in our experiments). We additionally optimize directly a probability feature z^p_{K+1} which we associate to the empty sprite. Given a feature f predicted by the encoder, we predict the probability p_k of each sprite $\mathbf{s}_k$ by computing the dot product between the probability features $z^p_{1:K+1}$ and a learned projection of the feature $\pi_\theta(f)$, and applying a softmax to the result:

$$p_{1:K+1}(f) = \text{softmax}\left(\lambda z^p_{1:K+1} \cdot \pi_\theta(f)^T\right), \tag{2}$$

where $\cdot$ is the dot product applied to each element of the sequence, $\lambda = 1/\sqrt{D}$ is a scalar temperature hyper-parameter, and the softmax is applied to the resulting vector. We use these probabilities to combine the sprites into the weighted average $\mathbf{s} = \sum_{k=1}^{K} p_k g_\theta(z_k)$. During inference, we simply select the sprite $g_\theta(z_k)$ with the highest probability p_k instead of computing a weighted average. Note that this compositing can be interpreted as attention operation [47]:

$$\mathbf{s} = \text{attention}(\bar{Q}, \bar{K}, \bar{V}) = \text{softmax}\left(\frac{\bar{Q}\bar{K}^T}{\sqrt{D}}\right)\bar{V}, \tag{3}$$

with $\bar{Q} = \pi_\theta(f)$, $\bar{K} = p_\theta(z_{1:K+1})$, $\bar{V} = g_\theta(z_{1:K+1})$, D the dimension of the features, and by convention $g_\theta(z_{K+1})$ is the empty sprite and $p_\theta(z_{K+1}) = z^p_{K+1}$. In fact, we show that directly optimizing $z^p_{1:K}$ instead of learning to predict the probability features $z^p_{1:K}$ from the sprite latent codes $z_{1:K}$, as in MarioNette [42], yields similar results. Note that we learn a probability code z^p_{K+1} to compute the probability of empty sprites instead of having a separate mechanism as in MarioNette [42] because it is critical for our supervised loss (see Section 3.3).

Positioning and coloring. The final step of our module is to position the selected sprite in a canvas of size $H \times W$ and to adapt its color. We implement this operation as a sequence of a spatial transformer [22] and a color transformation, similar to DTI-Sprites [37]. More specifically, the feature f is given as input to a network t_θ that predicts three parameters for the color of the sprite and three parameters for isotropic scaling and 2D-translation that are used by a spatial transformer [22] to deform $\mathbf{s}$. Finally, using the location x associated with the feature f, we paste the deformed colored sprite onto a background canvas of size $H \times W$ at position x to obtain a reconstructed RGBA image layer $\mathbf{o}$. Positioning the sprites with respect to the position of the associated local features helps us obtain results co-variant to translations of the text lines and independent of the line size. To produce the background canvas, each of the features $f_{1:T}$

is first passed through a shared MLP b_θ, to produce a vector of T background colors $b_{1:T}$. We then use bi-linear interpolation to upscale this vector to the full size of the input image x. Specific details concerning the parametrization of the transformation networks can be found in the supplementary material.

3.3 Losses and training details

Our system is designed in an analysis-by-synthesis spirit and thus relies mainly on a reconstruction loss. This reconstruction loss can be complemented by a loss on the selected sprites in the supervised setting where each text line is paired with a transcription. In the following, we define these losses for a single text line image and its transcription, using the notations of the previous section.

Reconstruction loss. Our core loss is a simple mean square error between the input image $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$ predicted by our system, as in Section 3.1:

$$\mathcal{L}_{\text{rec}}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2. \tag{4}$$

In the unsupervised setting, we don't train with any additional regularization.

Weakly supervised loss. The intrinsic ambiguity of the sprite decomposition problem may result in sprites that do not correspond to individual characters.

Using line-level annotation is an easy way to resolve this ambiguity. We find that simply adding the classical CTC loss [15] computed on the sprite probabilities to our reconstruction loss is enough to learn sprites that exactly correspond to characters. More specifically, we chose the number of sprites as the number of different characters and associate arbitrarily each sprite with a character and the empty sprite with the blank token of the CTC. Then given the one-hot line-level annotation y and $\hat{y} = (p_{1:K+1}(f_1), ..., p_{1:K+1}(f_T))$ the predicted sprite probabilities, we optimize our system's parameters by minimizing:

$$\mathcal{L}_{\text{sup}}(\mathbf{x}, y, \hat{\mathbf{x}}, \hat{y}) = \mathcal{L}_{\text{rec}}(\mathbf{x}, \hat{\mathbf{x}}) + \lambda_{\text{ctc}}\mathcal{L}_{\text{ctc}}(y, \hat{y}) \tag{5}$$

where λ_{ctc} is a hyper-parameter and $\mathcal{L}_{\text{ctc}}(y, \hat{y})$ is the CTC loss computed between the ground-truth y and the predicted probabilities $\hat{y}$. We use $\lambda_{\text{ctc}} = 0.01$ in all our experiments, increased only for the less challenging Google1000 to $\lambda_{\text{ctc}} = 0.1$.

Implementation and training details. We train on the Google1000 [48] and Fontenay [6] datasets with lines of height $H = 64$ and on the Copiale dataset [27] with $H = 96$. The generated sprites $\mathbf{s}_{1:K}$ are of size $H/2 \times H/2$. In the supervised setting, we use as many sprites as there are characters, and in the unsupervised, we set $K = 60$ for Google1000 and $K = 120$ for the Copiale cipher. In the supervised case we train for 100 epochs on Google1000 and for 500 epochs on Copiale with a batch size of 16, and we select the model that performs best on the validation set for evaluation. In the unsupervised setting, we use line crops

of width $W = 2H$ and train for 1000 epochs on Google1000 and for 5000 on the Copiale cipher with a batch size of 32 and use the final model. The number of epochs is much higher in the unsupervised case than in the supervised case because the network sees only a small crop of each line at each epoch, but each epoch is much faster to perform. To always avoid learning sprites that reconstruct the background instead of the actual characters, we warm start the training process by only training the background MLP for 3000 gradient steps.

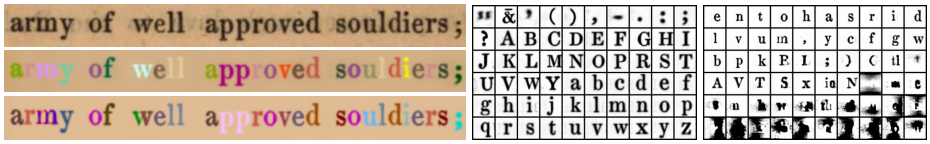
Our encoder network is a ResNet-32-CIFAR10 [20], that is truncated after layer 3 with a Gaussian feature pooling described in supplementary material. For our unsupervised experiments, we use as generator g_θ the U-Net architecture of Deformable Sprites [51] which converged quickly, and for our supervised experiments a 2-layer MLP similar to MarioNette [42] which produces sprites of higher quality. The networks π_θ and p_θ are single linear layers followed by layer-normalization. We use the AdamW [34] optimizer with a learning rate of 10^{-4} and apply a weight-decay of 10^{-6} to the encoder parameters.

4 Experiments

4.1 Datasets and metrics

Datasets. We experiment with four datasets with different characteristics: Google1000 [48], MFGR [41], the Copiale cipher [27] and Fontenay manuscripts [6]:

– *Google1000.* The Google1000 dataset contains scanned historical printed books, arranged into Volumes [48]. We use the English Volume 0002 which we process with the preprocessing code of [19], using 317 out of 374 pages and train-val-test set with 5097-567-630 lines respectively. This leads to a total number of 83 distinct annotated characters. Although supervised printed font recognition is largely considered a solved problem, and the annotation for this dataset is actually the result of OCR, this document is still challenging for an analysis-by-synthesis approach, containing artifacts such as ink bleed, age degradation, as well as variance in illumination and geometric deformations due to digitization.
– *MFGR.* The ICDAR-2024 "Multi Font Group Recognition and OCR challenge" dataset [41], contains text lines that were printed with a set of 8 distinct typefaces: *antiqua, bastarda, fraktur, gotico-antiqua, italic, rotunda, schwabacher, textura.* We focus only on lines that contain fonts from a single group. This dataset is similar to Google1000, but comes with the challenges of older prints, such as non-fully printed letters and allographs. With 12K-45K training lines for each of the 8 different typefaces, it serves as an ideal test-bed to assess the robustness of our model in learning meaningful sprites across a variety of historical prints.
– *Copiale cipher.* The Copiale cipher is an oculist German text dating back to an 18th-century secret society [27]. Opposite to Baro et al. [2] which uses a binarized version of the dataset, we train our model on the original text-line images, which we segmented using docExtractor [35] and manually assigned to their annotations, respecting the train-val-test split of Baro et al. [2] with 711-156-908 lines each. The total number of distinct annotated characters is 112. This dataset is more



(a) Semantic Segmentation (↑ sup. ↓ unsup.) (b) Sprites (sup.) (c) Sprites (unsup.)

Fig. 3: **Results on a printed document (Google1000).** In both the supervised and unsupervised setting our method produces meaningful sprites and accurate reconstructions **(a)**. We show the 60 most used sprites in alphabetic ordering in the supervised setting **(b)** and ordered by frequency in the unsupervised one **(c)**. See text for details and the supplementary material for more reconstructions.

challenging than printed text because of the handwritten morphological variance of a historical manuscript, and its large number of characters.

– *Fontenay manuscripts.* The Fontenay dataset contains digitized charters that originate from the Cistercian abbey of Fontenay in Burgundy (France) [6] and were created during the 12th and early 13th centuries. Each document has been digitized and each line has been manually segmented and transcribed. For our experiments, we selected a subset of 14 different documents sharing a similar script which falls into the family of praegothica scripts. These correspond to 163 lines, using 47 distinct characters. While they were carefully written and preserved, these documents are still very challenging (Figure 6). They exhibit degradation, clear intra-document letter shape variations, and letters can overlap or be joined by ligature marks. Moreover, each document represents only a small amount of data, e.g., the ones used in Figure 6 contain between 8 and 25 lines.

Metrics. Our goal is to capture the shape of all characters and position them precisely in each text line. Such fine-grained annotation is however not available in existing datasets. Instead, to provide a quantitative evaluation of our models, we report mean squared reconstruction error ("Rec." in our tables) and Character Error Rate (CER). CER is the standard metric for Optical Character Recognition (OCR). Given ground-truth and predicted sequences of characters, σ and $\hat{\sigma}$, it is defined as the minimum number of substitutions S, deletions D, and insertions I of characters needed to match the predicted sequence $\hat{\sigma}$ to the ground truth sequence σ, normalized by the size of the ground truth sequence $|\sigma|$:

$$CER(\sigma, \hat{\sigma}) = \frac{S + D + I}{|\sigma|}. \tag{6}$$

For simplicity, we ignore spaces. Predictions are obtained by selecting at every position the character associated to the most likely sprite. In the supervised setting, the association between sprites and characters is fixed at the beginning of training. In the unsupervised setting, we associate every sprite to a single character using a simple assignment strategy described in supplementary material.

(a) ↑ sup. ↓ unsup. (b) Sprites (sup.) (c) Sprites (unsup.)

Fig. 4: **Results on the Copiale cipher [27].** Despite the high number of characters and their variability, our method learns meaningful sprites and performs accurate reconstructions in both settings **(a)**. We show the 108 most used sprites sorted by frequency in the supervised **(b)** and the unsupervised **(c)** settings.

4.2 Qualitative results

Examples of semantic segmentation and sprites in the supervised and unsupervised setting on Google1000 and Copiale are shown in Figures 3,4 respectively. In the **unsupervised** setting, several sprites (Figures 3c,4c) can be used to reconstruct a single character. For example in Google1000, the "n" and "m" sprites are joined to better reconstruct "m". To account for appearance variation, multiple sprites are learned to reconstruct the most frequent character, e.g. "e" for Google and "c" in Copiale. These effects are even stronger in the handwritten Copiale dataset, where generic sub-character strokes are learned and used to better model characters' variations. In both datasets, some of the least used sprites do not correspond to characters, as they are never selected by the network, and thus are not properly optimized. These behaviors are expected in a completely unsupervised setting, because of the highly unbalanced statistics of the character frequencies and the ambiguity of reconstruction: without additional supervision, there is a clear benefit for the network to model well the variations of common characters, and to approximate or discard rare ones. This is a core limitation of existing unsupervised image decomposition approaches and a motivation for the introduction of our weakly supervised setting. In the (weakly) **supervised** setting, the sprites (Figure 3b, 4b) closely correspond to the characters, except for very rare characters like the capital letter 'Z' for Google1000 (as can be seen in supplementary material), while reconstruction is of very high quality and each character is reconstructed with the expected sprite.

Historical fonts reconstruction In Figure 5 we compare our learned sprites to manually extracted and binarized exemplars, where we observe that the learned sprites are mostly similar to the exemplars. Typefaces are sorted according to the average similarity between all the learned sprites and the manually extracted exemplars (between a-z and A-Z) that is computed using SSIM, as is a standard practice for font comparison [46]. SSIM is the highest for the antiqua font (0.745) and the lowest for the gotico-antiqua font (0.676). This may be related to the number of allographs that are present in each typeface. Antiqua is simple and standard, whereas gotico-antiqua is a hybrid between two visually distinct fonts, hence as our model learns a single sprite it fails to summarize them (e.g., see "T", "G", "I", "E", "F"). We believe that these results showcase the robustness of our approach, which is crucial for it to be applied in historical print analysis.

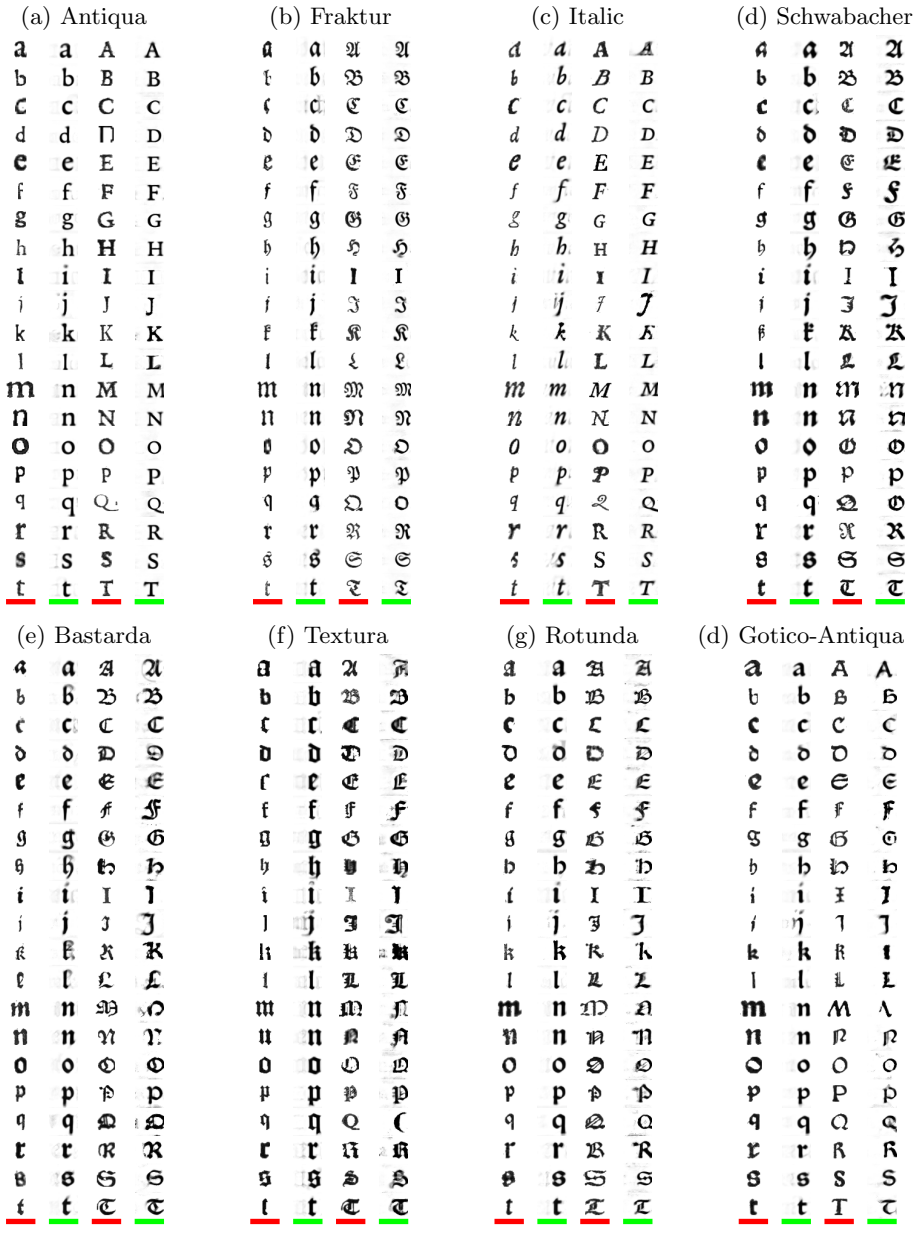


Fig. 5: **Qualitative Evaluation on MFGR.** We compare a-t, A-T between learned sprites (ours), and manualy extracted exemplars. Fonts are sorted by descending SSIM, computed on post-processed sprites (see details in supplementary material). Note, that although fonts come from a single family, they may present *allographs*. For example, *Italic* contains different variants of "Q", where a random exemplar can significantly differ from the one summarized using our method.

Table 1: **Quantitative results and ablation on Google1000 [48].** We report CER and mean squared reconstruction error for all the different approaches. For our method, we report the average of 5 runs and their standard deviation.

Method	Type	Rec. $\times 10^3$	CER
DTI-Sprites [37]	unsup.	2.54	18.4 %
FontAdaptor [52]	1-shot	-	6.7 %
ScrabbleGAN [11]	sup.	-	0.6 %
Learnable Typewriter	sup.	3.5 ± 0.1	$0.85 \pm 0.03\%$
w\o shared z_k	sup.	3.3 ± 0.1	$0.89 \pm 0.06\%$
w\o p_θ	sup.	3.5 ± 0.1	$0.99 \pm 0.05\%$
w\o g_θ	sup.	3.4 ± 0.1	$0.88 \pm 0.04\%$
Learnable Typewriter	unsup.	7.1 ± 0.4	$7.7 \pm 0.6\%$
w\o shared z_k	unsup.	7.4 ± 0.4	$8.0 \pm 0.2\%$
w\o p_θ	unsup.	7.0 ± 0.3	$7.7 \pm 2.0\%$
w\o g_θ	unsup.	10.5 ± 0.7	$27.0 \pm 2.2\%$

4.3 Quantitative results

Our quantitative results and ablations for Google1000 and Copiale are reported in Tables 1,2 respectively.

For Google1000, the CER in the supervised setting is close to perfect, while it is 7.7% for the unsupervised setting. To provide baselines for these performances, we train and evaluate on our version of the dataset (i) ScrabbleGAN [11], a supervised method with a standard recognizer and an additional generator module, (ii) FontAdaptor [52], a recent 1-shot method that learns to match single character exemplars to text lines, and (iii) an adaptation of the unsupervised DTI-Sprites [36] to text lines which we detail in the supplementary material (where we also show that vanilla MarioNette [42] has significantly worse results). Our unsupervised approach performs clearly better than our adaptation of DTI-Sprites and is almost on par with the 1-shot FontAdaptor, while our weakly supervised approach is almost on par with ScrabbleGAN. Our adaptation of DTI-Sprites is better at reconstructing images, but the learned sprites are much less meaningful, as shown by the poor CER performance. Interestingly, reconstruction is much better when using supervision, which hints that a better optimization scheme could improve unsupervised performances. We also evaluate the effect of varying the number of sprites K in the unsupervised setting. For K smaller than the actual number of characters (83), namely $K = 21$ and $K = 41$, we have a significant performance drop of 10% and 26% CER respectively, while increasing the number of characters to 166 and 332 doesn't significantly boost performances.

On the Copiale dataset, we compare our results with HTRbyMatching [43], a few-shot approach developed specifically for cipher recognition, using the same train/val/test splits. HTRbyMatching was evaluated on a wide range of few-shot scenarios, ranging from a scenario similar to FontAdaptor where a single

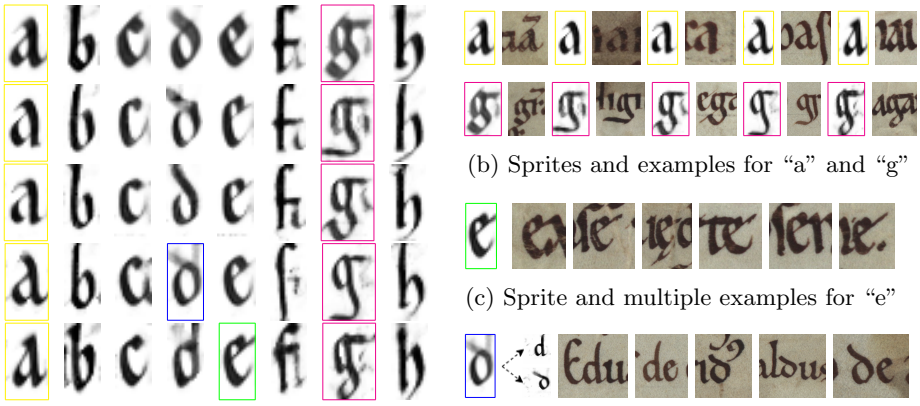
Table 2: **Quantitative results on Copiale [27].** We report CER and reconstruction error for a baselines and our method. For our method, we report average over 5 runs and standard deviation. *See text for details.

Method	Type	Rec. $\times 10^2$	CER
HTRbyMatching [43]	few-shot	-	$10-47\%^*$
Learnable Typewriter	sup.	1.81 ± 0.01	$4.2 \pm 0.3\%$
w\o shared z_k	sup.	1.79 ± 0.01	$4.0 \pm 0.1\%$
w\o p_θ	sup.	1.77 ± 0.02	$4.7 \pm 0.1\%$
w\o g_θ	sup.	1.96 ± 0.07	$4.2 \pm 0.2\%$
Learnable Typewriter	unsup.	1.93 ± 0.02	$52.6 \pm 1.7\%$
w\o shared z_k	unsup.	1.89 ± 0.02	$47.6 \pm 2.8\%$
w\o p_θ	unsup.	1.81 ± 0.06	$51.9 \pm 2.0\%$
w\o g_θ	unsup.	3.99 ± 0.14	$80.6 \pm 0.9\%$

exemplar is available for every character, to one where 5 exemplars are available for each character together with 5 completely annotated pages. Reported results are only for confident character predictions with different confidence thresholds, but summing the error rate of the predicted symbols and the percentage of non-annotated symbols, one can estimate the CER to vary between 10% and 47% depending on the scenario. This is consistent with the quantitative results we obtain with our approach, which are much better in the supervised setting (4.2%) and worse in the completely unsupervised one (52.6%). The low performance of the unsupervised approach is consistent with the qualitative results: given that many characters are reconstructed by sub-character sprites, one would have to associate sprite bi-grams to characters in order to obtain good CER performances. Interestingly, the reconstruction error is similar in the supervised and unsupervised setting, hinting that for this specific dataset, optimizing the reconstruction quality might not be enough to obtain relevant decomposition without additional priors. These results enable us to quantify and analyze a limitation of unsupervised image decomposition approaches on a more challenging dataset.

Note that the goal of our approach is not to boost CER performances - which in any case would be futile for Google1000 where the ground truth is already the result of an OCR model - but instead to learn character models and image decomposition. All these comparisons should be thus considered as sanity checks. Yet, it is possible to design post-processing algorithms to improve CER. We tested a simple algorithm where we assign a new sprite to the most frequent bi-grams and tri-grams, which leads to an improved CER for Copiale of 29.9%. However, we think it is more interesting to see this metric as a tool for evaluating the raw output of unsupervised image decomposition models.

In particular, we performed on both datasets an ablation of the architecture to better understand which design choices are critical. Interestingly, our results show that both in the supervised and the unsupervised setting, not sharing the



(a) Sprites from document finetuning

(b) Sprites and examples for “a” and “g”

(c) Sprite and multiple examples for “e”

(d) Learning dual sprites for “d” allographs

Fig. 6: **Sprites learned for similar documents of a Praegothica script.** On the left each line corresponds to a different document. Looking at any column, one can notice the small differences that characterise the handwriting in each document. Colored boxed correspond to cases analysed in more detail on the right: sprites summarize the key attributes of a character in each specific document, averaging its variations. Note the complexity of the documents: characters can overlap or be connected ligature, the parchment is often stained, and there are important intra-document character variations.

latent codes z_k between the generation network and the sprite selection and even completely removing the probability network p_θ has limited influence on the performance, and these design choices of MarioNette [42] are not critical. Conversely, removing g_θ and directly learning prototypes as network parameters similar to DTI-Sprites [37] has little impact in the supervised case, but leads to a significant drop in performance in the unsupervised one. A more detailed analysis of training curves reveals that in the unsupervised case, training is slower leads to overfitting. While it might be possible to fix this issue by adapting the learning scheme for the prototypes, this shows that it is easier to learn the prototypes through a generator network than to optimize them directly.

4.4 Application to palaeography

To test our approach in a more challenging case and demonstrate its potential for palaeographic analysis, we applied it to a collection of 14 historical charters from the Fontenay abbey [6]. While they all use similar scripts from the Praegothica type, they also exhibit clear variations. One of the goals of a palaeographic analysis would be to identify and characterize these variations. We focus on the variations in the shape of letters, their *morphology*, which is quite challenging to describe with natural language. One solution would be to choose a specific example for each letter in each document or to ask a palaeographer to manually

draw a 'typical' one. However, this is very time-consuming and might reflect the priors or biases of the palaeographer in addition to the actual variations. Instead, we propose to fine-tune our Learnable Typewriter approach on each document and visualize the sprites associated with each character and each document. Due to the difficulty of the dataset, we focus on the results of our supervised setting.

Figure 6a visualizes the sprites obtained for five different documents from the characters "a" to "h" and Figure 6 highlights different aspects of the results. Figure 6b emphasizes the fact that the differences in the learned sprites correspond to actual variations in the different documents, whether subtle, such as for the "a" sprite, or clearer, such as for the descending part of the "g" sprite. Figure 6c shows how a sharp sprite can be learned for the character "e", summarizing accurately its shape despite variations in the different occurrences. Finally, Figure 6d shows the case of a document in which two types of "d" co-exist. In this case, the learned sprite, shown on the left, reassembles an average of the two, with both versions of the ascending parts visible with intermediate transparency. Such a limitation could be overcome by learning several sprites per character. We thus experimented with learning two per character, simply by summing their probabilities when optimizing the CTC-loss. We find that when different appearances of the same letter exist, the two sprites learn two different appearances, and we show the example of the two different learned "d" sprites on the right of the original one.

Our approach could benefit palaeographic analysis in more ways than simply analyzing the characters' shapes. Indeed, our model also gives access to the position and scale variation for each letter. This would enable a quantitative analysis of more global text appearance factors, such as the space between letters or their respective size variations. Because they are tremendously tedious to annotate, such variations have rarely been quantified, and their analysis could open new research topics, for example, the study of the handwriting evolution of a single writer copying a book across several months. Another natural application of our approach is a font or writer classification, which could be achieved either using a single model to compare error statistics for the different letters or relative positions of bi-grams, or by training different models for different fonts or writers. The main advantage compared to most existing approaches would be the high interpretability of the predictions, which a user could easily validate.

5 Conclusion

We have presented a document-specific generative approach to document analysis. Inspired by deep unsupervised multi-object segmentation methods, we extended them to accurately model standard printed documents as well as much more complex ones, such as a handwritten ciphered manuscript or ancient charters. We outlined that a completely unsupervised approach suffers from the ambiguity of the decomposition problem and the imbalanced character distributions. Therefore, we extended these approaches using weak supervision to obtain robust, high-quality results. Finally, we demonstrated the potential of our Learnable Typewriter approach for a novel application: palaeographic analysis.

References

1. Baird, H.S.: Model-directed document image analysis. In: Proceedings of the Symposium on Document Image Understanding Technology (1999)
2. Baró, A., Chen, J., Fornés, A., Megyesi, B.: Towards a Generic Unsupervised Method for Transcription of Encoded Manuscripts. In: Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage (2019)
3. Berg-Kirkpatrick, T., Durrett, G., Klein, D.: Unsupervised Transcription of Historical Documents. ACL (2013)
4. Bluche, T., Messina, R.: Gated convolutional recurrent neural networks for multilingual handwriting recognition. In: 2017 14th IAPR international conference on document analysis and recognition (ICDAR). vol. 1, pp. 646–651. IEEE (2017)
5. Burgess, C.P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., Lerchner, A.: MONet: Unsupervised scene decomposition and representation. arXiv preprint arXiv:1901.11390 (2019)
6. Camps, J.B., Vidal-Gorène, C., Stutzmann, D., Vernet, M., Pinche, A.: Data diversity in handwritten text recognition: challenge or opportunity? Digital Humanities (2022)
7. Crawford, E., Pineau, J.: Spatially invariant unsupervised object detection with convolutional neural networks. AAAI (2019)
8. Deng, F., Zhi, Z., Lee, D., Ahn, S.: Generative scene graph networks. ICLR (2020)
9. Emami, P., He, P., Ranka, S., Rangarajan, A.: Efficient iterative amortized inference for learning symmetric and disentangled multi-object representations. ICML (2021)
10. Eslami, S.M.A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Kavukcuoglu, K., Hinton, G.E.: Attend, Infer, Repeat: Fast Scene Understanding with Generative Models. Advances in Neural Information Processing Systems (2016)
11. Fogel, S., Averbuch-Elor, H., Cohen, S., Mazor, S., Litman, R.: ScrabbleGAN: Semi-supervised varying length handwritten text generation. CVPR (2020)
12. Garrette, D., Alpert-Abrams, H., Berg-Kirkpatrick, T., Klein, D.: Unsupervised code-switching for multilingual historical document transcription. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (2015)
13. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. NeurIPS (2014)
14. Goyal, K., Dyer, C., Warren, C., G'Sell, M., Berg-Kirkpatrick, T.: A probabilistic generative model for typographical analysis of early modern printing. In: ACL (2020)
15. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. ICML (2006)
16. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. NeurIPS (2008)
17. Greff, K., Kaufman, R.L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., Lerchner, A.: Multi-object representation learning with iterative variational inference. ICML (2019)
18. Greff, K., Van Steenkiste, S., Schmidhuber, J.: Neural expectation maximization. NeurIPS (2017)
19. Gupta, A., Vedaldi, A., Zisserman, A.: Learning to read by spelling: Towards unsupervised text recognition. arXiv:1809.08675 [cs] (2018)

20. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2016)
21. Hochberg, J., Kelly, P., Thomas, T., Kerns, L.: Automatic script identification from document images using cluster-based templates. IEEE Transactions on Pattern Analysis and Machine Intelligence (1997)
22. Jaderberg, M., Simonyan, K., Zisserman, A.: Spatial Transformer Networks. NeurIPS (2015)
23. Jiang, J., Ahn, S.: Generative neurosymbolic machines. NeurIPS (2020)
24. Kahle, P., Colutto, S., Hackl, G., Mühlberger, G.: Transkribus-a service platform for transcription, recognition and retrieval of historical documents. In: ICDAR (2017)
25. Kang, L., Riba, P., Rusiñol, M., Fornés, A., Villegas, M.: Pay attention to what you read: Non-recurrent handwritten text-line recognition. Pattern Recognition (2022)
26. Karazija, L., Laina, I., Rupprecht, C.: ClevrTex: A Texture-Rich Benchmark for Unsupervised Multi-Object Segmentation. NeurIPS Datasets and Benchmarks (2021)
27. Knight, K., Megyesi, B., Schaefer, C.: The Copiale Cipher. In: Proceedings of the ACL Workshop on Building and Using Comparable Corpora (2011)
28. Kopec, G.E., Lomelin, M.: Document-specific character template estimation. In: Document Recognition III (1996)
29. Kopec, G.E., Lomelin, M.: Supervised template estimation for document image decoding. IEEE Transactions on Pattern Analysis and Machine Intelligence (1997)
30. Kopec, G.E., Said, M.R., Popat, K.: N-gram language models for document image decoding. In: Document Recognition and Retrieval IX (2001)
31. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation (1989)
32. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE (1998)
33. Li, M., Lv, T., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., Wei, F.: Trocr: Transformer-based optical character recognition with pre-trained models. arXiv preprint arXiv:2109.10282 (2021)
34. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. ICLR (2019)
35. Monnier, T., Aubry, M.: docExtractor: An off-the-shelf historical document element extraction. ICFHR (2020)
36. Monnier, T., Groueix, T., Aubry, M.: Deep Transformation-Invariant Clustering. NeurIPS (2020)
37. Monnier, T., Vincent, E., Ponce, J., Aubry, M.: Unsupervised Layered Image Decomposition into Object Prototypes. ICCV (2021)
38. Nolan, J.C., Filippini, R.: Method and apparatus for creating a high-fidelity glyph prototype from low-resolution glyph images (2010), uS Patent 7,702,182
39. Puigcerver, J.: Are multidimensional recurrent layers really necessary for handwritten text recognition? ICDAR (2017)
40. Reddy, P., Guerrero, P., Mitra, N.J.: Search for concepts: Discovering visual concepts using direct optimization. arXiv preprint arXiv:2210.14808 (2022)
41. Seuret, M., van der Loop, J., Weichselbaumer, N., Mayr, M., Molnar, J., Hass, T., Christlein, V.: Combining ocr models for reading early modern books. ICDAR (2023)
42. Smirnov, D., Gharbi, M., Fisher, M., Guizilini, V., Efros, A.A., Solomon, J.: MarioNette: Self-Supervised Sprite Learning. NeurIPS 2021 (2021)

43. Souibgui, M.A., Fornés, A., Kessentini, Y., Tudor, C.: A few-shot learning approach for historical ciphered manuscript recognition. CoRR (2020)
44. de Sousa Neto, A.F., Bezerra, B.L.D., Toselli, A.H., Lima, E.B.: Htr-flor: a deep learning system for offline handwritten text recognition. In: SIBGRAPI (2020)
45. Srivatsan, N., Vega, J., Skelton, C., Berg-Kirkpatrick, T.: Neural representation learning for scribal hands of linear b. ICDAR 2021 Workshops (2021)
46. Srivatsan, N., Wu, S., Barron, J., Berg Kirkpatrick, T.: Scalable font reconstruction with dual latent manifolds. EMNLP (2021)
47. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. NeurIPS (2017)
48. Vincent, L.: Google Book Search: Document Understanding on a Massive Scale. ICDAR (2007)
49. Xu, Y., Nagy, G.: Prototype extraction and adaptive OCR. IEEE Transactions on Pattern Analysis and Machine Intelligence (1999)
50. Yang, Y., Chen, Y., Soatto, S.: Learning to manipulate individual objects in an image. CVPR (2020)
51. Ye, V., Li, Z., Tucker, R., Kanazawa, A., Snavely, N.: Deformable sprites for unsupervised video decomposition. CVPR (2022)
52. Zhang, C., Gupta, A., Zisserman, A.: Adaptive Text Recognition through Visual Matching. ECCV (2020)